Comparing Inference Algorithms for Grounding Trajectories

Miles Eldon
miles_eldon@brown.edu
Kurt Spindler
kurt_spindler@brown.edu
Brown University
Providence, RI 02912

Abstract

One challenge in robotics is the speed at which a robotic agent can accurately respond to human commands to perform a complex task. We examine the Generalized Grounding Graph (G3) framework, which maps spoken commands to trajectories that satisfy the command. This system currently runs an exhaustive search over the given trajectory space using to find the trajectory that best matches the given preposition. We demonstrate an improvement to the Rapidly-Exploring Random Trees (RRT) algorithm that allows us to achieve near instantaneous results in most trajectory scenarios. Additionally, we demonstrate that the Random Forest learning algorithm greatly reduces overfitting in this domain, allowing us to achieve much more legible trajectories.

Introduction

We are reaching a stage where autonomous agents can successfully complete varied and more complex tasks. Robots can fold towels, navigate mazes, and even help humans in baking [1] [7]. With these advances, we also hope for the most natural interface to allow these autonomous agents to be controlled by humans: voice commands and natural language understanding. The Generalized Grounding Graph (G3) framework [2] is a recent framework that provides a very robust framework for language understanding, one that is resilient to variation in language choices. The system undergoes supervised learning in order to learn a mapping from a natural language sentence to an action sequence, as represented by a feature-space. At inference time, the system is given a language command and infers a most probable action sequence. One of the major drawbacks to any such system is that to be truly effective within the human robot interaction domain is that it must operate as close to instantaneously as possible. G3 has a minimum command processing speed of approximately ten seconds, which would be annoying, if not inapplicable to many day to day scenarios.

We began our research with this framework and sought to to answer two questions. First, can we improve on the exhaustive search of the trajectory space that the G3 framework uses to discover the maximum likelihood trajectory? Second, can we improve the accuracy of the trajectories produced by modifying both the features and algorithm used for learning the desired mapping? Doing this would allow the G3 system to operate more accurately and with speeds much closer to real time,

Related Work

The G3 framework, learns a mapping from commands to trajectories. It operates on an annotated training set containing graphical models of three dimensional trajectories paired with the command they satisfy. These trajectories are featurized and then trained using logistic regression.

Among search algorithms, we first considered simple hill climbing search. Unfortunately, our objective function is not convex, meaning that this method would rapidly become stuck at a local optima. While we considered heuristic-based search, it seems any heuristic would be sufficiently domain-specific as to impact the generality of the G3 system. The final method we implement is Rapidly-Exploring Random Trees (RRTs), a random search method with a bias to explore away from already explored nodes. [3]

Methodology

The next question to address is in what order paths should be evaluated. We compare three methodologies: an exhaustive search as baseline, Thresholded Rapidly-exploring Random Trees (TRRT), and Gradient Thresholded Rapidly-exploring Random Trees (GTRRT). We now explain the latter two algorithms.

The RRT algorithm begins from a starting point, and weights each neighbor of an explored state by the squared percentage of unexplored neighbors. These weights are then normalized into a probability distribution, and a neighbor to evaluate is selected according to this distribution. Our RRT search terminates when it finds a point with an associated path that the path evaluator assigns a probability above a certain threshold τ. From this point, we use local gradient ascent search to find a local optimum. For our tests, we used $\tau = 0.7$.

We improve the RRT algorithm to be biased by gradient, hence the GTRRT algorithm. Before normalization, the node weight is multiplied by the maximum likelihood of the state's neighbors. The weights are normalized and neighbors selected according to the RRT schema.

Lastly, we assumed that no gradient would be so steep that it's neighbor would not be relatively close to it. Therefore, we enforce that the RRT will only search along the diagonals of our xy-plane, effectively cutting our search space in half. While TRRTs generally exhibit behavior similar to this probabilistically, they will search the space exhaustively if we do not reach the threshold, so we enforce this sparsity as an invariant.

