Learning Actions from Events Using Agent Motions

Nikhil Krishnaswamy, Tuan Do, and James Pustejovsky

nkrishna@brandeis.edu • tuandn@brandeis.edu • jamesp@brandeis.edu

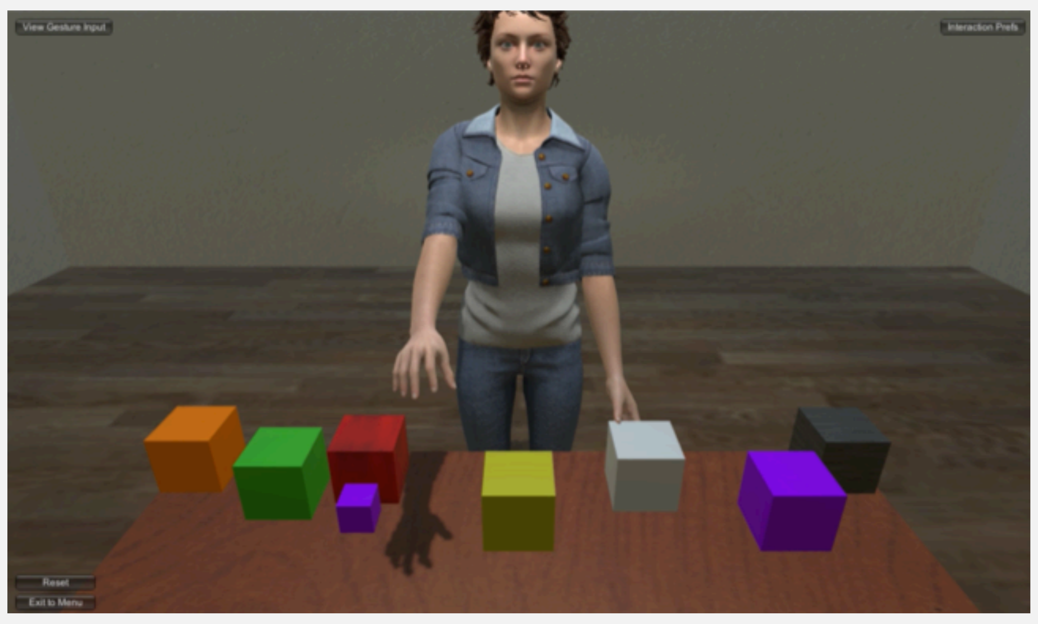
Overview

Work in event *visualization* from natural language (Coyne and Sproat, 2001; Siskind, 2001; Chang et al., 2015) often struggles with underspecified parameters in events. These parameters may be inherent to the event itself (e.g., speed, direction, etc.), or properties of the object argument(s) (e.g., axis of rotation, geometrical concavity, etc.). Should a computational visualization system use an inappropriate value for one of these parameters, it may generate a visualization for a given event that does not comport with a human viewer's understanding of what that event is.

Event *recognition* provides a venue to explore "learning from observation," and as a domain has achieved recent relevance in human communication with robotic agents (Yang et al., 2015b; Paul et al., 2017). Learning can abstract away the parameters that vary across instances of the same motion class, making those parameters underspecified as well, as in the aforementioned visualization problem. For an embodied agent to interact with objects, the agent must use its hands, and the hand motions effect forces upon the object. Thus, we expect that the same parameter abstraction approach can be used for the agent's hand motions, creating a path toward action recognition from hand gestures only.

Causal events are composed of an *object model*, which captures the change an object is undergoing over time, and an *action model*, which characterizes the activity that inheres in the causing agent (Pustejovsky and Krishnaswamy, 2016). We present results from an event visualization system using multimodal simulations and methodology from an event learning and composition system to introduce a framework for learning action recognition from the movements of the *agent* rather than the object. We expect such a framework may be useful for recognizing and evaluating the actions denoted by agent motions enacted without attached objects, e.g., by gestures.

