Efficient Activity Retrieval through Semantic Graph Queries

Gregory Castañón
Boston University
gdc1@bu.edu

Yuting Chen
Boston University
yutingch@bu.edu

Ziming Zhang
Boston University
zzhang14@bu.edu

Venkatesh Saligrama
Boston University
srv@bu.edu

ABSTRACT

We present an efficient retrieval approach for activity detection in large surveillance video datasets based on semantic graph queries. Unlike conventional approaches, our zero-shot retrieval method does not require knowledge of the activity classes contained in the video. We propose a novel user-centric approach that models queries through the creation of sparse semantic graphs based on attributes and discriminative relationships. We then pose search as a ranked subgraph matching problem and leverage the fact that the attributes and relationships in the query have different levels of discriminability to filter out bad matches. Rather than solving the NP-hard exact subgraph matching problem, we develop a novel maximally discriminative spanning tree (MDST) as the relaxation of a given query graph, and then describe a matching algorithm that recovers matches to this tree in linear time using maximally discriminative subgraph matching (MDSM). We utilize the MDST to minimize the number of possible matches to the original query while guaranteeing that the best matches are within this set. We test this algorithm on two large video datasets: the 35-GB Virat Ground dataset and a 1-TB aerial data collection from Yuma. These datasets yield graphs with 200,000 nodes and 1 million nodes, respectively, with an average degree of 5. Our approach finds complex, large-scale queries in seconds while maintaining comparable precision and recall to slower current approaches.

1. INTRODUCTION

As cameras become cheaper and more ubiquitous, algorithms that can reason over large video corpora are becoming increasingly vital. An important specialization of large-scale data reasoning is video search, which aims to recover topics of interest from a large volume of video. Video search has applications in many areas such as online video navigation, retail store surveillance, and forensic search of aerial imagery.

The main challenge of surveillance video search is to navigate a large corpus to quickly find matches to an a priori unknown query. Unlike typical approaches to search [13, 21, 25], we do not utilize exemplar videos or attempt to learn activity categories a priori. In video search, the *complex vocabulary of activities* renders activity model training both computationally infeasible and practically impossible due to a lack of training examples.

Our approach can be interpreted as zero-shot activity retrieval, a cousin of zero-shot learning [27]. It does not rely on training data or exemplars, nor does it exploit multimedia contextual information such as text or audio since these are not frequently available for surveillance videos. We accomplish zero-shot retrieval by leveraging the fact that activities in certain surveillance videos can be characterized by a relatively small attribute vocabulary $\mathcal{A}$ and a limited relationship vocabulary $\mathcal{R}$ to organize them. Furthermore, this limited vocabulary of attributes and relationships can be easily understood by a user, autonomously extracted in real-time, and stored efficiently using several well-known hashing schemes. Rather than relying on complex attributes, we make use of the structural relationships of an activity to formulate a query Q as a semantic graph, associating attributes with vertices and relationships with edges. Our search algorithm then tries to match this graph to attributes and relationships in the video data using approximate subgraph matching.

Subgraph matching is an NP-hard [40] problem that is computationally infeasible on large datasets even when the subgraph is relatively small [7]. Our approach is based on data reduction and in this sense is complementary to any large scale video search algorithm. First, we design coarse detectors for attributes and relationships to create a coarse graph C from the archive data, which precludes attributes and relationships that individually could not match vertices and edges in the query graph. The run-time for these detectors scales with $|C|$, the size of the coarse graph, as opposed to the size of the archive data. Second, we leverage the fact that not all attributes and relationships are equally discriminative, and that their discriminability can be calculated in real time. We use this discriminability to calculate the *Maximally Discriminative Spanning Tree* (MDST) T^* for the query graph in $O(|E^q|\log(|V^q|))$ time, where V^q and E^q are the vertices and edges in the query graph, respectively. This is the spanning tree that is expected to yield the smallest possible coarse graph C. We maximize the discriminability of this tree by modeling the distributions of attributes within graphs and solving a spanning tree optimization. Third, we propose a new tree-matching algorithm that finds a set of solutions in C that match T^*. This process is shown in Figure 1, and is accomplished without the explicit construction of the coarse graph.

Intuitively, our approach is effective on large datasets with different fields of view and a priori unknown activities because we create simple models from semantic features instead of creating complex models based on high-dimensional features. Thus, while

MM'15, October 26–30, 2015, Brisbane, Australia.
© 2015 ACM. ISBN 978-1-4503-3459-4/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2733373.2806229.

our models are less tuned than more complex models, they are flexible, easy to generate and have a high probability of detection. In addition, our approach takes advantage of the sparsity of surveillance queries. In order to find an activity, we do not have to model each detail - just a series of high-level features which differentiate it from other activities in the dataset. While sliding-window methods [23] struggle to model large-scale complex activities because of the amount of unrelated activity in a broad spatio-temporal window, our subgraph matching approach remains unaffected.

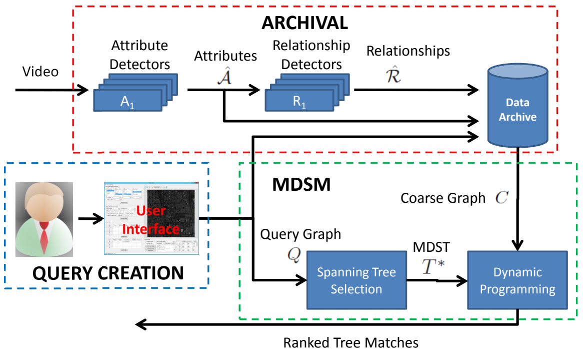

Figure 1: In the archival step, we take in incoming data, extract attributes and relationships and store them in hash tables. In the query creation step, a user utilizes our GUI to create a query graph that is used to extract the coarse graph C from archive data. In the Maximally Discriminative Subgraph Matching (MDSM) step, we calculate the maximally discriminative spanning tree (MDST) T^* from the query graph, retrieve matches to it, and assemble them into ranked search results for the user.

Our contributions are:

Zero-Shot Retrieval: We propose a zero-shot, user-centric approach to model acquistion through the creation of sparse semantic graphs based on attributes and discriminative relationships. This process requires no training data or training step, allowing for the description of novel classes.

Graph Representation: We introduce a graphical approach to query representation. While standard systems learn complete models for each activity, we provide a user with a series of simple semantic concepts and arrange them in a graph. In this graph, nodes represent attributes of an object or scene and edges represent relationships. We use this query in a sub-graph matching approach to identify activity, allowing our algorithm to effectively ignore confuser events and clutter that happen near the activity of interest. This graphical representation also makes the approach relatively agnostic to the duration of the event, allowing it to detect events that take place over several minutes.

MDST: We introduce the calculation of the Maximally Discriminative Spanning Tree, T^* based on the statistics of archive data stored in our indices. We not only exploit, as others do [23, 37], the sparsity of individual elements, but calculate an optimal combination of elements to maximally reduce the archive data using a novel Maximally Discriminative Subgraph Matching (MDSM) algorithm. We show that this approach results in a dramatically smaller corpus over which to search, and prove that the corpus must contain all high-ranking matches to the original graph.

We show that our approach, outlined in Figure 1, works in both airborne and building-mounted surveillance modalities, outperforms previous work and produces results comparable in quality to a brute-force search in a realistic time.

1.1 Related Work

There is extensive work on image and video classification [41, 45, 21, 42, 2] that follows a conventional pipeline approach by learning classifiers based on training data. These approaches extract, aggregate and code millions of low-level feature vectors for each image or video clip before learning classifiers for activity classes. Significant attention has also been specifically devoted to activity modeling in videos. Early datasets [35] created an emphasis on bags of local descriptors [13, 21, 25] to model activities. These datasets consist of single clips of isolated actions, with a single class of action appearing in every clip. Later work has moved away from bag of words models and focused on the spatio-temporal relationships of these words [31, 34, 4]. The most recent approaches have extended the focus on relationships to more complex datasets [15] with more activity classes and background noise.