In the learning domain, we compare two different learning algorithms: logistic regression and random forests. As mentioned, these algorithms are both trained in a supervised fashion using the dataset from [6]. At training time, they are provided prepositional phrases and associated paths, as well as true and false labels whether those paths are accurate. Logistic regression is the result of a sigmoid function applied to the dot product between a learned weights vector and the values of a list of features. Random Forests are a collection of n decision trees. These are trained on the training data by taking random subsets of the data and continuously dividing the subset according to the Gini importance criterion (a measure of entropy). Probability of correctness of the path is then the average classification of all trees. We use $n = 100$ and $n = 360$. The logistic regression is implemented as a single-node (degenerate) conditional random field using the MALLET Java package [4]. The random forest is implemented using the scikit-learn Python library [5].

Our prepositional phrases are of the form "<preposition> the square", with <preposition> taking the values "to", "towards", "past", "through", "down", "around", and "along". The state space is an x-y plane ranging from $x \in [-5, 20]$ and $y \in [-10, 10]$. The robotic agent is a square with side length 2 centered at $(0,0)$. The "square" is a rectangle between coordinates $(1, 9)$ and $(-1, 10)$.

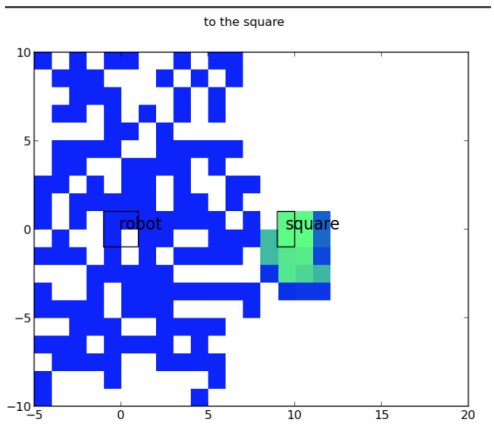
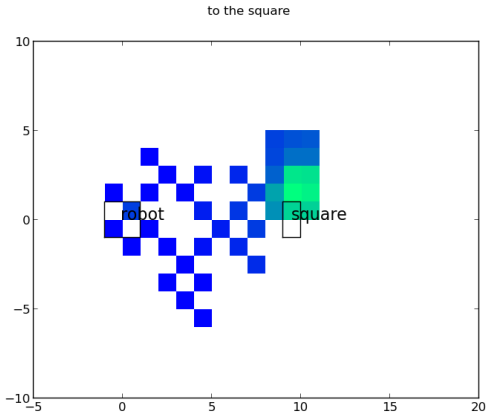
Results

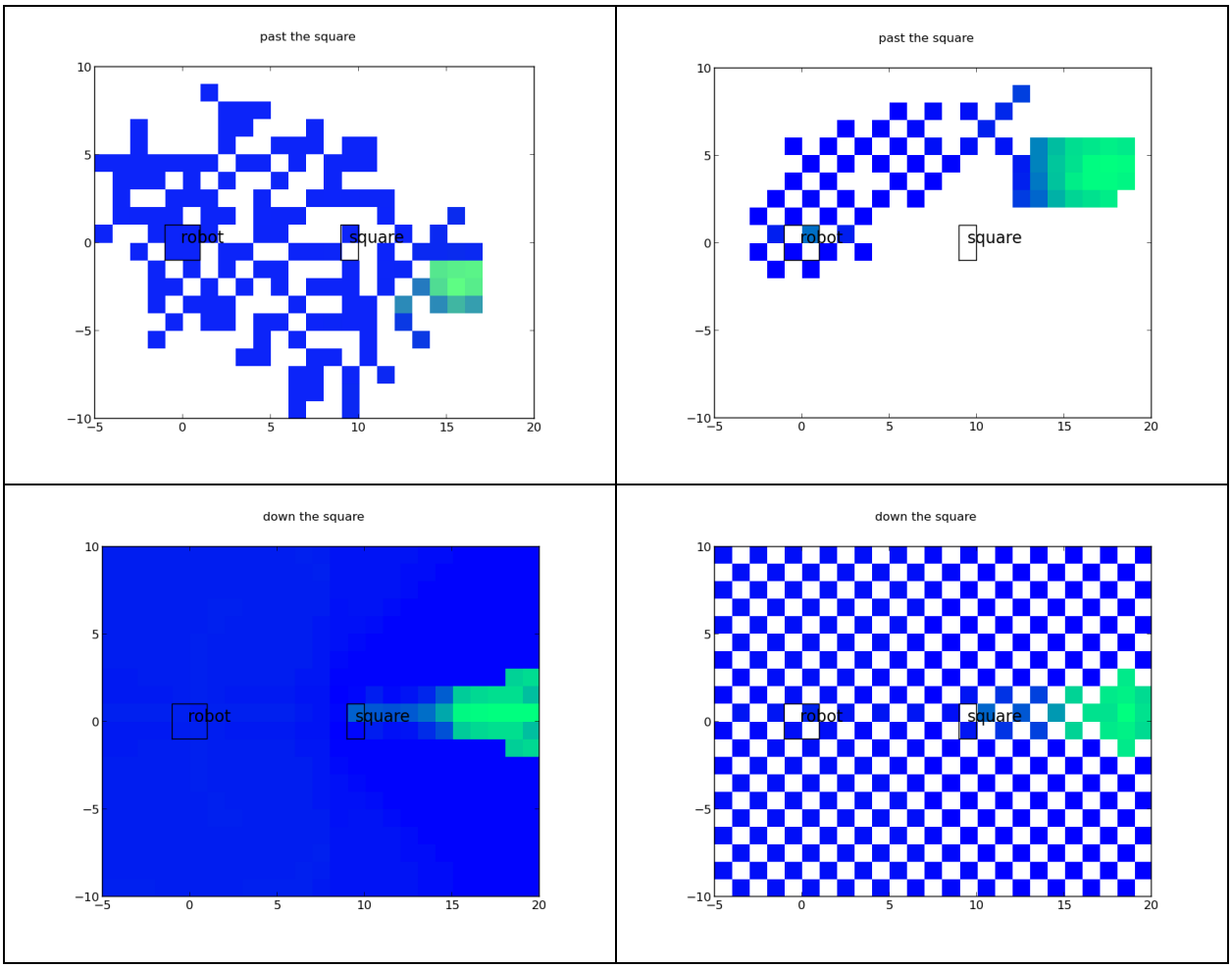
The initial, exhaustive baseline took approximately 10.5 seconds to complete on all propositions, yielding the optimal result with 100% accuracy.

