Querying For Actions Over Videos

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

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

ABSTRACT

Several recent works address challenges concerning declarative query processing over videos. Data research encompasses issues related to the presence of certain types of objects on video frames. In this paper, we introduce algorithms that enable queries involving objects as well as actions over videos. We first consider the case of queries involving a number of objects and an action over a streaming video in the absence of preprocessing. In this case, we introduce algorithms (SVAQ and SVAQD) for identifying video segments that satisfy the query predicates on the fly. We then consider the offline case, during which a number of videos are amenable to a single-time preprocessing in order to answer adhoc queries of the type discussed in this paper. In this case, we detail the types of preprocessing required and introduce algorithm RVAQ that produces effectively top-k video segments satisfying the query for user specified ranking functions. Our overall approach and algorithms are grounded on sound statistical principles, being adaptive to the underlying video properties, while utilizing state-of-the-art black box object and action detection models. We present the results of a thorough experimental evaluation utilizing real benchmark videos and real-world movies. Our results indicate that our algorithms attain superior accuracy while offering substantial performance advantages compared to other applicable approaches.

1 INTRODUCTION

Video data are ubiquitous. Video streaming dominates internet traffic (nearly 80% of internet traffic is video equivalent to 282 EB per month, increasing rapidly [47]). Cameras that are prevalent in computing devices have transformed video production into a commodity. As a result, it is easy to generate, upload, and disseminate video data, whose automated analysis becomes a pressing concern.

Given the importance and significance of video data, numerous recent research efforts aim to develop suitable algorithms and techniques to analyze and query videos [10, 12, 26, 29–31]. Advances in machine learning (ML) are offering highly sophisticated algorithms to classify video frames [40], detect and track objects across video frames [21, 24, 25, 37, 38, 48, 55], identify and classify actions and interactions among objects present in video frames [6, 8, 17, 18, 22, 42, 44]. These are fairly active research areas in the computer vision (CV) community. At the same time, in the data management community, there has been increasing research interest in the development of declarative query processing[1] techniques over videos [10–12, 26, 28, 29, 29–31, 49, 50]. These techniques typically introduce a declarative query interface (e.g. SQL-like languages) that is utilized to enable users to specify the desired video data retrieval from video repositories, without concerning the exact execution process. Execution frameworks are proposed to enhance the execution process by augmenting both its accuracy and speed. The basic idea is to expose the outcome of sophisticated ML vision algorithms (e.g. object detection) as first class citizens to a declarative query interface. Such queries are executed either in an *online* manner (as the video is streaming, assuming no pre-processing) or in an *offline* manner (over one or more long videos suitably pre-processed). In either case, the query semantics are the same. In the online case, the query identifies the frames or frame sequences where the query condition(s) are true and in the offline case, the query reports the associated frames or frame sequences where the query is true in one or more videos.

In this paper, we aim to overcome the challenges associated with making *actions* in videos a first class citizen for declarative query processing. Action recognition [8, 52] has been an active research area in the ML vision community. Such algorithms accept a sequence of video frames and classify the frame sequence as containing a specific action out of a list of possible actions the model is trained on. For example, they can classify a frame sequence as containing a human jumping or a human playing guitar, etc. Equivalently, action detection models [5, 17, 20, 53] can detect the precise frame sequence that an action is taking place (typically labeling the frames at the start and the end of the action sequence). Such models are fairly mature in the ML community but still constitute active research topics.

Action recognition/detection models are typically trained end to end. As such they require numerous datasets with suitable types of actions for training. The ML community over the years has developed a plethora of such datasets containing different types of actions to train the models. A significant challenge to utilizing action detection models as part of query processing lies in the interaction of the action detection model with other related query predicates. Consider for example a query seeking to detect a frame sequence that depicts a robot dancing while a car is visible in the frame sequence (as depicted in Figure 1):

```
SELECT frameSequence FROM (PROCESS inputVideo PRODUCE
    frameSequence, det USING VisionModel)
WHERE det = Action('robot_dancing', 'car', 'human')
```

From an action recognition perspective, the typical models for detecting actions are trained on the actions themselves (e.g., robot dancing) and are not aware of other objects. Thus, an action recognition module that is trained to recognize a robot dancing cannot be used to answer such a query in an effective manner (as it necessitates a post-processing step). One could in principle train a model to recognize actions that also contain a car in the frame sequence. Such an approach is not scalable, however, as it would require a model for any possible combination of query predicates present in queries, which is clearly impractical.

A different practical approach would be to decouple the detection of the action from the detection of the other objects mentioned in a query. Namely, one could detect a sequence of frames containing the desirable action using an action recognizer, then utilize an object detector to detect frame sequences that contain

[1] Declarative query processing emphasizes providing broad instructions regarding the task to be completed, rather than the specifics on how to complete it imperatively. SQL is widely considered a declarative query language.

the desired objects in the query and intersect them. In this manner, the SQL-like query statements presented in the preceding document transform into the following:

```
SELECT MERGE(clipID) AS Sequence
FROM (PROCESS inputVideo PRODUCE clipID, obj USING
     ObjectDetector, act USING ActionRecognizer)
WHERE act='robot_dancing' AND obj.inc('car', 'human')
```

Such an approach, however, requires several parameters/thresholds to be decided apriori. For example, for how many frames the frame sequence containing the action should overlap the frame sequences containing objects in order to declare a query match? Object detection algorithms are typically noisy (yield false positives and negatives), so how would such noise affect the thresholds? Is it possible or even feasible to choose or decide such thresholds before a query executes or to tune such thresholds for each different query predicate? In this paper, we place this problem into perspective and we propose solutions for both an online setting in which query results have to be reported as the video streams and an offline setting in which queries are issued against a specified repository of videos that have been suitably pre-processed.

For the online case, we present a streaming algorithm, called SVAQ, to identify segments (parts) of a **V**ideo **S**tream that satisfy the **Q**uery predicates both in terms of the **A**ction specified and in terms of any additional query specified object predicates. To eliminate the burden of setting query thresholds manually, and cope with the inaccuracies inherent to action and object detection models, we introduce a theoretically grounded approach based on scan statistics [23, 35, 45]. This approach first estimates the distribution of predictions made by each individual model involved in the query when the query predicates are not satisfied. Then, for each query predicate, it computes what constitutes a significantly large number (a critical value, k_{crit}) of positive predictions (the query predicate is satisfied) conducted by each individual model in a sequence of frames. These are utilized to synthesize an answer and determine whether a query is satisfied in a sequence of frames. Intuitively, if the number of positive predictions across the models utilized in the query exceeds k_{crit} in a sequence of frames, then this frame sequence has a higher probability of satisfying the query. In addition, instead of determining such statistical properties statically, we propose a **D**ynamic method to adjust them as the video stream evolves. Our proposed method, called SVAQD, is able to detect sudden changes in the video stream properties and adjust its estimations accordingly while capable of ignoring gradual changes over time, thus dealing with concept drift in a natural manner [45].

For the offline case, in which queries are issued against video repositories, we present a framework to **R**ank the relevant **V**ideo segments in the answers of the **A**ction **Q**uery. Our ranking framework, called RVAQ, can easily adopt any monotonic ranking scheme. We present a sample one incorporating various query specific signals such as the length of the answer video segments and the number of the detected objects and actions across relevant video parts. Our proposal consists of two steps: an ingestion phase and a query phase. During the ingestion phase which is executed only once when videos are added to the repository (i.e., a pre-processing step), several metadata (e.g., clip score tables) are extracted in a query independent manner, which are later utilized during query processing. During the query phase, our algorithm first extracts query specific metadata (e.g., query specific result sequences) from each video (materialized in the ingestion phase) and utilizes a top-k query processing framework to produce the

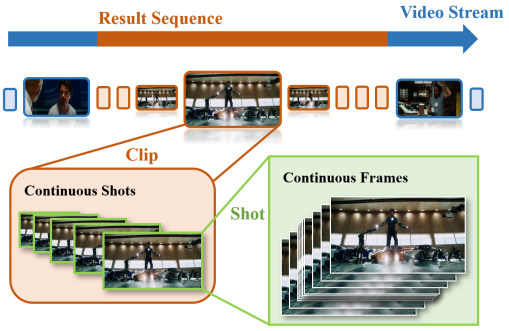


Figure 1: An example video stream. Green rectangles represent shots; rounded rectangles represent clips.

k most relevant results to the query. Given the specific nature of our problem, we demonstrate that a very different query processing methodology is required, compared to traditional top-k algorithms [16] for best performance.

In this paper, we make the following contributions:

- We formalize problems in the context of processing queries on video (streams) containing objects and actions.
- For the case of processing such queries over streaming videos, we introduce SVAQD that dynamically adjusts its estimates as the stream evolves.
- For the case of processing queries over videos that are amenable to pre-processing, we present RVAQ, which is able to process queries involving objects and any action supported by object/action detection models and efficiently return top-k frame sequences (for a user specified k) that satisfy the query.
- We conduct experiments utilizing real videos demonstrating the accuracy and performance benefits of our overall approach.

This paper is organized as follows. §2 presents background material and introduces terminology utilized in the remainder of the paper. §3 presents our solution and associated algorithms for the online case. In §4 we present algorithm RVAQ and detail our proposal for processing top-k queries involving objects and actions over pre-processed videos. §5 presents our experimental evaluation. §6 reviews related work, followed by §7 which concludes the paper and points to interesting problems for future work.

2 BACKGROUND

A video is a sequence of frames $V=\{v_1, \ldots v_{|V|}\}$. The number of frames (length) of a video $|V|$ can be fixed or unbounded. If $|V|$ is unbounded we refer to V as a video stream. A video repository is a collection of videos each of fixed length. In a video V we provide the following definitions.

(1) Frame, v: this is the building block of a video, as well as an input unit for object detection models. Each frame in V has an index (e.g., v_i) that indicates its position in V.
(2) Shot, s: a continuous (w.r.t. frame index) collection of frames of set length (see Figure 1). Action recognition models [8] accept a shot as input and yield action predictions the shot includes.
(3) Clip, c: continuous (w.r.t. frame index) collection of shots of set length (see Figure 1). Clips contain several shots that may yield multiple action and object predictions.
(4) Sequence, z: continuous collection of clips. Sequences are the results of our query evaluation and can be of any length as determined by our algorithms. Query result sequences are

represented as $P = \{(c_l, c_r)\}$, which is a set of the pairs of start and end clip identifiers for each result sequence returned by our algorithms.

Consider the example of Figure 1, depicting a movie with more than 100k frames. The video is divided into non-overlapping clips, each of which contains a number (e.g., fifty continuous) of frames. Then each clip is divided into a number (e.g., five) of shots with each shot containing a number of continuous frames. The length of shots in frames is decided by action recognition algorithms [8, 52] (typical values in the literature range from 10-30). The clip length is a parameter in our setting. Smaller clip lengths may fragment a long result sequence into multiple sequences; while a larger clip length may not. We will evaluate the effect of the length of the clips in our experiments.

Object Detection. A frame may hold a variety of object instances of different types, which can be detected by an object detection model. Each detected object instance is assigned a score that indicates how confident the detector is for this prediction and an estimated label indicating the object type of this instance. Let O be the set of all the object types that can be recognized by the deployed detector. For each object type $o_i \in O$, we denote ${}_{\max}S_{o_i}{}^{(v)}$ as the maximum score for all the instances of o_i on frame v given by the detector,

$$ {}_{\max}S_{o_i}{}^{(v)} = \max\left\{S^* \in O(o_i|v)\right\}, $$

where $O(o_i|v)$ represents the set of the scores of each detection of the instance of object type o_i at frame v. In accordance with prior work [24, 38, 49], one imposes a threshold $\mathcal{T}_{\text{obj}}$, on the scores to filter out the predictions of all object instances that the detector assigns a score below the threshold, keeping only detections with sufficient score. Such a threshold can be configured per object type but to ease notation, without loss of generality, we fix a threshold across all object types an object detector supports. We denote $\mathbb{1}_{o_i}{}^{(v)}$ as the prediction indicator of the object type o_i on frame v provided by the object detection model,

$$ \mathbb{1}_{o_i}{}^{(v)} = \mathbb{1}\left[{}_{\max}S_{o_i}{}^{(v)} \geq \mathcal{T}_{\text{obj}}\right], $$

where $\mathbb{1}[\cdot]$ is an indicator function. We say that the object type o_i has a positive prediction on frame v, iff $\mathbb{1}_{o_i}{}^{(v)}$=1.

