Learning the Abstract Motion Semantics of Verbs from Captioned Videos

Stefan Mathe Afsaneh Fazly Sven Dickinson Suzanne Stevenson
Department of Computer Science, University of Toronto
{mstefan, afsaneh, sven, suzanne}@cs.toronto.edu

Abstract

We propose an algorithm for learning the semantics of a (motion) verb from videos depicting the action expressed by the verb, paired with sentences describing the action participants and their roles. Acknowledging that commonalities among example videos may not exist at the level of the input features, our approximation algorithm efficiently searches the space of more abstract features for a common solution. We test our algorithm by using it to learn the semantics of a sample set of verbs; results demonstrate the usefulness of the proposed framework, while identifying directions for further improvement.

1. Introduction

Extracting meaningful information from perceptual features generally involves a process of abstraction. While some interpretation problems—e.g., detecting and identifying a specific exemplar within a 2D image—may be reducible to a process of feature selection, interpreting a visual scene in terms of general categories of objects or actions necessitates *constructing* the appropriate features to represent the distinguishing properties. Low-level visual input features, such as the contours extracted from an exemplar (in an image) or the detailed trajectory of an exemplar (in a video), must be segmented, grouped, and abstracted in some manner to support the interpretation of a particular object or action category in the visual scene. Determining the common features of a category across a set of images or videos requires searching this space of more abstract features to find the appropriate level of abstraction to capture the depicted object/action. Unfortunately, the input features may give rise to an exponential number of feature groupings/abstractions, making the problem intractable.

This situation is unavoidable when learning the semantics of a verb from videos depicting a variety of objects moving and interacting in a manner expressed by the verb. If the object motions and interactions were highly regular across the training set for a verb, learning the verb semantics might be reducible to an input feature selection problem. Consider, for example, a set of videos of different people throwing an object, each video captioned with the sentence "[Name-X] throws [Object-Y] on [Place-Z]." Furthermore, in each video, the person throws the object on a place with an underhand movement. Provided that our input feature set included horizontal and vertical motion primitives and relative motion between objects, then the videos could be time-segmented and "aligned" to yield a categorical model consisting of a number of states. In this case, there are four states, including a joint up-and-forward motion of a hand and an object (in contact), followed by an up-and-forward motion of the object (moving away from the hand), followed by a down-and-forward motion of the object, followed by zero object motion while the object is in close proximity to a place. Such a model correctly captures the regularities of the input training data.

However, the motion and/or interaction of objects described by a verb generally precludes a one-to-one correspondence at the input feature level. For example, let's add a few videos to our training set. In one video, the person is running while throwing the object; in another video, the person throws the object overhand, while still in another, the person throws the object while moving his/her hand sideways. If we insist on one-to-one input feature correspondence across the training set, there is no solution, i.e., there does not exist a segmentation (in time) of each input video such that corresponding intervals agree in terms of their input features. The correct solution—i.e., the person's hand and the object move together while in contact (regardless of how they move), followed by the object moving away from the hand (regardless of how the object moves, and whether the hand moves or remains still), followed by the convergence of the object and the place—does not exist at the level of the input features. We must address the combinatorial problem of grouping and abstracting input features to find commonality at some higher level of abstraction.

The problem lies not with defining the abstract motion/interaction features, for we could easily define a "scale space" of motion/interaction primitives, ranging from highly detailed (at the lowest levels) to highly abstract (at the highest levels). One could envision computing such a

scale space for each video and then searching for the lowest levels at which commonality occurs. Unfortunately, pre-computing such a scale space is problematic, for a primitive is computed over an interval in time, and its correct value will be attained when the time interval spans one of the states in the verb model (e.g., the duration over which the hand and the object move while in contact). Not only do we not know a priori the state boundaries in each video, we don't even know the appropriate number of states needed to model the verb semantics. Thus pre-computing such a scale space for a given video amounts to enumerating all possible time segmentations of the video, and computing each feature over each interval—a process that is clearly intractable.

To address this issue, we adopt an approach inspired by [8] who formulate the learning of a categorical shape model as a search for the *lowest common abstraction* among a set of increasingly-abstract representations of the training images. We begin with a set of candidate interval (state) boundaries in each of a set of input videos. Considering the space of possible alignments of these boundaries across the videos, we generate abstract features over the possible intervals and search for an alignment that maximizes the shared features across corresponding states. Moreover, we seek an alignment that maximizes the feature overlap among the input videos, yielding a lowest common abstraction. Since the space of possible alignments is intractable, we propose a polynomial-time approximation that converges on the lowest common motion/interaction abstraction over a set of training videos. Returning to our two example scenarios of throwing an object, we would expect to generate the much more detailed model in the first case, and the more abstract model in the second case.

