Deep compositional robotic planners that follow natural language commands

Yen-Ling Kuo
ylkuo@mit.edu
CSAIL, MIT

Boris Katz
boris@mit.edu
CSAIL, MIT

Andrei Barbu
abarbu@mit.edu
CSAIL, MIT

Abstract

We demonstrate how a sampling-based robotic planner can be augmented to learn to understand a sequence of natural language commands in a continuous configuration space to move and manipulate objects. Our approach combines a deep network structured according to the parse of a complex command that includes objects, verbs, spatial relations, and attributes, with a sampling-based planner, RRT. A recurrent hierarchical deep network controls how the planner explores the environment, determines when a planned path is likely to achieve a goal, and estimates the confidence of each move to trade off exploitation and exploration between the network and the planner. Planners are designed to have near-optimal behavior when information about the task is missing, while networks learn to exploit observations that are available from the environment, making the two naturally complementary. Combining the two enables generalization to new maps, new kinds of obstacles, and more complex sentences that do not occur in the training set. The model provides a level of interpretability through the use of attention maps allowing users to see its reasoning steps despite being an end-to-end model.

1 Introduction

When you carry out a command uttered in natural language, you combine your knowledge about the task to be performed and how it was carried out in the past with reasoning about the consequences of your actions. Thinking about the task allows you to choose actions that are likely to make progress, and it is most useful when the path forward is clearly understood in an environment that has been experienced before. Thinking about the consequences of your actions allows you to handle new environments and obstacles, and it is most useful where a task must be performed in a novel way. Generally, prior work has excelled at one of these but not both. Powerful models can control agents but do so from moment to moment without planning complex actions (Blukis et al., 2018; Misra et al., 2018; Shah et al., 2018). Planners on the other hand efficiently explore configuration spaces, often by building search trees (Hsu et al., 1997; Karaman and Frazzoli, 2011), but require a target final configuration (Chen et al., 2019; Kuo et al., 2018; Lee et al., 2018) or a symbolic specification of constraints (Tellex et al., 2011; Paul et al., 2017).

We demonstrate an end-to-end model that both reasons about a task and plans its action in a continuous domain resulting in a robot that can follow linguistic commands. It integrates a planner with a compositional hierarchical recurrent network. The recurrent neural network (RNN) learns which actions are useful toward a goal specified in natural language while the planner provides resilience when the situation becomes unclear, novel or too complicated. This process frees the network from having to learn the minutia of planning and allows it to focus on the overall goal while gaining robustness to novel environments.

To execute a command, the model proceeds as a traditional sampling-based planner with an additional input of a natural language command; see fig. 1. A collection of networks are arranged in a hierarchy that mirrors the parse of the command. This encodes commands into the structure of the model. A search tree is created through a modified RRT (LaValle, 1998) which explores different configurations of the robot and their effect on the environment. The search procedure is augmented by

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

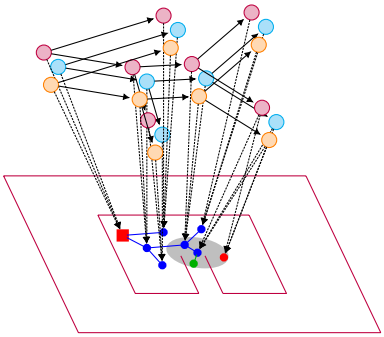


Figure 1: We augment a sampling-based planner, RRT, with a hierarchical recurrent network that encodes the meaning of a natural-language command the robot must follow. Just as with a traditional planner, the robot mentally explores the configuration space building a search tree from a start location to find a good path. Unlike a traditional planner, we do not specify a goal as a location, but instead rely on a neural network to score how likely any position in the configuration space is to be an end state while considering the past history of the robot's actions and its observation of the environment. The structure of the RNNs mirrors that of the search tree. At each time step, the RNNs observe the environment (the grey shadow), and can adjust the sampling process of the planner to avoid moving in undesirable locations (in this case, the tree is not expanded toward the red circle, and instead adjusted to go down the passageway through the green circle). See fig. 2 for details on the structured RNNs and how they encode the structure of sentences as relationships between recurrent models.

the hierarchical network which can influence the nodes being expanded and the direction of expansion. As the search tree splits into multiple branches, the hierarchical recurrent network similarly splits following the tree. This encodes the reasoning and state of the robot if it were to follow that specific set of actions. At each time point, the network predicts the likelihood that the action satisfies the command. In the end, much as with a classical sampling-based planner, a search tree is created that explores different options, and a path in that tree is selected to be executed.

