Teaching Virtual Agents to Perform Complex Spatial-Temporal Activities

Tuan Do, Nikhil Krishnaswamy, and James Pustejovsky
Department of Computer Science
Brandeis University
Waltham, MA 02453 USA
{tuandn, nkrishna, jamesp}@brandeis.edu

Abstract

In this paper, we introduce a framework in which computers learn to enact complex temporal-spatial actions by observing humans, and outline our ongoing experiments in this domain. Our framework processes motion capture data of human subjects performing actions, and uses qualitative spatial reasoning to learn multi-level representations for these actions. Using reinforcement learning, these observed sequences are used to guide a simulated agent to perform novel actions. To evaluate, we render the action being performed in an embodied 3D simulation environment, which allows evaluators to judge whether the system has successfully learned the novel concepts. This approach complements other planning approaches in robotics and demonstrates a method of teaching a robotic or virtual agent to understand predicate-level distinctions in novel concepts.

Motivation

The community surrounding "learning from (human) observation" (LfO) studies how computational and robotic agents can learn to perform complex tasks by observing humans (Young and Hawes 2015). Work in this area can be traced back to reinforcement learning studies by (Smart and Kaelbling 2002) or (Asada, Uchibe, and Hosoda 1999), which closely resembles the way humans learn. Children, as early as 14 months old, can imitate adults in a variety of tasks, such as *turning on and off a light-box*, and can even interpret the intentions behind actions and consider all constraints involved (Gergely, Bekkering, and Király 2002).

Most robots developed in the previous decades have shipped with pre-installed programs, limited to a set of predefined functionalities. Learning approaches in the robotics community seek to move toward smarter and more adaptable robots, for the following reasons, among others:

- Consumer desire for mobile or household assistant robots that can perform multiple tasks with a flexible apparatus, such as multiple grasping arms (Bogue 2017). Robots with behavioral robustness can learn from a wider range of experiences by interacting with humans in a dynamic environment (Hawes et al. 2017).
- Advances in deep learning have afforded robotic agents a high-level understanding of embedded semantics in multiple modalities, including language, gesture, object recognition, and navigation. This increases the circumstances and modalities available for robotic learning.

Event recognition and classification have achieved recent relevance in human communication with robotic agents (Paul et al. 2017). Meanwhile, lexical computational semantic approaches to events (e.g., Pustejovsky (1995), Pustejovsky and Moszkowicz (2011)) make it clear that event semantics are compositional with their arguments.

We have previously presented an approach toward facilitating human communication with a computational agent, using a rich model of events and their participants (Pustejovsky, Krishnaswamy, and Do 2017). Formally, we have devised a semantic framework using *Multimodal Semantic Simulations (MSS)*, which can be used to encode events as programs in a dynamic logic with an operational semantics. Computationally, we have been looking at event representation through sequential modeling, using data from 3-dimensional video captures, to distinguish between different event classes (Do and Pustejovsky 2017a). In this work, we aim to bridge the gap between these two lines of research by proposing a methodology to learn programmatic event representations from linguistic and visual event representations.

Linguistic event representation in our framework is modeled as a verbal subcategorization in a frame theory, a la Framenet (Baker, Fillmore, and Lowe 1998), with thematic role arguments. However, we also account for *extra-verbal factors* in our event type distinction. For example, we consider *A moves B toward C* and *A moves B around C* to be different event types and we learn each event type as a separate action.

Our visual event representation comprises visual features extracted from tracked objects in captured videos or virtual object positions saved from a simulation environment. Both types of feature represent information visible to humans and observable by a machine in an object state. Using these data points and sequences, machines can observe humans performing actions through processing captured and annotated videos, while humans can observe machines performing actions through watching simulated scenes.

Programmatic event representation can be based on formal event semantics or on features that can direct simulated or robotic agents to perform an action with an object of given properties. From a human perspective, the distinction between learning to recognize and learning to perform an action might be obvious. However from a machine's perspective, these two tasks might require different learning methods. Our work aims to demonstrate that given an appropriate framework, it is feasible to map between them, in a manner similar to the way humans actually learn: by matching actions to observations.

