Simultaneously Learning Transferable Symbols and Language Groundings from Perceptual Data for Instruction Following

Nakul Gopalan*, Eric Rosen†, George Konidaris† and Stefanie Tellex†
*Georgia Institute of Technology, †Brown University
Email: *nakul_gopalan@gatech.edu, †{eric_rosen, george_konidaris, stefanie_tellex}@brown.edu

***Abstract*—Enabling robots to learn tasks and follow instructions as easily as humans is important for many real-world robot applications. Previous approaches have applied machine learning to teach the mapping from language to low dimensional symbolic representations constructed by hand, using demonstration trajectories paired with accompanying instructions. These symbolic methods lead to data efficient learning. Other methods map language directly to high-dimensional control behavior, which requires less design effort but is data-intensive. We propose to first learning symbolic abstractions from demonstration data and then mapping language to those learned abstractions. These symbolic abstractions can be learned with significantly less data than end-to-end approaches, and support partial behavior specification via natural language since they permit planning using traditional planners. During training, our approach requires only a small number of demonstration trajectories paired with natural language—without the use of a simulator—and results in a representation capable of planning to fulfill natural language instructions specifying a goal or partial plan. We apply our approach to two domains, including a mobile manipulator, where a small number of demonstrations enable the robot to follow navigation commands like "Take left at the end of the hallway," in environments it has not encountered before.**

I. Introduction

Humans can easily learn novel tasks given paired demonstrations and verbal instructions; children can learn novel concepts and ground novel words to describe known or unseen objects very quickly, sometimes with only one sample [1]. It would be useful for robots to have the same ability to learn reusable knowledge from just a few demonstrations, enabling them to learn continuously and to use that learned knowledge to follow natural language instructions given by human users.

Previous work to map instructions to policies can be split in two categories. Firstly, end-to-end approaches [2], [3] directly map language to actions, and have been shown to be effective (but data intensive) in simulations. For example, Blukis et al. [4] use an end-to-end approach to map natural language to policies on a real robot via a simulator. However, the agent requires thousands of demonstrations in simulation, along with hundreds of real robot trajectories, to learn these policies. The second category of methods map language to handcrafted state abstractions, where states of importance are pre-specified [5], [6], [7], [8]. These methods are limited because we need to handcraft state abstractions, yet these same abstractions do not generalize well to novel environments.

We propose first learning transferable abstractions from demonstration trajectories, and then mapping the accompanying language to sequences of these learned abstractions. Such an approach allows the agent to learn abstractions useful for planning, and then ground these abstractions to novel maps of the environment. Our approach learns two types of abstractions from demonstrations: first, motor skills, which abstract low level robot control into high level actions, such as going straight until a wall or turning 90° left; second, perceptual symbols representing the set of states where these skills cease execution, that is, sub-goals, which are sufficient for planning using the motor skills [9].

For example, a trajectory of a robot to "left at the end of the hallway" can be segmented into three skill segments: the robot moving straight, the robot turning left, and the robot going straight again. The terminating states of these skill segments can be clustered to create termination conditions for generalizable skills. These skill termination conditions are the symbolic abstractions used to ground language to plans. From the example above the termination condition for skills that allow the robot to travel straight, and turn left can be learned from the states where the skills finish executing. Images of such a trajectory along with their associated sub-goals and LIDAR data is shown in Fig. 1.

Further, the resulting skills and symbols are learned in the agent's frame of reference, which supports transfer to new environments [10], [11]. Critically, at training time only language paired with trajectories in the robot's low-level sensory-action space are required, rather than needing to provide an abstract space of symbols. We then map the language instructions collected with the demonstrations to this symbolic abstraction.

At runtime, the learned mapping can be used to map language to symbols that partially specify the desired behavior by specifying the sub-goals needed to achieve the desired behavior. These symbolic sub-goals might not be immediately reachable, but the robot can use an off-the-shelf planner and generate a plan to reach the termination conditions of each of these symbols sequentially. When a new language command of, for example, "go left at the next intersection" is provided to the robot in a previously unseen location it maps the command to a sequence of sub-goals (learned previously), and then plans to reach the intersection, turn left and move forward.