2. Previous work

The earliest attempts at human action recognition have looked at low-level appearance-based features. These could be as simple as the background-subtracted images which were then time-warped to the input data, as in [3], or vector-quantized versions of the background-subtracted silhouettes, which were used to train Hidden Markov Models (HMMs), as in [11]. Others use motion histories of the 2D silhouettes ([1],[12]) or the 3D volumes ([10]) as global models for action. All these approaches assume a one-to-one correspondence between the low-level image and model features. This is reflected in the low within-class variability of the action classes they target, like human gestures, tennis strokes, and ballet movements.

Using features similar to ours, Siskind ([9]) trains HMMs over relative positions and velocities of body parts and objects to model simple manipulative human actions. As with previous HMM approaches, the number of states in the model has to be known in advance. The approach does not abstract away the actual velocities and accelerations in each subevent. The level of abstraction is lifted in [4] by using force-dynamic features describing interactions between objects in the world, such as attachment, contact, and support. These features are grouped into states, then into sequences of states, and finally into parallel timelines. While handling the grouping problem over features, these works fail to acknowledge that abstraction needs to be carried out over time intervals as well. This makes the systems sensitive to noise and variation in the manner of execution of a particular action.

Other methods to deal with the problem of abstraction have been proposed. Stochastic parsing techniques are used by [7] to abstract away structure and order of subevents. Possibly overlapping subevents are detected using HMMs. A hand-built probabilistic grammar is used to model human gestures with subevents occurring in varying orders. While looking for the most probable parse, the parser minimizes the total overlap between subevents. Unlike [4], the system handles noise by insertions, deletions, and misdetections explicitly encoded in the grammar. Although the possibility is mentioned, no grammar learning capabilities are included. Grammar induction is used in [5] to detect co-occurring patterns in the joint-angles of humans performing body movements, but the grammars are not used to abstract away structure in these patterns. The WordNet semantic lexicon is used as a hand-built source of abstraction in [6] to find high-level concepts matching low-level features detected in videos.

As alluded to earlier, in the domain of object recognition, a system for finding a common abstraction from a set of oversegmented objects is presented in [8]. The approach explores a lattice obtained by grouping and abstracting regions in two exemplars in all possible ways. In order to compute the lowest common abstraction of a set of exemplars, least common abstractions over all possible pairs of inputs are generated and a minimum distance criterion is used to choose the best solution. Because there are an exponential number of increasingly-abstract representations, a top-down polynomial-time approximation algorithm is proposed. Our system is based on the same least common abstraction paradigm and approximation approach, but adapted to the particular requirements of action recognition.

3. Overview of our Approach

Verbs are commonly used in natural language to express events or actions taking place in the environment. Our goal is to learn the semantics of certain (motion) verbs from a set of videos showing an action (e.g., a person throwing a ball on a book) labeled with a full sentence describing the action (e.g., *[Name-X] throws the ball on the book*). We define the meaning of such a verb to be the sequence of motions and/or interactions of objects that are in common across the training examples for the verb. We represent an object motion or

interaction maintained over time as a *state* in which relevant features hold over the object(s); for example, $\downarrow d(a, o)$ indicates that objects a and o are near in space. State boundaries occur when the values of the features change, indicating a change in the object motions or interactions. Learning the semantics of a verb thus means finding state boundaries in each training video such that a consistent sequence of states holds over the set of videos.

In addition to indicating what verb is to be learned, the association of a video with a sentence describing the depicted action helps us to establish correspondences between information gleaned from the sentence and from the video. These linkings are crucial to forming the appropriate semantics of the verb. Consider the above video–sentence pair again. There are three participants in the action expressed in this example, which appear in the video as three visual objects that move around and interact with each other, and in the sentence as three noun phrases (NPs), *[Name-X]*, *ball*, and *book*. We first apply a context-free parser to each sentence that uses general rules of the language to extract two pieces of information: the NPs to associate with objects in the video, and the semantic roles that those objects will play in the definition of the verbal semantics. In this case, the parser determines that *[Name-X]*, *ball*, and *book* are the relevant objects, and that, in the verb semantics, they will take on the roles of AGENT, OBJECT, and PLACE, respectively.

We can now use this language information to guide interpretation of the video. For each identified NP, we assume an object recognition component can locate the object of that type in the video, and that some process (either manual or automatic) initializes a convex hull around each rigid subpart of the object (or entire object, if non-articulating). We then label each of these subparts with the semantic role indicated by the parse of the sentence (e.g., AGENT). Now tracking the subparts over the course of a video (i.e., their 2D image positions as a function of time) will provide the input features that capture the motions and interactions of the semantic participants in the action. This enables us to extract meaningful representations from the video; e.g., in the feature mentioned earlier, $\downarrow d(a, o)$, we assert a relation between the AGENT and the OBJECT in the component state of the verb meaning, abstracting away from the specific entities involved in a particular video.