Extracting Actions from Events



Where captured instances contain multiple object configurations or permutations under the same label (for example, building rows of varying numbers of blocks or putting two objects near each other in various orientations), the LSTM learns event progress by changes in object relations, such as the number and relative orientation of EC or "touching" relations between objects in a row. This allows the REINFORCE algorithm to generalize a concept (e.g., row) to set of common relations across all captured or simulated instances without a set number of blocks. This makes the parameters that vary across the captured instances underspecified.

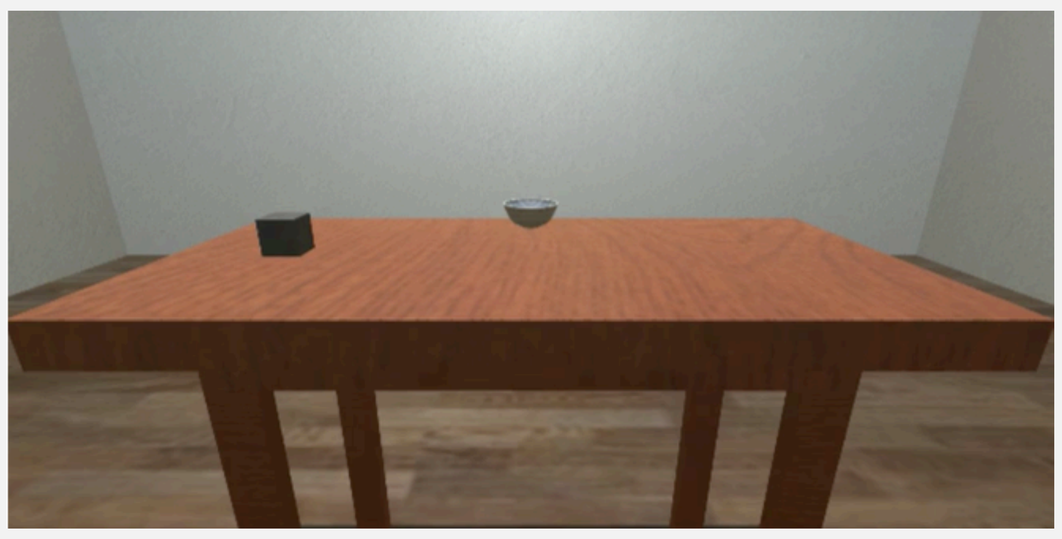
As we have shown that underspecified motion features appear to be strong signals of event class for objects moving in isolation, we expect the same principle holds for objects being manipulated by an agent, especially as one of the goals of our reinforcement learning pipeline is to abstract away those parameters whose values vary across the performed or simulated example actions.

For instance, let us return to the semantics of "slide" presented in Figure 1. One of the requirements is that at all times the moving object is kept EC (externally connected) with the supporting surface. Since in a 3D environment, all motions eventually break down into a series of translations and rotations, all relations between objects can be represented as relative offsets and orientations, as in the reinforcement learning trials. Thus, if "sliding" motions of various speeds and moving in various directions all return roughly equal rewards as long as the object remains attached to the supporting surface (as the LSTM should produce high values of event progress for all these motions given enough performed examples), the REINFORCE algorithm should be able to generate an event sequence wherein many values for these parameters can be sampled from the Gaussian distribution, and the action, when performed by an agent with those values, should satisfy an observer's judgment given the "slide" label. Thus the high variance of motion speed and motion direction comport with those parameters' status as strong signals of the "slide" event class.

Since in the 3D simulated world with the agent, objects are manipulated by attaching them to the agent's "graspers" or hands, so that the motion of the hand controls the motion of a grasped object, it is the motion of the hand that dictates what class of action is being undertaken. Thus in the above example, if the hand motion may take a wide variety of values of speed and direction but always maintains a constant or near-constant vertical offset with the surface (representing the height of the object being moved), then this motion may be interpreted as representing a "slide," regardless of whether or not any actual object is being moved. If no object is moved along with the hand, this "action model" becomes a "mime" or gestural representation of the action in question.