The TRRT search, with a threshold of 0.7, took slightly under 1.5 seconds for the preposition “to”, a little over 1 second for “towards”, and approximately 2.5 seconds for “past”, a 76-91% speedup. On these three tests, the TRRT method generally achieved half of the ten most likely trajectories. On all other propositions, the threshold was higher than any likelihood, and the RRT modeled an exhaustive search, matching the results of the exhaustive baseline.

The GTRRT method, again with a threshold of 0.7, took approximately 0.6 seconds for “to”, 0.5 seconds for “towards, and just under 1.3 seconds for “past”, while achieving the same level of optimality in results. The GTRRT method manages to match prepositions with a near optimal trajectory in near instantaneous time, 88-95% speedup, a vast improvement over the exhaustive baseline. It also cut the time for the exhaustive baseline in half by only examining states along the diagonals due to the weak assumption discussed earlier.

We demonstrate various heatmaps where the prepositional phrase is displayed as the title, the robot’s and landmark’s positions are outlined, and the color at a particular location is the probability of correctness of a straight-line path going from the robot’s position to the coordinate in the map.

TRRT $\tau = 0.7$	GTRRT $\tau = 0.7$
	

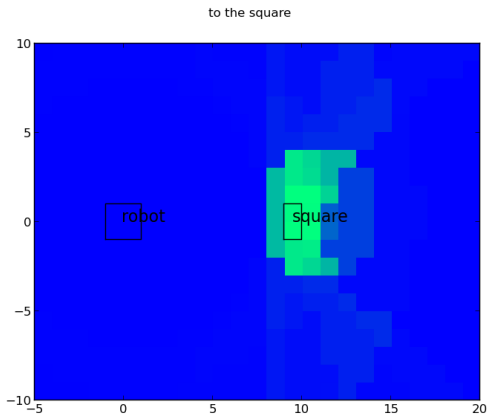
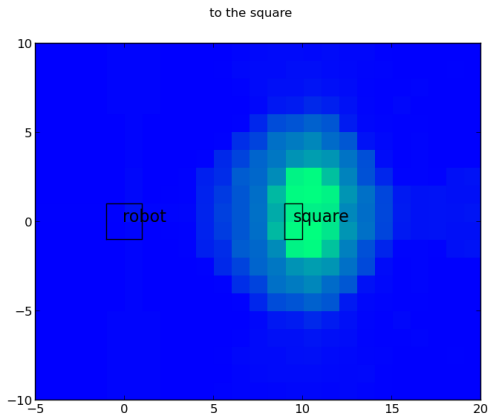
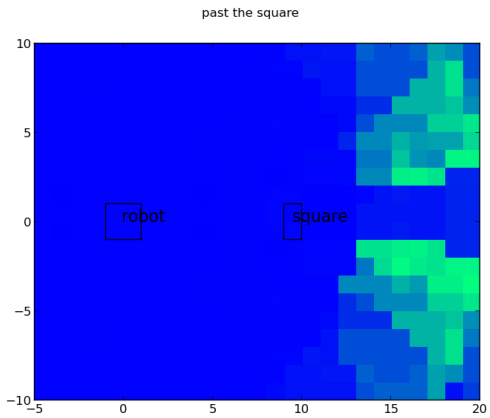
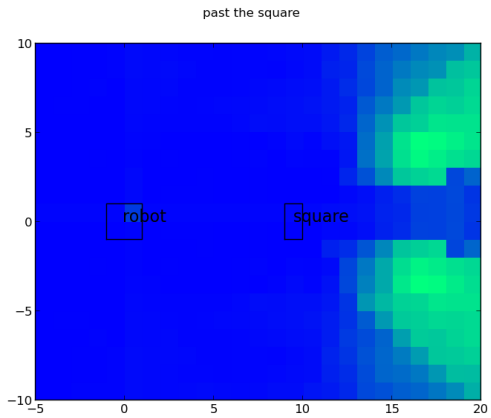
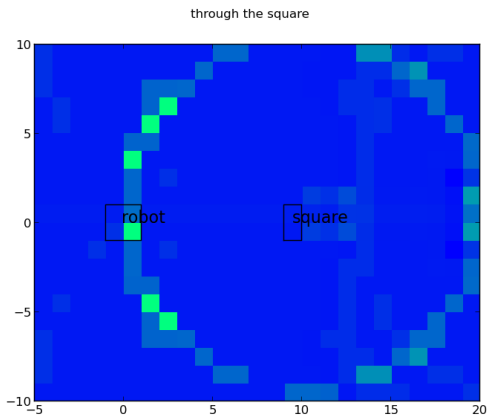
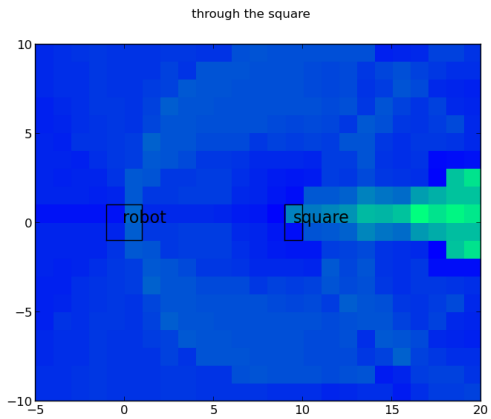


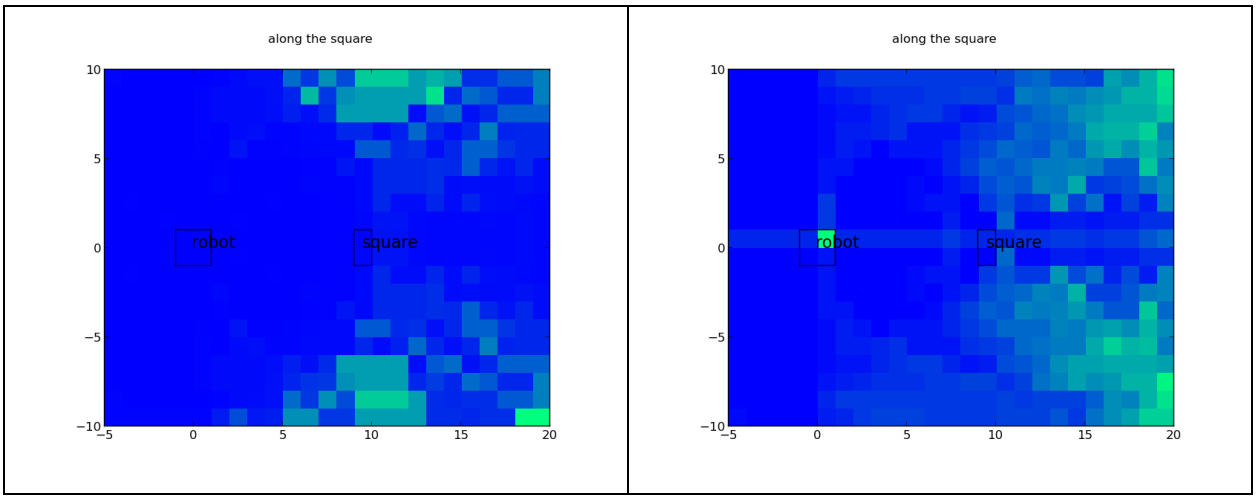
We envision this algorithm could be further improved with a degrading threshold. The threshold would begin at 1 and decrease based on the the magnitude of the likelihoods of the states evaluated. While tuning this threshold degradation ratio would be difficult, we theorize that it would improve both scalability and effectiveness of the GTRRT method.

