To appear, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

Efficient Activity Detection with Max-Subgraph Search

Chao-Yeh Chen and Kristen Grauman
University of Texas at Austin
chaoyeh@mail.utexas.edu, grauman@cs.utexas.edu

Abstract

We propose an efficient approach that unifies activity categorization with space-time localization. The main idea is to pose activity detection as a maximum-weight connected subgraph problem over a learned space-time graph constructed on the test sequence. We show this permits an efficient branch-and-cut solution for the best-scoring—and possibly non-cubically shaped—portion of the video for a given activity classifier. The upshot is a fast method that can evaluate a broader space of candidates than was previously practical, which we find often leads to more accurate detection. We demonstrate the proposed algorithm on three datasets, and show its speed and accuracy advantages over multiple existing search strategies.

1. Introduction

The activity detection problem entails both *recognizing* and *localizing* categories of activity in an ongoing (meaning "untrimmed") video sequence. Reliable activity detection would have major practical value for applications such as video indexing, surveillance and security, and video-based human computer interaction.

While the recognition portion of the problem has received increasing attention in recent years, state-of-the-art methods largely assume that the space-time region of interest to be classified has already been identified (e.g., [25, 14]). However, for most realistic settings, a system must not only name what it sees, but also partition out the temporal or spatio-temporal extent within which the activity occurs. The distinction is non-trivial; in order to properly recognize an action, the spatio-temporal extent usually must be known *simultaneously*.

To meet this challenge, existing methods tend to separate activity detection into two distinct stages: the first generates space-time candidate regions of interest from the test video, and the second scores each candidate according to how well it matches a given activity model (often a classifier). Most commonly, candidates are generated either using person-centered tracks [17, 19, 27, 11] or using exhaustive sliding window search through all frames in the video [9, 5, 21]. Both face potential pitfalls. On the one hand, a method reliant on tracks is sensitive to tracking failures, and by focusing on individual humans in the video, it overlooks surrounding objects that may be discriminative for an activity (e.g., the car a person is approaching). On the other hand, sliding window search is clearly a substantial computational burden, and its frame-level candidates may be too coarse, causing clutter features to mislead the subsequent classifier. In both cases, the scope of space-time regions even considered by the classifier is artificially restricted, e.g., to person bounding boxes or a cubic subvolume.

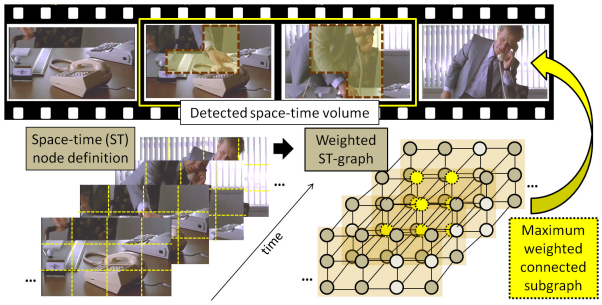


Figure 1. Our approach constructs a space-time video graph, and efficiently finds the subgraph that maximizes an activity classifier's score. The detection result can take on non-cubic shapes (see dotted shapes in top frames), as demanded by the action.

Our goal is to unify the classification and localization components into a single detection procedure. We propose an efficient approach that exploits top-down activity knowledge to quickly identify the portion of video that maximizes a classifier's score. In short, it works as follows. Given a novel video, we construct a 3D graph in which nodes describe local video subregions, and their connectivity is determined by proximity in space and time. Each node is associated with a learned weight indicating the degree to which its appearance and motion support the action class of interest. At this point, we show the detection problem is equivalent to solving a *maximum-weight connected subgraph* problem, meaning to identify the subset of connected nodes whose total weight is maximal; for our setting, this in turn is reducible to a prize-collecting Steiner tree problem,

for which practical branch-and-cut optimization strategies are available. This means we can efficiently identify both the spatial and temporal region(s) within the sequence that best fit a learned activity model. See Figure 1.

The proposed approach has several important properties. First, for the specific case where our space-time nodes are individual video frames, the detection solution is equivalent to that of exhaustive sliding window search, yet costs orders of magnitude less search time due to the branch-and-cut solver. Second, we show how to create more general forms of the graph that permit "non-cubic" detection regions, and even allow hops across irrelevant frames in time that otherwise might mislead the classifier (e.g., due to a temporary occluding object). This effectively widens the scope of candidate video regions considered beyond that allowed by any prior methods; the upshot is improved accuracy. Finally, the method accommodates a fairly wide family of features and classifiers, making it flexible as a general activity detection tool. To illustrate this flexibility, as a second contribution, we introduce a novel high-level descriptor amenable to subgraph search that reflects human poses and objects as well as their relative temporal ordering.

We validate the algorithm on three challenging datasets. The results demonstrate its clear speed and accuracy advantages over both standard sliding window search as well as a state-of-the-art branch-and-bound solution.

2. Related Work

We focus our literature review on methods handling action *detection* in video. There is also a large body of work in activity *recognition* (from either a sequence or a static frame) where one must categorize a clip/frame that is already trimmed to the action of interest. Representations developed in that work are complementary and could enhance results attainable with our detection scheme.