In general it has been recognized that the number of training examples cannot match the increasing complexity of classifiers and activities [33, 20]. While approaches have been developed [32] to artificially generate training examples, these methods require a sliding window through time and space. The difficulty of model estimation makes activity search particularly challenging - some image search methods [28] rely first on classifying the archive data, then classifying a single exemplar provided by the user and returning archived objects from the same class as the exemplar. In summary many of these approaches do not generalize to unseen activities and furthermore their performance degrades in the limit of small training data.

A recent approach that seeks techniques for learning without training data is zero-shot learning [27, 29]. Many of the existing approaches rely on semantic concepts to help generalize to novel classes. These include ontology [30], knowledge transfer [29, 14] and max-margin based methods [44]. In general, these techniques rely on extrapolating activity models from an existing set of classes, while our approach has users summarize their knowledge about activities for search.

There are a number of recent works for video classification and retrieval [1, 19, 9, 46, 49] based on video semantic similarity with no prior knowledge or training data. Many of these works are based on exploiting and leveraging multiple modalities such as text, audio and OCR in combination with video. These works attempt to determine semantic similarity between a textual query and a video by employing a multimedia concept lexicon [9, 46] and combining this multimedia information without prior knowledge of the event classes. Unfortunately, this multimedia information is extremely scarce in surveillance video, so our current implementation does not take it into account. Our method, however, can be easily extended into multimedia information retrieval by adding attributes and relationships in those domains.

There is an extensive literature on video retrieval for surveillance applications (see [18, 48, 36, 38, 22, 24]). Much of this literature deals with semantic activity modeling and indexing, where the indexing for motion features is done using keywords and is typically manual. The goal is to perform keyword based retrieval for later use. In contrast, our method does not require human annotation of activities. Our work is closely related to [3] who also create an attribute vocabulary and use the temporal ordering of those attributes to determine likely locations for an activity in the video. However, their approach is extremely limited as they focus primarily on temporal order and lack the ability to accurately model many activities that rely on spatial relationships for discrimination. In contrast our approach allows for arbitrary relationships between objects or attributes and models the uncertainty in those relationships.

Sub-graph and graph matching are also commonly used in image search applications [6, 11] as well as image duplicate detection [50]. In image search and image duplicate detection, the goal is the same: to match the graph extracted from a query image to the graphs extracted from each image in a corpus. Subgraph matching is used to allow these algorithms to be robust to small amounts of clutter. Some papers [5] have adapted this technology to detecting videos with similar events by building a graphical model for the elements of each video. This problem differs significantly from ours because their corpus is divisible into a series of relatively small graphs, one for each image or video, it can be solved efficiently by a series of subgraph matching problems.

In certain video domains, like news broadcasts, the corpus is easily partitionable into segments that can be reasoned over in parallel. In surveillance video, this is rarely the case, as a camera continually produces information, rendering partitioning unrealistic. Current approaches to graph and subgraph matching include semidefinite programming [39], graduated assignment [16] and spectral matching [8]. While the approximations used to solve subgraph matching problems work well on small problems (<1000 vertices per graph), their NP-scaling becomes problematic on a graph with the millions of vertices found in a typical representation of a single large surveillance video. Recent works [37, 12] have explored the idea of using advances in cloud computing to perform massive parallelization. These approaches are promising but have not been extended to attributed relational graphs (ARGs). In contrast, our approach focuses on rapid reduction of the search space to render graph-matching approaches not only feasible, but practical while maintaining similar quality to exact subgraph matching solutions.

2. MODEL

Our model is composed of four parts:

Vocabulary: We define an attribute vocabulary $\mathcal{A}$ and a relationship vocabulary $\mathcal{R}$. Our attribute vocabulary consists of pyramidal-binned histograms of pixel-level features, object detectors and low-level motion features extracted from tracked data. Our relationship vocabulary consists of local, pairwise descriptors such as "near to" and "shortly after".

Query: Our query graph, $Q = G(V^q, E^q, A^q, R^q, L^q)$ is a function of five elements. For each vertex $v \in V^q$ in the graph, A^q_v is the set of attributes from $\mathcal{A}$ associated with v, and L^q_v is the set of tolerances for each of those attributes. For each edge $(v_1, v_2) \in E^q$, $R^q_{(v_1,v_2)}$ is the set of relationships associated with (v_1, v_2), and $L^q_{(v_1,v_2)}$ is the set of tolerances for each of those relationships. Value and tolerance pairs specify an expected value and the variance in that value that is acceptable to the user.

Scoring: We define similarity functions $K_{a,\ell}(a')$ and $K_{r,\ell}(r')$ for the purposes of comparing attributes a and a' with tolerance ℓ, and relationships r and r' with tolerance ℓ.

Storage: We construct an archive representation which stores $\mathcal{A}$ and $\mathcal{R}$ given a maximal tolerance l_{max} for each attribute or relationship. This archive allows us to quickly recover the set of attributes or relationships which could match a query, independent of the specified tolerance.

2.1 Vocabulary

The video modality for our datasets is street surveillance, which dictates a specific set of components that a user might use to construct queries (see Sec. 2.2). We employ four vocabulary elements: three types of attributes in addition to replationships.

1. **Local Descriptors**: These are local descriptors of pixel-level attributes, like those used in [3]. These attributes, like color, size, coarse motion direction, persistence and activity can be computed over tiles made of B by B pixel squares and histogrammed to provide rough descriptors of foreground image properties. This approach is doubly beneficial; histogramming and quantization naturally mitigate low-level measurement noise while simultaneously producing low-dimensional features for efficient storage.

2. **Object Detections**: The second type of attribute that we utilize comes from object detectors, such as the Viola-Jones detector [43], which work particularly well in video surveillance due to stationary points of view. These detectors are used to find objects, people, and vehicles. These features are simple "type" attributes attached to the location where they were generated.

3. **Tracked data**: The third is tracked data [47], from which we can extract properties like identity, object size, direction of motion, and appearance/disappearance. Tracked data serves a dual purpose in our system. First, it enables us to extract motion features in extremely low-resolution/noisy video where accurate pixel-level optical flow is prohibitively difficult to compute. It also allows us access to new features indicating that an object appeared or disappeared at a given point in time. We use these descriptors to detect people getting in and out of vehicles, as well as in other scenarios.

4. **Relationships**: In addition to extracting attributes, we compute a relationship vocabulary $\mathcal{R}$ over pairs of extracted attributes $\mathcal{A} \times \mathcal{A}$. We do not, however, store all possible relationships, as the number of potential relationships scales as the square of the number of attributes. Moreover, storing all relationships would be counterproductive to search - a search algorithm is interested in attributes and relationships which discriminate between the activity described in the query and other activities present in the archive data. Instead, we employ a simple visual intuition: important relationships are both sparse and discriminative. They are sparse in that they are relatively rare compared to the number of potential relationships, and discriminative for that same reason. Because of this utility, we precompute sparse relationships like “near” and “the same as”. We use a spatiotemporal index [17] to efficiently calculate spatiotemporal relationships, and pairwise calculations to evaluate the presence of other relationships. For our datasets, our relationships describe relative positions in space (including proximity), time, and feature space. We also have an identity relationship, indicating if two attributes likely represent the same object.

Some queries may include non-discriminative relationships. The relationship “far” is a good example of a non-discriminative relationship, as most objects in a long video are far from each other in space or time. Likewise, it’s inefficient to calculate every possible type of spatiotemporal displacement. When we receive queries containing relationships that we have not precalculated we first use other precalculated relationships to reduce the data before looking for matches to relationships which we must calculate in real-time.