Robustness to new environments is achieved by trading off the planner against the hierarchical network. The influence of the network is proportional to its confidence. When new obstacles, or map features, or other difficulties are encountered (for example, not immediately seeing a goal), the algorithm can temporarily devolve into a traditional RRT. This is a desirable feature because algorithms like RRT make optimal decisions when other guidance is not available. Unlike planners, uncertain or untrained networks generally make pathologically bad decisions in such settings. This issue is often alleviated with techniques such as ϵ-greedy learning (Watkins, 1989) which provide random moves rather than the near-optimal exploration that sampling-based planners engage in.

We ensure that the model provides a level of interpretability in two ways. First, the structure of the sentence is encoded explicitly into the structure of the recurrent network; see fig. 2. Inspecting the network reveals which subnetworks are connected together and the topology of the connections mirrors that of natural language. Second, the internal reasoning of the model is highly constrained to operate through attention maps. Rather than allowing each component the freedom to pass along any information up the hierarchy in order to make a decision, we constrain all components to communicating via a grayscale map that is multiplied by the current observation of the environment. Inspecting these attention maps reveals information about which areas each network is focused on and can provide a means to understand and explain failures.

2 Deep Planners with Language

Robotic planners are efficient at searching the configuration spaces of robots. Here we describe how to augment them with networks that efficiently learn language and guide the planning process. This is related to the approach of Kuo et al. (2018), DeRRT, which introduced deep sequential models for sampling-based planning. The original DeRRT planner maintains a search tree and a corresponding RNN with the goal of reaching a fixed destination in the configuration space. In this work, we use a collection of networks (as shown in fig. 1) and have the networks determine the final configuration based on the command rather than explicitly providing it.

At each step, the planner chooses a node to extend by the standard RRT mechanism: sampling a point in space and finding the nearest point in the tree to that sample. It then proposes a new tree node between the selected tree node and the sampled point. The RNN takes as input the state at the current node, any visual observations at the current node, and the proposed extension to the tree.

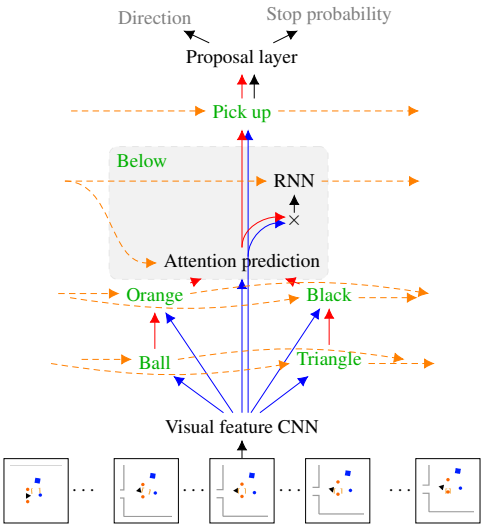


Figure 2: The structure of the model interpreting and following the command *Pick up the orange ball from below black triangle*. As the search tree described in fig. 1 is constructed, this model interprets the state of each tree node being expanded. It predicts the direction to expand the node in and whether the node completes the plan being followed. Each word is a module in the network; each module contains two neural networks (shown in black — the module for *below* is expanded). Each word's hidden state is updated at each time step using an RNN. The structure of the network is derived automatically from a parse produced by the NLTK coreNLP parser (Bird, 2006). Visual features are extracted and provided to each word model. Attention maps are predicted by each word by a combination of visual features, the attention maps of any words directly below in the hierarchy, and the hidden state of that word. The attention map of the final word and the output of its RNN are used to predict the direction of movement and the success probability. Using attention maps as the mechanism to forward information in the network provides a level of interpretability.

Observations are processed with a co-trained CNN that is shared among all words. The network makes its own prediction about how to extend the tree at the current node along with a confidence value. A simple mixture model selects between the planner and the network proposed directions. Once the tree is constructed, in this case after a fixed number of planning steps, the node that is considered most likely to be an end state of the described command is chosen and the path between the start state and that node is generated.

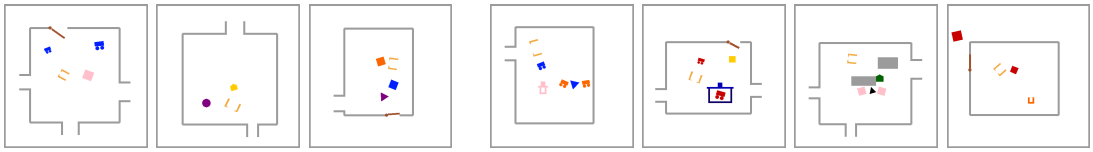
2.1 Language and Deep RRT

The model described thus far uses a single RNN to guide the planner. Technically, this is serviceable, as the network can in principle learn to perform this task. Practically, generalizing to new sentences and complex sentential structures is beyond the abilities of this simple model.

Using a collection of networks, rather than a single network can help generalization to new sentences. Just as one network can guide a planner, multiple networks can also guide it. Each network can make a prediction. A direction can be sampled from the posterior distribution over all of the predictions by all of the networks.