One class of methods tackles detection by explicitly tracking people, their body parts, and nearby objects (e.g., [17, 19, 11]). Tracking "movers" is particularly relevant for surveillance data where one can assume a static camera. However, relying on tracks can be limiting; it makes the detector sensitive to tracking errors, which are expected in video with large variations in backgrounds or rapidly changing viewpoints (e.g., movies or YouTube video). Furthermore, while good for activities that are truly person-based—like handwaving or jumping jacks—a representation restricted to person-tracks will suffer when defining elements of the action are external to people in the scene (e.g., the computer screen a person is looking at, or the chair he may sit in).

Conscious of the difficulty of relying on tracks, another class of methods has emerged that instead treats activity classes as learned space-time appearance and motion patterns. The bag of space-time interest point features model is a good example [14, 22]. In this case, at detection time the classifier is applied to features falling within candidate subvolumes within the sequence. Typically the search is done with a sliding window over the entire sequence [9, 5, 21], or in combination with person tracks [11].

Given the enormous expense of such an exhaustive search, some recent work explores branch-and-bound solutions to efficiently identify the subvolume that maximizes an additive classifier's output [29, 28, 3]. This approach offers fast detection and can localize activities in both space and time, whereas sliding windows localize only in the temporal dimension. However, in contrast to our approach, existing branch-and-bound methods are restricted to searching over *cubic* subvolumes in the video; that limits detections to cases where the subject of the activity does not change its spatial position much over time. Our results demonstrate the value of the more general detection shapes allowed by our method.

An alternative way to avoid exhaustive search is through voting algorithms. Recent work explores ways to combine person-centric tracks or pre-classified sequences with a Hough voting stage to refine the localization [27, 16], or to use voting to generate candidate frames for merging [26]. Like any voting method, such approaches risk being sensitive to noisy background descriptors that also cast votes, and in particular will have ambiguity for actions with periodicity. Furthermore, in contrast to our algorithm, they cannot guarantee to return the maximum scoring space-time region for a classifier.

Rather than pose a detection task, the multi-class recognition approach of [7] uses dynamic programming to select the temporal boundaries per action. Like our technique it jointly considers recognition and segmentation. However, unlike our method, it localizes only in the temporal dimension, assumes a multi-class objective where all parts of the sequence will belong to some pre-trained category (thus requiring one to learn a "background" activity class), and cannot detect multiple activities occurring at the same time.

The branch-and-cut algorithm we use to optimize the subgraph has also been explored for object segmentation in static images [24]. In contrast, our approach addresses activity detection, and we explore novel graph structures relevant for video data.

3. Approach

Our approach first trains a detector using a binary classifier and training examples where the action's temporal extent is known. Then, given test sequences for which we have no knowledge of the start and end of the activity, it returns the subsequence (and optionally, the spatial regions of interest) that maximizes the classifier score. This works by creating a space-time graph over the entire test sequence where each node's weight indicates its features' contribu-

tion to the classifier's score. Then, the best scoring subsequence is equivalent to the maximum weight connected subgraph. This subgraph can be efficiently computed using an existing branch-and-cut algorithm, allowing optimal solutions without exhaustive search.

We first define the classifiers accommodated by our method (Sec. 3.1), and the features we use (Sec. 3.2). Then we describe how the graphs are constructed (Sec. 3.3). Finally, we briefly explain the maximum subgraph problem and branch-and-cut search (Sec. 3.4).

3.1. Detector Training and Objective

We are given labeled training instances of the activity of interest, and train a binary classifier $f : S \rightarrow \mathbb{R}$ to distinguish positive instances from all other action categories. This classifier can score any subvolume S of a novel video according to how well it agrees with the learned activity. To perform activity detection, the goal is to determine the subvolume in a new sequence Q that maximizes the score

$$S^* = \arg\max_{S \in Q} \; f(S). \tag{1}$$

If we were to restrict the subvolume in the spatial dimensions to encompass the entire frame, then S^* would correspond to the output of an exhaustive sliding window detector. More generally, the optimal subvolume S^* is the set of contiguous voxels of arbitrary shape in Q that returns the highest classifier score.

Our approach requires the classifier to satisfy two properties. First, it must be able to score an arbitrarily shaped set of voxels. Second, it must be defined such that features computed within local space-time regions of the video can be combined *additively* to obtain the classifier response for a larger region. The latter is necessary so that we can decompose the classifier response across the nodes of the space-time graph, and thereby associate a single weight with each node. Suitable additive classifiers include linear support vector machines (SVM), boosted classifiers, or Naive Bayes classifiers computed with localized space-time features, as well as certain non-linear SVMs [23].

Our results use a linear SVM with histograms (bags) of quantized space-time descriptors. The bag-of-features (BoF) representation has been explored in a number of recent activity recognition methods (e.g., [14, 10, 18]), and, despite its simplicity, offers very competitive results. We consider BoF's computed over two forms of local descriptors. The first consists of low-level histograms of oriented gradients and flow computed at space-time interest points; the second consists of a novel high-level descriptor that encodes the relative layout of detected humans, objects, and poses. Both descriptors are detailed below in Sec. 3.2.