We discuss specific examples of $\mathcal{A}$ and $\mathcal{R}$ in Section 4.

2.2 Semantic Graph Query

Effective zero-shot retrieval must bridge the semantic gap between a desired activity and a representation that can be associated to the data with a particular score. To accomplish this, we enable the user to arrange attributes A^q, relationships R^q and uncertainty parameters L^q into query graph $Q = G(V^q, E^q, A^q, R^q, L^q)$. The vertex properties correspond to attributes in $\mathcal{A}$, while the edge properties correspond to relationships in $\mathcal{R}$.

In addition to specifying the attributes and relationships involved in the graph structure, we allow a user to specify sets of parameters L^q_v for the attributes A^q_v in vertex $v \in V^q$ to indicate confidence in an attribute or tolerance for error. These tolerances are used in the functions that calculate the similarity between pairs of attributes a

and a' and pairs of relationships r and r', $K_{a,\ell}(a')$ and $K_{r,\ell}(r')$, respectively, defined below in Equation 7. When an attribute $a \in A_v^q$ is something like color or size, we embed it in a real vector space as μ_a. Then we measure the similarity based on RBF kernels $N(\mu_a, \ell^2)$ or employ indicators on the interval $[\mu_a - \ell, \mu_a + \ell]$ with tolerance $\ell \in L_v^q$. For attributes such as object type where a distance metric is not as obvious, these functions merely indicate whether or not the attributes are identical.

Loading Example: Consider the activity of a person loading an object into a car, showing in Figure 2, a user specifies that a vertex has the "car" attribute, and an edge with the "near" relationship with a vertex containing the "small" and "object" attributes. If the user also wanted to see that object earlier, he would add another "small" "object" vertex, with the "same as" and "before" relationships. He might specify an uncertainty parameter for the "before" relationship, indicating that anything within a minute is acceptable. In this way, he uses relationships to filter out the vast majority of potential attribute matches.

While our attribute and relationship vocabularies, $\mathcal{A}$ and $\mathcal{R}$, may change based on application, we find that a relatively small vocabulary is sufficient to characterize a wide range of interesting actions with a surprising degree of nuance.

Graph Creation: The user assembles his or her attributes and relationships using a query GUI. In order to make this assembly simple and intuitive, we ensure that all of the attributes in our attribute vocabulary $\mathcal{A}$ have semantic meaning. This differentiates our approach significantly from others [1, 19, 9, 46], which use Histogram of Gradient (HOG) or spatio-temporal interest point (STIP) features to describe their actions. While it is easy for a user to create a graph out of cars, colors, or people, it requires significant expertise to create a meaningful assembly out of HOG or STIP features. Our relationships, like "near" and "same as", also represent clear semantic concepts. For common variables like color and motion, we provide intuitive means of user input such as selecting from a color panel or click-and-drag to indicate motion. Anecdotally, this system has been used by members of the armed forces, students and surveillance professionals with little instruction, suggesting that it does not require undue expertise.

Advantages: The advantages of this representation are its simplicity and representational power. Compared to other graphical representations [5], this representation creates far smaller graphs for identical activities. Relative to representations that do not use relationships [23], it is better able to represent long-term activities and events. By reducing an activity to the relevant components, we sparsify activity search.

Meeting example: As evidence, consider an example that would challenge most modern search algorithms: "Three people meet together for five minutes". This is a challenging problem because an activity that spans five minutes or more represents a significant amount of data, and the vast majority of data within that window likely has nothing to do with the activity in question. However, the graphical representation of such a query can be made relatively simple (Figure 3) - all that matters is that the same three people who were all near each other in the beginning remain near eachother five minutes later. Because we are going to solving a sub-graph matching problem, this allows us to treat unrelated activity that happens during our activity of interest exactly like we treat unrelated activity the rest of the time, and ignore it. This enables us to search for long-term events without worrying about confuser activity over that time period.

The meeting example highlights both a strength and a weakness of the approach: the graph shown in Figure 3 is not affected by what happened in the intervening five minutes between the first meet-up and the second meet-up. The group of three could have separated and reconvened, or stayed together the whole time. However, our approach to search focuses not on the perfect modeling of the event we are searching for, but instead discriminating it from other events - and for that, this graph is sufficient.

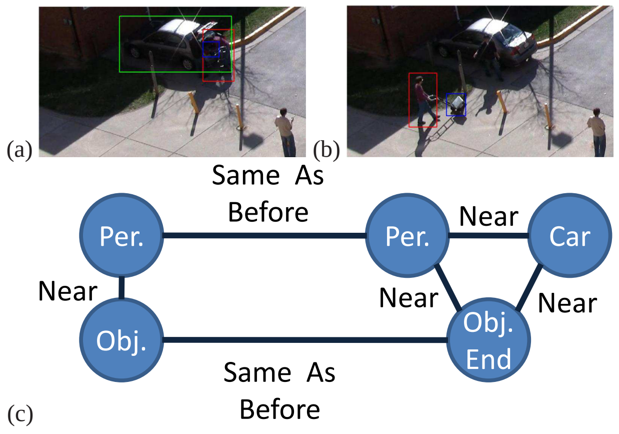

Figure 2: The graphical representation of an "object deposit" event.

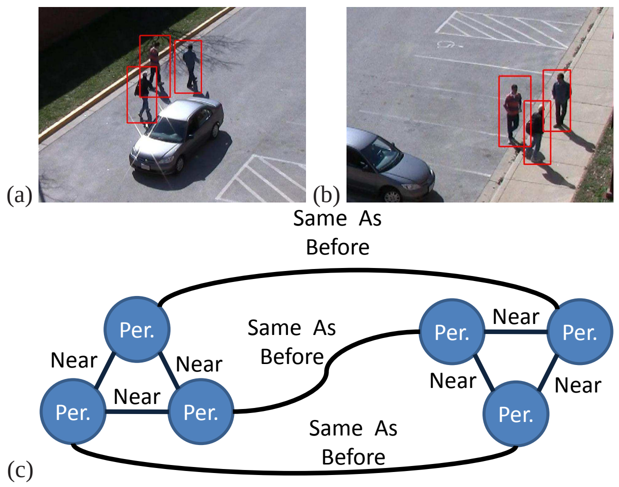

Figure 3: The graphical representation of a "meeting" event.

2.3 Efficient Indexing

Discriminative attributes and relationships are powerful if you can efficiently use their sparsity to rule out parts of a video that don't match a provided query. In this section, we describe how to store the attributes and relationships efficiently so that we can rapidly recover elements that might match a given query $Q = G(V^q, E^q, A^q, R^q, L^q)$.

To accomplish this recovery, we independently hash the locations and values of every attribute we find in the video archive to an inverted index. We use Locality Sensitive Hashing (LSH) [10] for attributes like size and color which have computable distances, and employ a standard hash table for discrete attributes like object type. Fuzzy hashing renders our search robust to minor discrepancies in query model, detection model, and storage by retrieving all archived attributes with a certain radius of the query attribute. By computing a hash code for each of a vertex's attributes, A_v^q, and taking the intersection of the bins corresponding to those codes, we can efficiently retrieve the set of matching locations to vertex $v \in V^q$.

We apply the same logic to the set of relationships. When a relationship exists between a particular pair of locations, we hash it (and its value), to a fuzzy or normal hash table depending on the relationship type. This allows us to retrieve the set of pairs of locations which approximately satisfy a particular relationship in time proportional to the number of satisfying pairs.

The power of this approach is that it allows us to efficiently downsample the full set of archive data to the set of potentially relevant attributes and relationships, in time proportional to the number of attributes and relationships returned, as opposed to the size of the original data.