In this paper, (1) we discuss related work in AI that focuses on the learning of action and object models, including our own past studies; (2) we discuss several technologies and machine learning methodologies that provide the foundation for our experiments; (3) we discuss our ongoing experiment to learn actions; (4) we discuss our evaluation scheme and possible extensions to our framework.

Related Work

Work on action and object representation can generally be divided into two types of approaches: bottom-up approaches and top-down approaches.

Bottom-up approaches include both unsupervised and supervised feature-based learning. Work such as (Duckworth et al. 2016; Alomari et al. 2017) aims for unsupervised co-learning of object and event representations in the same step, and introduced the notion of a learned *concept* as an abstraction of feature spaces. In such a framework, "learnable" concepts are any distinctions meaningful to a human, such as a facial expression, color, object property, or action distinction, and these categories can then be assigned labels based on their commonly-occurring features. Notable supervised learning studies include (Koppula, Gupta, and Saxena 2013), which jointly models the human activities and object *affordances*, or attached behaviors which the object either facilitates by its geometry (which we term Gibsonian) (Gibson, Reed, and Jones 1982), or for which it is intended to be used (which we term "telic") (Pustejovsky 1995). Such a model could be used to distinguish longer activities by means of labeling sub-activities and object affordances: for example, labeling a "meal preparation" and its different subtasks based on understanding the objects involved at each step.

The foundation of our embodied event simulation is the modeling language known as VoxML (Visual Object Concept Modeling Language) (Pustejovsky and Krishnaswamy 2016). We encode verbal programs into a dynamic logic format from which we can conduct programmatic planning of complex events from atomic subevents. This is a top-down approach in which verbs are encoded with their subevent structures into programmatic "voxemes," or visual instantiations of lexemes which can then be visualized and enacted by an agent in a virtual environment. Subevent programs may themselves be linked to other voxemes, allowing for condition satisfaction, as in Figure 1, where "touching" is defined as the EC (externally connected) relation in RCC (Region Connection Calculus (Randell et al. 1992)). This is underspecified and may be further constrained by relative orientations between the two objects involved: x and y.

We aim to unify the two broad types of approaches outlined above using a form of *apprenticeship learning*, wherein a learning model observes an expert demonstrating the task that we want it to learn to perform. We propose a model, cf. (Abbeel and Ng 2004), in which reinforcement learning is used as a backbone for planning, while estimating a reward function as measuring the progression of the event-actions to be learned.

$$\begin{bmatrix} \textbf{touching} & \\ \text{LEX} = & \begin{bmatrix} \text{PRED} = \textbf{touching} \end{bmatrix} \\ \text{TYPE} = & \begin{bmatrix} \text{CLASS} = \textbf{config} \\ \text{VALUE} = \textbf{EC} \\ \text{ARGS} = \begin{bmatrix} \text{A}_1 = \textbf{x:3D} \\ \text{A}_2 = \textbf{y:3D} \end{bmatrix} \\ \text{CONSTR} = \textbf{nil} \end{bmatrix} \end{bmatrix}$$

Figure 1: Sample voxeme: [[TOUCHING]]

Background

Simulators

VoxSim Our simulated environment is built in **VoxSim** (Krishnaswamy and Pustejovsky 2016), a semantically-informed visual event simulator built on top of the Unity game engine (Goldstone 2009). VoxSim contains a 3D agent capable of manipulating objects in the virtual world by creating parent-child relationships between the objects and its joints to simulate grasping. Assuming the simulated agent's skeleton is isomorphic to the joint structure of a physical robot, this then allows us to simulate events in the 3D world that represent real-world events (such as moving the virtual robot around a virtual table that has blocks on it in a configuration that is generated from the positioning of real blocks on a real table). The embodied agent can perform a set of simple actions:

- $ENGAGE$: grasp object near its end-effector.
- $MOVE(x)$: move end-effector (hand) to 3D point x, with parent limb motions calculated using inverse kinematics
- $DISENGAGE$: ungrasp current object, and retract the agent to standing position.