We implement our approach in two domains: a driving

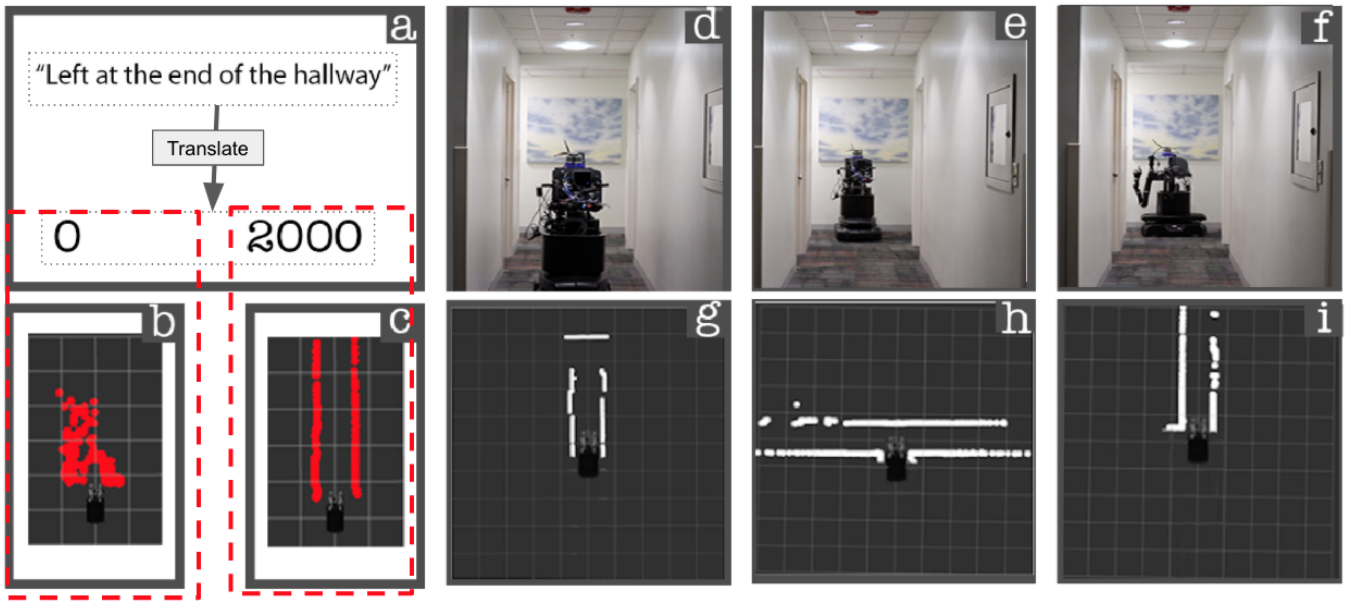


Fig. 1: A robot following the natural language instruction "left at the end of the hallway" using our pipeline. Our translation model converts the natural language input into the sequence of symbols which the robot uses as sub-goals to plan over. The first two symbols being: `0`, `2000`. (a) Translation of the sentence "left at the end of the hallway" into the symbol sequence "`0 2000`." Panels (b) and (c) show the median of the LIDAR data used to create the symbol classifiers `0` and `2000` respectively. Panels (d), (e) and (f) show the robot executing the command, with the lower row panels (g), (h) and (i), in order, showing the LIDAR data at the corresponding points during the execution. (d) (with the corresponding LIDAR data in (g)) The robot at the start of execution. The robot plans to reach to the first symbol, `0`, in the translated symbol sequence. (e) (with the corresponding LIDAR data in (h)) The robot reaches the LIDAR data matching symbol `0` by moving forward, where it terminates the corresponding skill. The robot then plans for the skill attached to the next symbol, that is `2000`, which causes it to execute a skill that turns left. (f) (with the corresponding LIDAR data in (i)) The robot's LIDAR data matches the `2000` symbol terminating the plan execution successfully. The robot plans to satisfy a partial plan specification defined by the termination symbols which in turn are specified by natural language.

domain using a high dimensional LIDAR dataset collected on a car [12], and a mobile robot scenario. We present the accuracy results of grounding natural language to our learned symbols in the first domain. We then demonstrate our complete system in the second domain, with language grounding accuracy results and planning on a real robot. Only 75 trajectory demonstrations paired with natural language commands are required to allow generalization and planning in novel maps. This is the first work—to the best of our knowledge—that learns symbolic abstractions and their mappings to language from demonstrations. Our approach allows learning from a small number of demonstrations, shows generalization to novel environments, and does not require a simulator.