Different instances of each object type can appear in the same frame. In our framework, we enable an object tracking model [55] that can track the same object instance across different frames. Each object instance gets assigned a unique tracking identifier by the object tracking model the first time it is detected. The identifier remains the same as the object is detected across frames for as long as it is present. We denote $S^t_{o_i}{}^{(v)}$ as the score of the instance of object type o_i with identifier t on the frame v given by the tracking model.

Action Recognition. Action recognition takes place on shots; utilizing an action recognition model [8], each shot is predicted to have zero or more actions from a universe of action categories the model is trained on. We denote A the set of all the action categories that can be recognized by the deployed action recognition model. The action recognizer will provide scores for all action categories predicted on each shot. For each action type $a_j \in A$, we denote $S_{a_j}{}^{(s)}$ the score of a_j detected on shot s. For action recognition, a threshold, $\mathcal{T}_{\text{act}}$, will also be set to filter out the action predictions with scores below the threshold [7, 8, 44]. We denote $\mathbb{1}_{a_j}{}^{(s)}$ as the prediction indicator of the action type a_j on shot s,

$$ \mathbb{1}_{a_j}{}^{(s)} = \mathbb{1}\left[S_{a_j}{}^{(s)} \geq \mathcal{T}_{\text{act}}\right]. $$

The action a_j has a positive prediction on the shot s, iff $\mathbb{1}_{a_j}{}^{(s)}$=1.

Query. Generally, a query specification encompasses several predicates, which can be specific actions (e.g., robot dancing from Kinetics-700 [7]), presence of objects in frames (e.g., human, car), or relationships between objects in frames (e.g., human left of the car). In this paper, without loss of generality and to simplify our notation, we consider queries encompassing conjunctions of query predicates, with the predicates involving the presence of both an action and some objects. Formally, a query q is:

$$ q : \{o_1, ..., o_I \in O; a \in A\}, $$

where I is the number of query predicates involving object types, o_i represents the type of the i_{th} query object, and a represents the query action. The forthcoming proposal can be easily adapted to accommodate more complicated queries that might incorporate other types of predicates (e.g., object relationships[2], multiple actions[3], disjunction of predicates[4]).

For example, for the query requesting a human jumping while a car is present in the scene: q:{o_1=human; o_2=car; a=jumping}. A SQL-like query for the online (video streaming) case that we will support in this work would be of the form:

```
SELECT MERGE(clipID) AS Sequence
FROM (PROCESS inputVideo PRODUCE clipID, obj USING
     ObjectDetector, act USING ActionRecognizer)
WHERE act='jumping' AND obj.include('car', 'human')
```

The query employs an object detector and an action recognizer to PROCESS the video, producing predictions of human, car and jumping, and then MERGEs the continuous clips that contain all query predicates to form result sequences. Similarly, an SQL-like query that we will support over a video repository (offline case) is:

```
SELECT MERGE(clipID) AS Sequence, RANK(act, obj)
FROM (PROCESS inputVideo PRODUCE clipID, obj USING
     ObjectTracker, act USING ActionRecognizer)
WHERE act='jumping' AND obj.include('car', 'human')
ORDER BY RANK(act, obj) LIMIT K
```

The offline query first PROCESSs the video and MERGEs the adequate clips to generate sequences. Then, it groups together the detection scores (e.g., $S_{a_j}{}^{(s)}$) in each sequence with a user-defined aggregation function, RANK, to provide an overall ranking score for each sequence. The top-K sequences will be returned ordered by their ranking scores. inputVideo in the FROM clause can refer to one or more videos suitably pre-processed as we detail in §4.

3 ONLINE CASE

In this section, we first propose an online model, namely SVAQ, for processing queries with complex predicates on video streams, and then propose an improved online model, namely SVAQD, equipped with dynamic parameter adjustment yielding a more robust processing framework.

3.1 Algorithm SVAQ

For the online case addressed in our work, algorithm SVAQ, is responsible for processing a query q:{o_1; ...; o_I; a}. For each object type o_i in q, we consider the positive object detection ($\mathbb{1}_{o_i}{}^{(v)}$=1)

[2]To support the predicate involving a constraint of object relationships, our proposal will yield a binary output indicating the presence of each relationship per frame based on the object detection outcomes, which technology is orthogonal to our approach, similar to the approach undertaken in [31].

[3]To support the constraint of multiple actions, our proposal will yield a binary output (indicators) for each queried action per clip based on the action recognition outcomes, and combine the indicators using a conjunction operation for all predicates.

[4]To support the queries involving a disjunction of predicates, one could first transform the predicates into conjunctive normal form, and then calculate the indicator for each clause in each clip separately, and then combine these indicators to answer the query.

on frames as an event in a probabilistic sense. In order to cope with object detection model noise (inaccuracies introducing false positives and negatives) and also determine what constitutes a suitable number of object detections of a specific type in a sequence of frames, we will estimate (§3.2) a statistically sound number of detections ($k_{\text{crit\_}o_i}$, for $1 \le i \le I$). This will enable us to compute all the clips considered to contain the object type o_i. The indicator of o_i on clip c is defined as,

$$\mathbb{1}_{o_i}{}^{(c)} = \mathbb{1}\left[\sum_{v \in V(c)} \mathbb{1}_{o_i}{}^{(v)} \geq k_{\text{crit\_}o_i}\right], \quad (1)$$

where $V(c)$ represents the set of frames in clip c.

Similarly, for the action type a specified in the query, we consider a positive action classification ($\mathbb{1}_a{}^{(s)}$=1) on shots as an event. We will also determine (§3.2) a statistically sound number of action classifications ($k_{\text{crit\_}a}$) to deal both with classification inaccuracies and also infer what constitutes a suitable number of action classifications of a specific type a in a sequence of shots. That way we will obtain the clips that are considered to contain action a. The indicator of a on clip c then is,

$$\mathbb{1}_a{}^{(c)} = \mathbb{1}\left[\sum_{s \in S(c)} \mathbb{1}_a{}^{(s)} \geq k_{\text{crit\_}a}\right], \quad (2)$$

where $S(c)$ represents the set of shots in clip c.

For the query q, a clip c is considered to contain all the query specified object types and action (referred to as *positive clips*) if all have positive predictions on clip c, that is,

$$\mathbb{1}_q{}^{(c)} = \mathbb{1}_{o_1}{}^{(c)} \wedge \cdots \wedge \mathbb{1}_{o_I}{}^{(c)} \wedge \mathbb{1}_a{}^{(c)}. \quad (3)$$

The positive clips are merged to produce result sequences; let P_q denote the set of result sequences for query q,

$$P_q = \left\{(c_l, c_r) \mid \forall c \in [c_l, c_r] : \mathbb{1}_q^{(c)}=1; \mathbb{1}_q^{(c_l-1)}=0; \mathbb{1}_q^{(c_r+1)}=0\right\}, \quad (4)$$

where (c_l, c_r) represents a result sequence that starts from the clip c_l and ends at c_r.

Algorithm SVAQ is presented as Algorithm 1. The input consists of continuous sequence of frames $\mathcal{X}$, the query q, and the values $k_{\text{crit\_o\_init}}/k_{\text{crit\_a\_init}}$. The derivation of values $k_{\text{crit\_o\_init}}/k_{\text{crit\_a\_init}}$ will be discussed in §3.2. For purposes of exposition in Algorithm 1, we initialize all values $k_{\text{crit\_}o_i}$ the same, but each may have its own initial values. The algorithm outputs a set of result sequences P_q. The algorithm determines clips from the video stream that satisfy the query (Lines 4-6) and then merges them if they are continuous into result sequences (Line 7).

When a new clip is considered, SVAQ conducts object detection and action recognition on its frames and shots respectively and determines whether the clip satisfies query q. Algorithm 2 depicts the process of determining whether a clip c satisfies the query. For each object type specified in the query, the algorithm determines the indicator of this object type on clip c (Lines 2-5). Next, it determines the indicator of the query action type on clip c (Lines 9-12). Finally, it determines whether clip c satisfies the query (Line 13). As query predicates are evaluated sequentially, a new predicate is evaluated only when the former predicate is determined to be positive; otherwise this clip can be skipped[5] (Lines 6-8).

[5]We observe that the positive rates among different predicates vary, implying their distinct selectivity. The order in which predicates are evaluated could influence the overall efficiency. A thorough investigation into the impact of the predicate order will be conducted in future research. In this paper, the predicate order is determined based on user expertise.

Algorithm 1: SVAQ

Input: video stream $\mathcal{X}$; q: $\{o_1; ...; o_I; a\}$; $k_{\text{crit\_o\_init}}$, $k_{\text{crit\_a\_init}}$;
Output: set of result sequences P_q;

1 $k_{\text{crit\_}o_1}, k_{\text{crit\_}o_2}, ..., k_{\text{crit\_}o_I} \leftarrow k_{\text{crit\_o\_init}}$.
2 $k_{\text{crit\_}a} \leftarrow k_{\text{crit\_a\_init}}$.
3 $P_q \leftarrow \emptyset$. # initialize the result sequence set.
4 **while** $!\mathcal{X}.end()$ **do**
5 $c \leftarrow \mathcal{X}.next()$. # grab the next clip in video stream $\mathcal{X}$.
6 $\mathbb{1}_q{}^{(c)} \leftarrow$ get the indicator of clip c through Algorithm 2 w.r.t. values $k_{\text{crit\_}o_1}, ..., k_{\text{crit\_}o_I}$ and $k_{\text{crit\_}a}$.
7 $P_q \leftarrow \left\{(c_l, c_r) \mid \forall c \in [c_l, c_r] : \mathbb{1}_q{}^{(c)}=1; \mathbb{1}_q{}^{(c_l-1)}=0; \mathbb{1}_q{}^{(c_r+1)}=0; \right\}$.

Algorithm 2: Clip Indicator

Input: query q: $\{o_1; ...; o_I; a\}$; object detection model O and action recognition model $\mathcal{A}$; thresholds $\mathcal{T}_{\text{obj}}$ and $\mathcal{T}_{\text{act}}$; clip c;
Output: indicator of clip c: $\mathbb{1}_q{}^{(c)}$;

1 **for** $i = 1, ..., I$ **do**
2 **for** *each frame v in clip c* **do**
3 ${}_{\max}S_{o_i}{}^{(v)} \leftarrow \max\{S^* \in O(o_i|v)\}$.
4 $\mathbb{1}_{o_i}{}^{(v)} \leftarrow \mathbb{1}\left[{}_{\max}S_{o_i}{}^{(v)} \geq \mathcal{T}_{\text{obj}}\right]$.
5 $\mathbb{1}_{o_i}{}^{(c)} \leftarrow \mathbb{1}\left[\sum_{v \in V(c)} \mathbb{1}_{o_i}{}^{(v)} \geq k_{\text{crit\_}o_i}\right]$.
6 **if** $\mathbb{1}_{o_i}{}^{(c)} = 0$ **then**
7 $\mathbb{1}_q{}^{(c)} \leftarrow 0$.
8 **Return** $\mathbb{1}_q{}^{(c)}$.
9 **for** *each shot s in clip c* **do**
10 $S_{a_j}{}^{(s)} \leftarrow \mathcal{A}(a|s)$,
11 $\mathbb{1}_a{}^{(s)} \leftarrow \mathbb{1}\left[S_{a_j}{}^{(s)} \geq \mathcal{T}_{\text{act}}\right]$.
12 $\mathbb{1}_a{}^{(c)} \leftarrow \mathbb{1}\left[\sum_{s \in S(c)} \mathbb{1}_a{}^{(s)} \geq k_{\text{crit\_}a}\right]$.
13 $\mathbb{1}_q{}^{(c)} \leftarrow \mathbb{1}_{o_1}{}^{(c)} \wedge \cdots \wedge \mathbb{1}_{o_I}{}^{(c)} \wedge \mathbb{1}_a{}^{(c)}$.
14 **Return** $\mathbb{1}_q{}^{(c)}$.