The challenge is to find a sequence of states that holds consistently across all the training videos for a verb. Two problems arise. First, we must determine where to locate state boundaries in each video; that is, we need to partition each video into a sequence of time intervals over which a number of features hold. Note that there is no prior information about how many states are needed to encode the verb semantics, so we must consider partitions of varying sizes. Moreover, given the possible partitions for each video, we must find an *alignment*—a set of partitions, one per video,

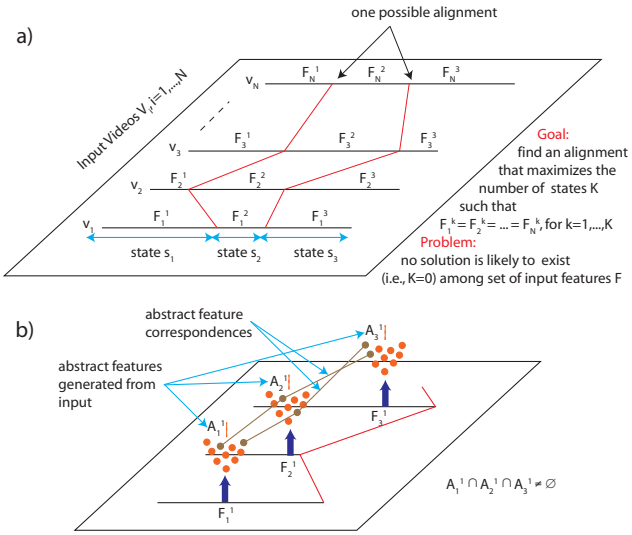

Figure 1. (a) From a set of N input videos, we seek an alignment of the partitions, one per video, where each partition has the same number of intervals or states (in this case, three), and corresponding states across the videos have similar features. (b) While no feature commonality may exist at the input feature level, commonality may exist at a more abstract level. For the first state in the first three videos, two abstract features are in common.

where each partition has the same number of intervals—in which the features within corresponding time intervals have consistent values. See Figure 1: the input is a set of N videos, each yielding a set of time-varying features over convex subparts of objects. In Figure 1(a), an example alignment is shown in which the N input videos are each partitioned into three states. Recall our first "throw" example in Section 1, in which the input videos were highly regular. Such a highly constrained set of training videos may indeed yield an alignment with the same input features (or perhaps a subset of the features) across all the videos.

However, this raises the second issue: given more realistic training data for a typical action verb, there may not be an alignment that provides a non-empty set of consistent input feature values across corresponding partitions of the videos. That is, the input features that we extract are unlikely to capture the higher-level commonality across the videos that abstracts away from very specific motion and interaction primitives. In order to form a successful alignment, we need to find the right level of abstraction of the input features that yields a consistent sequence of states representing the meaning of the verb. Figure 1(b) illustrates this more ambitious formulation, where we have focused on the first state in the first three input videos in Figure 1(a). If we could generate an appropriate set of abstract features, then our strategy of searching for a non-empty intersecting set of features across corresponding intervals would yield a common abstraction.

It is important to note here that one useful way to generate abstract features is to summarize the values of local input features over an interval of time. Therefore the computation of such abstract features depends on the time interval over which they are computed—that is, the abstract features cannot be pre-computed. Thus our strategy is to hypothesize an alignment, compute abstract features for each interval (in each video) induced by the alignment, and search for a non-empty intersection of the abstract features.

A successful alignment will yield a sequence of states that each represent a subpart of the meaning of the verb. For example, our system may learn the following meaning of *throw* from several videos showing a "throwing" action:

$$\begin{array}{lccc} & \text{STATE}_1 & \text{STATE}_2 & \text{STATE}_3 \\ \lambda o,p,a. & \overbrace{\downarrow d(a,o)}; & \overbrace{\uparrow d(a,o) \wedge \uparrow \left\| \frac{d}{dt}\, c(o) \right\|}; & \overbrace{\downarrow d(o,p)} \\ & \text{near}(a,o); & \text{far}(a,o) \wedge \text{moving}(o); & \text{near}(o,p) \end{array}$$

where *a* stands for the AGENT, *o* for the OBJECT, and *p* for the PLACE. The lambda expression is the output of our system, whose semantics will be explained in Section 4 below. (The text underneath the lambda expression indicates an intuitive label for each feature, for illustrative purposes only.) The lambda expression shows that the learned semantics of *throw* in this example consists of three successive states: First, the AGENT *a* is in contact with the OBJECT *o*; second, *o* is not in contact with *a* and is moving (quickly); third, *o* is in the proximity of the PLACE *p*.

4. The Abstract Feature Space

To produce our abstract feature space, we extend the set of input features (extracted as explained above), using operators that generate simpler, higher-level, and/or combined features. These abstract/combined features are intended to capture some higher-level semantics of object motion and/or interaction (compared to the low-level input features), e.g., the speed of the movement of an object (irrespective of the manner of motion), and the relative distance between two objects (regardless of their absolute positions).