Event Classification

Using the VoxSim simulation environment (Krishnaswamy and Pustejovsky, 2016; Krishnaswamy, 2017), we generated three visualizations for input sentences of the imperative form VERB x (or VERB x RELATION y). Amazon Mechanical Turk workers were shown a single animated movie of an event and asked them to select, out of three heuristically-generated captions (one of which was the original input sentence, the best one.



Sample VoxSim capture for "move the block"

Values assigned to the verb's underspecified features during visualization were saved in feature vectors used to train classifiers to select the verb of the input sentence that produced that vector (ML analogue to the MTurk task for a **restricted** set of 3 verbs or an **unrestricted** set of 16). We trained a baseline MaxEnt and 8 variants of a multi-layer perceptron on this task:

Classifier	μ Accuracy (restricted set)	μ Accuracy (unrestricted set)
Baseline	0.4850	0.1662
DNN variant 1	0.9788	0.9514
DNN variant 2	0.9788	0.9547
DNN variant 3	0.9800	0.9550
DNN variant 4	**0.9895**	**0.9707**
DNN variant 5	0.9615	0.9150
DNN variant 6	0.9600	0.9144
DNN variant 7	0.9615	0.9675
DNN variant 8	0.9871	0.9150

All neural network variant exceeded 90% accuracy for verb selection given underspecified features

The best performing network was trained on only the **presence or absence** of a given feature, independent of value, showing that the mere existence of a feature is a strong predictor of motion class.

$$\begin{bmatrix} \textbf{slide} & \\ \text{LEX} = & \begin{bmatrix} \text{PRED} = \textbf{slide} \\ \text{TYPE} = \textbf{process} \end{bmatrix} \\ \text{TYPE} = & \begin{bmatrix} \text{HEAD} = \textbf{process} & \\ \text{ARGS} = & \begin{bmatrix} \text{A}_1 = \textbf{x:agent} \\ \text{A}_2 = \textbf{y:physobj} \\ \text{A}_3 = \textbf{z:surface} \end{bmatrix} \\ \text{BODY} = & \begin{bmatrix} \text{E}_1 = grasp(x, y) \\ \text{E}_2 = [while(hold(x, y), \\ \quad while(EC(y, z), \\ \quad move(x, y)))] \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

VoxML semantics for [[SLIDE]]. The absence of speed and direction parameters in E_2 indicates underspecification.

Conclusion & Future Directions

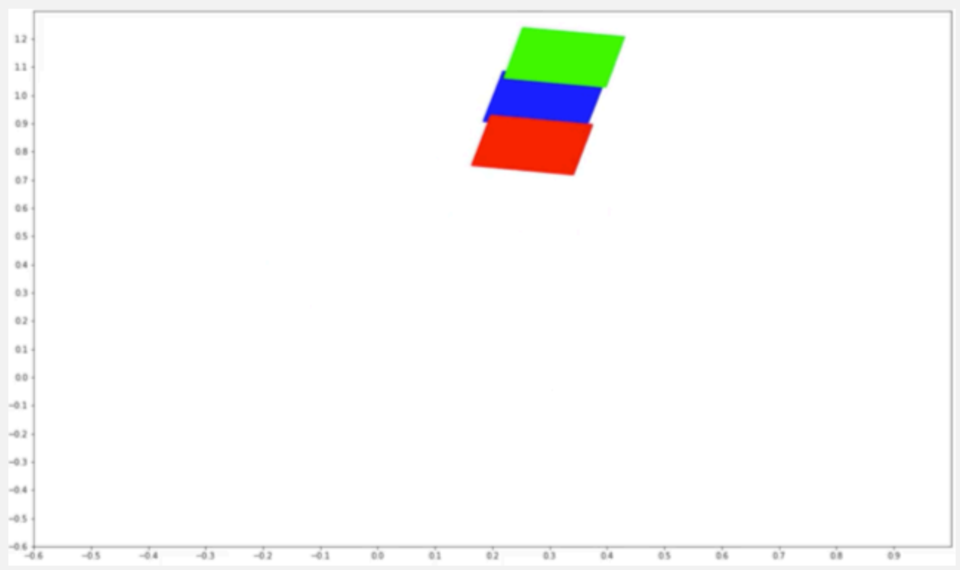
We have argued and presented evidence that underspecified parameters associated with motion events can serve as reliable indicators of a particular event class. We have also presented a framework for action learning that relies on abstracting away those motion parameter values that may vary across individual instances and performances of events. These two avenues naturally combine to create a pipeline for action recognition by a computational agent using information from visual and linguistic modalities (cf. (Yang et al., 2014; Yang et al., 2015), and for using action performance and gestural representations of actions as a learnable communicative modality between humans and computers.