3.2 Statistical Properties of Event Sequences

The detection of objects taking place in frames is considered a statistical event. Every time an object specified in the query is detected on a frame, we consider the presence of this object in the frame as an event associated with this object type occurring on the frame. Similarly, when a shot is classified to contain the action specified by the query, we consider this an event associated with the action taking place at the shot. As a result, as the video stream progresses, we generate sequences of events associated with the objects and actions included in the query.

Our discussion in §3.1 left unspecified formal derivation of a statistically significant number of events (objects or action) in a sequence, which we will specify in this section. Scan statistics [23, 35] is a powerful method to detect unusually high rates of events in a large sequence. The primary idea is to compare the observed distribution of events within a specific scanning window against the expected distribution derived from statistical assumptions. In our setting, we can deploy principles of scan statistics to detect a high concentration of positive predictions by the object detector ($\mathbb{1}_{o_i}{}^{(v)}$=1) and action recognizer ($\mathbb{1}_{a_j}{}^{(s)}$=1).

We refer to the lowest granularity at which an event may occur (object detected or action predicted) as the *occurrence unit* (OU, which can be a frame or a shot in our case). Let N be the total number of OUs. Since OU is discrete, we model event occurrence by a Bernoulli distribution. Thus, the occurrence of events can be modeled as N Bernoulli trials with a set background probability of success on a given trial p.

Let $n_{y,y+w}$ denote the number of successes in trials y, y+1, y+2, ..., y+w−1. After N OUs have been observed, define $\mathcal{S}_w(N)$ as the maximum number of OUs to be found in any *scanning interval* (sequence of OUs) of length w, $\mathcal{S}_w(N) = \max_{1\le y\le N-w+1} n_{y,y+w}$. When $\mathcal{S}_w(N)\ge k$, we say that a quota of at least k successes within some w consecutive trials has occurred. Following [35], we can approximate the distribution $P(\mathcal{S}_w(N)\ge k|p,w,L)$[6], where L=N/w and p is a background probability. Based on the approximation, assuming that $p = p_0$ (for a given p_0) we can compute the smallest k, a *critical value*, for objects and actions for which the following holds:

$$P(\mathcal{S}_w(N) \ge k_{\text{crit}}|p_0, w, L) \le \alpha, \tag{5}$$

where α is the significance level. If the number of positive predictions in a scanning interval is greater than the critical value, then we say that at significance level α the event (e.g., object type, an action) is present at the scanning interval. For the case of object types, which are part of the query, the OU is a frame, and for the case of actions, it is a shot. Using the suitable OU in each case (frame for object and shot for actions), we can compute critical values independently for objects $k_{\text{crit\_o\_init}}$ (one per object used in a query) and actions $k_{\text{crit\_a\_init}}$ using this methodology (as utilized in Algorithm 1).

As a result, we can now define indicator functions for objects and action to be present at a clip c as follows:

$$\sum_{v\in V(c)} \mathbb{1}_{o_i}{}^{(v)} \ge k_{\text{crit\_}o_i} \Rightarrow \mathbb{1}_{o_i}{}^{(c)} = 1,$$
$$\sum_{s\in S(c)} \mathbb{1}_{a_j}{}^{(s)} \ge k_{\text{crit\_}a} \Rightarrow \mathbb{1}_{a_j}{}^{(c)} = 1,$$

where $V(c)$ and $S(c)$ is the set of all frames and all shots in a clip. A query will then be satisfied on a clip as per Equation 3. We note that although our modeling is based on Bernoulli trials that are assumed to be independent and identically distributed, following the approach of [27] the entire analysis can be extended to incorporate random variables that have known Markov dependencies[7]. From a practical point of view, such dependencies are not easy to determine apriori, especially in a streaming scenario. We will demonstrate experimentally in §5 that our analysis constitutes a viable proxy for determining high rates of events in real videos.

So far, however, the background probability (either a single one across all object predicates and the action or one per object predicate separately and the action) has to be set apriori. This is not easy to do in practice as one has no context or guidance on how to set them. We will remove this constraint next.

3.3 Dynamic Parameter Updates (SVAQD)

We assumed so far, that the Bernoulli probability of positive predictions, p_0, is fixed across predicates of query q (objects and action) or equivalently that each object predicate and the action have their own (possibly different) fixed probability of positive predictions set apriori. In practice, however, the background probability p_0 can change suddenly. For instance, a surveillance camera at a crossroad can experience peak traffic at certain times of the day, that is, the probability of a vehicle being detected varies over time. Thus, deciding the value of p_0 for the entire video is not acceptable.

We aim to estimate the suitable value of p utilizing data from observations over a period of time. We present our derivation assuming the granularity of an event is an OU (as in §3.2). The maximum likelihood estimate of p is $\frac{N^*}{N}$, where N^* is the number of events (positive predictions for an object type or action) and N is the total number of OUs. An intuitive method to estimate p is to define a view length $\mathcal{V}$. When updating the value of p, only the events in $\mathcal{V}$ OUs before and after the current OU are considered. However, this will introduce new parameters. Thus, we use a more feasible parameter update method to update p automatically, while ignoring gradual enough changes.

Let t represent an occurrence unit and $\mathcal{R}$ a region around it; such a region will contain occurrence units before and after t in sequence. We observe that the probability of an event, drawn from a Binomial distribution $p(t)$, will fall in a region $\mathcal{R}$: $P = \sum_{i\in\mathcal{R}} p(t_i)$, where t_i is the i-th occurrence unit in region $\mathcal{R}$. If we have N^* events drawn from Binomial distribution $p(t)$, the probability that k of these N^* events fall in the region $\mathcal{R}$ is $\mathbb{P}(k) = \binom{N^*}{k}P^k(1-P)^{N^*-k}$. It is known [4, 45] that the expectation and variance of $\frac{k}{N^*}$ is $\mathbb{E}[\frac{k}{N^*}] = P$, $Var[\frac{k}{N^*}] = \frac{P(1-P)}{N^*}$. Thus, when $N^* \to \infty$, $P \approx \frac{k}{N^*}$. If we assume $\mathcal{R}$ is so small that $p(t)$ does not vary appreciably within it, then $P = \sum_{i\in\mathcal{R}} p(t_i) = p(t)u$, where u is the volume enclosed by region $\mathcal{R}$. Thus, we obtain

$$\hat{p}(t) = \frac{k}{N^*u} = \frac{1}{N^*u}\sum_{n=1}^{N^*} K\left(\frac{t-t_n}{u}\right),$$

K is a kernel function; N^* is the number of events; t_n is the occurrence unit that the n_{th} event occurs.

By choosing a smooth exponential kernel, for example, $K(\frac{t-t_n}{u}) = \exp(-\frac{t-t_n}{u})$, the parameter update can be made very efficient even for a large N^*. Given the estimate $\hat{p}(t)$, we can update it after Δt OUs:

$$\hat{p}(t+\Delta t) = \frac{1}{N^*u}\sum_{n=1}^{N^*}\exp\left(-\frac{t+\Delta t-t_n}{u}\right) = \exp\left(-\frac{\Delta t}{u}\right)\hat{p}(t).$$

If an event occurs when updating the estimate $\hat{p}(t)$, a bias will be generated. Edge correction [14] can help to remove the bias,

$$\hat{p}(t) = \frac{1}{N^*u}\sum_{n=1}^{N^*}\exp\left(-\frac{t-t_n}{u}\right)\Big/\sum_{j=1}^{N}\exp\left(-\frac{t-t_j}{u}\right)$$
$$= \frac{1}{N^*u}\sum_{n=1}^{N^*}\exp\left(-\frac{t-t_n}{u}\right)\frac{1-\exp\left(-\frac{1}{u}\right)}{1-\exp\left(-\frac{t}{u}\right)},$$

where N is the total number of OUs observed. This estimator is unbiased in the case when the background probability is constant. The overall update equations with edge correction

$$\hat{p}(t+\Delta t) = \hat{p}(t)\frac{1-\exp\left(-\frac{t}{u}\right)}{\exp\left(\frac{\Delta t}{u}\right)-\exp\left(-\frac{t}{u}\right)} + \frac{1-\exp\left(-\frac{1}{u}\right)}{u\left(1-\exp\left(-\frac{t+\Delta t}{u}\right)\right)}. \tag{6}$$

This equation refines the background probability by smoothing events over time. Thus for each object type specified in the query and the action, we can now adjust the background probability dynamically over time.

The resulting algorithm, named SVAQD, employs the method of dynamic parameter adjustment for each object type present in query predicates and for the action predicate (Equation 6). For the case of object types, the occurrence unit is fixed to a frame, and for the action, it is a shot. The process is shown in Algorithm

[6]The approximation, $P(\mathcal{S}_w(N)\ge k|p,w,L) \approx 1 - Q_2(Q_3/Q_2)^{L-2}$, is omitted here due to space constraints. For details, please refer to [33].

[7]An approximation approach of $P(\mathcal{S}_w(N)\ge k)$, developed using the finite Markov chain embedding (FMCE) technique [19], can be accommodated for approximating the distribution of the scan statistics defined on Markov-dependent Bernoulli trials. In detail, FMCE first introduces the compound pattern to embed event sequences that satisfy $\mathcal{S}_w(N)\ge k$, which are also considered as states within the Markov chain. Then, it computes the probability transition matrix for the Markov chain (with known first-order stationary distribution) and uses it to approximate $P(\mathcal{S}_w(N)\ge k)$. The details are beyond the scope of the current paper; a future publication will focus on this, providing a basic description of the extension we have outlined.

Algorithm 3: SVAQD

Input: video stream $\mathcal{X}$; query q: $\{o_1; ...; o_I; a\}$; initialized background probabilities p_{obj_0}, p_{act_0};
Output: set of result sequences P_q;

1 $p_{o_1}, ..., p_{o_I} \leftarrow p_{\text{obj}_0}$.
2 $p_a \leftarrow p_{\text{act}_0}$.
3 $k_{\text{crit\_}o_1}, ..., k_{\text{crit\_}o_I}, k_{\text{crit\_}a} \leftarrow$ calculate critical values through Scan Statistics (Equation 5) w.r.t. $p_{o_1}, ..., p_{o_I}, p_a$.
4 $P_q \leftarrow \emptyset$.
5 **while** $!\mathcal{X}.end()$ **do**
6 $\quad c \leftarrow \mathcal{X}.next()$. ${\mathbb{1}_q}^{(c)} \leftarrow$ get indicator of clip c through Algorithm 2 w.r.t. $k_{\text{crit\_}o_1}, ..., k_{\text{crit\_}o_I}$ and $k_{\text{crit\_}a}$.
7 $\quad$ **if** ${\mathbb{1}_q}^{(c)} = 1$ **then**
8 $\quad\quad p_{o_1}, ..., p_{o_I}, p_a \leftarrow$ update background ps (Equation 6).
9 $\quad\quad k_{\text{crit\_}o_1}, ..., k_{\text{crit\_}o_I}, k_{\text{crit\_}a} \leftarrow$ calculate the new critical values through Scan Statistics w.r.t. $p_{o_1}, ..., p_{o_I}, p_a$.
10 $P_q \leftarrow \left\{(c_l, c_r) \mid \forall c \in [c_l, c_r] : \mathbb{1}_q^{(c)}=1; \mathbb{1}_q^{(c_l-1)}=0; \mathbb{1}_q^{(c_r+1)}=0; \right\}$.