We also compare the learned representation of the logistic regression model and the random forest model. Since we hope to learn the intuitive meaning that humans assign to prepositions, we have no numeric metric of the accuracy of these results. Furthermore, we did not give the results to test subjects to rate their effectiveness. Nonetheless, we report our results in the images below. Significantly, the “to” and “past” plots of the random forest are much smoother than their logistic regression counterparts, which improves the performance of gradient-based methods such as our GTRRT method. We report the greatest improvement on the “through” preposition. Despite some noise in the heatmap, the optimal paths clearly lead the robot “through” the square.

The random forest is not a silver bullet, however, as the “along” preposition demonstrates. We posit that improved training data or a more complex search space which searched over curved paths would further improve these results. While we report n=360 below, the model shows marginal, not dramatic, improvements above approximately n=100 trees. We

also note the issue of the high-value point of remaining stationary in the “along” random forest heatmap. We are currently looking to address this issue, but believe it to be a bug in the code.

Logistic Regression	Random Forest (n=360)
	
	
	



Contributions

Our research yielded a new search algorithm ideal for rapidly exploring costly and large spaces. GTRRTs can easily be extended to any domain with somewhat convex nature, quickly and effectively eliminating the need for complex heuristics, exhaustive searches, or large computations. By adding the additional degrading threshold discussed previously GTRRTs could be made robust and generic to varying scenarios and uses with little to no modification.

Furthermore, our research showed that Random Forests generate a better, less overfitted model for mapping language to trajectories, allowing for improvement in the ability for robotic agents to accurately follow spoken commands, driving the goal of fast and seamless human-robot interaction.

Conclusion

We demonstrate two improvements to the G3 framework. The first is a search algorithm which rapidly explores the space in order to find areas of “high heat” and then performs local search to optimize it to find a path of high likelihood, achieving near optimal results in all cases considered. The second is the usage of random forests to improve the learned understanding of prepositions, which are demonstrated to produce much smoother gradients and in general better represent the semantic meaning of the prepositions. We hope that the search algorithms could be applied to real-world robotic systems to dramatically speed their rate of inference and allow their search of more complex search spaces, and we hope the models could be used to improve the semantic representations robots use in the understanding of natural language commands, enabling them to better understand commands and dialogue with human interlocutors.

References

[1] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei and Pieter Abbeel. Cloth Grasp Point Detection based on Multiple-View Geometric Cues with Application to Robotic Towel Folding.

[2] Thomas Kollar, Stefanie Tellex, Matthew R. Walter, Albert Huang, Abraham Bachrach, Sachi Hemachandra, Emma Brunskill, Ashis Banerjee, Deb Roy, Seth Teller, Nicholas Roy. (2013). Generalized Grounding Graphs: A Probabilistic Framework for Understanding Grounded

Language. Journal of Artificial Intelligence Research.
[3] Steven M. Lavalle. 2006. Planning Algorithms, pp. 228-237.
[4] McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu. 2002.
[5] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[6] Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S., and Roy, N. (2011b). Approaching the symbol grounding problem with probabilistic graphical models. AI Magazine, 32(4):64–76.
[7] Bollini, Mario, Jennifer Barry, and Daniela Rus. "Bakebot: Baking cookies with the pr2." The PR2 Workshop: Results, Challenges and Lessons Learned in Advancing Robots with a Common Platform, IROS. 2011.