VoxSim: A Visual Platform for Modeling Motion Language

Nikhil Krishnaswamy
Brandeis University
415 South Street
Waltham, MA 02453 USA
nkrishna@brandeis.edu

James Pustejovsky
Brandeis University
415 South Street
Waltham, MA 02453 USA
jamesp@brandeis.edu

Abstract

Much existing work in text-to-scene generation focuses on generating static scenes. By introducing a focus on motion verbs, we integrate dynamic semantics into a rich formal model of events to generate animations in real time that correlate with human conceptions of the event described. This paper presents a working system that generates these animated scenes over a test set, discussing challenges encountered and describing the solutions implemented.

1 Introduction

The expressiveness of natural language is difficult to translate into visuals, and much work in text-to-scene generation has focused on creating static images, e.g., Coyne and Sproat (2001) and Chang et. al (2015). Our approach centers on motion verbs, using a rich formal model of events and mapping from an NL expression, through Dynamic Interval Temporal Logic (Pustejovsky and Moszkowicz, 2011), into a 3D animated simulation. Previously, we introduced a method for modeling motion language predicates in three dimensions (Pustejovsky and Krishnaswamy, 2014). This led to VoxML, a modeling language to encode composable semantic knowledge about NL entities (Pustejovsky and Krishnaswamy, 2016), and a reasoner to generate simulations involving novel objects and events (Krishnaswamy and Pustejovsky, 2016). Our system, **VoxSim**, uses object and event semantic knowledge to generate animated scenes in real time without a complex animation interface. The latest stable build of VoxSim is available at `http://www.voxicon.net`. The Unity project and source is at `https://github.com/nkrishnaswamy/voxicon`.

2 Theoretical Motivations

Dynamic interpretations of event structures divide motion verbs into "path" and "manner of motion" verbs. Path verbs reassign the moving argument's position relative to a specified location; for manner verbs, position is specified through prepositional adjunct. Thus, *The spoon falls* and *The spoon falls into the cup* result in different "mental instantiations," or "simulations" (Bergen, 2012). In order to visualize events, a computational system must infer path or manner information from the objects involved or from their composition with the predicate.

Visual instantiations of lexemes, or "voxemes" (Pustejovsky and Krishnaswamy, 2016), require an encoding of their situational context, or a *habitat* (Pustejovsky, 2013; McDonald and Pustejovsky, 2014), as well as afforded behaviors that the object can participate in, that are either *Gibsonian* or *telic* in nature (Gibson, 1977; Gibson, 1979; Pustejovsky, 1995). For instance, a cup may afford containing another object, or being drunk from. Many event descriptions presuppose such conditions that rarely appear in linguistic data, but a visualization lacking them will make little sense to the observer. This linguistic "dark matter," conspicuous by its absence, is thus easily exposable through simulation.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pages 54–58, Osaka, Japan, December 11-17 2016.

3 Architecture

VoxSim uses the Unity game engine (Goldstone, 2009) for graphics and I/O processing. Input is a simple natural language sentence, which is part-of-speech tagged, dependency-parsed, and transformed into a simple predicate-logic format. These NLP tasks may be handled with a variety of third-party tools, such as the ClearNLP parser (Choi and McCallum, 2013), SyntaxNet (Andor et al., 2016), or TRIPS (Ferguson et al., 1998), which interface with the simulation software using a C++ communications bridge and wrapper. 3D assets and VoxML-modeled entities are loaded externally, either locally or from a web server. Commands to the simulator may be input directly to the software UI, or may be sent over a generic network connection or using **VoxSim Commander**, a companion iOS app.