3. Different from SVAQ, SVAQD is initialized with values p_{obj_0} (can be the same for all objects present in the query or different) and p_{act_0} as input. The update of p can be performed every time a new event occurs (Lines 7-9), or after processing a fixed number of clips. This algorithm will eliminate the influence of p_{obj_0} and p_{act_0} naturally, adjusting them as the statistical properties of the video change, increasing the robustness of the online model.

4 OFFLINE CASE

We now focus on answering queries over one or more videos in an offline manner. In this case, each video is amenable to be pre-processed to extract metadata which is fully available during query processing. During an *ingestion phase* we process each video utilizing object detection and action detection models. Since the queries are unknown, we process each frame extracting metadata for all possible object and action types, supported by the deployed object detection model and action recognition model. At query time, when desired objects and actions are specified, the applicable metadata are utilized for query processing.

We propose algorithm RVAQ to process queries utilizing metadata extracted during the ingestion phase. In order to effectively limit the number of results returned, RVAQ uses scoring functions to rank the results, returning only the K highest scoring ones for a user specified K. We first detail a general class of scoring functions supported by RVAQ in §4.1, and then present the ingestion phase in §4.2, followed by algorithm RVAQ in §4.3 and §4.4.

4.1 Scoring Functions

The scoring functions utilized in our work map a result sequence z to a score (a real value). The scoring functions utilize the score provided by the object detection/tracking models (${S_{o_i}^t}^{(v)}$ defined in §2) and the action recognition model (${S_{a_j}}^{(s)}$ also defined in §2) at the frame and shot level respectively. Our entire framework is agnostic on the specific object detection/tracking and action recognition models; any model can be utilized for this purpose.

The score of a result sequence z is concerned with the query content, where we assume that the query is $q : \{o_1; ...; o_I; a\}$ and the sequence $z = \{c_1, c_2, ..., c_{|z|}\}$. We denote ${S_q}^{(z)}$ the target score of the sequence z under the query q. For each clip in this sequence z, we denote ${S_q}^{(c)}$ the score of a clip c under the query q. The score of every type of queried object or action in a clip can be also calculated. We denote ${S_{o_i}}^{(c)}$ the score of object type o_i on a clip c , and similarly ${S_a}^{(c)}$ the score of action type a on the clip c.

In a clip c, for any object type o_i, ${S_{o_i}}^{(c)}$ should be calculated by combining all the scores related to this object o_i given by the object detection model embedded in our algorithm. Let h represent the scoring function[8].

$$ {S_{o_i}}^{(c)} = h({S_{o_i}^t}^{(v)} | \forall t \in T_{o_i}(c), \forall v \in V(c)), \tag{7} $$

where $T_{o_i}(c)$ represents the set of all tracking IDs of the object type o_i that appear in clip c. Analogous, in a clip c, for any action type a_j, ${S_{a_j}}^{(c)}$ could be calculated by combining all the scores related to this action,

$$ {S_{a_j}}^{(c)} = h({S_{a_j}}^{(s)} | \forall s \in S(c)). \tag{8} $$

The overall score of a clip c under the query q, ${S_q}^{(c)}$, could combine the scores of all queried objects and the queried action. Let g represent the mapping function,

$$ {S_q}^{(c)} \rightarrow g\left({S_{o_1}}^{(c)}, {S_{o_2}}^{(c)}, ..., {S_{o_I}}^{(c)}, {S_a}^{(c)}\right). \tag{9} $$

Furthermore, the overall score of a sequence z under the query q, ${S_q}^{(z)}$, could combine the scores of all clips in the sequence. Let f represent the scoring function,

$$ {S_q}^{(z)} \rightarrow f\left({S_q}^{(c_1)}, {S_q}^{(c_2)}, ..., {S_q}^{(c_{|z|})}\right). \tag{10} $$

Our proposal is agnostic of the specifics of scoring functions adopted. In particular, any function f, g, can be utilized in our framework as long as they satisfy the following properties. In particular for the sequence score function f:

- Score of a sequence z is monotonic to the score of each clip in z: $\partial {S_q}^{(z)} / \partial {S_q}^{(c_i)} \geq 0,\ \forall 1 \leq i \leq |z|$.
- Score of a sub-sequence of a sequence z is always lower than that of z: ${S_q}^{(z)} \geq {S_q}^{(z')},\ \forall z' \subseteq z$.
- If a sequence is divided into two non-overlapping sub-sequences, we can obtain the score of the original sequence by manipulating the scores of its sub-sequences. Let $\odot$ represent the aggregation operator on scores:

$$ \text{if } z_1 \cup z_2 = z, z_1 \cap z_2 = \phi, \quad {S_q}^{(z)} = {S_q}^{(z_1)} \odot {S_q}^{(z_2)}. \tag{11} $$

For the clip score function g:

- Score of a clip c is monotonic to the score of each predicate in c,

$$ \partial {S_q}^{(c)} / \partial {S_{o_i}}^{(c)} \geq 0, \quad \forall o_i \in q; \qquad \partial {S_q}^{(c)} / \partial {S_a}^{(c)} \geq 0. $$

We impose no constraints for h, the clip score function of a specific object or action type. Our algorithms will be described using an abstract representation for functions f, g and h. We will detail sample scoring function choices used in our experiments in §5.

4.2 Ingestion Phase

Clip Score Tables. During the ingestion phase, metadata are extracted from each video in a query independent manner. We describe the process for a single video to simplify notation. Multiple videos are handled in the same manner by associating a *video identifier* to each clip identifier (cid) below. We process the video one clip c at a time and calculate the scores for each object type and each action type on the clip, materializing them in the following tables,

$$ table_{o_i} : \{cid, Score\} \quad \forall o_i \in O; \quad table_{a_j} : \{cid, Score\} \quad \forall a_j \in A. $$

For each object type detected by the object detection model, we calculate ${S_{o_i}}^{(c)}$ by Equation 7, and store it in $table_{o_i}$ as $(c, {S_{o_i}}^{(c)})$.

[8]The function h for every object or action can be different. We omit the subscript for brevity.

Similarly, For each action type that is recognized by the action recognition model, we calculate ${S_{a_j}}^{(c)}$ by Equation 8, and store it in $table_{a_j}$ as $(c, {S_{a_j}}^{(c)})$. The tuples in all these tables are ordered by *Score*. Although we present the tables for a single video for brevity, notice that it is very easy to add more videos or delete videos in this setting, by manipulating the information in these tables. We just associate a video identifier for each *cid* in the tables.

Individual Sequences. Utilizing algorithm SVAQD (§3.3), for each object type and action type, we determine the positive clips utilizing Equations 1 and 2 respectively. As a result, for each object and for each action type, the entire video is divided into sequences by merging the corresponding consecutive positive clips. Let P_{o_i} denote the individual sequences for object type o_i, P_{a_j} denote the individual sequences for action type a_j. Let $C(V)$ be the total number of clips in video V; then for $1 \le c_i \le C(V)$ we have:

$$P_{o_i} = \{(c_l, c_r) \mid \forall c \in [c_l, c_r] : {\mathbb{1}_{o_i}}^{(c)} = 1; {\mathbb{1}_{o_i}}^{(c_l-1)} = {\mathbb{1}_{o_i}}^{(c_r+1)} = 0\};$$

$$P_{a_j} = \{(c_l, c_r) \mid \forall c \in [c_l, c_r] : {\mathbb{1}_{a_j}}^{(c)} = 1; {\mathbb{1}_{a_j}}^{(c_l-1)} = {\mathbb{1}_{a_j}}^{(c_r+1)} = 0\}.$$

For each object and action type, we first traverse all the clips in the target video, and compute the indicators on each clip (Equation 1 and 2). Then, utilizing the equation above, we traverse the indicators on all clips to extract the individual sequences for each object type and each action type. The resulting sequences are stored as $\{(c_l, c_r)\}$, a set of the pairs of start and end clip identifiers. That way P_{o_i} and P_{a_j} are materialized during ingestion for each object type i and action type j in the deployed object detection and action recognition models. This is easily accomplished with a single pass over all the clips of a video. A clip c satisfies query q, if all the query object types and the action have positive predictions on this clip (Equation 3). Let $\otimes$ denote the *intersection* of two individual sequences defined in this paper. For two individual sequences P_1 and P_2, $P_1 \otimes P_2$ represents new sequences that contain the clips both in P_1 and P_2,

$$P_1 \otimes P_2 = \{(c_l, c_r) \mid \forall c \in [c_l, c_r] : c \in C(P_1) \cap C(P_2); \\ c_l - 1 \notin C(P_1) \cap C(P_2); c_r + 1 \notin C(P_1) \cap C(P_2)\}$$

where $C(P_1)$ and $C(P_2)$ represent the set of all the clip identifiers that appear in the sequences of P_1 and P_2 respectively. Thus, one can obtain sequences that satisfy query q, as:

$$P_q = P_a \otimes P_{o_1} \otimes P_{o_2} \otimes \ldots \otimes P_{o_I} \tag{12}$$

For any query q, P_q can be computed very efficiently; since all the P_{o_i} and P_a involved in q are set of clip identifier ranges, each range can be viewed as an interval. All intervals can be sorted by their start point and P_q can be computed in a single pass by an interval sweep [46]. Next, we will introduce sequence scores and how to obtain the top K sequences efficiently.

A straightforward way to identify the K sequences in P_q with the highest score is to retrieve each clip in the sequences of P_q, compute each sequence score (Equation 10) and report the K sequences with the highest score. This however will involve a large number of random accesses to the underlying clip score tables $table_{o_i}, table_{a_j}$. We will propose next a more efficient algorithm to identify the result sequences that effectively prunes early unnecessary sequences effectively limiting the number of clips that need to be accessed.

4.3 The RVAQ Algorithm

The basic idea of RVAQ is to estimate bounds (upper and lower) on the scores of each sequence in P_q and refine them progressively so that the top K sequences with the highest scores can be computed much faster without the need to inquire about the scores of all clips across all sequences in P_q.

The efficient access of the clips in P_q is conducted with the help of an iterator TBClip (Algorithm 5) that progressively delivers clips of the sequences in P_q along with their scores only; clips not belonging to sequences in P_q are excluded. In particular, TBClip returns two clips each time (with scores calculated by Equation 9): c_{top}, which is the clip in P_q with the largest score among those clips not already processed by the iterator, and c_{btm}, which is the clip in P_q with the lowest score among those not already processed. Each time we obtain new values for $c_{\text{top}}, c_{\text{btm}}$, the upper and lower bounds of the scores for all sequences in P_q are re-estimated; the process continues until a stopping condition is encountered and the K sequences with the highest scores are identified.