In either case, we compute a vocabulary of K visual words by quantizing a corpus of features from the training images. A video subvolume with N local features is initially described by the set $S = \{(\boldsymbol{x}_i, \boldsymbol{v}_i)\}_{i=1}^N$, where each $\boldsymbol{x}_i = (x_i, y_i, t_i)$ refers to the 3D feature position in space and time, and $\boldsymbol{v}_i$ is the associated local descriptor. Then the subvolume is converted to a K-dimensional BoF histogram $h(S)$ by mapping each $\boldsymbol{v}_i$ to its respective visual word c_i, and tallying the word counts over all N features.

We use the training instances to learn a linear SVM, which means the resulting scoring function has the form: $f(S) = \beta + \sum_i \alpha_i \langle h(S), h(S_i) \rangle$, where i indexes the training examples, and α, β denote the learned weights and bias. This can be rewritten as a sum over the contributions of each feature. Let $h^j(S)$ denote the j-th bin count for histogram $h(S)$. The j-th word is associated with a weight $w^j = \sum_i \alpha_i h^j(S_i)$, for $j = 1, \ldots, K$. Thus the classifier response for a subvolume S is:

$$f(S) \;\; = \;\; \beta + \sum_{j=1}^{K} w^j h^j(S) \;\; = \;\; \beta + \sum_{i=1}^{N} w^{c_i}, \tag{2}$$

where again c_i is the index of the visual word that feature $\boldsymbol{v}_i$ maps to, $c_i \in [1, K]$. By writing the score of a subvolume as the sum of its N features' "word weights", we now have a way to associate each local descriptor occurrence with a single weight—its contribution to the total classifier score.

This same property of linear SVMs is used in [12] to enable efficient subwindow search for object detection, whereas we exploit it to score non-cubic subvolumes in video for action detection. We stress that our method is not limited to linear SVMs; alternative additive classifiers with the properties described above are also permitted.

3.2. Localized Space-Time Features

We consider two forms of localized descriptors for the $\boldsymbol{v}_i$ vectors above: a conventional low-level gradient-based feature, and a novel high-level feature.

Low-level descriptors For low-level features, we use histograms of oriented gradients (HoG) and histograms of optical flow (HoF) computed in local space-time cubes [14, 10]. The local cubes are centered at either 3D Harris interest points [13] or densely sampled. These descriptors capture the appearance and motion in the video, and their locality lends robustness to occlusions. See [13, 14, 10] for details.

High-level descriptors We introduce a novel descriptor for an alternative high-level representation. While low-level gradient features are effective for activities defined by gestures and movement (e.g., running vs. diving), many interesting actions are likely better defined in terms of the semantic *interactions* between people and objects. For example, "answering phone" should be compactly describable in terms of a person, a reach, a grasp of the receiver, etc.

To this end, we compose a descriptor that encodes the objects and poses occurring in a space-time neighborhood.

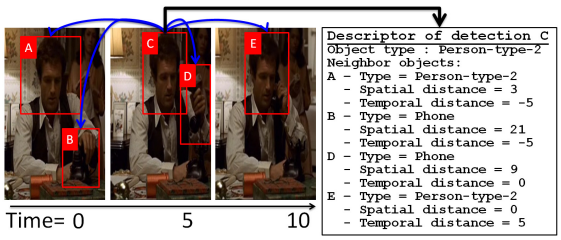


Figure 2. Left: detected objects surrounding the person detection C. Right: info captured in the high-level descriptor, relative to C.

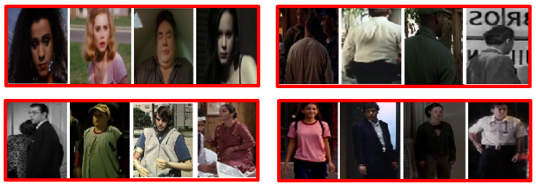
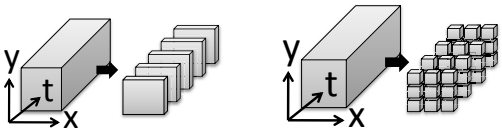


Figure 3. Instances from four discovered person types.

First, we run a bank of object detectors [6] and a bank of mid-level "poselet" detectors [2] on all frames. To capture human *pose*, we categorize each detected person into one of $P = 15$ "person types". These types are discovered from person detection windows in the training data: for each person window we create a histogram of the poselet activations that overlap it, and then quantize the space of all such histograms with k-means to provide P discrete types. Each reflects a coarse pose—for example, a seated person may cause upper body poselets to fire, whereas a hugging person would trigger poselets from the back. Figure 3 shows four example person types discovered on the Hollywood data.

Given the sparse set of bounding box detections in a test sequence, we form one neighborhood descriptor per box. This descriptor reflects (1) the type of detector (e.g., person type #3, car) that fired at that position, (2) the distribution of object/person types that also fired within a 50-frame temporal window of it, and (3) their relative space-time distances. See Figure 2. To quantize this complex space into discriminative high-level "words", we design a twist on the standard random forest technique. When training the random forest, we choose spatial distance thresholds, temporal distance thresholds, and object types to parameterize semantic questions that split the raw descriptor inputs so as to reduce action label entropy. Each training and testing descriptor is then assigned a visual word according to the indices of the leaf nodes it reaches when traversing each tree in the forest. Essentially, this reduces each rich neighborhood of space-time object relationships to a single quantized descriptor, i.e., a single index c_i in Eqn. 2.