3. SEARCH

The goal of search is to rapidly recover from the indexed data the set of attributes and relationships that best match the query graph Q. We accomplish this in three steps: First, we independently look up each attribute and relationship in Q in our data store to retrieve a coarse graph C. Then we calculate discriminative weights for each of the relationships in the query, then calculate the maximally discriminative spanning tree T^* for that query. Finally, we create a matching graph H to encode the possible matches in the coarse graph C for the spanning tree T^* and use a Maximally Discriminative Subgraph Matching (MDSM) approach to recover the ranked matches to T^*.

3.1 Coarse Graph Construction

Given a query graph Q, we construct the coarse archive graph $C = G(V^c, E^c, A^c, R^c)$ in two steps. First, for every vertex $v \in V^q$, we use our data store to retrieve the set of locations where all attributes in A^q_v are present. We add a vertex v_i to V^c for each of those locations and store the attribute values in $A^c_{v_i}$. Second, for every discriminative (hashed) edge $(v_1, v_2) \in E^q$, we retrieve the set of attribute pairs that satisfy every relationship in $R^q_{(v_1,v_2)}$ and whose attributes are in V^c. For each of these pairs, we add an edge to E^c and store the associated relationships in R^c. The computational complexity of this step is proportional to the number of vertices and edges that end up matching the query graph, $O(|V^c| + |E^c|)$.

3.2 Tree Selection

Downsampling the data to the set of potentially relevant vertices and edges provides incredible cost savings, but the result is still frequently a coarse graph with hundreds of thousands of vertices. Modern subgraph matching algorithms for attributed relational graphs only scale to hundreds of vertices [8], absent approximation by massive parallelization. As such, we are obliged to downsample the coarse graph further.

The optimal downsampled graph, C^*_q would contain only vertices and edges that were present in high-ranked matches to query graph Q so as to minimize the time spent performing an expensive search. We approximate C^*_q by selecting a spanning tree T of Q and downsampling the coarse graph C to C_t, the set of vertices and edges involved in high-ranking solutions to that tree. Because the tree effectively relaxes constraints (in the form of edges) of the optimal solution to the query graph, C_t is guaranteed to contain all of the solutions which match the query graph, with added false alarms. To generate such trees, we first compute a set of weights indicating the discriminative power of each attribute and relationship in Q, then calculate the spanning tree T which maximizes that discriminative power.

3.2.1 *Weight Computation*

The choice of which spanning tree to select has significant runtime implications. Its creation involves the removal of edges from Q, and not all edges are created equal. During the archival process, we assign probabilities $p(a)$ and $p(r)$ to each attribute and relationship that we store. These functions denote the probability that a randomly-chosen attribute or relationship in the archive is a match to the attribute a or relationship r. For example, in a dataset that surveils a highway, the "car" attribute is not particularly discriminative, because almost everything on a highway is a car. The "bicycle" attribute in the same dataset would be extremely discriminative, as it is rare for people to bicycle on the highway. We show the probabilities $p(a)$ for attributes used in the VIRAT [26] data set in Figure 4.

Relationships, in particular, have greater power to be discriminative because the set of potential relationships is $|\mathcal{A}|^2$. Take Figure 2; while the "Person near car" relationship is normally very discriminative, 80% of the dataset is shot in parking lots, where people are frequently near cars. Objects disappear near cars far less less frequently - thus, a tree rooted at the "object disappears" vertex and connecting through the "near" edge to the "car" vertex is more discriminative than one starting elsewhere.

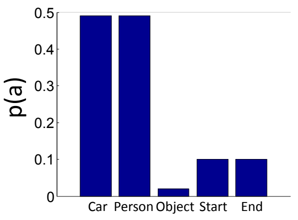

Figure 4: The probabilities associated with five of the attributes in the VIRAT dataset. Being an object, as opposed to a car or a person, is the most unlikely attribute and thus the most discriminative.

We compute empirical values for $p(a)$ and $p(r)$ during the archival process by computing the percentage of attributes (and relationships) that are returned by the hash tables described in Section 2.3. If relationships have not been hashed, they are assumed to be nondiscriminative and assigned values of $p(r) = 1$. If it is later determined that these relationships are discriminative, we can revise our estimate of $p(r)$.

3.2.2 *Maximally Descriminative Spanning Tree*

Absent additional information indicating the distribution of relationships and attributes in the video corpus, we assume that all attributes and relationships are generated independently. This model is clearly imperfect, as certain pairs of attributes are more likely to have a particular relationship. A speed attribute of 60 miles per hour is far more likely to be found in the same location as a car object, as opposed to a person object. Given a diverse set of video corpora, one could learn these correlations and adjust our model accordingly, but in this work we assume independence.

Given a query tree $T = G(V^t, E^t, A^t, R^t)$, we compute weights $p(v)$ for all $v \in V^t$ and $p(e)$ for all $e \in E^t$ as the product of the matching probabilities of the attributes in node v or relationships in edge e. That is,

$$p(v) = \prod_{a \in A^t_v} p(a), \quad p(e) = \prod_{r \in R^t_e} p(r) \tag{1}$$

Since nodes and edges in tree T are independent, the total matching probability of the tree $p(T)$ to the archive is:

$$p(T) = \prod_{v \in V^t} p(v) \prod_{e \in E^t} p(e) \tag{2}$$

As noted, we are going to do a search using T instead of the original query graph Q in order to reduce the coarse graph C. We therefore select a tree T which results in the fewest possible matches so as to generate the maximal reduction in the coarse graph. That is our novel maximially discriminative spanning tree.

Definition (Maximially Discriminative Spanning Tree (MDST))**.** *We call a tree T^* an MDST of query graph Q with node weights $p(v) \forall v \in V^q$ and edge weights $p(e) \forall e \in E^q$, if the tree satisfies*

$$T^* = \operatorname*{argmin}_{T \in \mathcal{T}} p(T), \tag{3}$$

where $\mathcal{T}$ denotes the set of all possible trees induced from query graph Q.

Lemma 3.1. *Assume that there is no such node in query graph Q that the matching probabilities of the node and all its connected edges are 1. Then the MDST of a query graph Q is equivalent to the minimum spanning tree of Q.*

Proof. Based on the definition of MDST, we can rewrite Eq. 3 as follows:

$$\begin{aligned} T^* &= \operatorname*{argmin}_{T \in \mathcal{T}} p(T) = \operatorname*{argmin}_{T(V^t, E^t) \in \mathcal{T}} \prod_{v \in V^t} p(v) \prod_{e \in E^t} p(e) \\ &\equiv \operatorname*{argmin}_{T(V^t, E^t) \in \mathcal{T}} \left\{ \sum_{v \in V^t} \log p(v) + \sum_{e \in E^t} \log p(e) \right\} \end{aligned} \tag{4}$$

Since $\forall p(v), \forall p(e)$ are real numbers between 0 and 1, $\forall \log p(v)$ and $\forall \log p(e)$ are all non-positive. Therefore, in order to minimize Eq. 4, all the nodes in Q will be included, and all the specific edges will be included as well, which (1) form a tree together with all the nodes in Q, and (2) achieve the minimum summation. Deleting any node/edge from the tree will either increase the objective value in Eq. 4 or destroy the tree structure. This is exactly the same as the definition of minimum spanning tree. □

Lemma 3.1 provides us with a convenient way to calculate the MDST that should produce the fewest possible matches. Notice that by matching with MDST, our method can return all the solutions which satisfy all the nodes and some of the edges in query graph Q. Given that $T^* \neq Q$, we will prove that the set of matches to MDST contains all of the matches to the original query graph.