Let $({c_l}^{(i)}, {c_r}^{(i)}) \in P_q$ be a sequence ($1 \le i \le |P_q|$); we denote $B_{\text{up}}^{{c_l}^{(i)}}$, $B_{\text{lo}}^{{c_l}^{(i)}}$ the estimates for its upper and lower bound scores. Given c_{top} and c_{btm}, we refer to the clips in the sequence that have larger/lower scores than c_{top} and c_{btm} respectively as top/bottom *processed* clips in this sequence; we refer to all other clips in this sequence as *not processed*. Let $L_{\text{up}}^{{c_l}^{(i)}}$ and $L_{\text{lo}}^{{c_l}^{(i)}}$ represent the length of the sequence minus the number of *processed* top/bottom clips belonging to this sequence,

$$L_{\text{up}}^{{c_l}^{(i)}} = c_r^{(i)} - c_l^{(i)} - \left|\{\text{processed top clip } c \in [{c_l}^{(i)}, {c_r}^{(i)}]\}\right|, 1 \le i \le |P_q|;$$

$$L_{\text{lo}}^{{c_l}^{(i)}} = c_r^{(i)} - c_l^{(i)} - \left|\{\text{processed bottom clip } c \in [{c_l}^{(i)}, {c_r}^{(i)}]\}\right|, 1 \le i \le |P_q|.$$

where $S_{\text{up}}^{{c_l}^{(i)}}$ and $S_{\text{lo}}^{{c_l}^{(i)}}$ represent the overall scores of top/bottom clips that have been processed in sequence $({c_l}^{(i)}, {c_r}^{(i)})$,

$$S_{\text{up}}^{{c_l}^{(i)}} = f({S_q}^{(c)} | \text{processed top clip } c \in [{c_l}^{(i)}, {c_r}^{(i)}]), \quad 1 \le i \le |P_q|,$$

$$S_{\text{lo}}^{{c_l}^{(i)}} = f({S_q}^{(c)} | \text{processed bottom clip } c \in [{c_l}^{(i)}, {c_r}^{(i)}]), \quad 1 \le i \le |P_q|,$$

where f is the sequence score function defined in Equation 10. After obtaining a new c_{top} from the iterator, the upper bounds for all the sequences in P_q will be re-estimated,

$$B_{\text{up}}^{{c_l}^{(i)}} = \underbrace{f({S_q}^{(c_{\text{top}})}, \ldots, {S_q}^{(c_{\text{top}})})}_{L_{\text{up}}^{{c_l}^{(i)}}} \odot S_{\text{up}}^{{c_l}^{(i)}}, \quad 1 \le i \le |P_q|. \tag{13}$$

where ${S_q}^{(c_{\text{top}})}$ represents the score of clip c_{top} and $\odot$ represents the aggregation operator on sub-sequence scores defined in Equation 11; this is also the highest score of a clip among the not processed top clips of all result sequences. If $c_{\text{top}} \in [{c_l}^{(j)}, {c_r}^{(j)}]$, for $1 \le j \le |P_q|$, $L_{\text{up}}^{{c_l}^{(j)}}$ and $S_{\text{up}}^{{c_l}^{(j)}}$ will also be updated for the sequence $({c_l}^{(j)}, {c_r}^{(j)})$,

$$L_{\text{up}}^{{c_l}^{(j)}} = L_{\text{up}}^{{c_l}^{(j)}} - 1, \quad S_{\text{up}}^{{c_l}^{(j)}} = S_{\text{up}}^{{c_l}^{(j)}} \odot f({S_q}^{(c_{\text{top}})}),$$

Similarly, after obtaining a new, c_{btm} from the iterator, the estimation of the lower bound for each sequence $({c_l}^{(i)}, {c_r}^{(i)})$ will also be refined:

$$B_{\text{lo}}^{{c_l}^{(i)}} = \underbrace{f({S_q}^{(c_{\text{btm}})}, \ldots, {S_q}^{(c_{\text{btm}})})}_{L_{\text{lo}}^{{c_l}^{(i)}}} \odot S_{\text{lo}}^{{c_l}^{(i)}}, \quad 1 \le i \le |P_q|. \tag{14}$$

We also update $L_{\text{lo}}^{{c_l}^{(j)}}$ and $S_{\text{lo}}^{{c_l}^{(j)}}$ for the sequence $({c_l}^{(j)}, {c_r}^{(j)})$ if $c_{\text{btm}} \in [{c_l}^{(j)}, {c_r}^{(j)}]$: $L_{\text{lo}}^{{c_l}^{(j)}} = L_{\text{lo}}^{{c_l}^{(j)}} - 1$, $S_{\text{lo}}^{{c_l}^{(j)}} = S_{\text{lo}}^{{c_l}^{(j)}} \odot f({S_q}^{(c_{\text{btm}})})$. For each successive invocation of the iterator TBClip, the upper and lower bounds of all sequences are refined and converge to

the exact values unless this process halts earlier triggered by a stopping condition.

The algorithm utilizes two priority queues to efficiently determine whether a stopping condition is satisfied. Let PQ_{lo}^{K} be a priority queue containing the K result sequences in P_q with the highest lower bounds, organized in non-decreasing order of the bound estimates.

$$PQ_{\text{lo}}^{K} = \left\{ \left({c_l}^{(i)}, {c_r}^{(i)} \right) \text{ with } K \text{ highest } B_{\text{lo}}^{{c_l}^{(i)}} \mid \left({c_l}^{(i)}, {c_r}^{(i)} \right) \in P_q \right\}.$$

Each time a new lower bound estimate for a sequence (Equation 14) is available, we update the priority queue PQ_{lo}^{K} reflecting the new score. Similarly, let $PQ_{\text{up}}^{\neg K}$ be a priority queue for all result sequences not in PQ_{lo}^{K} ranked in non-decreasing order of their upper bounds,

$$PQ_{\text{up}}^{\neg K} = \left\{ \left({c_l}^{(i)}, {c_r}^{(i)} \right) \text{ ranked by } B_{\text{up}}^{{c_l}^{(i)}} \mid \left({c_l}^{(i)}, {c_r}^{(i)} \right) \in \left(P_q \setminus PQ_{\text{lo}}^{K} \right) \right\}.$$

Each time a new upper bound estimate is available (Equation 13), we update $PQ_{\text{up}}^{\neg K}$. If a sequence z is inserted in PQ_{lo}^{K} removing sequence z' from it, z is removed from $PQ_{\text{up}}^{\neg K}$ and z' is inserted instead. Let B_{lo}^{K} be the minimum of the lower bounds among the sequences in PQ_{lo}^{K},

$$B_{\text{lo}}^{K} = \min \left\{ B_{\text{lo}}^{{c_l}^{(i)}} \mid \left({c_l}^{(i)}, {c_r}^{(i)} \right) \in PQ_{\text{lo}}^{K} \right\}.$$

Let $B_{\text{up}}^{\neg K}$ be the maximum of the upper bounds of all the result sequences not currently in PQ_{lo}^{K},

$$B_{\text{up}}^{\neg K} = \max \left\{ B_{\text{up}}^{{c_l}^{(i)}} \mid \left({c_l}^{(i)}, {c_r}^{(i)} \right) \in PQ_{\text{up}}^{\neg K} \right\}.$$

Formally, the stopping condition becomes

$$B_{\text{lo}}^{K} \geq B_{\text{up}}^{\neg K}. \tag{15}$$

The whole procedure, RVAQ, is presented in Algorithm 4.

Skipped Clips. During the execution of algorithm RVAQ, a lot of information is gained regarding which of the not processed clips are still relevant in estimating the upper/lower bounds and which should be *skipped* by TBClip. The set C_{skip} dynamically adjusts as the algorithm executes containing clips that the iterator TBClip can safely skip as they cannot contribute or alter the final result. The set contains two types of clips:

- The clips that are in the target video but not in P_q. At the beginning of Algorithm 1 (Line 2), C_{skip} will be initialized as $C(\mathcal{X}) \setminus C(P_q)$, where $C(\mathcal{X})$ and $C(P_q)$ represent the clips in the target video and P_q respectively.
- As algorithm RVAQ progresses, some sequences are placed in the top-K result conclusively and some are conclusively excluded from the top-K result. More specifically, for each sequence $({c_l}^{(i)}, {c_r}^{(i)}) \in P_q$, if its upper bound is smaller than B_{lo}^{K}, it will never be a top-K sequence; all the clips in this sequence are added into C_{skip} (Lines 13-14 Algorithm 1):

$$B_{\text{up}}^{{c_l}^{(i)}} < B_{\text{lo}}^{K} \Rightarrow C_{\text{skip}} = C_{\text{skip}} \cup \text{range}\left({c_l}^{(i)}, {c_r}^{(i)} \right),$$

where $\text{range}({c_l}^{(i)}, {c_r}^{(i)})$ is the set of all clips in this sequence. Similarly, if the lower bound of a sequence is larger than $B_{\text{up}}^{\neg K}$, it is one of the top-K sequences; if the final exact scores of top-K sequences are not required, all the clips in this sequence will be added into C_{skip} (Lines 19-20),

$$B_{\text{lo}}^{{c_l}^{(i)}} > B_{\text{up}}^{\neg K} \Rightarrow C_{\text{skip}} = C_{\text{skip}} \cup \text{range}\left({c_l}^{(i)}, {c_r}^{(i)} \right).$$

Algorithm 4: RVAQ

Input: query q: $\{o_1; ...; o_I; a\}$; $\forall i \in [1, I]$: P_{o_i}; P_a; K;
Output: top-K sequences: $P_{\text{fix}}^{\text{top}}$;

1 $P_q \leftarrow P_a \otimes P_{o_1} \otimes P_{o_2} \otimes ... \otimes P_{o_I}$.
2 $C_{\text{skip}} \leftarrow C(\mathcal{X}) \setminus C(P_q)$.
3 $B_{\text{lo}}^{K}, B_{\text{up}}^{\neg K} \leftarrow -1, \infty$.
4 **for** *each* $({c_l}^{(i)}, {c_r}^{(i)})$ *in* P_q **do**
5 $\quad B_{\text{up}}^{{c_l}^{(i)}}, B_{\text{lo}}^{{c_l}^{(i)}}, S_{\text{up}}^{{c_l}^{(i)}}, S_{\text{lo}}^{{c_l}^{(i)}} \leftarrow \infty, -1, 0, 0$.
6 $\quad L_{\text{up}}^{{c_l}^{(i)}}, L_{\text{lo}}^{{c_l}^{(i)}} \leftarrow {c_r}^{(i)} - {c_l}^{(i)}, {c_r}^{(i)} - {c_l}^{(i)}$.
7 $PQ_{\text{lo}}^{K} \leftarrow \{({c_l}^{(i)}, {c_r}^{(i)})$ with K highest $B_{\text{lo}}^{{c_l}^{(i)}} \mid ({c_l}^{(i)}, {c_r}^{(i)}) \in P_q\}$.
8 $PQ_{\text{up}}^{\neg K} \leftarrow \{(c_l^{(i)}, c_r^{(i)})$ ranked by $B_{\text{up}}^{c_l^{(i)}} \mid (c_l^{(i)}, c_r^{(i)}) \in (P_q \setminus PQ_{\text{lo}}^{K})\}$.
9 **repeat**
10 $\quad c_{\text{top}}, S_q{}^{(c_{\text{top}})}, c_{\text{btm}}, S_q{}^{(c_{\text{btm}})} \leftarrow$ get the next top/bottom clip through iterator TBClip (Algorithm 5) w.r.t. C_{skip}.
11 $\quad$ **for** *each* $({c_l}^{(i)}, {c_r}^{(i)})$ *in* P_q **do**
12 $\quad\quad B_{\text{up}}^{{c_l}^{(i)}} \leftarrow f(S_q{}^{(c_{\text{top}})}, ...) \odot S_{\text{up}}^{{c_l}^{(i)}}$, update PQ_{lo}^{K} and $PQ_{\text{up}}^{\neg K}$.
13 $\quad\quad$ **if** $B_{up}^{{c_l}^{(i)}} < B_{lo}^{K}$ **then**
14 $\quad\quad\quad C_{\text{skip}} \leftarrow C_{\text{skip}} \cup \text{range}({c_l}^{(i)}, {c_r}^{(i)})$.
15 $\quad {c_l}^{(j)} \leftarrow {c_l}^{(i)}|_{({c_l}^{(i)}, {c_r}^{(i)}) \in P_q:\ {c_l}^{(i)} \leq c_{\text{top}} \leq {c_r}^{(i)}}$.
16 $\quad S_{\text{up}}^{{c_l}^{(j)}}, L_{\text{up}}^{{c_l}^{(j)}} \leftarrow S_{\text{up}}^{{c_l}^{(j)}} \odot f(S_q{}^{(c_{\text{top}})}), L_{\text{up}}^{{c_l}^{(j)}} - 1$.
17 $\quad$ **for** *each* $({c_l}^{(i)}, {c_r}^{(i)})$ *in* P_q **do**
18 $\quad\quad B_{\text{lo}}^{{c_l}^{(i)}} \leftarrow f(S_q{}^{(c_{\text{btm}})}, ...) \odot S_{\text{lo}}^{{c_l}^{(i)}}$, update PQ_{lo}^{K} and $PQ_{\text{up}}^{\neg K}$.
19 $\quad\quad$ **if** $B_{lo}^{{c_l}^{(i)}} > B_{up}^{\neg K}$ **then**
20 $\quad\quad\quad C_{\text{skip}} \leftarrow C_{\text{skip}} \cup \text{range}({c_l}^{(i)}, {c_r}^{(i)})$.
21 $\quad {c_l}^{(j)} \leftarrow {c_l}^{(i)}|_{({c_l}^{(i)}, {c_r}^{(i)}) \in P_q:\ {c_l}^{(i)} \leq c_{\text{btm}} \leq {c_r}^{(i)}}$.
22 $\quad S_{\text{lo}}^{{c_l}^{(j)}}, L_{\text{lo}}^{{c_l}^{(j)}} \leftarrow S_{\text{lo}}^{{c_l}^{(j)}} \odot f(S_q{}^{(c_{\text{btm}})}), L_{\text{lo}}^{{c_l}^{(j)}} - 1$.
23 $\quad B_{\text{lo}}^{K} \leftarrow \min\{B_{\text{lo}}^{{c_l}^{(i)}} \mid ({c_l}^{(i)}, {c_r}^{(i)}) \in PQ_{\text{lo}}^{K}\}$.
24 $\quad B_{\text{up}}^{\neg K} \leftarrow \max\{B_{\text{up}}^{{c_l}^{(i)}} \mid ({c_l}^{(i)}, {c_r}^{(i)}) \in PQ_{\text{up}}^{\neg K}\}$.
25 **until** $B_{lo}^{K} \geq B_{up}^{\neg K}$;
26 **Return** PQ_{lo}^{K}.