In contrast to the low-level features, this descriptor encodes space-time ordering, demonstrating that our max-subgraph scheme is clearly not limited to pure bag-of-words representations. Furthermore, it yields sparser video graphs, since the number of detected objects is typically much fewer than the number of space-time interest points.



(a) Temporal only (T) (b) Spatio-temporal (ST)

Figure 4. The two node structures we consider.

3.3. Definition of the Space-Time Graph

So far we have defined the training procedure and features we use. Now we describe how we construct a space-time graph $G = (V, E)$ for a novel test video, where V is a set of vertices (nodes) and E is a set of edges. Recall that a test video is "untrimmed", meaning that we have no prior knowledge about where an action(s) starts or ends in either the spatial or temporal dimensions. Our detector will exploit the graph to efficiently identify the most likely occurrences of a given activity. We present two variants each for the node and link structures, as follows.

Node structure Each node in the graph is a set of contiguous voxels within the video. In principle, the smallest possible node would be a pixel, and the largest possible node would be the full test sequence. What, then, should be the scope of an individual node? The factors to consider are (1) the granularity of detection that is desired (i.e., whether the detector should predict only when the action starts and ends, or whether it should also estimate the spatial localization), and (2) the allowable computational cost. Note that nodes larger than individual voxels or frames are favorable not only for computational efficiency, but also to aggregate neighborhood statistics to give better support when the detector considers that region for inclusion.

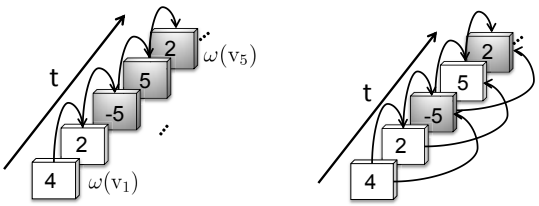
With this in mind, we consider two possible node structures. The first breaks the video into frame-level slabs, such that each node is a sequence of F consecutive frames. The second breaks the video into a grid of $H \times W \times F$ space-time cubes. In all our results, we set $F = 5$ or 10, and let H and W be $\frac{1}{3}$ of the frame dimensions.[1] See Figure 4. At detection time, the two forms yield a *temporal subgraph* (**T-Subgraph**) and *spatio-temporal subgraph* (**ST-Subgraph**), respectively. Note that a T-Subgraph will be equivalent to a sliding window search result with a frame step size of F. In contrast, a ST-Subgraph will allow irregular, non-cubic detection results. See the first and last images in Figure 6.

After building a graph with either node structure for a test video, we compute the weight for each node v:

$$\omega(\mathrm{v}) = \sum_{\boldsymbol{x}_j \in \mathrm{v}} w^{c_j}, \tag{3}$$

where $\boldsymbol{x}_j$ is the 3D coordinate of the j-th local descriptor falling within node $\mathrm{v} \in V$, and c_j is its quantized feature

[1] Rather than space-time cubes, one could use space-time *segments* from a bottom-up grouping algorithm, though at greater expense.



(a) Neighbors only (b) Neighbors + "Jump"

Figure 5. The two linking strategies we consider. Numbers shown on nodes indicate weights; white nodes indicate those that would be selected under either linking strategy (see text).

index. Note that $\boldsymbol{x}_j$ is the space-time interest point position for our low-level features, while $\boldsymbol{x}_j$ is the center of the originating object detection window for our high-level features. Intuitively, nodes with high positive weights indicate that the activity covers that space-time region, while nodes with negative weights indicate the absence of the activity.

Linking strategies The connectivity between nodes also affects both the shape of candidate subvolumes and the cost of subgraph search. We explore two strategies. In the first, we link only those neighboring nodes that are temporally (and spatially, for the ST node structure) adjacent (see Figure 5 (a)). In the second, we additionally link nodes that are within the first two temporal neighbors (see Figure 5 (b)); we call this variant **T-Jump-Subgraph**. Since at test time we will seek a maximum scoring *connected* subgraph, the former requires detection subvolumes to be strictly contiguous in time (and thus equates to the options available to a sliding window), while the latter allows subvolumes that "jump" over an adjacent neighbor in time.

By allowing jumps, we can ignore misleading features that may interrupt an otherwise good instance of an action. For example, Figure 5 depicts some temporal nodes and their associated weights $\omega(\mathrm{v}_i)$'s, under either connectivity scheme. The max subgraph *without* jumps in (a) is the first two nodes only; in contrast, for the same node weights, the max subgraph *with* jumps in (b) extends to include the fourth node, yielding a higher weight subgraph (4+2+5 vs. 4+2). This can be useful when the skipped node(s) contain noisy features, such as an object temporarily blocking the person performing the activity. Like the space-time nodes presented above, the use of temporal jumps further expands the space of candidate subvolumes our method can search, at some additional computational cost. One might consider more complicated linking strategies which could provide even greater flexibility for detection, but at a higher computational cost; we leave this as future work.