The simulation environment is used to demonstrate the agent's understanding of learned behavior, by enacting new behaviors over a set of virtual objects. Scenes generated by VoxSim will be used to evaluate performance of the system, as discussed later.

Simplified Simulator For the updating loops in our reinforcement learning algorithm, we want to simulate observational data similar to the real captured data faster than real-time for effective computation. As a real-time, graphics heavy simulator, VoxSim is not feasible for this portion of the task. We are aware of a few other physical simulation environments such as Gazebo[1], but as we do not focus on physical constraints in this study, so we implemented our own simplified simulator in Python.

Our set of learnable actions is limited to ones that can be easily approximated in 2D space. 3D captured data is transformed into simplified simulator space by projecting it onto a 2D plane defined by the surface of the table used for

[1]http://gazebosim.org/

performing the captured interaction. Our 2D simulator has the following features:

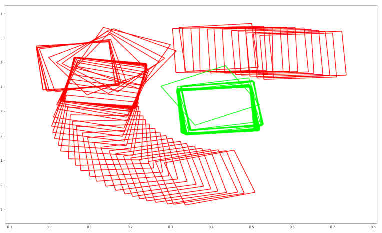

Figure 2: An event "Move A around B" projected into simulator. A is projected as a red square, B as a green square

- Each object is represented as a polygon (or square), with a $transform$ object that stores its position, rotation, and scale.
- The space is constrained so objects do not overlap.
- Speed can be specified so that object movement can be recorded as a sequence of feature vectors interpolated from frame to frame.

Qualitative Spatial Reasoning

Qualitative spatial reasoning (QSR), a sub-field of qualitative reasoning, is considered to be formally akin to the way humans understand geometry and space, due to the cognitive advantages of conceptual neighborhood relations and its ability to draw coarse inferences under uncertainty (Freksa 1992). It is also considered a promising framework in robotic planning (Cohn and Renz 2001). QSR allows formalization of many qualitative concepts, such as *near*, *toward*, *in*, *around*, and facilitates learning distinctions between them (Do and Pustejovsky 2017b). QSR has many methods of accounting for relative vs. absolute relations, such as allowing *near* to be thresholded relative to an existing reference point (Renz and Nebel 2007), which reinforces the intuition that predicates such as *near* are inherently relative (Peters 2007). The use of qualitative predicates ensure that scenes which are semantically close have very similar feature descriptions. We use the following QSR types for feature extraction.

- CARDINAL DIRECTION measures relations between two objects as compass directions (north, northeast, etc.)
- MOVING or STATIC measures whether a point is moving or not.
- QUALITATIVE DISTANCE CALCULUS discretizes the distance between two moving points, following (Yang and Webb 2009).
- QUALITATIVE TRAJECTORY CALCULUS is a representation of motions between two objects by considering them as two moving point objects (MPOs).

Event Annotation Framework

We use an event capture and annotation tool developed in our lab, ECAT (Do, Krishnaswamy, and Pustejovsky 2016), which employs Microsoft Kinect® to capture performers interacting with objects in a blocks world environment. Objects are tracked using markers fixed to their sides. They are then projected into three dimensional space using Depth of Field (DoF). Performers are also tracked using the Kinect® API, which provides three dimensional inputs of their joint points (e.g., wrist, palm, shoulder).

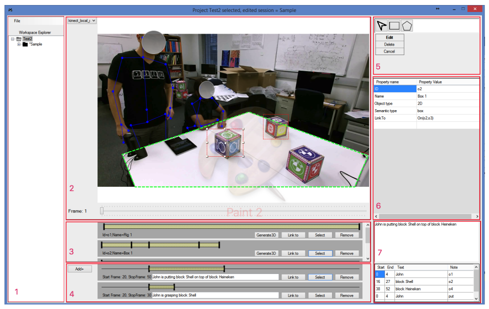

Figure 3: ECAT GUI showing performer interacting with recognized and annotated objects.