Lemma 3.2. *Let M denote a match between a query graph $Q = (V^q, E^q)$ and a coarse graph C. Let $S(\cdot, C, M)$ denote the matching score between any query (e.g. graphs, trees, attributes, relationships) and C, and let $Q' = Q - T^* = (\emptyset, E^q - E^t)$ denote the residual of graph Q. We further define $S(Q, C, M) = \sum_{v \in V^q} S(v, C, M) + \sum_{e \in E^q} S(e, C, M)$. Then we have*

$$\begin{aligned} |S(Q, C, M) - S(T^*, C, M)| &= \left| \sum_{e \in Q'} S(e, C, M) \right| \\ &\leq \sum_{e \in Q'} |S(e, C, M)| \leq |Q'| \cdot \max_{e \in Q'} \max_{\tilde{M} \in \mathcal{M}} |S(e, C, \tilde{M})|, \end{aligned} \tag{5}$$

where $|Q'|$ denotes the number of edges in Q', and $\mathcal{M}$ denotes the set of possible matches between Q' and C.

Proof. $\max_{e \in Q'} \max_{\tilde{M} \in \mathcal{M}} |S(e, C, \tilde{M})|$ denotes the best possible absolute matching score for an arbitrary edge in Q', which leads to $\forall e \in Q', |S(e, C, M)| \leq \max_{e \in Q'} \max_{\tilde{M} \in \mathcal{M}} |S(e, C, \tilde{M})|$. Together with inequality properties, we can easily prove the lemma. □

Lemma 3.2 shows that the score difference between using query graph Q and using its MDST T^* is upper-bounded. Using this, we can set our parameters to make sure that all good matches to the query graph Q are present in the set of matches to the MDST T^*. We prove this in Lemma 3.3.

Lemma 3.3. *By setting proper thresholds, we guarantee that the matches returned by MDST cover all the matches for the original query graph.*

Proof. Lemma 3.2 shows that the score difference between using query graph Q and using its MDST T^* is upper-bounded. Let $S_Q = \min_{M_Q \in \mathcal{M}_Q} S(Q, C, M_Q)$, where $\mathcal{M}_Q$ denotes the set of possible matches between Q and C, then in the worst case, we can set the matching score threshold δ for returning solutions as

$$\delta = S_Q - \Delta_{Q'} = S_Q - |Q'| \cdot \max_{e \in Q'} \max_{\tilde{M} \in \mathcal{M}} |S(e, C, \tilde{M})|, \tag{6}$$

which is guaranteed to return all the solutions for query graph Q. □

As defined in Lemma 3.3, δ ensures that every perfect match will be present in the coarse graph C_t. If a user is tolerant of error, they can also set the threshold higher to $\delta + \tau$, guaranteeing that all matches within τ of perfect are preserved. This is the performance/speed tradeoff of the MDST: a higher τ allows for solutions with slightly more error, but it results in a larger coarse graph C_t and thus slower matching. In practice, we have found τ stable and simple to tune.

3.3 Maximally Discriminative Subgraph Matching (MDSM)

3.3.1 Scoring Attributes & Relationships

Because of visual ambiguity, compositional variance and spatiotemporal distortion, we do not want to search for identical matches to the query graph. Not only is it unreasonable to expect a perfect description of an activity from a user, it is improbable that all examples of that activity within the dataset fit a particular description. To this end, we look for a score function with some tolerance for error. This error manifests in two parts - a kernel function on the vertex matches, K_v, and a kernel function on the edges, K_e. For vertex attributes, our similarity functions are the sum of similarities for each attribute belonging to that vertex. Similarity functions for relationships are also the sum of the similarities associated with that edge. For notational convenience, we express vertex attributes, edge relationships, and their corresponding tolerances in terms of binary vectors $A_v \in \{0,1\}^{|\mathcal{A}|}$, $R_e \in \{0,1\}^{|\mathcal{R}|}$, $L_v \in [0,1]^{|\mathcal{A}|}$, $L_e \in [0,1]^{|\mathcal{R}|}$ for any arbitrary vertex v and edge e.

Let v_1 be a vertex in query graph Q, whose value and tolerance for the j-th attribute in $\mathcal{A}$ are given by $a_1(j)$ and $\ell_1(j)$, respectively. Let v_2 be a vertex from the archive graph, whose j-th attribute is given by $a_2(j)$. Let e_1,e_2 represent edges in the archive and query graphs, with values $r_1(j)$ and $r_2(j)$ with tolerance $\ell_1(j)$.

$$K_v(v_1, v_2) = \sum_{j=1}^{|\mathcal{A}|} K_{a_1(j),\ell_1(j)}(a_2(j))$$
$$K_e(e_1, e_2) = \sum_{j=1}^{|\mathcal{R}|} K_{r_1(j),\ell_1(j)}(r_2(j)) \qquad (7)$$

Note that we are only provided with tolerance for a given vertex or edge in the query, because the other vertex or edge generally represents archive data.

Given a coarse graph C and the MDST T^*, we want to select a matching $M : V^t \to V^c$. We select the matching that maximizes our objective function $S(Q, C, M)$ comprised of two parts, vertex and edge potentials

$$S(Q, C, M) = \sum_{v_q \in V^q} K_v(v_q, M(v_q)) + \qquad (8)$$
$$\lambda \sum_{(v_{q_1}, v_{q_2}) \in E^q} K_e(v_{q_1}, v_{q_2}, M(v_{q_1}), M(v_{q_2}))$$

where λ is a weighting parameter determining the relative value of matching vertices and edges.

3.3.2 Matching Graph Creation

We solve for the optimal match between the MDST T^* and archive graph C in two steps. In the first step, we build a *matching graph* $H = G(V^h, E^h)$, where each vertex $v \in V^h \subseteq V^t \times V^c$ is a tuple denoting a proposed assignment between a vertex in T^* and a vertex in C, and each edge $e \in E^h$ denotes a relationship between the two assignments. We create H by first adding matches for the root, then adding in matches to its successors which satisfy both vertex and edge relationships described in T^*. We then define score thresholds τ_v and τ_e to be the minimum score for vertices and edges. This process in described in Algorithm 1, and the expected number of vertices scales as a product of $p(T^*)$ and the size of the archive data.

Algorithm 1 Create Matching Graph

```
 1: procedure CREATE MATCHING GRAPH($T^*, C, K_v(v_1, v_2), K_e(v_1, v_2, v_3, v_4)$)
 2:     $H = G(V^h, E^h) \leftarrow \emptyset$
 3:     Iterate from root to leaves
 4:     for all $v^t \in V^t$ do
 5:         Compute the matches to this vertex
 6:         $N_{v_t} \leftarrow (v_t, v_c)$ where $K_v(v_t, v_c) > \tau_v$
 7:         if $Parent(v_t) \neq \emptyset$ then
 8:             $E \leftarrow \emptyset$
 9:             for all $(v_{t_p}, v_{c_p}) \in N_{Parent(v_t)}$ do
10:                 $E \leftarrow E \cup (v_t, v_c)$ where $K_e(v_{t_p}, v_{c_p}, v_t, v_c) > \tau_e$
11:             end for
12:             $N_{v_t} \leftarrow N_{v_t} \cap E$
13:             $V^h \leftarrow E^h \cup N_{v_t}$
14:             $E^h \leftarrow E^h \cup E$
15:         else
16:             $V^h \leftarrow V^h \cup N_{v_t}$
17:         end if
18:     end for
19: end procedure
```

3.3.3 Retrieval with MDSM

Having traversed T^* from root to leaves to create a matching graph, we traverse it from leaves to root to determine the optimal solution for each root match. For each leaf vertex in T^*, we merge vertices in H with their parents, keeping the one which has the best score. We repeat this process until only matches to root vertices are left, and then sort these root vertices by the score of the best tree which uses them. This process is described in Algorithm 2, and has complexity of $O(|E^h|)$.