3.4. Searching for the Maximum Weight Subgraph

Having defined the graph constructed on an untrimmed test sequence, we are ready to describe the detection procedure to maximize $f(S)$ in Eqn. 1. Our detection objective is an instance of the maximum-weight connected subgraph problem (MWCS): *Given a connected undirected, vertex-weighted graph $G = (V, E)$ with weights $\omega : V \rightarrow \mathbb{R}$, find a connected subgraph $T = (V_T \subseteq V, E_T \subseteq E)$ of G, that maximizes the score $W(T) = \sum_{\mathrm{v} \in V_T} \omega(\mathrm{v})$.* The best-scoring subgraph is the subvolume in the video most likely to encompass the activity of interest.

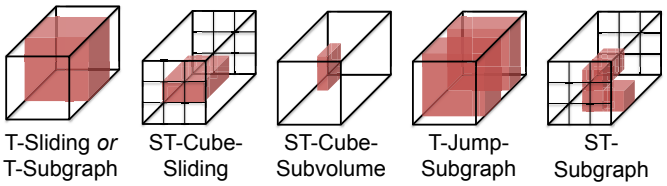


Figure 6. Sketch of candidate subvolume types considered by different methods, ordered approximately from least to most flexible.

Dataset	Features	Num test videos	Ave length (#frames)	Ave length of action	Node/step size (#frames)
UCF-Concat	Dense+HoG3D	12	589	13%	5
Hollywood uncropped	STIP+HoG/HoF or high-level	211	474	62%	10
MSR Action	STIP+HoG/HoF	16	756	10%	10

Table 1. Properties of the three datasets.

With both positive and negative weights, the problem is NP-complete [8]; an exhaustive search would enumerate and score all possible subsets of connected nodes. However, MWCS can be transformed into an instance of the prize-collecting Steiner tree problem [4], which is solvable by transforming the graph into a directed graph and formulating an integer linear programming (ILP) problem with binary variables for every vertex and edge. Then by relaxing the integrality requirement, the problem can be solved with linear programming using a branch-and-cut algorithm (see [15]). This method gives optimal solutions and is very efficient in practice for the space-time graphs in our setting.

The max-subgraph approach specifically seeks the subvolume within the entire test sequence that maximizes the activity classifier's output. To return *multiple* top-scoring detections, we follow the protocol of [29]. Essentially, this entails iteratively running detection for the best-scoring subvolume, then removing the nodes participating in the detection subvolume (equivalently, zeroing out their weights). Note that with the space-time node structure, this still allows for detecting instances that overlap in time.

4. Experimental Results

Datasets We validate on three datasets: a concatenated form of UCF Sports [20], the uncropped Hollywood videos [14], and MSR Actions [29]. See Table 1 for an overview of the dataset properties. On average, the action of interest occupies only 18% of the total test sequence, making detection (as opposed to classification) necessary. We use the authors' code for HoG3D/HoG/HoF [14, 10], with default parameter settings. For all datasets, we train a binary SVM to build a detector for each action.

Baselines We compare to three baselines: (1) **T-Sliding**: a standard temporal sliding window. This is the status quo method in the literature, e.g., [9, 5, 21]. Its results are equivalent to our T-Subgraph variant (using temporal linking structure), but computed with exhaustive search. (2) **ST-Cube-Sliding**: a variant of sliding window that searches all cuboid subvolumes having any *rectangular* combination of the spatial-nodes used by our method. Its search scope is similar to our ST-Subgraph, *except* that it lacks all possible spatial links, meaning the detected subvolume cannot shift spatial location over time. (3) **ST-Cube-Subvolume**: the state-of-the-art branch-and-bound method of [29]. It considers *all possible* cube-shaped subvolumes, and returns the one maximizing the sum of feature weights inside. Its scope is more flexible than ST-Cube-Sliding. Its objective is identical to ours, *except* that it is restricted to searching cube-shaped volumes that cannot shift spatial location over time. We use the authors' code.[2]

We consider three variants of our approach: T-Subgraph, T-Jump-Subgraph, and ST-Subgraph, as defined in Sec. 3. To recap, T-Subgraph *provides equivalent accuracy to T-Sliding, but is faster*.[3] T-Jump-Subgraph and ST-Subgraph provide more flexibility for detection, *allowing temporal discontinuities and spatial changes not permitted by any of the above methods*. Figure 6 depicts the scope of the regions searched by each method.

Evaluation metrics We use the *mean overlap accuracy* for evalution, following [27, 11, 29]. Whether performing temporal or full spatio-temporal detection, this metric computes the intersection of the predicted detection region with the ground truth, divided by the union. We use detection time (on our 3.47GHz Intel Xeon CPUs) to evaluate computational cost.

We stress that our approach is a new strategy for *detection*; results in the literature focus largely on *classification*, and so are not directly comparable. The sliding window and subvolume baselines are state-of-the-art methods for detection, so our comparisons with identical features and classifiers give clear insight into our method's performance.