II. Skills and Symbols Background

We translate natural language instructions into a sequence of learned symbols. We model motor skills using the option framework [13], as follows:

Definition 1. Skill – A skill is a triple: $\langle \mathcal{I}, \sigma, \pi \rangle$ where,

1) $\mathcal{I} : \mathcal{S} \longrightarrow [0, 1]$ is the initiation condition that defines the set of states in $\mathcal{S}$ where a skill can start executing.
2) $\sigma : \mathcal{S} \longrightarrow [0, 1]$, is the termination condition defining the set of states where the option terminates.
3) $\pi : \mathcal{S} \longrightarrow Pr(\mathcal{A})$ is the skill's stochastic policy, defining probability of actions over states.

We can learn the policy for these skills or use a planner to plan for the policy given the initiation and termination conditions. The termination and initiation sets themselves can be modelled as classifiers indicating the states where the option can execute and terminate [9]. Following Konidaris et al. [9], we formally define a symbol as a set of skill termination states, σ, represented by a classifier. In our paper we assume all skills can be initialized at every location.

III. Mapping Language to Plans via Learned Symbolic Abstractions

Our method for learning to follow instructions is split in two stages: training and runtime. During training the robot is given multiple egocentric trajectories ($\{\tau\}_{i=0}^{N}$) paired with natural language instructions for each trajectory $\{I\}_{i=0}^{N}$. Each trajectory is a sequence of observed states and actions in the demonstrations, that is, $\tau = \{(s_0, a_0), (s_1, a_1)....(s_T, a_T)\}$.

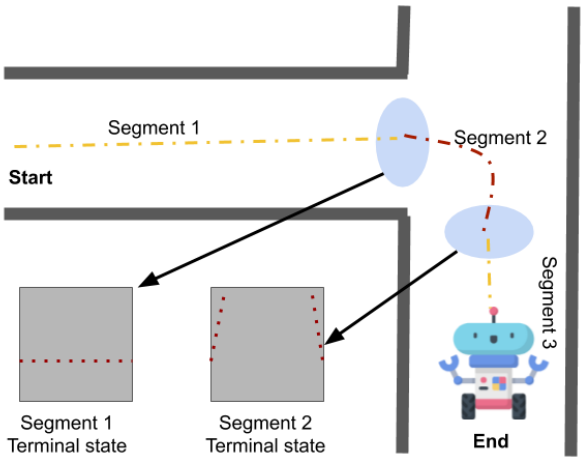


Fig. 2: Our symbol learning methodology. We drive the robot around and collect trajectories associated with a natural language command. Here the natural language command is "turn right at the intersection and go straight." These trajectories are processed with a change point detection algorithm to obtain skill segments `1`, `2`, and `3`. The HDP-HMM skill segmentation algorithm automatically assigns skill segments `1` and `3` to the same skill. We then collect the terminal states of the resulting skills—illustrated here as the egocentric LIDAR readings of the robot when facing a wall, at the termination of skill segment `1`, and when facing the corridor at the termination of skill segment `2`. Termination states for instances of the same skill policy are clustered to learn all possible skill terminations sets for the skill policy, and used to create classifiers from each cluster to represent a symbol.

Moreover, every trajectory is paired with an instruction I_n, a command sentence given by a human user in natural language. Training results in a set of symbols $\sigma_i \in \Sigma$ that can be used for planning, and a learned mapping from natural language to (sequences of) these symbols.