Algorithm 2 Solve for Optimal Matches

```
 1: procedure OPTIMIZE MATCHES($T^*, H$)
 2:     $Score(v) \triangleq K_v(v[0], v[1])$
 3:     $Score(v_1, v_2) \triangleq K_e(v_1[0], v_1[1], v_2[0], v_2[1])$
 4:     Iterate from leaves to root
 5:     for all $v_t \in T^*$ do
 6:         if $Parent(v_t) \neq \emptyset$ then
 7:             for all $v \in V^h$ where $v[0] == Parent(v_t)$ do
 8:                 $Score(v) += max_{c \in Children(v)}(Score(c) + Score(v, c))1(c[0] == v_t)$
 9:             end for
10:         end if
11:     end for
12: end procedure
```

This process yields a set of matches, M_{T^*}, for each potential activity - generally on the order of the number of true matches in the data. We then iterate through each match $m \in M_{T^*}$ and compute $S(Q, C, m)$, filtering matches that have poor scores for the edges which were not included in the MDST. This allows us to have the speed of the MDSM approach and the effective quality of the full graph-matching result.

4. EXPERIMENTATION

The novelty of this method is its ability to reason over events that are not necessarily extremely time-localized. Other approaches [22, 3, 23] are limited to events which span relatively short amounts of time (seconds), whereas our subgraph matching formulation is agnostic to activity duration. For the purposes of comparison, we compare performance on both local events like dismounts and object loading as well as longer events like group meetings.

4.1 Comparisons

The most comparable approach to our algorithm is the system described in [3]. The authors formulate the optimization purely as a dynamic program and assume that the input graph Q is a line graph. Given a line graph representing a series of temporal events, the system expects events to happen sequentially, but employs dynamic time warping to allow for flexible duration.

Specifically, given a sequence of vertices $V_1, V_2, ...V_i$, they score activities using a Markov model with three types of fixed transition probabilities. When an activity stays in the same state, the transition probability $P_{i,i}$ is a continuation or a missed detection, depending on evidence support. When the activity moves to the next state, the transition probability $P_{i,i+1}$ is either a match or a deletion, depending on whether or not an archive match exists. Given that their model is a line, they can recover the set of optimal matches in polynomial time.

While this works well for sequential events over relatively short periods of time that always have something happening, it fails at detecting events that have a graph structure, multiple salient objects at a given point in time, or minutes of duration. Specifically, the linear decay in likelihood that occurs from not seeing the next vertex is a poor model when the next piece of evidence could be minutes away. We compare to the baseline established in this method, and note that we expect comparable performance on short-term chains of events and improved performance in other areas.

We compare this baseline against two other approaches, both utilizing the MDST reduction. In the first, we use MDSM to rank matches to the MDST, T^*. We select the top few of these matches, and re-score them using the query graph instead of the MDST. In the second, we evaluate every subgraph in the matching graph H and rank it using query graph Q. By comparing quality against a brute-force approach, we show the error due to modeling.

4.2 Datasets

We evaluate the performance of our algorithm on two data sets representing different surveillance modalities. The first is a wide-area persistent surveillance sensor flying over Yuma, Arizona. This dataset has an extremely low framerate (.66 FPS), combined with large black-and-white frames (64 megapixels) over a large area, totalling to 1 TB of imagery, or a graph with 1 million nodes and 5 million edges. Our hash table stores features averaging .5 MB per GB of data. Because of the size of the area, there are relatively few pixels on a target. Vehicles have roughly 100 pixels and people have around 10. See an example of a u-turn activity in Figure 5.

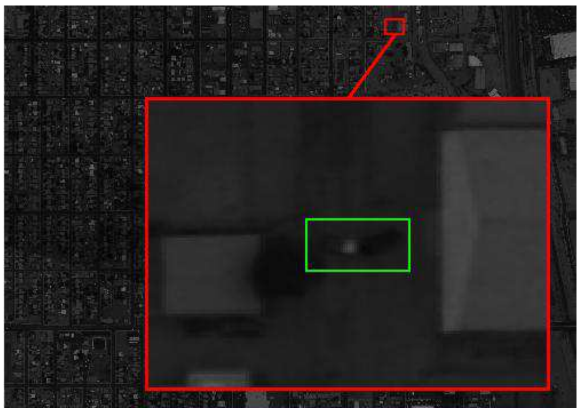

Figure 5: The YUMA video data set features high-resolution imagery taken at significant elevation, leaving few pixels on targets as small as people or vehicles.

Because of the low resolution on target, pixel-level features are extremely noisy in the YUMA dataset. For this dataset, we rely on a very simple tracking approach to generate track IDs, bounding boxes, time-stamps and size. We extract from these tracks a series of locations with the features of object size, motion orientation and magnitude and a start/end attribute denoting the beginning and termination of the track that serve as our attribute vocabulary $\mathcal{A}$, similar to the raw attributes extracted in [18, 3]. We employ object size as a way of disambiguating cars and people - carried objects are not visible at this resolution. The discriminative relationship vocabulary $\mathcal{R}$ contains a “near” operator and an identity operator which provides a confidence that two elements are from the same object. Because these videos contain no camera zoom, “near” can be defined in units of pixels. Spatiotemporal and feature displacement are the non-discriminative relationships that are used in queries, but not hashed. For the VIRAT dataset, which has significantly higher target resolution, we also include object type as a discrete attribute, as object type can reliably be detected.

The other surveillance modality we explored was street surveillance from building-mounted cameras. For this, we utilized the VIRAT 2.0 Ground Dataset [26], a popular surveillance dataset containing 35 GB of video and represented in a graph of 200,000 nodes and 1 million edges. These are relatively standard 2 megapixel surveillance cameras shooting at 30 frames per second, and we extract 2.5 MB of features per GB of data. Because of the smaller field of view, there are far more pixels on target, enabling basic object recognition. As such, we define $\mathcal{A}$ to include object type (person, vehicle, bag) as well as the attributes used in the YUMA dataset. The relationship vocabulary $\mathcal{R}$ is identical. To further differentiate it from the YUMA dataset, the VIRAT ground dataset contains 315 different videos covering 11 scenes rather than a single large video covering one scene.

Query	MDST Reduction	Baseline[3]	Brute	Brute (MDST)	DP (MDST)
Long Meeting	85	6.5	*3e25*	4610	2.3
Short Meeting	86	6	*3e25*	10671	2.5
Deposit	<1	12.5	*4.5e16*	5	<1
Removal	<1	14.4	*4.7e16*	4.9	<1
Mount	<1	6.3	*3.7e8*	<1	<1
Dismount	<1	7.7	*3.7e8*	<1	<1
UTurn	<1	2.0269	*6.6e8*	<1	<1
Dismount	<1	<1	<1	<1	<1
Mount	<1	<1	<1	<1	<1
Suspicious Stop	<1	<1	32.2820	<1	<1

Table 1: The run times for baseline [3], brute force and DP algorithms on the VIRAT (top) and YUMA (bottom) datasets. When a brute force algorithm is infeasible, we estimate runtime based on a sub-set of solutions.

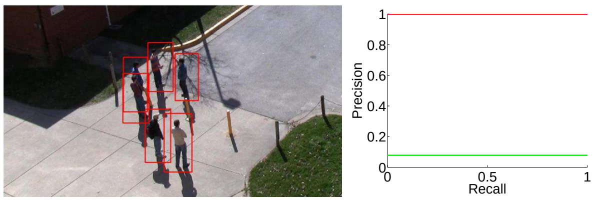

Figure 6: Precision/recall curves for the long-term meeting activities in the VIRAT video dataset. Dynamic programming and brute force recover all three instances as top returns.