We use these operators in our current system:

- $c(o)$: computes the centroid of a moving object o as a 2D signal.
- $d(o, o')$: computes the distance between the convex hulls of two moving objects, o and o', as a scalar signal.
- $rp(o, o')$: computes a vector representing the relative position of object o with respect to object o'. Because both o and o' can have several parts, the relative position is defined as the vector connecting the centroids of the parts of o and o' which are closest to each other.
- $\frac{d}{dt}(s)$: evaluates the time derivative of a 1D or 2D signal.
- $\|s\|$: returns the norm of a 1D or 2D signal at each time instant.
- $\uparrow_\sigma$, $\downarrow_\sigma$, $\uparrow^+_\sigma$, $\uparrow^-_\sigma$: soft threshold operators on a 1D signal. Operator $\uparrow$ will respond whenever the magnitude of its argument is large, while operator $\downarrow$ will respond whenever its magnitude is small. Operators $\uparrow^+$ and $\uparrow^-$ respond to large positive (respectively negative) values of their scalar argument. We use the Geman-McClure estimator to evaluate these operators:

$$\uparrow^+(s) = \frac{s^2}{\sigma^2 + s^2} \cdot [s \geq 0] \qquad \uparrow(s) = \frac{s^2}{\sigma^2 + s^2}$$
$$\uparrow^-(s) = \frac{s^2}{\sigma^2 + s^2} \cdot [s \leq 0] \qquad \downarrow(s) = \frac{\sigma^2}{\sigma^2 + s^2}$$

- *holds*(s): computes a global measure of how close the signal s (in the range [0,1]) is to the true value (1) over an entire interval. The comparison is done by calculating the squared error between s and the constant signal 1, and then using a fixed threshold to yield a single boolean value.

In order to generate meaningful abstract motion features, we use the following grammar rules:

$$\begin{array}{rcl} \text{OBJECT} & \rightarrow & a \mid o \mid p \\ \text{DISTANCE} & \rightarrow & d(\text{OBJECT}, \text{OBJECT}) \\ \text{POS} & \rightarrow & c(\text{OBJECT}) \\ \text{VEL} & \rightarrow & \frac{d}{dt}\, c(\text{OBJECT}) \\ \text{REL\_POS} & \rightarrow & rp(\text{OBJECT}, \text{OBJECT}) \\ \text{REL\_VEL} & \rightarrow & \frac{d}{dt}\,(\text{REL\_POS}) \\ \text{REL\_ACC} & \rightarrow & \frac{d}{dt}\,(\text{REL\_VEL}) \\ \text{VECTOR} & \rightarrow & \text{VEL} \mid \text{REL\_VEL} \mid \text{REL\_ACC} \\ \text{SCALAR} & \rightarrow & \text{DISTANCE} \mid \|\text{VECTOR}\| \mid \text{VECTOR}_x \mid \text{VECTOR}_y \\ \text{SCALAR}^- & \rightarrow & \text{VECTOR}_x \mid \text{VECTOR}_y \\ \sigma_{\text{M}} & \rightarrow & 0.125 \mid 0.25 \mid 0.50 \mid 1.0 \mid 2.0 \mid 4.0 \\ \text{MAG} & \rightarrow & \uparrow_{\sigma_{\text{M}}} \text{SCALAR} \mid \downarrow_{\sigma_{\text{M}}} \text{SCALAR} \mid \uparrow^+_{\sigma_{\text{M}}} \text{SCALAR}^- \mid \uparrow^-_{\sigma_{\text{M}}} \text{SCALAR}^- \\ \text{QUAL} & \rightarrow & \textit{holds}(\text{MAG}) \end{array}$$

The first 10 rules constrain the possible combinations of the operators. The σ_{M} nonterminal generates the set of scales used when applying the soft threshold operators (those subscripted with σ).[1] The MAG nonterminal expands to soft-thresholded magnitude features. The start symbol is QUAL, which expands to an evaluation, *holds*(MAG), of all possible abstract features. The language of this grammar thus defines the set of abstract features, which are the elements out of which our higher-level verb semantics are constructed.

[1] We choose six values for σ_{M} because they cover a wide range of scales occurring in practice. For example, we can capture movements as slow as 0.25 pixels/frame and as fast as 8 pixels/frame.

5. Learning Verb Meanings

In this section, we present a formal statement of our search problem (Section 5.1), reformulate it into a search over a graph (Section 5.2), and propose an efficient approximation algorithm for finding a solution (Section 5.3).