Learning Framework

Sequential Learning In this study, we consider a version of Long-short term memory (LSTM) (Hochreiter and Schmidhuber 1997) that processes sequential inputs to a sequence of output signals. LSTM has found utility in a range of problems involving sequential learning, such as speech and gesture recognition. Inputs are the feature vectors taken from action captures or from the simplified simulator and output is a function that corresponds to the progress of an event. In particular, we create a function that takes a sequence S of feature vectors, current frame i and action e: $f(S, i, e) = 0 \leq q_i \leq 1$

The training set of sequential captured data is passed through an LSTM network, which is fitted to predict a linear progressing function. At the start or outside of an event span, the network produces 0, whereas at the end, it produces 1.

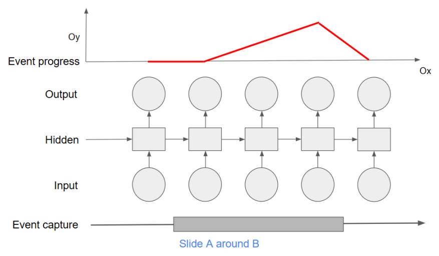

Figure 4: LSTM network producing event progress function

Reinforcement Learning The objective of the embodied agent is to generate a sequence of actions to attain a maximum reward, whereas our reward corresponds to how closely the produced object movement resembles movement of objects in the training data. Visual (tracked) information is used to evaluate performance of the system.

Currently, the action space is continuous. Therefore, planning is carried out by selecting the action at step k (u_k) based on the current state of the system ($X_k \in R^n$). A stochastic planning step is parameterized by policy parameters $\theta : u_k \sim \pi_\theta(u_k|x_k)$.

This type of parameterized reinforcement learning policies is best solved by using policy gradients (Gullapalli 1990; Peters and Schaal 2008). Here, we use the REINFORCE algorithm (Williams 1992), for its effectiveness in policy gradient learning.

We consider two versions of REINFORCE, which carry out planning in continuous and discrete search spaces, respectively. For continuous space, we propose using a Gaussian distribution policy $\pi_\theta(u|x) = Gaussian(\mu, \sigma)$. For simplicity, the dimensions of μ and σ are the same as the degrees of freedom in our simplified simulator (2 dimensions for position and 1 dimension for rotation). An artificial neural network (ANN) will be used to produce values μ and σ. The set of weights in our ANN is the parameter θ from the REINFORCE algorithm, learned with gradient descent.

For discrete space, we again use a qualitative reasoning method. Specifically, the searching space for the *transform* of the target location could be separated into two spaces, for (X, Y) coordinates and rotation r. The searching space for (X, Y) could be discretized according to cardinal direction and quantized distance.

A searching method employing simple random search with back-up is used as baseline to evaluate performance of the progress learner. We will present some preliminary results from this searching method.

Experiments

Here we describe our experimental setup and evaluation plans.

Experiment

We aim to use the learning framework outlined above for teaching an agent to perform a set of actions where it interacts directly with a single object while the other objects stay relatively static and the interaction takes place over a continuous span.

1. An agent moves {object A} **closer to** {object B}
2. An agent moves {object A} **away from** {object B}
3. An agent moves {object A} **past** {object B}
4. An agent moves {object A} **next to** {object B}
5. An agent moves {object A} **around** {object B}

This set of actions differ only in their prepositional adjuncts, which describe different motion trajectories. Thus for this experiment, the learning problem is reduced to one of motion paths.

These actions are, however, generally classified into different event types. Using the treatment from (Pustejovsky 1991), an action such as “moves {object A} **next to** {object B}” is an *achievement*, which means it has a logical culmination or duration. Other actions do not have a defined ending, though for “moves {object A} **closer to** {object B},” this action is ended at the point when “{object A} is **next to** {object B}.” From a cognitive point of view, recognition of these action types, except possibly for **move next to**, requires consideration of the trajectory as well as the start and ending points of the objects involved. For example, **closer to** conceptually involves change of distance between the start and the ending position of the moving object relative to the static object, but a complex motion path could lead to misinterpretation of the action. **Closer to**, therefore, strongly indicates a trajectory of the moving object toward the static object.