At runtime, the robot is placed in a novel environment and given a natural language instruction. The agent first uses its trained model to map from the language to a sequence of symbols. This sequence of symbols provides a partial plan in the form of sub-goals, but the agent must still plan to reach these sub-goals. Planning is required as writing a controller to satisfy sub-goals in novel maps is non trivial. Off the shelf planners or the learned skill policies can be used to satisfy every sub-goal in the symbol sequence in order. The state space in this work is LIDAR data that the robot observes in the world and the actions are continuous robot joint torque values. However, our methods are generic enough to allow for other state representations like images and point cloud data for states.

A. Learning Skills and Symbols from Demonstration

During learning our approach requires robot trajectories (in the form of actions and observations in egocentric frame) paired with natural language commands. We learn an abstraction of the world in the form of a collection of skills extracted from the demonstrations, and the symbols representing their termination conditions. We first extract the skills latent to the demonstration trajectories using changepoint detection, and then learn their termination conditions using clustering and classification. This process is illustrated in Fig. 2. At this point each trajectory demonstration is abstracted to a sequence of symbolic sub-goals. We next learn a function to map natural language to these abstracted sequences of symbols. Finally, to achieve instruction following, at runtime we translate natural language to a sequence of these learned symbols. We then use planning to reach every symbol or sub-goal in the sequence, in sequential order, to satisfy the given instruction. The entire pipeline is illustrated in Fig. 3; we now describe each part in more detail.

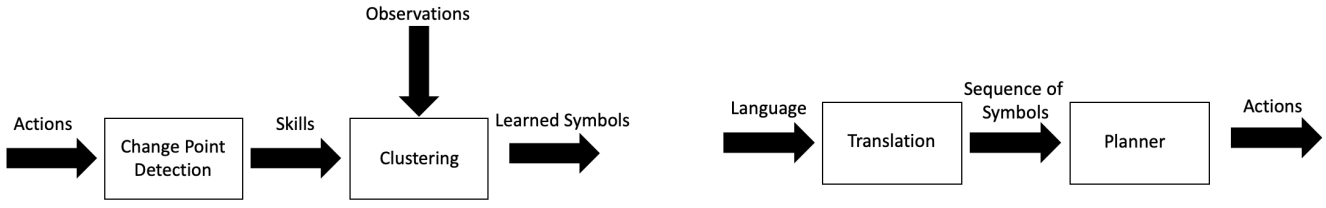
1) Symbols: Our goal in symbol learning is to identify the termination conditions of reusable skills obtained via segmentation, which are used as grounding classifiers for our symbolic representation. Our key insight is showing that the equivalence of option termination conditions and symbols—first pointed out by Konidaris et al. [9]—can enable a robot to learn to map natural language to goal-based planning with only trajectories as supervision. The skill policy can be learned from the segmented demonstration using a framework like Dynamic Movement Primitives (DMPs) [14], or we can plan to skill terminations if the transition model of the world is known.

2) Egocentric Representation: A key goal of our approach is to learn symbols that generalize to new environments. To achieve this goal, we use an egocentric state space representation, which is known to be sufficient for transfer [15]. The classifier of each symbol is trained with states observed in egocentric space. These egocentric symbols can be grounded in the global frame of reference by checking if the egocentric observations of a location in the map or pose of the robot is similar to the egocentric observations of the symbol during learning. A more complete treatment of learning symbols in the egocentric state and transferring them to a global map is provided by James et al. [11].

The egocentric representation in our case has observations in the form of Light Detection and Ranging (LIDAR) data, and the actions in the form of torque input to the agent. For other platforms the state could be an egocentric camera input. The egocentric representation, is non-Markov in nature as multiple states might look the same to the agent.

There is a separate global frame of reference [15], which is the state of the external world under the given instruction, and is sufficient for planning. We learn the abstract symbols in local egocentric frame, and plan in a global Markov state space, which in this case is a map of the world and the robot's location in it. We ground symbols learned in the egocentric state space onto the physical map of the world and then plan over them. For this work we assume the transition dynamics of the world to be known, allowing point-to-point navigation on the map.