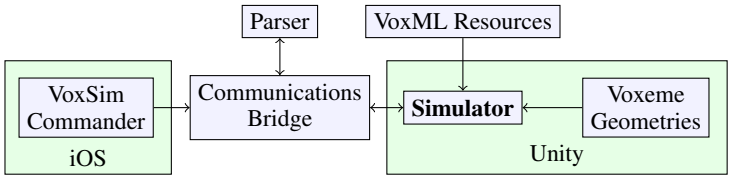

Figure 1: VoxSim architecture schematic

3.1 Processing Pipeline

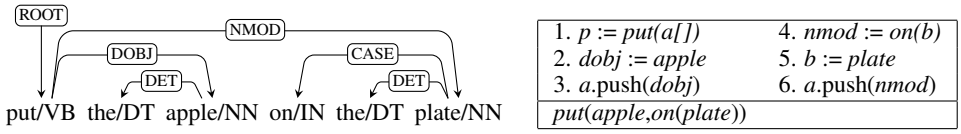

Figure 2: Dependency parse for *Put the apple on the plate* and transformation to predicate-logic form.

Given a tagged and dependency parsed sentence, we can transform it into predicate-logic format using the root of the parse as the VoxML PROGRAM, which accepts as many arguments as are specified in its type structure, and subsequently enqueuing arguments that are either constants (i.e. VoxML OBJECTs) or evaluate to constants at runtime (all other VoxML entity types). Other non-constant VoxML entity types are treated similarly, though usually accept only one argument.

4 Semantic Processing and Compositionality

Rather than relying on manually-specified objects with identifying language, we instead procedurally compose voxemes' VoxML properties in parallel with their linked lexemes.

A VoxML entity's interpretation at runtime depends on the other entities it is composed with. A cup on a surface, with its opening upward, may afford containing another object, so to place an object $in(cup)$, the system must first determine if the intended containing object (i.e., the cup) affords containment by default by examining its affordance structure (figure 3).

If so, the object must be currently situated in a habitat which allows objects to be placed partially or completely inside it (represented by RCC relations PO, TPP, or NTPP, as shown in the VoxML for in). cup's VoxML TYPE shows a concave object with rotational symmetry around the Y-axis and reflectional symmetry across the XY and YZ planes, meaning that it opens along the Y-axis. Its HABITAT further situates the opening along its positive Y-axis, meaning that if the cup's opening along its +Y is currently unobstructed, it affords containment. Previously established habitats, i.e., "The cup is flipped over," may activate or deactivate these and other affordances.

Finally, the system must check to see if the object to be contained can fit in the containing object in its current configuration. If so, it is moved into position. If not, the system attempts to rotate the contained object into an orientation where it will fit inside the container. If it can, the object is rotated into that orientation and then moved. If no such orientation exists, the system returns a message stating that the requested action is impossible to perform.

$$\left[\begin{array}{l}\textbf{put}\\ \text{TYPE} = \left[\begin{array}{l}\text{HEAD} = \textbf{process}\\ \text{ARGS} = \left[\begin{array}{l}\text{A}_1 = \textbf{agent}\\ \text{A}_2 = \textbf{physobj}\\ \text{A}_3 = \textbf{location}\end{array}\right]\\ \text{BODY} = \left[\begin{array}{l}\text{E}_1 = grasp(A_1, A_2)\\ \text{E}_2 = [while(hold(A_1, A_2),\\ \quad move(A_2))]\\ \text{E}_3 = [at(A_1, A_3) \rightarrow\\ \quad ungrasp(A_1, A_2)\end{array}\right]\end{array}\right]\end{array}\right]$$

$$\left[\begin{array}{l}\textbf{in}\\ \text{TYPE} = \left[\begin{array}{l}\text{CLASS} = \textbf{config}\\ \text{VALUE} = \textbf{ProperPart} \parallel \textbf{PO}\\ \text{ARGS} = \left[\begin{array}{l}\text{A}_1 = \textbf{x:3D}\\ \text{A}_2 = \textbf{y:3D}\end{array}\right]\end{array}\right]\end{array}\right]$$