By grouping the learning of different event types together, we aim to examine the capability of a single learning framework that to learn multiple event types. The reason is rather obvious: we, as humans, can learn all of these actions without prior knowledge of different action types.

For each action type, we are capturing 40 sessions of two different performers. Block positions are randomized at the start. We mark the beginning and end of the captured action and give it a textual description.

We generate frame-by-frame feature vectors by employing the set of aforementioned QSR features: cardinal direction and qualitative distance between objects' positions and frame-to-frame difference; qualitative trajectory for each object and frame-to-frame difference. These features are used only for the sequential model to predict event progress, whereas we use objects' parameters (positions and rotations) across consecutive frames as state of the system X_k.

Evaluation

Human evaluation will be carried out on action demonstrations generated by both the 2D simulator and our lab's 3D visualizer, VoxSim (Figure 5). In VoxSim, we create a testbed scene with blocks on a table, similar to the setup used in video captures. For each randomized configuration of objects (block positions and rotations), we command the virtual agent to perform one of the actions, and the scene is recorded for evaluators to judge its performance.

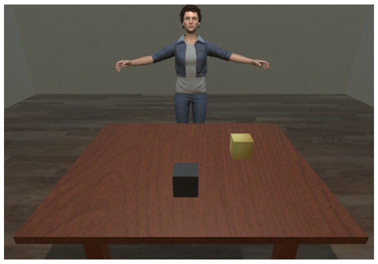
Figure 5: Visualizer implemented in Unity

Our human-driven evaluation method aims to help answer the following questions:

1. Does the virtual agent learn the concept in question? Reflected by average score given to a demonstration when annotators know the action label.
2. Can the virtual agent make distinctions between learned actions? Reflected by confusion matrix when annotators have to label the action performed in a scene.
3. Will evaluation scores on the 2D simulator significantly differ from those on the 3D visualizer?
4. Can we use the feedback from human evaluation to improve the learned model? Generated demonstrations with feedback scores complement real, captured data, and in some sense are better than learning by demonstration, in that they provide a rigorous way to include negative samples.

Evaluations of this type using VoxSim-generated scenes have already been conducted in (Krishnaswamy 2017; Krishnaswamy and Pustejovsky 2017), using Amazon Mechanical Turk to crowdsource judgments. Human judgments of a scene are given as "acceptable" or "unacceptable" relative to the event's linguistic description.

Preliminary results

Preliminary runs of the system with brute-force searching show that the progress learner can help to generate correct demonstrations (Fig. 6), but sometimes produces deviations (Fig. 7), probably because of the lack of negative training samples. We hope that incorporating feedback from evaluators will improve the overall performance of the learner.

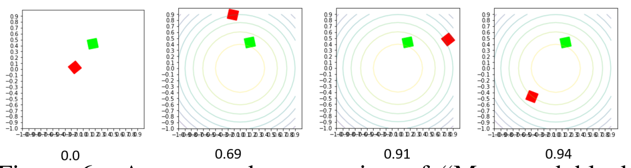

Figure 6: A correct demonstration of "Move red block around green block."

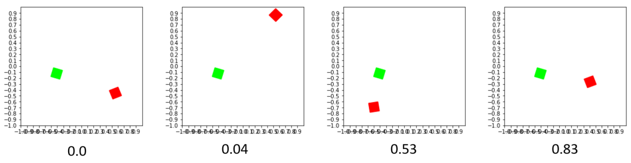

Figure 7: A wrong demonstration of "Move red block around green block." The value beneath each frame is value predicted by the progress learner.