3) Skill Segmentation: Our input consists of natural language paired with a robot trajectory. Our trajectory demonstra-



(a) Training (b) Runtime

Fig. 3: Our methodology for learning a symbolic representation and grounding language to it for instruction following. a) Demonstration trajectories are segmented using change point detection to generate skills. The observations that occur at skill termination are clustered to generate our learned symbols. These symbols are used as input to to learn to mapping from the given natural language commands to sequence of learned latent sub-goal symbols present in the corresponding trajectory. b) At run time, natural language is input to our translation model, which outputs a sequence of symbols. These symbols are used as sub-goals for a planner, which generates robot actions.

tions are unstructured, but we assume that they have underlying latent skills that are present in multiple demonstrations. We need to find underlying skills and symbols that create the trajectory so as to learn a plannable abstraction. We use a hierarchical Bayesian changepoint detection approach to detect repeated instances of the same latent skill. Such approaches are more data efficient than similar deep neural network approaches, which require orders of magnitude more data.

We used the Hierarchical Dirichlet Process Hidden Markov Model (HDP-HMM) [16] to model our trajectories. This formulation recognizes the change points and the underlying latent skills generating the trajectory segments, allowing us to recognize when the same skill is being repeated across multiple demonstrations. Such an approach has been used previously elsewhere for similar purposes [17], [18], [19], [20].

The data for skill segmentation in our case consists of the observed actions $y = [y_0, y_1, ...y_{T-1}, y_T]$ generated by the demonstrator, where the data is d dimensional. We posit underlying latent skills which come from $k \in \{1, 2, ...\}$ possible labels. Under the HDP-HMM process latent skills are assumed to be generated from a first order Markov process represented by $z = \{z_0, z_1, ...z_T\}$. The latent first order Markov process is parameterized with initial state probabilities (π_0) and transition probabilities $(\{\pi_k\}_{k=1}^{\infty})$ from one latent skill to another. There is an additional parameter of $(\{\theta_k\}_{k=1}^{\infty})$ to govern the observation noise when observing the output trajectory, which is parameterzied by the hyperparameter λ. The mathematical formulation for a HDP-HMM as described by Fox et al. [21] is given by:

$$
\begin{aligned}
&y_t|z_t = k \sim \mathcal{N}(x_t|A_k x_{t-1}, \Sigma_k);\\
&z_{t+1}|\{\pi_k\}_{k=1}^{\infty}, z_t \sim \pi_{z_t};\\
&\pi_k|\alpha, \kappa, \beta \sim \text{DP}(\alpha + \kappa, (\alpha\beta + \kappa\delta_\kappa)/(\alpha + \kappa));\\
&\beta|\gamma \sim \text{GEM}(\gamma),
\end{aligned}
$$

where DP is a Dirichlet process and β is a random probability measure setting the transition probabilities between skills. GEM is the Griffiths, Engen and McCloskey distribution, which is used to generate β using a stick breaking model parameterized by γ. Parameter β in turn parameterizes the transition probabilities π_k along with the hyperparameters of α and κ. The goal of such a model is to estimate $Pr(\beta, \pi, \theta, \{z\}_{t=1}^{T}|\{x\}_{t=0}^{T})$, which is generally performed using variational inference. The Viterbi algorithm [22] is used to label each time step with a latent skill using the joint probability distribution. The observed sequence y here is treated as an auto-regressive process. We used the BNPY toolbox by Hughes et al. [23] for performing this HDP-HMM change point detection; for a more complete mathematical treatment of HDP-HMM please refer to Fox et al. [21] and Hughes et al. [24]. HDP-HMM labels the trajectory segments with the underlying latent skills. These trajectory segments can be used to learn a skill policy using DMPs [14].

In this work we are only segmenting the action trajectory of the agent, and not the state-action trajectory. This is different from previous methods [17], [18], where state and action information is used for segmentation. This is because our state data is not just the robot's joint space, but the actual high dimensional state of the environment visible to the agent. It is computationally intractable to segment trajectories with such high dimensions with the current state of the art Bayesian changepoint detection methods.