5.1. A Formal Specification of the Problem

Let us denote by $\mathcal{F}$ the set of abstract features (generated as explained in Section 4), by $\mathcal{V}$ the domain of input videos, and by $\mathcal{I}$ the set of time intervals. Given an input video $v \in \mathcal{V}$ and a time interval $[a, b] \in \mathcal{I}$, we can evaluate any feature $f \in \mathcal{F}$ yielding a truth value:[2]

$$val : \mathcal{F} \times \mathcal{V} \times \mathcal{I} \rightarrow \{true, false\}$$

We formally represent a state as a set of abstract features; the domain of states, $\mathcal{S}$, is thus the power set of $\mathcal{F}$. We interpret a state s as the conjunction of its component features f: a state evaluates to *true* iff all its features evaluate to *true*:

$$val(s, v, [a, b]) = \bigwedge_{f \in s} val(f, v, [a, b])$$

We represent an event as an ordered sequence of states, and use $\mathcal{R}$ to denote the set of all events, which also specifies the domain of representations for verb semantics:

$$\mathcal{R} = \mathcal{S}^*$$

Such a representation can only match an input video v with respect to a time partition (because features need to be evaluated over a time interval). Let us denote by $\Delta(v)$ the set of all time partitions for video v, defined formally as:

$$\Delta(v) = \{(t_1, t_2, \ldots, t_n) \in \mathbb{N}^* | t_1 = 1, t_n = l(v), t_i < t_{i+1}\}$$

where $l(v)$ is the length of video v.

A representation $r \in \mathcal{R}$ matches an input video v for a partition $\delta \in \Delta(v)$, if the number of time intervals in δ matches the number of states in r, and the evaluation of each state on the corresponding time interval yields *true*:

$$eval(r, v, \delta) = \begin{cases} \bigwedge_{i=1}^{|\delta|} val(r_i, v, [t_i, t_{i+1}]) & , r \text{ has } |\delta| \text{ states} \\ false & , \text{otherwise} \end{cases}$$

where r_i is the i^{th} state in the representation r. We define the set of all representations that match a given video as:

$$\mathcal{R}^{true}(v) = \{r \in \mathcal{R} | \exists \delta \in \Delta(v), eval(r, v, \delta) = true\}$$

[2]The start symbol of the feature grammar, QUAL, expands to a boolean expression by application of the *holds* operator to soft-thresholded features. Other means of generating boolean expressions using novel operators can be added without affecting our algorithm.

The size of a representation is the total number of features in its states:

$$|r| = \sum_{i=1}^{n} |r_i|$$

We can now reformulate our problem as finding, for a set of videos $V = \{v_1, \ldots, v_k\}$, a representation r^* of maximal size which matches all the videos:[3]

$$r^*_{(V)} = \underset{r \in \cap_{v \in V} \mathcal{R}^{true}(v)}{\operatorname{argmax}} |r|$$

To see if any given r is an element of $\mathcal{R}^{true}(v)$, a set of k partitions has to be determined, one for each input video. We thus have a two-fold search problem, where we need to simultaneously search over $\mathcal{R}$ and $\Delta(v)$. However, it can be shown that the maximization problem above, due to the nature of its objective function, can be formulated as a search over the set of partitions alone. Due to lack of space, here we only present a statement of this claim without proof.

A tuple of partitions (one from each of the input videos) having the same length defines an alignment. We define the set of all possible alignments for a set of inputs as:

$$\mathcal{A} = \{(\delta_1, \cdots, \delta_k) \in \Delta(v_1) \times \cdots \times \Delta(v_k) \mid |\delta_1| = \cdots = |\delta_k|\}$$

We define the set of all features that hold in input video v over interval $[a, b]$ as:

$$\mathcal{F}^{true}(v, [a, b]) = \{f \in \mathcal{F} | val\,(f, v, [a, b]) = true\}$$

For an alignment $\eta = (\delta_1, \ldots, \delta_k)$, an optimal representation $r(\eta)$ can be obtained by intersecting the sets of states that match the corresponding time intervals in each video:

$$r(\eta)_i = \bigcap_{v_j \in V} \mathcal{F}^{true}(v_j, [\delta_{j,i}, \delta_{j,i+1}])$$

where $[\delta_{j,i}, \delta_{j,i+1}]$ is the i^{th} interval in partition δ_j. Then, our optimization problem is equivalent to:

$$\eta^* = \underset{\eta \in \mathcal{A}}{\operatorname{argmax}} \; |r(\eta)| \tag{1}$$

with the optimal solution given by $r^* = r(\eta^*)$.

5.2. A Graph Search Formulation

Here, we present a graph search formulation of our problem. We assume that a set of candidate interval boundaries—which we call *cut points*—have been selected for each input video. (Section 5.3 explains how the cut points are chosen.)

[3]We aim for a maximal representation because that captures the most commonality among the input videos, avoiding a trivial solution of one state with a single feature, which simply says that the videos, in their entirety, are equivalent—i.e., all depict the same verb.