$$\left[\begin{array}{l}\textbf{cup}\\ \text{TYPE} = \left[\begin{array}{l}\text{HEAD} = \textbf{cylindroid[1]}\\ \text{COMPONENTS} = \textbf{surface,interior}\\ \text{CONCAVITY} = \textbf{concave}\\ \text{ROTATSYM} = \{Y\}\\ \text{REFLECTSYM} = \{XY, YZ\}\end{array}\right]\\ \text{HABITAT} = \left[\begin{array}{l}\text{INTR} = [2]\left[\begin{array}{l}\text{UP} = align(Y, \mathcal{E}_Y)\\ \text{TOP} = top(+Y)\end{array}\right]\end{array}\right]\\ \text{AFFORD\_STR} = \left[\begin{array}{l}\text{A}_1 = H[2] \rightarrow\\ \quad [put(x, on([1]))]\\ \quad support([1], x)\\ \text{A}_2 = H[2] \rightarrow\\ \quad [put(x, in([1]))]\\ \quad contain([1], x)\\ \text{A}_3 = H[2] \rightarrow\\ \quad [grasp(x, [1])]\end{array}\right]\end{array}\right]$$

Figure 3: VoxML typing, habitats, and affordances for "put", "in", and "cup"

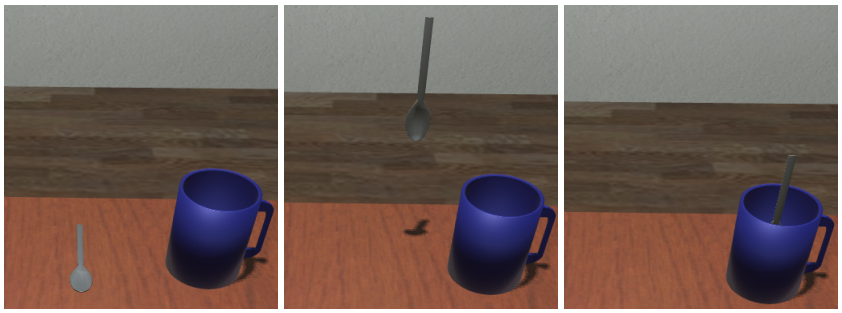

Figure 4: Execution of "put the spoon in the mug"

Currently VoxSim implements RCC relations (Randell et al., 1992; Galton, 2000; Albath et al., 2010), but can be extended to other QSR approaches, including the situation calculus (Bhatt and Loke, 2008), and Intersection Calculus (Kurata and Egenhofer, 2007).

We augment this approach with an embodied agent that simultaneously enacts the same program as the manipulated object, composing the object motion ("object model") and the agent motion ("action model") into a single "event model," allowing for both agent-free and agent-driven actions.

Figure 5: Execution of "put the apple on the plate" using embodied agent

Once all parameters requiring specification have values assigned to them, VoxSim executes the program over its arguments, rendering the visual result each frame, which provides a trace of the event from beginning to end. Note that the precise running time of the generated animation is variable and dependent on the values calculated for the aforementioned parameters, including the total distance an object must move from its starting position to its target, any preconditions that must be fulfilled before the commanded event can be executed, and others.

5 Conclusions

VoxSim provides a method not only for generating 3D visualizations using an intuitive natural language interface instead of specialized skillsets (a primary goal of programs such as WordsEye), but also a platform on which researchers may conduct experiments on the discrete observables of motion events while evaluating semantic theories, thus providing data to back up theoretical intuitions. Visual simulation provides an intuitive way to trace spatial cues' entailments through a narrative, enabling broader study of event and motion semantics.

VoxSim currently handles an expanding lexicon of voxemes (a "voxicon"), with many primitive objects and behaviors encoded in VoxML and available for composition into macro-entities. The current voxicon status is given in table 1. No distinction is made here between primitive and macro-entities.

Objects (18)	**Programs** (17)	**Relations** (6)	**Functions** (12)
block	grasp	on	edge
ball	hold	in	center
plate	touch	against	top
cup	move	at	bottom
disc	turn	support	back
spoon	roll	containment	front
book	slide		left
blackboard	spin		right
bottle	lift		corner
grape	stack		diagonal
apple	put		above
banana	lean		below
table	flip		
bowl	close		
knife	open		
pencil	reach		
paper sheet	push		
box			

Table 1: Current voxicon contents

Scene visualization work is not well-reflected in current evaluation, due to sparsity of datasets and lack of a general-domain gold standard(Johansson et al., 2005), so we are developing two human-driven evaluation methods, augmented by an automatic method. Human evaluation asks subjects to make a pairwise similarity judgement over a generated simulation and a set of possible labels, going both ways (i.e. a judgement on one simulation to many labels and on one label to many simulations). Automatic evaluation measures the vector distance from a generated simulation to a preconceived "prototype" of the input event descriptor. The results of these experiments are currently being evaluated.