4) Clustering: Each of the latent skills learned with changepoint detection could potentially have multiple termination conditions. For example, the going forward skill might terminate when the agent reaches a wall, or when the agent reaches the next room. The termination sets for both of these skills must be modelled separately. The number of effect sets associated with a skill policy are not known in advance. We use clustering as an unsupervised method to identify the possible termination sets of each learned skill. We use the DBSCAN [25] exclusively as the clustering method as it is a non-parametric method that does not need a pre-specified number of clusters or the diameters of possible clusters. We only specify a distance metric between clusters, Euclidean in our case; the minimum number of support points for a cluster so a cluster is not just a made of one or two points; and a noise parameter ϵ that allows the algorithm to identify outliers. We

choose the same noise and support parameters for all skills in a domain, that we pick such that we do not have any cluster of termination states made of relatively few support points.

5) Classification: A propositional symbol refers to a classifier that determines the states in which the symbol is true, and those in which it is false [9]. We therefore next convert each cluster of states into such a classifier For example, to detect the end of a corridor, the LIDAR state data directly ahead of the robot would show an obstacle at close range for a cluster skill termination states. This set of states is not enough to recognize a novel point on the map as the end of the corridor. We must learn a classifier with the clustered states as positive labels for states that represent the end of a corridor. We used Single Class SVMs [26] as they are efficient and robust to noise, which is prevalent in LIDAR data. Single Class SVMs were developed as an outlier detection method, and in our case, outlier states are those that do not satisfy the statistics of our termination states. We use a Sigmoid kernel as features for classification, as it helps distinguish LIDAR distance data well. The hyperparameters for the classifiers are chosen based on the number of dimensions of the data and the variance of the data as suggested by Scholkopf et al. [26]. After we train classifiers associated with each cluster, we merge classifiers that have similar output responses for each skill. To compare the output responses of the classifiers we test the points used to train one classifier using the other classifier, and vice-versa. If both classifiers label more than 85% of the points belonging to the other cluster as inliers, we then merge the cluster, as their inlier responses are similar, and the clusters have a large overlap, hence they might as well be the same symbol.

B. Mapping Language to Symbols

We next learn a mapping from natural language instructions, defined as a sequence of strings $i = [i_0, i_1, ..., i_{l-1}, i_l]$, to a sequence of symbols representing skill termination conditions, $[\sigma_0, .., \sigma_n]$. This is a machine translation problem, which is posed generically as mapping an input sequence to an output sequence.

The output target sentence is the order in which the symbols appear in the demonstration trajectory. We then train a deep Seq2Seq recurrent neural network [27] to take the sequences and output the paired sequence of skills using the negative log likelihood loss. The embedding vector size is 50 dimensional, and the hidden vector size is 256. Seq2Seq utilizes two recurrent neural networks, one which acts as an encoder E and the other as a decoder D. The encoder is trained to take the input sequence and map it to a vector of fixed dimension, and the decoder is trained to take the output and decode to the translated sequence. Our Seq2Seq model has Badhanau attention [28] to provide soft alignment between input and output symbol sequences. We also use GloVe [29] pre-trained embeddings as they improve Seq2Seq performance in low data scenarios.

C. Planning Over Symbol Sequences

At runtime we plan over the sequence of output symbols predicted by the Seq2Seq model in order, that is, we plan for the first symbol in the sequence, and then the next and so on until the terminal symbol is reached. The result is a sequence of skills, the policy for each of which could be a DMP or an out of the box planner motion planner. Our pipeline allows us to translate natural language to a sequence of learned transferable symbols that can be grounded on a novel map for planning, allowing a mapping from natural language to plans with generalizability that can be executed on a physical robot in different locations.