4.1. Temporal Detection on UCF Sports

Since the UCF clips are already cropped to the action of interest, we modify it to make it suitable for detection. We form 12 test sequences by concatenating 8 different clips each from different verbs. All test videos are totally distinct. We train the SVM on a disjoint set of cropped instances. We perform temporal detection only, since the activities occupy the entire frame.

Verbs	T-Sliding	ST-Cube-Subvol [29]	Our-T-Subgraph	Our-T-Jump-Subgraph
Diving	0.8106	0.7561	0.8106	**0.9091**
Lifting	0.7899	0.8058	0.7899	**0.8096**
Riding	**0.5349**	0.5075	**0.5349**	0.3888
Running	0.4602	0.3269	0.4602	**0.4705**
Skateboard	0.1407	0.1057	0.1407	**0.1803**
Swing-Bench	0.5520	**0.6259**	0.5520	0.4582
Swing-Side	0.6728	0.3478	0.6728	**0.7212**
Walking	0.4085	0.3462	0.4085	**0.4657**

Table 2. Mean overlap accuracy for the UCF Sports data.

Detection time (ms)	T-Sliding	ST-Cube-Subvol [29]	Our-T-Subgraph	Our-T-Jump-Subgraph
Mean	1.25×10^5	7.87×10^4	1.02×10^2	6.51×10^2
Stdev	7.52×10^3	3.17×10^4	5.35×10^1	3.17×10^2

Table 3. Search time for the UCF Sports data.

Verbs	T-Sliding	ST-Cube-Subvol [29]	Our-T-Subgraph	Our-T-Jump-Subgraph
AnswerPhone	0.3968	0.2905	0.3968	**0.3994**
GetOutCar	0.2276	0.2267	0.2276	**0.2921**
HandShake	0.3071	0.3390	0.3071	**0.3663**
HugPerson	0.3869	**0.4486**	0.3869	0.4150
Kiss	0.3822	0.4230	0.3831	**0.4412**
SitDown	**0.3612**	0.2861	**0.3612**	0.3550
SitUp	0.2592	0.2053	0.2592	**0.3255**
StandUp	0.3475	0.3013	0.3475	**0.3775**

Table 4. Mean overlap accuracy on uncropped Hollywood data.

Detection Time (ms)	T-Sliding	ST-Cube-Subvol [29]	Our-T-Subgraph	Our-T-Jump-Subgraph
Mean	3.71×10^3	1.70×10^5	6.63×10	5.69×10^2
Stdev	1.03×10^4	5.79×10^5	7.51×10	1.77×10^3

Table 5. Search time on uncropped Hollywood data.

Table 2 shows the accuracy results, and Table 3 shows the search times. For almost all verbs, our subgraph approaches outperform the baselines. Further, our T-Jump variant gives top accuracy in most cases, showing the advantage of ignoring noisy features (in this data, often found near the onset or ending of the verb). ST-Cube-Subvolume is often weaker than sliding window; we find it often fires on a small volume with highly weighted features when the activity changes in spatial location over time. However, it is best on "Swing-Bench", likely because the backgrounds are fairly static, minimizing misleading features. Both our subgraph methods are orders of magnitude faster than the baselines. (Note that the ST-Cube-Subvolume's higher cost is reasonable since here it is searching a wider space.)

4.2. Temporal Detection on Hollywood

We next test the Hollywood data. The dataset creators provide both the noisy "uncropped" versions of the sequences, which are only roughly aligned to the action and contain about 40% extraneous frames, as well as the "clean" or cropped versions of the sequences, which have been trimmed temporally to the action of interest. Existing work uses this data for classification, and so trains *and* tests with

[2] We found its behavior sensitive to its *penalty value* parameter, which is a negative prior on zero-valued pixels [29]. The default setting was weak for our data, so for fairest comparisons, we tuned for best results on UCF and then fixed it for the rest.

[3] For the special case of temporal search, one can obtain equivalent solutions using 1-D branch-and-bound search to detect the max subvector along the temporal axis [1]. In practice we find this method's run-time to be similar or slightly faster than T-Subgraph. Note, however, that it is *not* applicable for any other search scope handled by our approach.

Test sequence composition	Accuracy
Raw uncropped clips	24.83%
Output from T-Subgraph	29.66%
Manual ground truth	29.97%

Table 6. Recognition accuracy on Hollywood as test input varies.

the cropped versions. To perform temporal detection, we instead train with the cropped clips, and test with the uncropped clips.

Table 4 shows the accuracy results, and Table 5 shows the search times. Our T-Jump-Subgraph achieves the best accuracy for 6 of the 8 verbs, with even more pronounced gains than on UCF. This again shows the value of skipping brief negatively weighted portions; e.g., "AnswerPhone" can transpire across several shot boundaries, which tends to mislead the baselines. Our method is again significantly faster than the baselines. Our T-Jump-Subgraph is slower than our T-Subgraph search, given the higher graph complexity (which also makes it more accurate).