We also provide a quantitative breakdown of a small-scale human evaluation in Table 1. Two annotators (college students) are asked to give scores from 0 to 10 and are also asked to give comments on any video they graded between 3 and 7 (higher scores are considered better). **Evaluator Disparity** is the average of the absolute values of the differences between scores given by two annotators over the demonstrations of a particular action.

Action Type	Average Score	Evaluator Disparity
Slide Closer	5.4	1.57
Slide Away	6.48	2.37
Slide Next To	5.55	1.7
Slide Past	6.38	1.9
Slide Around	2.75	1.03

Table 1: Evaluation

Evaluator comments provide some insight into bad demonstrations. Typical comments on *Slide Next To* include "Need to be even closer", while on *Slide Closer To* a typical comment is "The blocks touched." That suggests some confusion between these two actions, which requires a method to help distinguish them. Three reasons are given by evaluators for low scores on *Slide Around* demonstrations: the movement being not smooth, one or more additional steps needed for completion, and many cases where the algorithm does not generate the proper trajectory.

Code, experimental and evaluation results can be found on GitHub[2]. Complete experimental results will be forthcoming at that address.

[2]https://github.com/tuandnvn/learn-to-perform/

Conclusion

Two different lines of research may be extended from this framework. One involves a learning mechanism for more complex actions, such as "make a row from given objects," and one involves learning the "manner of motion" of actions.

Learning complex actions from simpler actions requires an additional semantic framework for objects and actions. For example, to learn "make a row from given objects" given observations of 2-unit and 3-unit rows, the learner needs to be equipped with the concept of *recursion*, the concept of a composite object made from elementary objects (e.g. the size and shape of the composite object), and other abstract concepts, such as object axis and extension of a structure along said axis.

Learning the manner aspect of actions requires a finer-grained treatment of object affordances. For example, for the learner to distinguish "rolling a bottle" and "sliding a bottle," we need to equip it with a reasoning mechanism to determine how an object's pose and position dictate its affordances. VoxML, the underlying platform to the VoxSim system, supports modeling these types of affordance distinctions, so reference to the VoxML semantics of objects and events can provide the reasoner with the mechanism for distinguishing these behavior types, as illustrated by (Krishnaswamy and Pustejovsky 2016).

Acknowledgements

We would like to thank the reviewers for their helpful comments. This work is supported by a contract with the US Defense Advanced Research Projects Agency (DARPA), Contract W911NF-15-C-0238. Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We would like to thank Alex Luu and Dr. Pengyu Hong for their discussion and input on this topic. All errors and mistakes are, of course, the responsibilities of the authors.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, 1. ACM.

Alomari, M.; Duckworth, P.; Bore, N.; Hawasly, M.; Hogg, D. C.; and Cohn, A. G. 2017. Grounding of human environments and activities for autonomous robots. In *IJCAI-17 Proceedings*.

Asada, M.; Uchibe, E.; and Hosoda, K. 1999. Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development. *Artificial Intelligence* 110(2):275–292.

Baker, C. F.; Fillmore, C. J.; and Lowe, J. B. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguis-*

tics and 17th International Conference on Computational Linguistics-Volume 1, 86–90. Association for Computational Linguistics.

Bogue, R. 2017. Domestic robots: Has their time finally come? *Industrial Robot: An International Journal* 44(2):129–136.

Cohn, A. G., and Renz, J. 2001. Qualitative spatial representation and reasoning. 46:1–2.

Do, T., and Pustejovsky, J. 2017a. Fine-grained event learning of human-object interaction with lstm-crf. *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*.

Do, T., and Pustejovsky, J. 2017b. Learning event representation: As sparse as possible, but not sparser. *arXiv preprint arXiv:1710.00448*.

Do, T.; Krishnaswamy, N.; and Pustejovsky, J. 2016. Ecat: Event capture annotation tool. *Proceedings of ISA-12: International Workshop on Semantic Annotation*.

Duckworth, P.; Alomari, M.; Gatsoulis, Y.; Hogg, D. C.; and Cohn, A. G. 2016. Unsupervised activity recognition using latent semantic analysis on a mobile robot. In *IOS Press Proceedings*, number 285, 1062–1070.