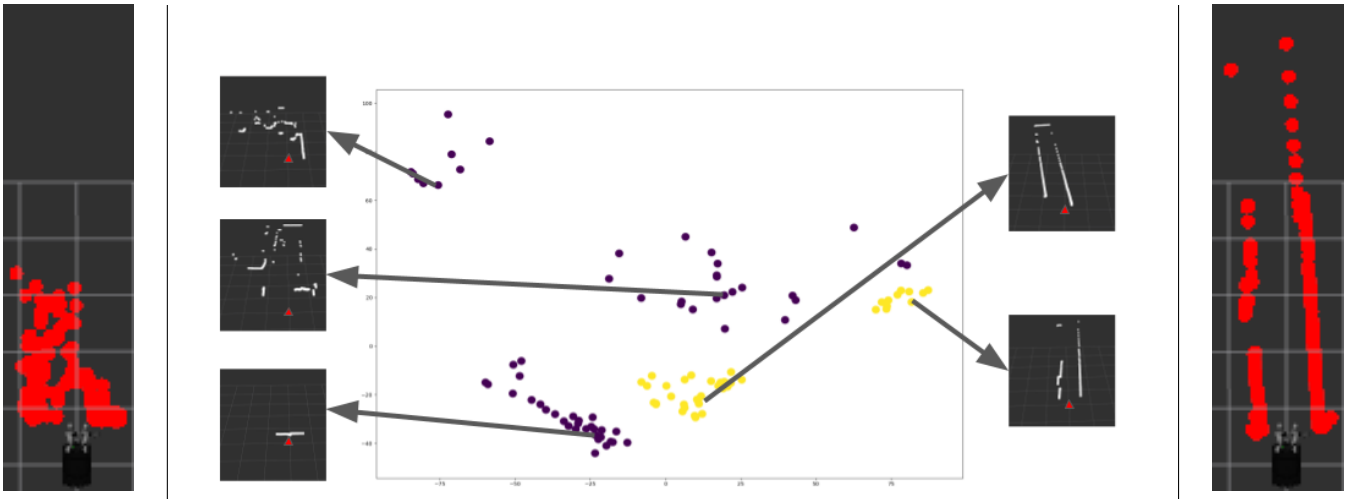
IV. Experiments

We evaluated our algorithm on two robot datasets: a real-world car LIDAR dataset collected by the Naval Postgraduate School (NPS) [12], and a robotics dataset we collected ourselves with a Kinova Movo mobile manipulator in an office building. For the NPS LIDAR dataset we report translation accuracy to the learned output symbol sequence. We compare our translation accuracy against a random baseline, which is the random chance of creating the right sequence, that is, permutation with repetition = $\frac{1}{L}^n$, where L is max length of an output sequence and n is total number of available tokens. Vanilla sequence-to-sequence methods that map language to actions directly would fail as the length of these output trajectories is greater than 1500 time steps and it is infeasible to use reinforcement learning for instruction following on a robot directly as the number of available samples is very small. For the mobile manipulator domain we do the same, and also demonstrate the complete pipeline where a robot is instructed via natural language to plan with the learned symbols in a novel environment.

A. NPS Car Dataset

We first wanted to test if our approach can be used on a high-dimensional domain, with continuous states and actions, to produce a much smaller set of symbols for instruction following. The NPS car dataset [12] consists of a car with 32-Laser Velodyne HDL-32e LIDAR and three Logitech c920 HD RGB cameras driving around a real neighborhood. The Velodyne runs at 10Hz and is collected from a full 360 degree view with a minimum distance of 0.9 meters and a maximum range of 130 meters. The RGB cameras function at 30Hz, and have a full-HD resolution of 1920×1080. We used this dataset because it had a lot of urban driving with plenty of turns, instead of highway driving. However, the data set does not contain sufficient information to test planning because no map information or world model was available.

Moreover, no action or IMU data was recorded in this dataset, and hence we used the LOAM velodyne [30] Robot Operating System (ROS) [31] package to localize the car from the Velodyne data. We then used the difference between sequential poses to compute the magnitudes and directions at each time step, which we treated as a 2 dimensional action vector.