We are also planning on building links to lexical semantic resources such as VerbNet (Kipper et al., 2006) to allow us to leverage existing datasets for macro-program composition, and to expand the semantic processing to event sequences, to simulate narratives beyond the sentence level.

Acknowledgements

We would like to thank the reviewers for their insightful comments. This work was supported by Contract W911NF-15-C-0238 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO). Approved for Public Release, Distribution Unlimited. The views expressed herein are ours and do not reflect the official policy or position of the Department of Defense or the U.S. Government. All errors and mistakes are, of course, the responsibilities of the authors.

References

Julia Albath, Jennifer L. Leopold, Chaman L. Sabharwal, and Anne M. Maglia. 2010. RCC-3D: Qualitative spatial reasoning in 3D. In *CAINE*, pages 74–79.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*.

Benjamin K. Bergen. 2012. *Louder than words: The new science of how the mind makes meaning*. Basic Books.

Mehul Bhatt and Seng Loke. 2008. Modelling dynamic spatial systems in the situation calculus. *Spatial Cognition and Computation*.

Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D. Manning. 2015. Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289*.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *ACL (1)*, pages 1052–1062.

Bob Coyne and Richard Sproat. 2001. Wordseye: an automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 487–496. ACM.

George Ferguson, James F Allen, et al. 1998. Trips: An integrated intelligent problem-solving assistant. In *AAAI/IAAI*, pages 567–572.

Antony Galton. 2000. *Qualitative Spatial Change*. Oxford University Press, Oxford.

J. J. Gibson. 1977. The theory of affordances. *Perceiving, Acting, and Knowing: Toward an ecological psychology*, pages 67–82.

J. J. Gibson. 1979. *The Ecology Approach to Visual Perception: Classic Edition*. Psychology Press.

Will Goldstone. 2009. *Unity Game Development Essentials*. Packt Publishing Ltd.

Richard Johansson, Anders Berglund, Magnus Danielsson, and Pierre Nugues. 2005. Automatic text-to-scene conversion in the traffic accident domain. In *IJCAI*, volume 5, pages 1073–1078.

Kara Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2006. Extensive classifications of english verbs. In *Proceedings of the 12th EURALEX International Congress*, Turin, Italy.

Nikhil Krishnaswamy and James Pustejovsky. 2016. Multimodal semantic simulations of linguistically underspecified motion events. *Proceedings of Spatial Cognition*.

Yohei Kurata and Max Egenhofer. 2007. The 9+ intersection for topological relations between a directed line segment and a region. In B. Gottfried, editor, *Workshop on Behaviour and Monitoring Interpretation*, pages 62–76, Germany, September.

David McDonald and James Pustejovsky. 2014. On the representation of inferences and their lexicalization. In *Advances in Cognitive Systems*, volume 3.

James Pustejovsky and Nikhil Krishnaswamy. 2014. Generating simulations of motion events from verbal descriptions. *Lexical and Computational Semantics (* SEM 2014)*, page 99.

James Pustejovsky and Nikhil Krishnaswamy. 2016. VoxML: A visualization modeling language. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, May. European Language Resources Association (ELRA).

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

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

James Pustejovsky. 2013. Dynamic event structure and habitat theory. In *Proceedings of the 6th International Conference on Generative Approaches to the Lexicon (GL2013)*, pages 1–10. ACL.

David Randell, Zhan Cui, and Anthony Cohn. 1992. A spatial logic based on regions and connections. In Morgan Kaufmann, editor, *Proceedings of the 3rd Internation Conference on Knowledge Representation and Reasoning*, pages 165–176, San Mateo.