Learning Complex Events

Even a simple event such as *put(x, near(y))* requires a series of translations that can be difficult for a computer to distinguish from other types of motions involving changing relations between two objects. For this is a sequential learning problem, we turn to LSTM (Hochreiter and Schmidhuber, 1997) to learn the sequence of primitive events that comprise a complex event. If the sequence can be effectively learned, it should be able to be reproduced by a virtual embodied agent, whose objective is to produce a sequence of actions that resembles movement of objects in the training data.

In the tool ECAT (Do et al., 2016), we replicate the virtual scenes generated with VoxSim, but with the presence of a real agent to manipulate the objects (blocks only). Video is captured and object tracked with Microsoft Kinect depth-sensing cameras, and three-dimensional coordinates of performer joints are also captured and annotated.

Captured object positions are then flattened to two dimensions in order to normalize any jitter in the capture and allow for easier evaluation of object relations relative to the table surface.



A sequence of feature vectors, *S*, which represent the qualitative spatial relations between the objects in the action captures, is fed to an LSTM network along with a frame number and an event. The network output *f(S, i, e)* = $0 \leq q_i \leq 1$ estimates the progress of *e* at frame *i*.

The virtual agent's objective is then to manipulate the objects in sequence, for a increasing reward as the generated sequence more closely approximates the movement of objects in the training data. We use the REINFORCE algorithm with a Gaussian distribution policy $\pi\theta(u|x) = Gaussian(\mu, \sigma)$, where dim($\mu$) is the degree of freedom in position (2 dimensions) and dim(σ) is the degree of freedom in orientation (1 dimension).

Planning is parameterized by $\theta : u_k \sim \pi\theta(u_k|x_k)$, where u_k is motion performed at step *k* and x_k is current set of relations between objects. After each atomic object manipulation u_k we use the LSTM network to estimate how fully u_k completes the event in question, then calculate the immediate reward as *f(S, k, e) - f(S, k-1, e)*.

The result is a sequence that can be executed by a virtual agent within the VoxSim environment.

Selected References

Chang, A., Monroe, W., Savva, M., Potts, C., and Manning, C. D. (2015). *Text to 3D scene generation with rich lexical grounding.*

Coyne, B. and Sproat, R. (2001). *WordsEye: an automatic text-to-scene conversion system.*

Do, T., Krishnaswamy, N., and Pustejovsky, J. (2016). *ECAT: Event capture annotation tool.*

Do, T., Krishnaswamy, N., and Pustejovsky, J. (2018). *Teaching virtual agents to perform complex spatial-temporal activities.*

Hochreiter, S. and Schmidhuber, J. (1997). *Long short-term memory.*

Krishnaswamy, N. and Pustejovsky, J. (2016). *VoxSim: A visual platform for modeling motion language.*

Krishnaswamy, N. (2017). *Monte-Carlo Simulation Generation Through Operationalization of Spatial Primitives.*

Peters, J. and Schaal, S. (2008). *Reinforcement learning of motor skills with policy gradients.*

Randell, D., Cui, Z., Cohn, A., Nebel, B., Rich, C., and Swartout, W. (1992). *A spatial logic based on regions and connection.*

Siskind, J. M. (2001). *Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic.*

Williams, R. J. (1992). *Simple statistical gradient-following algorithms for connectionist reinforcement learning.*

Yang, Y., Guha, A., Fermuller, C., and Aloimonos, Y. (2014). *A cognitive system for understanding human manipulation actions.*

Yang, Y., Aloimonos, Y., Fermuller, C., and Aksoy, E. E. (2015). *Learning the semantics of manipulation action.*