4.3 Results

We test our algorithm against a baseline feature-accumulation approach used in [3], as well as a brute-force enumeration of possibilities. Precision/recall curves for the YUMA dataset are shown in Figure 8; the VIRAT results are shown in Figures 7 and 6.

We also show the run-time of our approach in Table 1, demonstrating the futility of solving a large-scale graph search problem without performing intelligent reduction first. It should surprise nobody that an algorithm which must explore all $|V^q|$-sized subsets of the data will take forever to run on a large dataset. More relevant is how long it takes us to compute the MDST and downsample the data to the relevant subset: Less than a second with pre-hashed relationships (all examples except meetings), and 80 seconds when we do not have pre-hashed relationships. When we do not have hashed relationships, our algorithm must compute pair-wise relationships, which is expensive to do even when the data is reduced.

The "before" relationship that is used in the meeting events is not hashed because it is not particularly local. Many elements of a dataset satisfy the relationship that one of them comes significantly before the other. Therefore, this relationship is not particularly discriminative by itself, so we would have no a-priori reason to calculate it and store it. Here we run into a classic memory vs. runtime trade off, where hashing less-discriminative relationships may turn out to later be valuable, but is prohibitively memory expen-

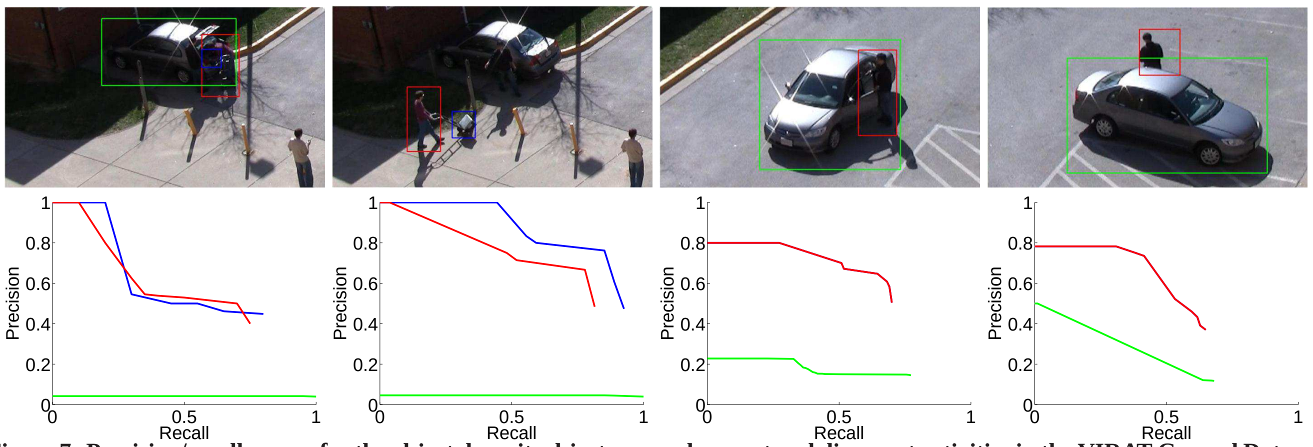

Figure 7: Precision/recall curves for the object deposit, object removal, mount and dismount activities in the VIRAT Ground Dataset. We show baseline [3] search results in green, MDSM results in red, and brute force results in blue when they diverge from MDSM results.

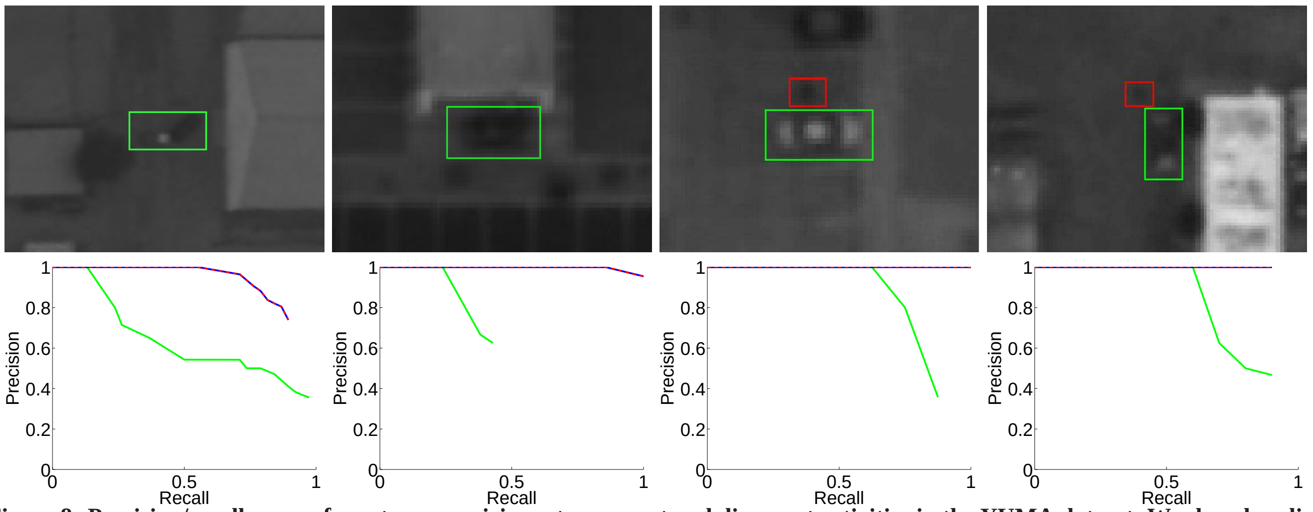

Figure 8: Precision/recall curves for u-turn, suspicious stop, mount and dismount activities in the YUMA dataset. We show baseline [3] search results in green, and unfiltered MDSM results in red. Because these queries are all trees, dynamic programming produces identical results to brute force, shown in the dashed blue and red line.

sive. The "before" attribute becomes discriminative when paired with the "same as" attribute so our algorithm performs successful reduction, albeit slower due to a lack of hashing.

The performance of our approach, compared to baseline and brute force solutions, is shown in Figures 7 and 8. Our algorithm performs extremely favorably relative to the baseline, and comparably to brute force in a number of scenarios. The performance gap for simple, time-localized graphs on the YUMA data is relatively small, so the baseline approach performs relatively well. In addition, the simple/time-localized graphs have relatively few cases of complex spatial relationships that are difficult to model with time-sequences like those in Figure 2.

The performance of the baseline tails off as the scale of the events begins to grow from pick-up/drop-off to object deposit and withdrawal and widens further as query duration increases. Over a large time scale, the number of unrelated events within the sliding window overwhelms the true information, creating false alarms. The subgraph matching approach remains agnostic to the additional clutter, and performs consistently across all queries.

These experiments convincingly showcase our approach's ability to represent events with multiple entities over multiple time scales while still maintaining strong performance in precision and recall.

5. CONCLUSIONS

In this paper, we presented a practical solution to video search in both run-time and performance quality. Our method has many advantages. First, it is flexible; we are able to model every activity present in our two video corpora, and many activities (such as long-term meetings) that were not intentionally present. Second, it is fast; we downsample the data using the MDST and solve most examples in a matter of seconds using maximally discriminative subgraph matching. Finally, use of a sparse graphical representation of complex activities makes it surprisingly flexible and robust. The subgraph matching formulation allows it to ignore foreground clutter and confusion to identify which elements of the archival data match the query optimally. These attributes combine to make it a powerful new tool in video search.

This material is based upon work supported in part by the U.S. Department of Homeland Security, Science and Technology Directorate,Office of University Programs, under Grant Award 2013-ST-061-ED0001, by ONR Grant 50202168 and US AF contract FA8650-14-C-1728. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the social policies, either expressed or implied, of the U.S. DHS, ONR or AF.