We build this collection of networks out of a lexicon of component networks. Given a training set of commands and paths, one component network is trained per word in the command that the robot is following. Given a test sentence, the words in the sentence determine the set of component networks which guide the planner. We call this the *bag of words* (BoW) model because there is no explicit relationship or information flow between the networks. Due to the lack of relationships between words, this model has fundamental difficulties representing the difference between *Grab the black toy from the box* and *Grab the toy from the black box*.

To address this limitation, we introduce a hierarchical network; see fig. 2. Given a sentence and the parse of the sentence derived from the NLTK coreNLP parser (Bird, 2006), we select the same set of component networks that correspond to the words in the sentence. Networks are arranged in a tree — naturally, such trees are rooted by verbs in most linguistic representations. The state at the current node informs the representation of each component network. Each component updates its own hidden state and forwards information to all of the components that are linked to it. The leaves only receive as input observation at the current state and their own hidden state. The root of the tree produces an output used by a linear proposal layer to predict the direction of movement and the likelihood that



(a) Examples of the training set (b) Examples of the test set

Figure 3: Examples of the (a) training set and of the (b) test set. The robot is shown in orange as a pair of L-shaped grippers. Objects are randomly positioned, with random properties and orientations. The training set is considerably simpler, with fewer objects on average, without cups that have lids, without the need to traverse doors or channels as all objects are inside the room, and without immovable obstacles (grey rectangles).

the current node has reached a goal. This approach has the ability to represent the earlier distinction about which noun the word *black* is attached to because different attachments result in different parse trees and thus different instantiations of the model.

We restrict nodes to communicating via attention maps rather than arbitrary vectors. This helps generalization as words which never co-occurred in the training set can be seen in the test set of the experiments we report. By ensuring that the representation shared between component networks is universal, such as the attention maps, component networks are encouraged to be more compatible with one another. Each word takes as input a set of attention maps, weighs the input image with each attention map independently, and combines this with the hidden state of that word. A new attention map is predicted and passed to subsequent words. Using this predicted attention map, an RNN takes as input the observed image weighted by the attention map and updates the hidden state of the word. In addition to encouraging generalization, attention maps can be interpreted by humans, and help speed up learning by being relatively low dimensional.

3 Experiments

A generative model creates new training and test maps conditioned on a target command which is sampled from a grammar. The space of possible maps is large and includes rooms of varying sizes, which can have between 0 and 4 narrow gaps, possibly contain a door to the outside, and may contain between two and eight objects with multiple properties (shape, color, and size); see fig. 3. The grammar which generates commands contains seven verbs (push, grab, approach, touch, open, leave, carry), seven nouns (block, cup, ball, triangle, quadrilateral, house, cart), eight colors (red, green, blue, pink, yellow, black, purple, orange), two sizes (big, small), nine spatial relations (left of, right of, top of, bottom of, on the left of, on the right of, near, above, below), and two prepositions (towards, away from). Given all of the possible objects, distractors, room size, doors, gateways, object locations, colors, rotations, and sizes, a random map is generated. We verify that the target plan is feasible on this map. The same map never appears in both the training and the test sets. This provides an immense space from which to generate maps and to test model generalization capabilities.

To execute a natural language commands generated by users, each of these linguistic constituents becomes a component neural network in a lexicon of networks. Sentences are parsed with the NLTK coreNLP parser (Bird, 2006) and unknown words in the sentences are mapped to nearby words using their distance in WordNet (Miller, 1995; Pedersen et al., 2004).

To evaluate our model, we develop several baselines by augmenting earlier work to perform this challenging task. The weakest baseline, *RNN-Only*, is our model without the planner but including the hierarchical neural network. A more powerful baseline, *BoW*, is created by augmenting the work in Kuo et al. (2018). A collection of neural networks represent the meaning of a sentence, but they do not interact with one another; these form the *bag of words*. Finally, we compare against a model, *RRT+Oracle*, which represents the performance that can be expected if the hierarchical network is operating well. This model uses the same underlying planner but the goals are manually specified through the use of an oracle.

3.1 User Study

We generated 500 map and command pairs and had the robot execute those commands. The executions of these commands, but not the commands themselves, were shown to four users recruited for this experiment. Users were asked to produce the instructions they would provide to the robot in order to elicit the behavior they observed. Out of 500 descriptions, 128 were impossible for the robot to follow due to user error, e.g., by mentioning objects that are physically not there, or could not be reasonably parsed. The 372 remaining descriptions had an average length of 9.04 words per sentence

Planner	Obstacles	Cup & Lid	Door
RNN-Only	0.12	0.08	0
BoW	0.32	0.08	**0.35**
Ours	**0.52**	**0.16**	0.30
RRT+Oracle	**0.52**	0.08	**0.35**

Figure 4: The success rate of different baselines and models when generalizing to maps that have properties which have never or rarely been experienced at training time.