Freksa, C. 1992. *Using orientation information for qualitative spatial reasoning*. Springer.

Gergely, G.; Bekkering, H.; and Király, I. 2002. Developmental psychology: Rational imitation in preverbal infants. *Nature* 415(6873):755.

Gibson, J. J.; Reed, E. S.; and Jones, R. 1982. *Reasons for realism: Selected essays of James J. Gibson*. Lawrence Erlbaum Associates.

Goldstone, W. 2009. *Unity Game Development Essentials*. Packt Publishing Ltd.

Gullapalli, V. 1990. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural networks* 3(6):671–692.

Hawes, N.; Burbridge, C.; Jovan, F.; Kunze, L.; Lacerda, B.; Mudrová, L.; Young, J.; Wyatt, J.; Hebesberger, D.; Kortner, T.; et al. 2017. The strands project: Long-term autonomy in everyday environments. *IEEE Robotics & Automation Magazine* 24(3):146–156.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Koppula, H. S.; Gupta, R.; and Saxena, A. 2013. Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research* 32(8):951–970.

Krishnaswamy, N., and Pustejovsky, J. 2016. Multimodal semantic simulations of linguistically underspecified motion events. In *Spatial Cognition X: International Conference on Spatial Cognition*. Springer.

Krishnaswamy, N., and Pustejovsky, J. 2017. Do you see what I see? effects of pov on spatial relation specifications. In *Proc. 30th International Workshop on Qualitative Reasoning*.

Krishnaswamy, N. 2017. *Monte-Carlo Simulation Generation Through Operationalization of Spatial Primitives*. Ph.D. Dissertation, Brandeis University.

Paul, R.; Arkin, J.; Roy, N.; and Howard, T. 2017. Grounding abstract spatial concepts for language interaction with robots. In *IJCAI-17 Proceedings*.

Peters, J., and Schaal, S. 2008. Reinforcement learning of motor skills with policy gradients. *Neural networks* 21(4):682–697.

Peters, J. F. 2007. Near sets. Special theory about nearness of objects. *Fundamenta Informaticae* 75(1-4):407–433.

Pustejovsky, J., and Krishnaswamy, N. 2016. VoxML: A visualization modeling language. In Chair), N. C. C.; Choukri, K.; Declerck, T.; Goggi, S.; Grobelnik, M.; Maegaard, B.; Mariani, J.; Mazo, H.; Moreno, A.; Odijk, J.; and Piperidis, S., eds., *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Paris, France: European Language Resources Association (ELRA).

Pustejovsky, J., and Moszkowicz, J. 2011. The qualitative spatial dynamics of motion. *The Journal of Spatial Cognition and Computation*.

Pustejovsky, J.; Krishnaswamy, N.; and Do, T. 2017. Object embodiment in a multimodal simulation. *AAAI Spring Symposium: Interactive Multisensory Object Perception for Embodied Agents*.

Pustejovsky, J. 1991. The syntax of event structure. *Cognition* 41(1):47–81.

Pustejovsky, J. 1995. *The Generative Lexicon*. Cambridge, MA: MIT Press.

Randell, D.; Cui, Z.; Cohn, A.; Nebel, B.; Rich, C.; and Swartout, W. 1992. A spatial logic based on regions and connection. In *KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, 165–176. San Mateo: Morgan Kaufmann.

Renz, J., and Nebel, B. 2007. Qualitative spatial reasoning using constraint calculi. In *Handbook of spatial logics*. Springer. 161–215.

Smart, W. D., and Kaelbling, L. P. 2002. Effective reinforcement learning for mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 4, 3404–3410. IEEE.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Yang, Y., and Webb, G. I. 2009. Discretization for naive-bayes learning: managing discretization bias and variance. *Machine learning* 74(1):39–74.

Young, J., and Hawes, N. 2015. Learning by observation using qualitative spatial relations. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 745–751. International Foundation for Autonomous Agents and Multiagent Systems.