6. REFERENCES

[1] Y. Aytar, M. Shah, and J. Luo. Utilizing semantic word similarity measures for video retrieval. *Computer Vision and Pattern Recognition*, 2008.

[2] Y. L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. *Computer Vision and Pattern Recognition*, pages 2559–2566, 2010.

[3] G. Castanon and A. Caron. Exploratory search of long surveillance videos. *ACM Multimedia*, 2012.

[4] O. Çeliktutan, C. Wolf, B. Sankur, and E. Lombardi. Real-time exact graph matching with application in human action recognition. *Human Behavior Understanding*, pages 17–28, 2012.

[5] T. E. Choe, H. Deng, F. Guo, M. W. Lee, and N. Haering. Semantic video-to-video search using sub-graph grouping and matching. *International Conference on Computer Vision*, 2013.

[6] W. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *PAMI*, 17(8), 1995.

[7] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *Pattern Analysis and Machine Intelligence*, 26(10):1367–72, Oct. 2004.

[8] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. *Advances in Neural Information Processing Systems*, 19:313, 2007.

[9] J. Dalton, J. Allan, and P. Mirajkar. Zero-shot video retrieval using content and concepts. *Information and Knowledge Management*, pages 1857–1860, 2013.

[10] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computation Geometry*, pages 253–262. ACM, 2004.

[11] F. De la Torre. Factorized graph matching. *Computer Vision and Pattern Recognition*, pages 127–134, June 2012.

[12] S. Demeyer, T. Michoel, J. Fostier, P. Audenaert, M. Pickavet, and P. Demeester. The Index-Based Subgraph Matching Algorithm (ISMA): Fast Subgraph Enumeration in Large Networks Using Optimized Search Trees. *PLoS ONE*, 8(4), 2013.

[13] P. Dollár and V. Rabaud. Behavior recognition via sparse spatio-temporal features. *Performance Evaluation of Tracking and Surveillance*, pages 65–72, 2005.

[14] M. Elhoseiny, B. Saleh, and A. Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. *International Conference on Computer Vision*, pages 2584–2591, 2013.

[15] U. Gaur, Y. Zhu, B. Song, and A. Roy-Chowdhury. A "string of feature graphs" model for recognition of complex activities in natural videos. *International Conference on Computer Vision*, 2011.

[16] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *Pattern Analysis and Machine Intelligence*, 1996.

[17] A. Guttman. R-trees: a dynamic index structure for spatial searching. *SIGMOD '84*, pages 47–57, 1984.

[18] W. Hu, D. Xie, Z. Fu, W. Zeng, and S. Maybank. Semantic-based surveillance video retrieval. *Transactions on Image Processing*, 2007.

[19] M. Y. Jung and S. H. Park. *Semantic Similarity Based Video Retrieval*. 2009.

[20] S. Kwak, B. Han, and J. H. Han. Multi-agent Event Detection: Localization and Role Assignment. *Computer Vision and Pattern Recognition*, pages 2682–2689, June 2013.

[21] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64:107–123, 2005.

[22] T.-L. Le, M. Thonnat, A. Boucher, and F. Brémond. Surveillance Video Indexing and Retrieval Using Object Features and Semantic Events. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(07):1439–1476, 2009.

[23] D. Lin, S. Fidler, C. Kong, and R. Urtasun. Visual Semantic Search : Retrieving Videos via Complex Textual Queries. *Computer Vision and Pattern Recognition*, 2014.

[24] S. Maybank. A Survey on Visual Content-Based Video Indexing and Retrieval. *IEEE Transactions on Systems, Man, and Cybernetics*, 41(6):797–819, 2011.

[25] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words. *International Journal of Computer Vision*.

[26] S. Oh, A. Hoogs, and A. Perera. A large-scale benchmark dataset for event recognition in surveillance video. *Computer Vision and Pattern Recognition*, (2), 2011.

[27] M. Palatucci and D. Pomerleau. Zero-shot learning with semantic output codes. *Neural Information Processing Systems (NIPS)*, 2009.

[28] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. *Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[29] M. Rohrbach, M. Stark, and B. Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. *Computer Vision and Pattern Recognition*, (June):1641–1648, 2011.

[30] O. Russakovsky and L. Fei-Fei. Attribute learning in large-scale datasets. *ECCV*, 6553 LNCS(PART 1):1–14, 2012.

[31] M. Ryoo and J. Aggarwal. Recognition of composite human activities through context-free grammar based representation. *Computer Vision and Pattern Recognition*, 2006(June), 2006.

[32] M. Ryoo and W. Yu. One video is sufficient? Human activity recognition using active video composition. *Workshop on Applications of Computer Vision*, (January), 2011.

[33] M. S. Ryoo and J. K. Aggarwal. Stochastic Representation and Recognition of High-Level Group Activities. *International Journal of Computer Vision*, 93(2):183–200, June 2010.

[34] S. Savarese, A. DelPozo, and J. C. Niebles. Spatial-Temporal correlatons for unsupervised action classification. *IEEE Workshop on Motion and Video Computing*, pages 1–8, Jan. 2008.

[35] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local SVM approach. *International Conference on Pattern Recognition*, pages 3–7, 2004.

[36] C. G. Snoek and M. Worring. Concept-Based Video Retrieval. *Foundations and Trends in Information Retrieval*, 2009.

[37] Z. Sun, H. Wang, B. Shao, and J. Li. Efficient subgraph matching on billion node graphs. *Proceedings of the VLDB Endowment*, pages 788–799, 2012.

[38] Y.-L. Tian, A. Hampapur, L. Brown, R. Feris, M. Lu, A. Senior, C.-F. Shu, and Y. Zhai. Event Detection, Query, and Retrieval for Video Surveillance. In *Artificial Intelligence for Maximizing Content Based Image Retrieval*. 2008.

[39] P. H. S. Torr. Solving Markov Random Fields using Semi Definite Programming. *Artificial Intelligence and Statistics*, 2003.

[40] J. R. Ullmann. An Algorithm for Subgraph Isomorphism. *Journal of the ACM*, 23(1):31–42, Jan. 1976.

[41] K. Van De Sande, T. Gevers, and C. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

[42] J. C. Van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. M. Geusebroek. Visual word ambiguity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1271–1283, 2010.

[43] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition*, 2001.

[44] B. H. S. V. N. Vishwanathan. Efficient Max-Margin Multi-Label Classification with Applications to Zero-Shot Learning. *Machine Learning*, 88(1-2):127–155, 2010.

[45] H. Wang, A. Kläser, C. Schmid, L. Cheng-Lin, and L. Cheng. Action Recognition by Dense Trajectories. *Computer Vision and Pattern . . .*, pages 3169–3176, 2011.

[46] S. Wu, S. Bondugula, F. Luisier, X. Zhuang, and P. Natarajan. Zero-Shot Event Detection Using Multi-modal Fusion of Weakly Supervised Concepts. *Computer Vision and Pattern Recognition*, pages 2665–2672, June 2014.

[47] T. Yang, S. Z. Li, Q. Pan, and J. Li. Real-time Multiple Objects Tracking with Occlusion Handling in Dynamic Scenes. *Computer Vision and Pattern Recognition*, (60172037), 2005.

[48] J. Yuan, Z.-J. Zha, Y.-T. Zheng, M. Wang, X. Zhou, and T.-S. Chua. Learning concept bundles for video search with complex queries. *ACM Multimedia*, 2011.

[49] J. Yuan, Y. Zhao, H. Luan, M. Wang, and T. Chua. Memory Recall Based Video Search: Finding Videos You Have Seen Before Based on Your Memory. *ACM Trans. Multimedia Comput. Commun. Appl.*

[50] D.-q. Zhang and S.-f. Chang. Stochastic Attributed Relational Graph Matching for Image Near-Duplicate Detection. *ACM Multimedia*, 2004.