Consider a set of k training videos, and let C_i be an ordered set, $\{c_i^1, \ldots, c_i^{n_i}\}$, of all candidate cut points in video v_i, with c_i^1 and $c_i^{n_i}$ corresponding to the beginning and end, respectively, of v_i. We construct a graph $G = (U, E)$, where each vertex corresponds to a tuple of candidate cut points, one from each training video, i.e.:

$$U = C_1 \times \cdots \times C_k$$

Considering two vertices, $u_a = (c_1^{a_1}, \ldots, c_k^{a_k})$ and $u_b = (c_1^{b_1}, \ldots, c_k^{b_k})$, we add an edge $e = (u_a, u_b)$ if and only if:

$$a_i < b_i,\ i = 1, \ldots, k \tag{2}$$

That is, the cut points corresponding to the video v_i in u_a and u_b respect the time ordering, such that the cuts in u_b always follow cuts in u_a.

G includes two special vertices: $u_{\text{start}} = (c_1^1, \ldots, c_k^1)$ that aligns the cuts at the beginning of all videos (with no incoming edges); and $u_{\text{end}} = (c_1^{n_1}, \ldots, c_k^{n_k})$ that aligns all cuts at the end of the input videos (with no outgoing edges). Any path from u_{start} to u_{end} defines an alignment of the videos. Furthermore, the set of all such paths covers the entire set of possible alignments, restricted to using only the candidate cut points.

Let P be a path corresponding to an alignment η. In order to compute the objective function $|r(\eta)|$ in (1) above, we assign a weight to each edge $e = (u_a, u_b) \in P$ that is the number of features that hold over the corresponding intervals in all videos:[4]

$$w(e) = \left| \bigcap_{i=1}^{k} \mathcal{F}^{\textit{true}}(v_i, [c_i^{a_i}, c_i^{b_i}]) \right|$$

It is easy to see that the sum of the weights of all the edges in P gives the size of the maximal representation $r(\eta)$ for the corresponding alignment η. Hence, the longest (maximum weight) path in G corresponds to the optimal solution we are looking for. Because G is a directed acyclic graph, dynamic programming can be used to find the longest path in time polynomial in the number of vertices. Next, we present two heuristics for reducing the number of vertices in the search graph.

5.3. Heuristics for Pruning the Search Space

The complexity of our full optimization problem is prohibitive because the hypothesized interval boundaries (candidate cuts) within a partition may appear at any time point in a video. We use heuristics for selecting a set of promising cut points, narrowing the search space substantially.

In practice, transitions between states are likely to occur when local minima or maxima are encountered in the feature values. Our first heuristic is to reduce the search space to only these local minima or maxima, which we detect by using a 1D version of the Canny edge detector [2]. This involves applying a derivative of Gaussian filter on the signal resulting from the evaluation of each feature. Non-maximal suppression is then carried out for each signal independently. Finally, the resulting extrema from all signals are clustered together using robust M-estimation, yielding a small set of candidate cut points. To bound the complexity of the subsequent computations, we limit our search space to the cuts corresponding to the N_{cuts} largest clusters.

For an average number n of candidate cuts across the videos, the graph constructed as explained in Section 5.2 has $O(n^k)$ vertices. The time for computing the longest path in this graph is thus exponential in the number of training videos, $O(n^{2k})$. The time ordering constraint (2) generates a total number of edges which is approximately half the number of vertex pairs. Computing the longest path requires evaluating all the features on all the edges, which is linear in the number of abstract features, $|\mathcal{F}|$. This gives a total running time for both generating the graph and computing the longest path of $O(|\mathcal{F}| \cdot n^{2k})$.

The search space can be further reduced by removing less likely alignments. Our second heuristic is based on the observation that the pattern of transition of features in the vicinity of aligned cuts should be similar. We define a cut profile that encodes the pattern of transition for all features in video j around a given cut $c \in C_j$. The profile is obtained by computing the slope of the feature values over a set of different scales. For each scale and feature, we classify the slope as upwards (an increase in the feature value), downwards (a decrease in the feature value) or horizontal (no change). We then take the majority vote on these transition types over all scales to obtain a single transition pattern for each feature and cut. A cut profile is the vector of such transitions for all features.

We label each vertex $u = (c_1^{a_1}, \ldots, c_k^{a_k})$ with the average similarity among the profiles corresponding to the cuts in u. Similarity between a pair of cut profiles is measured using the Hamming similarity between the corresponding vectors of discrete transition patterns. We then restrict our graph to contain only the top T vertices (those with the highest similarity scores), and search for the longest path in this new graph. Pruning the original set of $O(n^k)$ vertices using a constant number of features ($|\mathcal{F}|$) leads to a complexity of $O(|\mathcal{F}| \cdot n^k)$, a significant reduction in the exponent from $O(|\mathcal{F}| \cdot n^{2k})$.