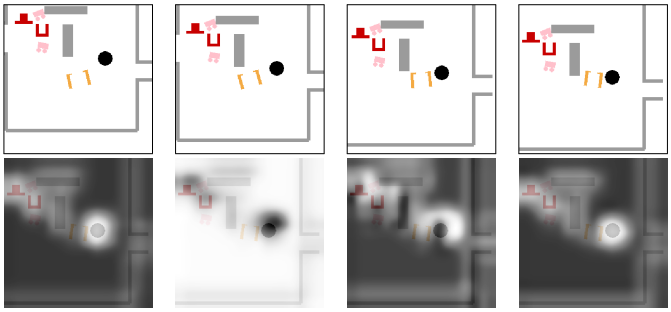


Figure 5: Example snapshots from the execution of the model (top; these are the local information available to the robot rather than the entire map) along with the attention maps (bottom) produced by each component network in the model at a critical time in the execution of the command which was not carried out correctly. Note that the polarity of the attention maps is irrelevant — models can communicate by either suppressing or highlighting features and neither carries any a priori valence. In this figure, the robot fails to pick up the correct object and heads to the circle instead. The failure can be explained by the poor detection of the cup seemingly confusing it with the circle.

standard deviation of 2.49. The baseline RNN-Only model achieved 17% success rate, the BoW model succeeded 40% of the time, while our model succeeded 49% of the time. The RRT+Oracle model had roughly the same performance as ours succeeding 51% of the time. This demonstrates that our approach scales to real-world user input.

3.2 Additional Obstacles and Preconditions

Robots must continually deal with new difficulties. To evaluate how models adapt to new problems, we further modify the test set to include other features not present at training time; see fig. 4. In particular, we add random fixed obstacles and require that the robot traverse a push-button-controlled door. In addition, the frequency of objects inside cups with lids is significantly increased.

The model which does not include a planner, the RNN-Only model, has difficulty generalizing to new scenarios. All the other models generalized better, with ours performing on par with the oracle. These results indicate that planners provide robustness when encountering new challenges. This is well known in symbolic planning but has not been exploited as part of an end-to-end approach before.

4 Discussion & Conclusion

We have demonstrated that a hierarchical recurrent network can work in conjunction with a sampling-based planner to create a model that encodes the meaning of commands. It learns to execute commands in challenging new environments that contain features not seen in the training set. We demonstrated that our approach scales to real-world sentences produced by users.

Our model provides a level of interpretability. The structure of the model mirrors that of the parse of a sentence making it easy to verify if a sentence has been incorrectly encoded. Attention maps are used throughout the hierarchical network to allow component parts to communicate with one another. These provide another means to understand which components caused a failure; see fig. 5 for example. In many cases, this provides both reassurances that errors will be pinpointed to the responsible part of the model and confidence in the chosen model. This level of transparency is unusual for end-to-end models in robotics.

In the future, we plan to extend the language learning capacities of the approach and explore ways to carry out more complex tasks.

References

Bird, S. (2006). NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Blukis, V., Misra, D., Knepper, R. A., and Artzi, Y. (2018). Mapping navigation instructions to continuous control actions with position visitation prediction. In *Proceedings of the Conference on Robot Learning*.

Chen, B., Dai, B., and Song, L. (2019). Learning to plan via neural exploration-exploitation trees. *arXiv preprint arXiv:1903.00070*.

Hsu, D., Latombe, J.-C., and Motwani, R. (1997). Path planning in expansive configuration spaces. In *ICRA*.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894.

Kuo, Y.-L., Barbu, A., and Katz, B. (2018). Deep sequential models for sampling-based planning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6490–6497. IEEE.

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.

Lee, L., Parisotto, E., Chaplot, D. S., Xing, E., and Salakhutdinov, R. (2018). Gated path planning networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*.

Miller, G. A. (1995). WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Misra, D. K., Bennett, A., Blukis, V., Niklasson, E., Shatkhin, M., and Artzi, Y. (2018). Mapping instructions to actions in 3d environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Paul, R., Barbu, A., Felshin, S., Katz, B., and Roy, N. (2017). Temporal grounding graphs for language understanding with accrued visual-linguistic context. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4506–4514. AAAI Press.

Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics.

Shah, P., Fiser, M., Faust, A., Kew, J. C., and Hakkani-Tur, D. (2018). FollowNet: Robot navigation by following natural language directions with deep reinforcement learning. *arXiv preprint arXiv:1805.06150*.

Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S., and Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Watkins, C. (1989). Learning form delayed rewards. *Ph. D. thesis, King's College, University of Cambridge*.