One might wonder whether a naive detector that simply classifies the entire uncropped clip could do as well. To check, we compare *recognition* results when we vary the composition of the test sequence to be either (a) the uncropped clip, (b) the output of our detector, or (c) the ground truth cropped clip. Table 6 shows the result. We see indeed that detection is necessary; using our output is much better than the raw untrimmed clips, and only slightly lower than using the manually provided ground truth.

4.3. Space-Time Detection on MSR Actions

The MSR dataset differs from the above in that test sequences may contain multiple simultaneous instances of different actions, and the actors change their position over time. This makes a good testbed to evaluate our ST-Subgraph with the node structure in Figure 4(b), and we link neighboring nodes both in space and time. Following [29], we train detectors using KTH data [22].

Table 7 shows the temporal detection accuracy. Even under the temporal criterion, our ST-Subgraph is most accurate, since it can isolate those nodes that participate in the action. However, that accuracy does come at the cost of longer search time (see Table 8). Figure 7 illustrates how our space-time node structure succeeds when the location of activity changes over time, whereas ST-Cube-Subvolume may be trapped in cube-shaped maxima.

Figure 8 further evaluates detection of multiple instances per clip, using the procedure described in Sec. 3.4. A detection is regarded as correct if the temporal overlap accuracy is greater than 1/8, following [29, 3]. Our method yields the highest average precision.

Table 9 shows the full space-time localization accuracy, evaluated under the provided ground truth (first numbers) and our own refined annotations (second numbers). The original ground truth labels only the hand regions (see Figure 7), whereas our ground truth labels the whole person performing the action; it is unclear which is more appropriate so we include both. Results are mixed between the methods, with a slight edge for our ST-Subgraph. Also, only the non-rectangular shape detection from our ST-Subgraph reflects the large spatial motions in actions.

Verbs	T-Sliding	ST-Cube-Sliding	ST-Cube-Subvol [29]	Our-T-Subgraph	Our-ST-Subgraph
Boxing	0.0541	0.0717	0.0794	0.0541	**0.0989**
Clapping	0.0982	0.0982	0.0602	0.0982	**0.1754**
Waving	0.2342	0.2204	0.2669	0.2342	**0.2926**

Table 7. Mean temporal overlap accuracy on the MSR dataset.

Detection Time (ms)	T-Sliding	ST-Cube-Sliding	ST-Cube-Subvol [29]	Our-T-Subgraph	Our-ST-Subgraph
Mean	4.2×10^3	5.5×10^4	3.0×10^5	2.8×10^2	3.1×10^6
Stdev	3.3×10^3	4.2×10^4	1.6×10^5	2.3×10^2	4.6×10^6

Table 8. Search time on the MSR dataset.

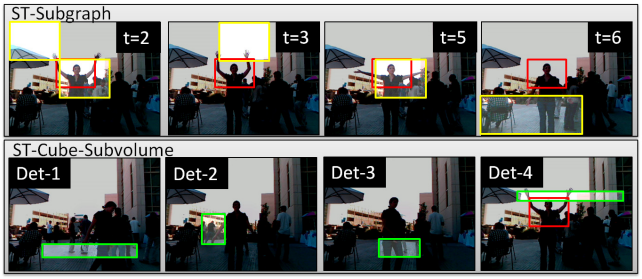
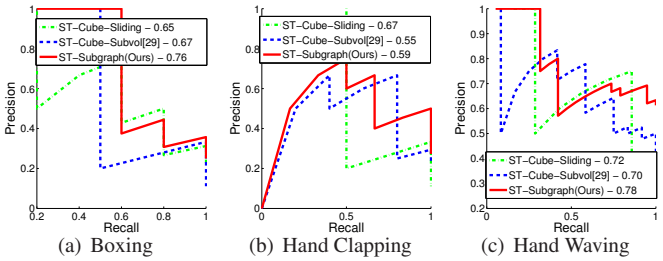


Figure 7. Example of ST-Subgraph's top output (top) and the top 4 detections from ST-Cube-Subvolume [29] (bottom). Red rectangles denote ground truth. Brighter areas denote detections.



(a) Boxing (b) Hand Clapping (c) Hand Waving

Figure 8. Detecting multiple instances per sequence on MSR. Numbers in legends indicate mAP.

Verbs	ST-Cube-Sliding	ST-Cube-Subvol [29]	Our-ST-Subgraph
Boxing	**0.0474/0.0478**	0.0456/0.0193	0.0309/0.0417
Hand Clapping	0.0261/0.0373	0.0173/0.0071	**0.0295/0.0630**
Hand Waving	0.0912/0.0851	**0.1013**/0.0581	0.0699/**0.1121**

Table 9. Mean space-time overlap accuracy on the MSR dataset; first value uses hand-based ground truth, second value uses person-based ground truth. (T-Sliding/T-Subgraph are omitted since they don't do spatial localization.)