6. Experimental Results

We perform experiments on four verbs; for each, we have a set of monocular videos of persons performing the corresponding manipulative action. The verbs include *put*, with 8 training videos, and *push*, *pull*, and *touch*, with 10 training

[4] To reduce the effect of outliers, in practice, we compute the weight as the number of features that are true in a majority of the videos.

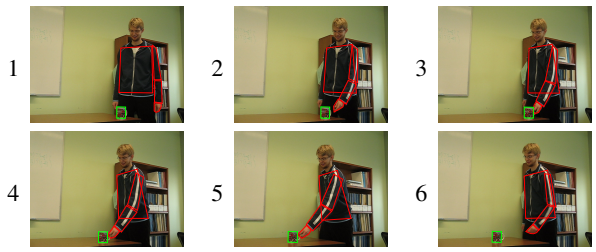

Figure 2. Six equally spaced frames from a video labeled *Calden pushes the cup*.

videos each. The contrasting semantic properties of these verbs allow us to investigate relevant aspects of our system.

We manually segment each video to determine the position of the upper limb involved in the action and the torso, as shown in Figure 2 (avoiding full-body segmentation due to time constraints). Candidate cut proposals are then generated for each video. We use a threshold of 0.01 for non-maximal suppression, a value of $\sigma = 4$ for the robust estimator, and a value of $N_{\text{cuts}} = 9$ (including the beginning and the end of the videos) for trimming the cut points. The average number of frames per video ranges from 58.2 (for *push*) to 102 (for *put*). The average number of cut points per video ranges from 6.6 (for *push*) to 8.4 (for *put*). Had it been generated, the size of the full graph using these candidate cuts would range from 2.83×10^6 to 3.61×10^8 vertices. Using the cut similarity heuristic, we limit our search to a graph consisting of only the best 1000 vertices. The solutions we obtain contain three states for *push* and *touch* and four states for *pull* and *put*, with an average of 47.3 features per state, selected out of a total number of 192 abstract features generated by our grammar.

6.1. Learned Models of Verbs

We display the semantic representation learned for a verb in a format as shown in Table 1, where the features are listed in the column titled FEATURE, and the final states of the solution are presented in columns titled STATE_s. Because the number of features in a solution may be large, the table only presents a subset of them.[5] The scalar feature that is abstracted is indicated in the FEATURE column of the table. The abstract features may have had differing scales (σ_{M}) and soft thresholding operators ($\uparrow$ or $\downarrow$) applied. Each colored rectangle in a STATE cell of the table corresponds to a different abstract feature whose value of the scale parameter σ_{M} ranges from .125 to 4 (left to right in the table). The color of the rectangle—blue or red—reflects which soft threshold operator ($\uparrow$ or $\downarrow$) is selected in the learned model; a blank rectangle indicates that no commonality was found for the feature at that scale. For example, the third cell of the

[5]Features involving accelerations and the $\uparrow^+$ and $\uparrow^-$ operators are not shown here for simplicity and space reasons, although they do carry intuitive semantic meaning.

Table 1. Semantic representation learned for *push*.

	FEATURE	STATE_1	STATE_2	STATE_3
1	$d\,(a,o)$			
2	$\frac{d}{dt}\,rp\,(a,o)_x$			
3	$\frac{d}{dt}\,rp\,(a,o)_y$			
4	$\left\|\frac{d}{dt}\,rp\,(a,o)\right\|$			
5	$\frac{d}{dt}\,c\,(o)_x$			
6	$\frac{d}{dt}\,c\,(o)_y$			
7	$\left\|\frac{d}{dt}\,c\,(o)\right\|$			

Table 2. Semantic representation learned for *touch*.

	FEATURE	STATE_1	STATE_2	STATE_3
1	$d\,(a,o)$			
2	$\frac{d}{dt}\,rp\,(a,o)_x$			
3	$\frac{d}{dt}\,rp\,(a,o)_y$			
4	$\left\|\frac{d}{dt}\,rp\,(a,o)\right\|$			
5	$\frac{d}{dt}\,c\,(o)_x$			
6	$\frac{d}{dt}\,c\,(o)_y$			
7	$\left\|\frac{d}{dt}\,c\,(o)\right\|$			

first row (Table 1) shows all the abstract features appearing in state number 2 of the solution, which have been derived from the relative distance between the agent and the object, namely $d(a,o)$, through thresholding at different scales.

Consider the first row of Table 1, containing the feature $d\,(a,o)$—the distance between the agent (a) and the object being pushed (o)—and its values (the color rectangles) selected in each state of the model representation. We can see that in STATE_2, the value of this feature is low (specified by the red color) at all scales, reflecting that a and o are in close contact while the object is being pushed by the agent (frames 3-5 in Figure 2). In STATE_1 and STATE_3, there is a non-zero distance between a and o, as shown by the blue color at small scales (frames 1, 2, and 6 in Figure 2).