C_{skip} is used by the iterator TBClip (detailed in §4.4) to return only clips relevant for further processing.

4.4 TBClip

We now discuss TBClip, an iterator presented by Algorithm 5 to return incrementally the highest and lowest ranking clips that satisfy query q. The iterator deploys a variant of the popular top-k query algorithm processing algorithm (e.g., [16]) with some important differences as detailed below.

After some initialization the first time it is invoked (Lines 1-5), the iterator proceeds in two steps. First, for each $table_{o_i}$, $table_{a_j}$ involved in q, conducts sorted access in parallel starting from the row in each table, accessed last in the previous invocation, until at least one *new* clip is identified in all the tables (Lines 8-13). Second, performs random accesses to obtain the scores of all retrieved clips (Lines 16-22) to fully compute their scores. At the same time, the iterator computes the clip with the current lowest score (Lines 23-24). The clip with the highest score along with the clip with the lowest score identified are returned by the iterator in this invocation.

Algorithm 5: iterator: TBClip

Input: query q: $\{o_1, ..., o_I, a\}$; $\forall i \in [1, I]$: $table_{o_i}$; $table_a$; C_{skip};
Output: next top and bottom clip identifiers and their scores: $c_{\text{top}}, S_{\text{top}}, c_{\text{btm}}, S_{\text{btm}}$

1 **if** *this iterator is initialized* **then**
2 $C_{\text{top}}, C_{\text{btm}} \leftarrow \emptyset, \emptyset$. # processed top/bottom clips.
3 ${C_{o_1}}^{\text{top}}, ..., {C_{o_I}}^{\text{top}}, {C_a}^{\text{top}} \leftarrow \emptyset, ..., \emptyset, \emptyset$.
4 ${C_{o_1}}^{\text{btm}}, ..., {C_{o_I}}^{\text{btm}}, {C_a}^{\text{btm}} \leftarrow \emptyset, ..., \emptyset, \emptyset$.
5 $stamp^{\text{top}}, stamp^{\text{btm}} \leftarrow 1, 1$.
6 ▷ **Step 1** (8-13) for each table, do sorted access in parallel from the $(stamp^{\text{top}})_{\text{th}}$ row until a *new* clip is found in all the tables.
7 **repeat**
8 ${c_{o_1}}^*, ..., {c_{o_I}}^*, {c_a}^* \leftarrow$ retrieve all the *CID*s that appear on the $stamp^{\text{top}}{}_{\text{th}}$ row of $table_{o_1}, ..., table_{o_I}, table_a$ respectively.
9 ${C_{o_1}}^{\text{top}}, ..., {C_{o_I}}^{\text{top}} \leftarrow {C_{o_1}}^{\text{top}} \cup \{{c_{o_1}}^*\}, ..., {C_{o_I}}^{\text{top}} \cup \{{c_{o_I}}^*\}$.
10 ${C_a}^{\text{top}} \leftarrow {C_a}^{\text{top}} \cup \{{c_a}^*\}$.
11 $C_{\cap}{}^{\text{top}} \leftarrow {C_a}^{\text{top}} \cap \bigcap_{i=1}^{I} {C_{o_i}}^{\text{top}} - C_{\text{top}} - C_{\text{skip}}$.
12 $stamp^{\text{top}} \leftarrow stamp^{\text{top}} + 1$.
13 **until** $|C_{\cap}{}^{top}| \geq 1$. ;
14 ▷ **Step 2** (16-22) perform random accesses to obtain the scores of all the seen clips and return the clip with the highest score.
15 $S_{\text{top}} \leftarrow -1$.
16 $C_{\cup}{}^{\text{top}} \leftarrow {C_a}^{\text{top}} \cup \bigcup_{i=1}^{I} {C_{o_i}}^{\text{top}} - C_{\text{top}} - C_{\text{skip}}$.
17 **for** *each clip c in* $C_{\cup}{}^{top}$ **do**
18 ${S_{o_1}}^{(c)}, ..., {S_{o_I}}^{(c)} \leftarrow$ retrieve *Score*s from $table_{o_1}, ..., table_{o_I}$ w.r.t. *CID c*.
19 ${S_a}^{(c)} \leftarrow$ retrieve the *Score* from $table_a$ w.r.t. *CID c*.
20 **if** ${S_a}^{(c)}(\sum_{i=1}^{I} {S_{o_i}}^{(c)}) > S_{top}$ **then**
21 $S_{\text{top}} \leftarrow g\left({S_a}^{(c)}, {S_{o_1}}^{(c)}, {S_{o_2}}^{(c)}, ..., {S_{o_I}}^{(c)}\right)$.
22 $c_{\text{top}} \leftarrow c$.
23 ▷ **Step 3** for each table, do reverse access from the bottom in parallel from the $(stamp^{\text{btm}})_{\text{th}}$ row and update ${C_{o_1}}^{\text{btm}}, ..., {C_{o_I}}^{\text{btm}}, {C_a}^{\text{btm}}$, until a *new* clip is found in all the tables.
24 ▷ **Step 4** perform random accesses to obtain the scores of all the seen clips and return the clip with the lowest score.
25 $C_{\text{top}}, C_{\text{btm}} \leftarrow C_{\text{top}} \cup c_{\text{top}}, C_{\text{btm}} \cup c_{\text{btm}}$.
26 **Return** $c_{\text{top}}, S_{\text{top}}, c_{\text{btm}}, S_{\text{btm}}$.

Thus, TBClip obtains both the highest ranking and the lowest ranking clips every time it is called. Moreover, while accessing the tables sequentially in parallel, we can *skip* (Lines 11 and 16) clips from further processing, if we are confident that these clips cannot contribute to sequences in the final top-K result. Such skipped clips belong to C_{skip} detailed in §4.3. Notice that set C_{skip} expands dynamically as the algorithm progresses. In TBClip, clips belonging to C_{skip} will only be accessed once during parallel sorted access, and be excluded from further processing (thus imposing no random access overhead in the ensuing execution).

5 EXPERIMENTAL EVALUATION

In this section, we present the results of our experimental evaluation of our proposals on a variety of datasets. For purposes of exposition, we utilize the following scoring functions in our offline case experiments (utilizing the notation of §4.1),

$$h : {S_{a_j}}^{(c)} = \sum_{s \in S(c)} {S_{a_j}}^{(s)}, \quad {S_{o_i}}^{(c)} = \sum_{v \in V(c)} \sum_{t \in T_{o_i}(c)} S_{o_i}^{t\ (v)};$$

$$g : {S_q}^{(c)} = {S_a}^{(c)} \left(\sum_{i=1}^{I} {S_{o_i}}^{(c)}\right); \qquad f : {S_q}^{(z)} = \sum_{c \in C(z)} {S_q}^{(c)}.$$

Functions f and h are additive on their operands, and function g is monotonic. Any function that adheres to the properties of §4.1 can be easily adopted.

5.1 Experimental Setup

Models. Our proposals are orthogonal to the underlying object/action detection and tracking models utilized. Our approach can work with any state-of-the-art model utilizing the best-in-kind models. In our experiments, for purposes of exposition, we select the following models as they have demonstrated solid performance. We stress however that our proposal can work with any model desired.

- MaskRCNN and YOLOv3. Mask R-CNN [24] stands as a two-stage object detector trained on dataset COCO; YOLOv3 [36, 37] is an object detector with capability extending to 9000 distinct types of objects.
- I3D. I3D [8] is an action recognizer trained on dataset Kinetics (encompassing 600 action types) [7].
- CenterTrack. CenterTrack [55] is a real-time object tracker that localizes objects and predicts their associations with the previous frames.
- Ideal Model. To evaluate our algorithms' accuracy without introducing errors in the detection models utilized, we also include experiments utilizing the *ideal models* that essentially generate detections matching the ground-truth labels exactly.

All models are set up as recommended in their papers.

Datasets. We use two video datasets in the experimental study.

- YouTube. ActivityNet [5] provides a large-scale YouTube video benchmark for human activity understanding. We select more than 600 videos in ActivityNet and authors manually label the range of different objects appearing in each activity for evaluating our proposal. We evaluate the algorithms on a fixed set of queries depicted in Table 1.
- Movies. We select well-known movies and form queries involving actions and objects appearing in them (Table 2) for the evaluation of RVAQ.

For annotation, we first divide all the YouTube videos into 12 sets according to their corresponding action types. For each set of videos, we only focus on the types of objects that will be queried, such as *faucet* and *Oven* for the videos about *washing dishes*. Then, for each queried object type, we label the temporal boundaries of the appearances of this object in each video. An object may appear multiple times in a video, and the temporal boundaries of each appearance will be annotated. The intersection of the temporal intervals of all the query-specified objects and the action will be considered as the result sequence that satisfies this query.

Metrics. For the evaluation of SVAQ and SVAQD, we first present the *F1 Score*, evaluating the effectiveness of the algorithms to identify the sequences compared to the ground truth, whose computation follows the subsequent procedure:

> We say that a sequence $\tilde{z}$ identified by our proposal matches a ground truth sequence z, iff the intersection over the union (IOU) of the clips of the two sequences is above a threshold, η. We set the threshold η to 0.5 in our evaluation signifying substantial overlap between the sequences[9]. A result sequence, $\tilde{z}$, is considered as a true positive if the IOU between $\tilde{z}$ and any sequence in the ground truth sequences is more than the threshold η; otherwise, it is a false positive. A ground truth sequence $\tilde{z}$, whose IOU with any result sequence is less than η, is considered a false negative.

In addition, detection models are prone to noise, leading to inaccurate query processing outputs (as indicated in §1), where we evaluate the effectiveness of SVAQD by quantifying the *proportion*

[9]The same IOU threshold (η=0.5) is widely utilized in numerous other research works, especially in object/action detection tasks [6, 24, 42, 52].

Table 1: Action and object types queried in evaluation on YouTube, alongside the total length (in min) of videos containing each specific action.