4.4. Summary of Trade-Offs in Results

There are three dimensions of trade-offs between all methods tested: search time, search scope, and detection accuracy. Figure 9 summarizes all trade-offs for all datasets, using more complex polygon symbols for the methods that search a wider scope of subvolume shapes (e.g., least complex for T-Sliding/T-Subgraph, most complex for ST-

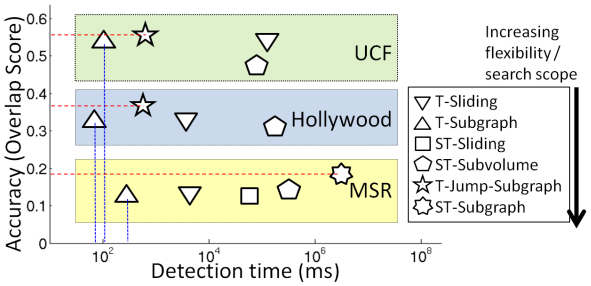


Figure 9. Overview of all methods on the three datasets.

Verbs	T-Subgraph (HoG/HoF)	T-Subgraph (high-level)
AnswerPhone	**0.3968**	0.1741
GetOutCar	**0.2276**	0.1447
HandShake	0.3071	**0.4194**
HugPerson	0.3869	**0.5292**
Kiss	0.3822	**0.4906**
SitDown	0.3612	**0.3753**
SitUp	0.2592	**0.3843**
StandUp	**0.3475**	0.2636

Table 10. Mean overlap accuracy on Hollywood for low-level features vs. the proposed high-level descriptors.

Subgraph). Importantly, we see that increased search scope generally boosts accuracy. In addition, the flexibility of the graph structure in our subgraph algorithm allows it to perform best per dataset in terms of *either* speed (see vertical blue dotted lines) or accuracy (see horizontal red dotted lines). We provide our source code and data in our project page: http://vision.cs.utexas.edu/projects/maxsubgraph.

4.5. Subgraph Search with High-level Features

Finally, we test our novel high-level descriptor on Hollywood, since its actions contain human-object interactions. We apply 6 object detectors—bus, car, chair, dining table, sofa, and phone—to every 5th frame, and use random forests with 10 trees. Table 10 shows the results, compared to our method using low-level features. For 5 of the 8 verbs, the proposed descriptor improves accuracy. It is best for verbs based on the interaction between two people (e.g., kiss) or involving an obvious change in pose (e.g., sit up), showing the strength of the proposed person types to capture pose. For other verbs with varied objects (answer phone, get out of car), it hurts accuracy due to object detector failures in this dataset.

5. Conclusions

We presented a novel branch-and-cut subgraph framework for activity detection that efficiently searches a wide space of temporal or space-time subvolumes. Compared to traditional sliding window search, it significantly reduces computation time. Compared to existing branch-and-bound methods, its flexible node structure offers more robust detection in noisy backgrounds. Our novel high-level descriptor also shows promise for complex activities, and makes it possible to preserve the spatio-temporal relationships between humans and objects in the video, while still exploiting the fast subgraph search.

Acknowledgements This research is supported in part by DARPA CSSG N11AP20004 and DARPA Mind's Eye.

References

[1] J. Bentley. Programming pearls: algorithm design techniques. *Commun. ACM*, 27(9):865–873, Sept. 1984.
[2] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, 2009.
[3] L. Cao, Z. Liu, and T. S. Huang. Cross-dataset action recognition. In *CVPR*, 2010.
[4] M. T. Dittrich, G. W. Klau, A. Rosenwald, T. Dandekar, and T. Mller. Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *Bioinformatics*, 2008.
[5] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009.
[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32:1627–1645, 2010.
[7] M. Hoai, Z.-Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011.
[8] T. Ideker, O. Ozier, B. Schwikowski, and A. F. Siegel. Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics*, 2002.
[9] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *ICCV*, 2005.
[10] A. Kläser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008.
[11] A. Kläser, M. Marszałek, C. Schmid, and A. Zisserman. Human focused action localization in video. In *International Workshop on Sign, Gesture, Activity*, 2010.
[12] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, 2008.
[13] I. Laptev. On space-time interest points. *IJCV*, 64(2):107–123, 2005.
[14] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.
[15] I. Ljubic, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Math. Prog.*, 2006.
[16] K. Mikolajczyk and H. Uemura. Action recognition with motion-appearance vocabulary forest. In *CVPR*, 2008.
[17] D. Moore, I. Essa, and M. Hayes. Exploiting human actions and object context for recognition tasks. In *CVPR*, 1999.
[18] J. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *IJCV*, 2008.
[19] D. Ramanan and D. Forsyth. Automatic annotation of everyday movements. In *NIPS*, 2003.
[20] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008.
[21] S. Satkin and M. Hebert. Modeling the temporal extent of actions. In *ECCV*, 2010.
[22] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICPR*, 2004.
[23] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.
[24] S. Vijayanarasimhan and K. Grauman. Efficient region search for object detection. In *CVPR*, 2011.
[25] H. Wang, M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.
[26] G. Willems, J. Becker, T. Tuytelaars, and L. V. Gool. Exemplar-based action recognition in video. In *BMVC*, 2009.
[27] A. Yao, J. Gall, and L. van Gool. A Hough transform-based voting framework for action recognition. In *CVPR*, 2010.
[28] G. Yu, J. Yuan, and Z. Liu. Unsupervised random forest indexing for fast action search. In *CVPR*, 2011.
[29] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *CVPR*, 2009.