The values of the last three features (the velocity of o) show that the object is not moving in the first and the final states, but it makes a horizontal movement at a slow, and bounded, speed (blue color at small scales, red color at large scales) during the second state (while being pushed by a). Looking at the values of the second and fourth features (the horizontal speed of a relative to o), we can see that in STATE_1 and STATE_3, a is moving (recall from the above explanation that o is not moving)—although the speed is considered high only at smaller scales (up to .5).

To summarize, the three states of the learned model for *push* correspond to the following: The agent comes into contact with the object (o is stationary, while a is moving relative to o in STATE_1), the agent pushes the object (a and o move while in contact in STATE_2), and finally the agent is no longer in contact with the object (o is stationary, while a is moving relative to o in STATE_3). For the verb *pull* (not shown here), the semantics is very similar to *push*, with

the exception that an intermediate state is present which encodes that there is a time interval in which both the agent and the object are at rest. The semantics of *touch* resembles that of *push* and *pull* (see Table 2); its distinguishing properties are discussed below. For the verb *put* (also not shown), the model naturally describes the agent holding the object (STATE_1 and STATE_2), the object reaching its final location (STATE_3), and finally the agent releasing the object (STATE_4). The first two states differ only in that humans in our videos tend to move the object faster at the beginning and then slower just before reaching the target place.

6.2. Analysis of the Learned Models

Many regularities of the training data are reflected in the model semantics learned for the four verbs. State 2 for *push* is an example where the system has bounded the speed of the object being pushed both from below and above, locking on the range of speeds at which the human subjects have executed the event (row 7 of Table 1). Also in the second state of *push*, the system has detected a horizontal movement as part of the model semantics because all our objects are pushed on a horizontal surface (row 5 of Table 1). Such "accidental" regularities in the training set will be captured by the solution. However, as more training videos are added, such regularities are less likely to persist across the entire data set, and will be factored out as the search for commonality is driven to higher levels of abstraction.

Conversely, there are cases where the learned models are too weak. For example, in the case of *put*, the proximity of the object to the target place at the final state is missed because no appropriate scale can be found (in the provided range) at which all (or most) videos show a similar pattern.

Comparing Tables 1 and 2, we see that the system successfully distinguishes between *push* and *touch* in the different values of the fifth and seventh features. In STATE_2, the representation of *push* includes movement of the object (the blue rectangles for these features), while the representation of *touch* has no movement (all red rectangles for those features). However, for *push* and *pull*, our system fails to identify a discriminating feature that reflects the direction of the movement of the object relative to the agent (in both cases the hand and the object move while in contact). This reveals an insufficiency in our feature space, which we need to augment with an operator for projecting the velocity of an object onto its relative position with respect to another object, indicating motion toward or away from the other object.

7. Conclusions

We have presented a graph theoretic formulation for learning the motion/interaction semantics of a verb from captioned training data. Our approach is novel in two important ways: we assume neither knowledge of the number of states representing an action nor that the training videos necessarily share some subset of input features. Determining the meaning of a verb may require searching an intractable space of abstract features for the lowest common abstraction capturing the maximum degree of regularity across the training examples. Focusing on feature discontinuities across scales, we effectively reduce this search space, yielding approximate solutions successfully demonstrated on four verbs. We currently are working on two extensions: modifying our learning algorithm to support classification of a video (to enable automatic annotation), and simultaneously learning the semantics of verbs (motions/interactions) and nouns (object appearance/shape).

8. Acknowledgements

We are grateful to NSERC, OCE and Idée, Inc. for financial support, and to Yulia Eskin and Calden Wloka for helping us generate and annotate the videos.

References

[1] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23:257–267, 2001.

[2] J. F. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8:679–714, 1986.

[3] T. Darrell and A. P. Pentland. Space-time gestures. In *CVPR*, 335–340, 1993.

[4] A. Fern, R. Givan, and J. M. Siskind. Specific-to-general learning for temporal events with application to learning event definitions from video. *Journal of Artificial Intelligence Research*, 17:379–449, 2002.

[5] G. Guerra-Filho and Y. Aloimonos. A language for human action. *IEEE Computer Magazine*, 40:60–69, 2007.

[6] A. Hoogs, J. Rittscher, G. Stein, and J. Schmiederer. Video content annotation using visual analysis and a large semantic knowledgebase. In *CVPR*, 327–334, 2003.

[7] Y. Ivanov and A. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22:852–872, 2000.

[8] Y. Keselman and S. J. Dickinson. Generic model abstraction from examples. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27, 2005.

[9] J. M. Siskind and Q. Morris. A maximum likelihood approach to visual event classification. In *European Conference on Computer Vision*, 347–360, 1996.

[10] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104:249–257, 2006.

[11] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-squential images using hidden markov model. In *CVPR*, 379–385, 1992.

[12] A. Yilmaz and M. Shah. A differential geometric approach to representing the human actions. *Computer Vision and Image Understanding*, 109:335–351, 2008.