	Action	Object	Len		Action	Object	Len		Action	Object	Len
q_1	Washing Dishes	Faucet; Oven	57	q_5	Volleyball	Tree	110	q_9	Doing Crunches	Chair	85
q_2	Blowing Leaves	Car; Plant	52	q_6	Playing Rubik Cube	Clock	89	q_{10}	Blow-drying Hair	Kid	138
q_3	Walking The Dog	Tree; Chair	127	q_7	Cleaning Sink	Faucet; Knife	84	q_{11}	Washing Hands	Faucet; Dish	113
q_4	Drinking Beer	Bottle; Chair	63	q_8	Kneeling	Tree	104	q_{12}	Archery	Sunglasses	156

Table 2: Action and object types queried on Dataset Movies.

Video	Action	Object	Length
Coffee and Cigarettes	Smoking	Wine Glass, Cup	1h 36min
Iron Man	Robot Dancing	Car, Airplane	2h 6min
Star Wars 3	Archery	Bird, Cat	2h 14min
Titanic	Kissing	Surfboard, Boat	3h 14min

of noise eliminated from the detection models by SVAQD.

For the evaluation of RVAQ, we measure the query *runtime* (or query latency), as well as the *number of random disk accesses* to the clip score tables required to answer the queries.

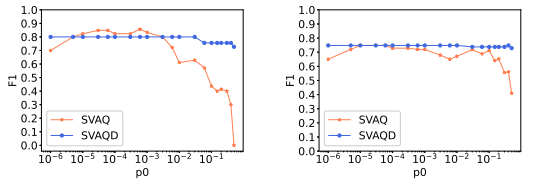
Algorithms Compared: Offline Case For the offline case algorithm RVAQ utilizes both a forward and a backward pass on the suitable tables to obtain top/bottom clips and utilizes a *skip* mechanism to avoid accessing clips unnecessarily. For purposes of exposition we also present a comparison with the following algorithms:

- We utilize Fagin's Algorithm [15] to produce top ranking clips according to equation 12. Each clip as produced is checked against the ranges of clips in P_q. If the clip is not in any of the clip ranges in P_q it is disregarded. The score of each sequence in P_q is computed as its clips are produced and the algorithm stops when the score of each sequence in P_q has been produced and the final K sequences are returned. We refer to this algorithm as *FA*.
- We also include a variant of algorithm RVAQ without activating the *skip* mechanism in order to quantify the effectiveness of our proposal. We refer to this algorithm as RVAQ-noSkip.
- Finally we consider the algorithm that accesses all clips in the sequences of P_q (Equation 12), calculates the scores of each sequence and returns the K sequences with the highest scores. This algorithm accesses only the clips in the result sequences. We refer to this algorithm as the P_q-Traverse.

Server. All algorithms were implemented in Python and run on a Linux server with Intel Xeon Gold 6244 3.60GHz CPU and 64GB memory and an NVIDIA RTX TITAN Xp GPU.

5.2 The Online Case

Impact of Background Probability. In our description of algorithms SVAQ and SVAQD, there is an initialization of the background probability p_0. We first explore the sensitivity of the algorithms to this initialization. In Figure 2, we illustrate the performance of the two algorithms for two queries on the YouTube dataset, varying the background probability. The depicted queries are (a): {a=blowing leaves; o_1=car} and (b): {a=washing dishes; o_1=faucet}. Similar results hold for the queries on other videos in Dataset YouTube and are omitted for brevity. For each experiment, we initialize both algorithms with the same value of p_0 and report the associated F1 score for each query. As is evident in the Figure, SVAQD has a very low dependency on the initial value of p_0 due to its adaptive design. In contrast, algorithm SVAQ has a high dependency on the background probability value and some initial values affect its accuracy substantially. Thus, the benefits of the adaptive design of SVAQD are clear as the dependency



(a) q:{a=blowing leaves;o_1=car} (b) q:{a=washing dishes;o_1=faucet}

Figure 2: F1 Scores of SVAQ/SVAQD on diff initial background prob.

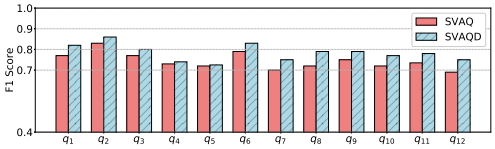


Figure 3: F1 scores of SVAQ and SVAQD for all the queries on YouTube.

on the initial value of the background probability is immaterial, making it of choice.

F1 Score. We present the F1 scores for the two algorithms across several diverse queries. As a basis for comparison, we observe that the F1 scores of SVAQ as p_0 varies in Figure 2 peaks in the range $[10^{-5}, 10^{-4}]$. So we fix p_0 for SVAQ to $p_0 = 10^{-4}$ in this and subsequent experiments. Figure 3 presents the F1 scores for all the queries on dataset YouTube, specified in Table 1, for SVAQ and SVAQD. It is evident that SVAQD, which dynamically updates p_0, provides superior performance on F1 Scores.

Impact of Predicate Correlation and Predicate Quantity. There are two important factors that affect the overall F1 score, namely the accuracy in the detection of the underlying object and action models deployed as well as how correlated the object and action predicates are. Table 3 presents the F1 scores for twelve different queries on the YouTube dataset with varying object predicates for the SVAQ and SVAQD algorithms. We observe that for queries containing multiple correlated predicates (i.e., the range of frames on which both predicates are true exhibit high overlap, such as the query about *blowing leaves* and *person*), the predicates with higher detection accuracy (such as *person*) may improve the precision of our model for the composite query, compared to queries without the predicate (i.e., only *blowing leaves*). This happens because predicates with higher accuracy of detection (due to the object/action model deployed) may mitigate the errors caused by predicates with lower detection accuracy. In general, however, queries involving multiple predicates result in slightly lower F1 scores since the likelihood of detection errors increases with the number of query predicates. In practice, utilizing the most accurate detectors for objects and actions is preferred; SVAQD is able to attain superior accuracy for queries of varying complexity.

Impact of Detection Noise. In Table 4, we present the F1 scores for the two algorithms adopting different detection models. As expected, lower accuracy models (e.g., YOLOv3 [37]) yield lower

Table 3: F1 scores of queries with varying object predicates.

Queries	SVAQ	SVAQD
a=blowing leaves	0.82	0.85
a=blowing leaves, o_1=person	0.90	0.93
a=blowing leaves, o_1=plant	0.84	0.87
a=blowing leaves; o_1=car	0.83	0.85
a=blowing leaves, o_1=person, o_2=car	0.83	0.88
a=blowing leaves, o_1=person, o_2=plant, o_3=car	0.81	0.85
a=washing dishes	0.86	0.88
a=washing dishes, o_1=person	0.89	0.93
a=washing dishes, o_1=oven	0.83	0.86
a=washing dishes, o_1=faucet	0.77	0.79
a=washing dishes, o_1=faucet, o_2=oven	0.77	0.80
a=washing dishes, o_1=person, o_2=faucet, o_3=oven	0.78	0.82

Table 4: F1 scores of SVAQ and SVAQD with different detection models for the query q: {a=blowing leaves; o_1=car}.

Methods in SVAQ	F1	Methods SVAQD	F1
SVAQ (MaskRCNN+I3D)	0.83	SVAQD (MaskRCNN+I3D)	0.85
SVAQ (YOLOv3+I3D)	0.80	SVAQD (YOLOv3+I3D)	0.82
SVAQ (Ideal Models)	1.00	SVAQD (Ideal Models)	1.00

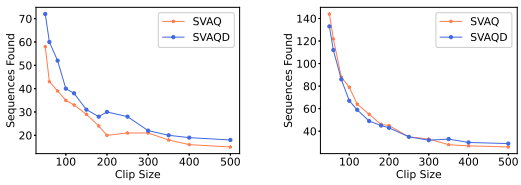
Table 5: False positive rate of action/objection detection without or with SVAQD.

Queries	FPR of Action		FPR of Object	
	w/o SVAQD	w/ SVAQD	w/o	w/
a=blowing leaves; o_1=car	0.10	0.05	0.31	0.18
a=washing dishes; o_1=faucet	0.16	0.01	0.18	0.04

overall F1 scores when adopted in our algorithms, versus more accurate models (e.g., Mask R-CNN [24]). We also present the accuracy of our algorithms assuming the *ideal models* that have a perfect detection accuracy (i.e., match the ground truth in all of their inferences). It is evident that the major source of inaccuracy for our algorithms is the errors introduced by the underlying detection models. When that source of inaccuracy is removed (i.e., utilizing better models) our algorithms improve their accuracy in terms of identifying all results of interest compared to ground truth.

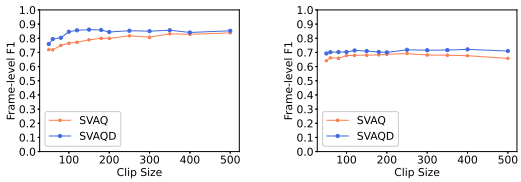
Effectiveness of Eliminating Detection Noise. Table 5 presents the false positive rates (FPR) of the action recognizer and the object detector over two queries on dataset YouTube, conducted without (w/o) or with (w/) SVAQD. It is evident that, by employing our proposed query identification and merging techniques, SVAQD can substantially reduce false positives produced by the detection models by 50-80%, eliminating their errors effectively.

Impact of Clip Size. We conduct experiments varying the clip size. The size of clips is a parameter in our setting and we wish to explore how its choice affects the results reported. In Figure 4, we report the number of result sequences identified by our algorithms with varying clip sizes for two different queries. As is depicted in Figure 4, it is evident that the prevailing trend is that with a smaller clip size, we generate more result sequences (of smaller length), whereas, with a larger clip size, we will get fewer result sequences (of larger length). However upon close examination of the actual number of frames reported in each case, irrespective of the clip size, the total number of frames reported for each clip size remains stable. Thus the choice of the size of the clip primarily affects the number of result sequences reported, not the actual content of the sequences compared to ground truth. In particular, if we evaluate the algorithms assessing the frame-level F1 score comparing the frames in the sequences retrieved to the ground truth, their accuracy exhibits low dependency on



(a) q:{a=blowing leaves;o_1=car} (b) q:{a=washing dishes;o_1=faucet}

Figure 4: # of Sequences found with diff clip sizes.



(a) q:{a=blowing leaves;o_1=car} (b) q:{a=washing dishes;o_1=faucet}

Figure 5: Frame-level F1 scores with diff clip sizes.

the clip size (as shown in Figure 5), which corroborates the fact that frames in the result sequences obtained by our algorithms are the same even if the clip size varies. Similar results hold for the queries on other videos in Dataset YouTube and are omitted for brevity.

Runtime Superiority. The query latency (or query runtime) for both SVAQ and SVAQD is dominated (>98%) by the inference time of the object detection and action recognition models. Taking query q_1 as an example, its overall query processing time is 171.8 minutes, with 168.7 minutes dedicated to model inference. Making a comparison between our proposal and the solution based on the end-to-end idea, we undertake the training of an end-to-end model (using the network architecture in [8]) for query q_1. Despite yielding little improvements (<0.05) in F1 Score, the end-to-end detection model has a training (fine-tuning) and query processing time of >60 hours, which is significantly worse than the runtime of SVAQD. This proves the statements made in §1 that attempting to train an end-to-end model for recognizing actions involving objects concurrently is not scalable nor applicable.

5.3 Performance of RVAQ

For the offline case, once the query is provided along with the number of results required, the main objective is to produce the results sequences with the highest score fast. Since the video sequences have been pre-processed, the execution of the algorithm involves accessing information in secondary storage to produce the results. As such we report on the *runtime* required to produce the final answer, as well as the *total number of random accesses* to secondary storage.

Table 6 presents the runtime and the total number of random disk accesses conducted by all approaches we consider (§5.1) for the movie, Coffee and Cigarettes, as per Table 2, as the number K of highest ranking results varies. Table 7 presents the performance of all approaches for the YouTube dataset with respect to K=5. Some observations are immediate; for algorithm *FA* as no lower bounds can be obtained as well as there is no way to skip unnecessary clips, the overall performance is substantially worse than all other approaches. In Algorithm RVAQ-noSkip, we observe that, although upper and lower bounds are obtained for the clips accessed, the inability to skip irrelevant clips is detrimental to the performance of the algorithm. For algorithm P_q-Traverse, the

Table 6: Performance on movie Coffee and Cigarettes.

Methods	Runtime (sec); Number of Random Accesses (×1000)					
	K=1	K=5	K=9	K=11	K=13	K=15
FA	246; 39	282; 50	293; 51	293; 51	349; 54	349; 54
RVAQ-noSkip	130; 24	180; 27	184; 27	185; 27	192; 27	195; 27
P_q-Traverse	56; -	56; -	56; -	56; -	56; -	56; -
RVAQ	14; 3.6	24; 3.7	34; 4.0	38; 5.2	42; 5.6	48; 6.0

Table 7: Performance on YouTube dataset (K=5).

Queries	Runtime (sec); # of Random Accesses (×1000)			
	FA	RVAQ-noSkip	P_q-Traverse	RVAQ
q_1	218; 33.2	63; 7.2	9.8; -	3.2; 0.62
q_2	96; 15.1	65; 7.3	32; -	2.6; 0.55

Table 8: Speedup of RVAQ against P_q-Traverse on 3 movies.

Datasets	K=1	K=3	K=5	K=7	K=9	K=11	max K
Iron Man	3.70x	3.03x	2.86x	2.78x	2.70x	2.33x	1.05x
Star Wars 3	3.23x	2.94x	2.78x	2.63x	2.63x	2.63x	1.03x
Titanic	2.78x	2.78x	2.78x	2.78x	2.78x	2.38x	1.11x

runtime and the number of random disk accesses is a constant irrespective of the value of K and proportional to the total number of clips contained in the result sequences. Overall, our proposed RVAQ is the most efficient and progressively its performance approaches closely that of P_q-Traverse when all result sequences are requested, as expected (this video has 21 ground truth result sequences). Since the performance of *FA* and RVAQ-noSkip is almost an order of magnitude worse than that of RVAQ for small values of K we do not consider these algorithms further.

Table 8 presents the speedup of RVAQ against P_q-Traverse for three popular movies as the number K of highest ranking results varies. The associated queries we issue are depicted in Table 2. We observe that RVAQ is 2.7 to 3.7 times faster compared to p_q-Traverse for a small number of results K. As K increases gradually, since the query requires accessing all the clips of top-K sequences to obtain their exact scores, the number of accesses required by RVAQ increases to reach progressively that of P_q-Traverse, when all the results sequences in the video are required in the final answer. Note that the last column depicts the largest value of K which contains all the results sequences for the query in each video.

In terms of accuracy for all these results, since the movies in our evaluation were not annotated by a third party, we manually annotated the results for the queries considered and inspected the ranked result sequences of RVAQ. In all cases for all sequences returned by RVAQ, the precision is above 81.0% and the associated F1 score is above 82.9%. These results are in line with the accuracy of SVAQD presented in the previous section (as compared to annotated ground truth in benchmark datasets). Moreover, in all cases, when K=10, that is for the top-10 sequences ranked by Equation 10, the precision and F1 score are both 1, attesting to the superior accuracy (and associated performance) attainable by our proposals.

6 RELATED WORK

Action Recognition and Object Detection. Emerging solutions for object detection can be roughly divided into two main categories: one-stage methods (such as YOLO [37] and SSD [32]) and two stage-methods (such as Faster R-CNN [39] and Mask R-CNN [24]). Mask R-CNN [24] is proposed to tackle pixel-wise object instance segmentation by extending Faster R-CNN. Several recent works address the problem of recognizing actions in videos [8, 17, 20, 53]. I3D [8] extends the network structure from two to three dimensions and proposes a two-stream inflated 3D ConvNet for action recognition. These works primarily train deep architectures of varying complexities end to end yielding models to classify actions in video frame sequences. Our work readily utilizes action recognition, object detection and tracking models [48, 55] in an online or offline manner to enable declarative query processing scenarios.

Automated Video Analytics in Data Management Community. In the data management community, several recent works [9, 10, 12, 13, 30, 51] have introduced declarative query interfaces that utilize frame content (objects, spatial locations in the frame, etc) as first class citizens, aiming to enhance the execution accuracy and speed, while enabling users to query without concerning the exact execution process. NoScope and BlazeIt [28, 29] utilize special-purpose-build neural networks (NNs) to detect objects accelerating queries via inference-optimized model search. Focus [26] implements low-latency search over large video datasets aiming to balance precision and query speed. SVQ [31, 49] provides a series of filters to accelerate video monitoring queries involving count and spatial constraints on objects present in the frames. [11, 50] present declarative query processing on video streams involving objects and their interactions. Our work follows this research thread proposing a framework to incorporate object detection and action recognition in a declarative framework for both streaming and offline videos.

Content-based Image Retrieval. The history of content-based image retrieval (CBIR) research in the multimedia context can be dated back to more than 20 years [33, 41, 43]. CBIR aims at searching for visual (or semantic) similar images given a query image according to their visual contents. Bartolini et al. [1–3] develop retrieval systems compositing multimedia retrieval models with multimedia databases. Recently, image representations based on CNN have attracted increasing interest in the community and demonstrated impressive performance [34, 54]. The CNNs with hashing approaches have also been adopted in the CBIR task for obtaining more compact image features to compute the similarity of pairwise images [34]. The scope of our research diverges from CBIR. In this paper, our approach centers on leveraging the outputs of pre-existing CV models to answer SQL-like queries involving actions and objects, where our system enhances query accuracy and reduces time expenditure by minimizing model invocations. Conversely, CBIR is dedicated to constructing representations of video content and addressing the searching through similarity computations [2].

7 CONCLUSIONS

We treated online and offline queries on videos incorporating objects and actions as first-class citizens for query processing. As off-the-shelf deep learning models for action recognition focus on the action itself and do not incorporate object constraints, we proposed a framework based on scan statistics to identify sequences of frames that contain all query-specified objects and actions in a statistically sound manner. We demonstrated our approaches' accuracy and performance benefits using real-world videos. This work raises a few avenues for follow-up work. First, adding other action detection models (like group action detection) to our framework should be investigated. Also, queries involving interactions between objects and actions in the video feed are worth investigating.

REFERENCES

[1] Ilaria Bartolini, Paolo Ciaccia, Vincent Oria, and M Tamer Özsu. 2007. Flexible integration of multimedia sub-queries with qualitative preferences. *Multimedia Tools and Applications* 33 (2007), 275–300.

[2] Ilaria Bartolini, Marco Patella, and Corrado Romani. 2013. SHIATSU: tagging and retrieving videos without worries. *Multimedia tools and applications* 63 (2013), 357–385.

[3] Ilaria Bartolini, Marco Patella, and Guido Stromei. 2011. The windsurf library for the efficient retrieval of multimedia hierarchical data. In *Proceedings of the International Conference on Signal Processing and Multimedia Applications*. IEEE, 1–10.

[4] Christopher M Bishop et al. 1995. *Neural networks for pattern recognition*. Oxford university press.

[5] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 961–970.

[6] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. 2018. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340* (2018).

[7] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. 2019. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987* (2019).

[8] Joao Carreira and Andrew Zisserman. 2017. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[9] Daren Chao, Kaiwen Chen, and Nick Koudas. 2023. SVQ-ACT: Querying for Actions over Videos. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 3599–3602.

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

[11] 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*. 2769–2772.

[12] Daren Chao, Nick Koudas, and Xiaohui Yu. 2023. Marshalling model inference in video streams. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1379–1392.

[13] Yueting Chen, Xiaohui Yu, and Nick Koudas. 2021. Evaluating Temporal Queries Over Video Feeds. *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data* (2021).

[14] Peter Diggle. 1985. A kernel method for smoothing point process data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 34, 2 (1985), 138–147.

[15] Ronald Fagin. 1999. Combining fuzzy information from multiple systems. *Journal of computer and system sciences* 58, 1 (1999), 83–99.

[16] Ronald Fagin. 2002. Combining fuzzy information: an overview. *ACM SIGMOD Record* 31, 2 (2002), 109–118.

[17] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. 2019. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision*. 6202–6211.

[18] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. 2016. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[19] James C Fu and Brad C Johnson. 2009. Approximate probabilities for runs and patterns in iid and Markov-dependent multistate trials. *Advances in Applied Probability* 41, 1 (2009), 292–308.

[20] Jiyang Gao, Zhenheng Yang, Kan Chen, Chen Sun, and Ram Nevatia. 2017. Turn tap: Temporal unit regression network for temporal action proposals. In *Proceedings of the IEEE International Conference on Computer Vision*. 3628–3636.

[21] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.

[22] 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.

[23] Joseph Glaz, Joseph Naus, and Sylvan Wallenstein. 2001. *Scan Statistics* (1st ed.). Springer.

[24] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[26] 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)*. 269–286.

[27] Brad Johnson. 2017. Approximations for Discrete Scan Statistics on i.i.d and Markov Dependent Bernoulli Trials. *Handbook of Scan Statistics* (2017).

[28] Daniel Kang, Peter Bailis, and Matei Zaharia. 2018. Blazeit: Fast exploratory video queries using neural networks. *arXiv preprint arXiv:1805.01046* (2018).

[29] 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.

[30] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2022. TASTI: Semantic Indexes for Machine Learning-based Queries over Unstructured Data. (2022).

[31] Nick Koudas, Raymond Li, and Ioannis Xarchakos. 2020. Video Monitoring Queries. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1285–1296.

[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.

[33] Yan Liu and Fei Li. 2002. Semantic extraction and semantics-based annotation and retrieval for video databases. *Multimedia Tools and Applications* 17 (2002), 5–20.

[34] Huimin Lu, Ming Zhang, Xing Xu, Yujie Li, and Heng Tao Shen. 2020. Deep fuzzy hashing network for efficient image retrieval. *IEEE transactions on fuzzy systems* 29, 1 (2020), 166–176.

[35] Joseph I Naus. 1982. Approximations for distributions of scan statistics. *J. Amer. Statist. Assoc.* 77, 377 (1982), 177–183.

[36] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[37] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[38] 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.

[39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence* 39, 6 (2016), 1137–1149.

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.

[41] Thomas Seidl and Hans-Peter Kriegel. 1997. Efficient user-adaptable similarity search in large multimedia databases. In *VLDB*, Vol. 97. 506–515.

[42] Karen Simonyan and Andrew Zisserman. 2014. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*. 568–576.

[43] Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. 2000. Content-based image retrieval at the end of the early years. *IEEE Transactions on pattern analysis and machine intelligence* 22, 12 (2000), 1349–1380.

[44] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 6450–6459.

[45] Ryan Turner, Zoubin Ghahramani, and Steven Bottone. 2010. Fast online anomaly detection using scan statistics. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*. IEEE, 385–390.

[46] MJ Van Kreveld. 1996. *Variations on sweep algorithms: efficient computation of extended viewsheds and class intervals*. Vol. 1996. Utrecht University: Information and Computing Sciences.

[47] VNI Cisco. 2018. Cisco visual networking index: Forecast and trends, 2017–2022. *White Paper* 1 (2018), 1.

[48] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. 2019. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1328–1338.

[49] Ioannis Xarchakos and Nick Koudas. 2019. Svq: Streaming video queries. In *Proceedings of the 2019 International Conference on Management of Data*. 2013–2016.

[50] Ioannis Xarchakos and Nick Koudas. 2021. Querying for Interactions. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*.

[51] Y. Chen, X. Yu, and N. Koudas. 2020. TQVS: Temporal Queries over Video Streams in Action. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2737–2740.

[52] Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen. 2019. A comprehensive survey of vision-based human action recognition methods. *Sensors* 19, 5 (2019), 1005.

[53] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. 2017. Temporal action detection with structured segment networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 2914–2923.

[54] Liang Zheng, Yi Yang, and Qi Tian. 2017. SIFT meets CNN: A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence* 40, 5 (2017), 1224–1244.

[55] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. 2020. Tracking objects as points. In *European Conference on Computer Vision*. Springer, 474–490.