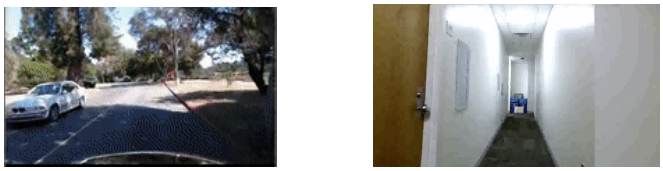


(a) (b) (c)

Fig. 4: Clustering results for the skill to go straight in the Movo robot navigation domain. There are two termination clusters learned in this domain. The center plot, Fig. 4b visualizes the 270 dimensional LIDAR data along its first 2 principal components along with their cluster label. (The distances between points in the center plot are not accurate representations of actual physical distances between these points.) Clustering was performed with the parameters $\epsilon = 88$ and minimum support $= 5$. The 2 clusters, shown in yellow and purple, show distinctly different types of termination states. States in the purple cluster (on the left) are states where the robot ends its trajectory in front of an obstacle. The states in the yellow cluster are those where the robot stops in the middle of a long corridor without an obstacle directly ahead. We show the median LIDAR data from each of the two clusters on next to their respective LIDAR states: Fig. 4a, shows the median LIDAR data for termination states with obstacles ahead and Fig. 4c shows the median LIDAR data for termination states where the trajectory ends in the middle of a corridor.

We selected a total of 10 trajectories with urban driving, uploaded the frontward facing RGB video of these trajectories to Amazon Mechanical Turk (AMT), and requested users to write down one sentence they would use to command the car to perform the observed behavior. From this, we collected 300 language annotations for 3 different behaviors that the car demonstrated: going straight and turning left at the intersection, going straight and turning right at the intersection, and going straight until the intersection. We cleaned the language dataset to 270 examples, with 163 novel words and a maximum sentence length of 148. The other 30 sentences were not related to the task as the annotators did not follow the instructions correctly. We segmented the behavior into the underlying skills of travelling straight, turning left and turning right, and used the sequence of these skills as our target language for translation. We converted the Velodyne data into a 360 dimensional vector that we use as our input states for skill segmentation and symbol learning. We then took the terminating laser scans from each skill segment and clustered them using DBSCAN [25] ($\epsilon = 20$ and minimum support $= 10$). This produced 3 symbols for the terminations of: turning left, turning right, and going straight.

We used the crowd-sourced language instructions as our source language for translation, and the output sequence of symbols from the trajectory segmented as our target language. We trained our network 5 times, and ran 5-fold cross validation for testing each time. We observed an average cross-validating exact match accuracy result of 96.51 ± 0.46. The random baseline for this test is 12.5% because there are 3 output symbols and a maximum sequence length of 2.



(a) (b)

Fig. 5: Our two domains. (a) The NPS Neighborhood dataset [12], and (b) the Movo mobile manipulator. We collected language and trajectory data for each dataset. We segmented trajectories of the high-dimensional LIDAR trajectories skills and their corresponding symbols, and mapped language to a sequence of these symbols.

Our behaviors were short in length, which made learning a mapping from language easy. We would have preferred to learn longer horizon behaviors, but our mapping and state-estimation package accumulated state-estimation errors over longer periods of time, limiting our train and test behaviors to be relatively short compared to our other experiment; this would not have been an issue if we had access to high-quality IMU or action data.

B. Mobile Robot Navigation

We next implemented our framework on a Kinova Movo robot with a built-in LIDAR sensor and forward-facing RGB camera. The LIDAR sensor operates at 15Hz, and actions were sent to the robot over ROS [31] with Twist messages at 45 Hz. The action data was 2-dimensional, encoding the linear x velocity and the angular z velocity.

For training, we went to 5 different locations in the same building and collected a total of 75 trajectories. For language collection, we uploaded the robot frontward facing RGB camera videos to AMT, and requested users to write down one sentence they would say to instruct the robot to perform the observed behavior. This resulted in 913 language annotations for 5 different behaviors of going straight and taking left, going straight and taking right, in-place holonomic left, in-place holonomic right, and going to the intersection. There were 347 novel words in this corpus with a maximum sentence length of 165. In the context of segmentation, the trajectories are noisy because the algorithm labels even small left and right movements as left and right turning skills. Hence we remove any latent skill shorter than half a second, as these movements might just involve course correction while providing demonstrations. The algorithm produced 3 different latent skills of turning left, turning right and going straight. The output now is a sequence of skills per demonstration trajectory.

For symbol learning, we filtered the LIDAR to only use the frontward facing 90 degree data, as the corridors were narrow and the LIDAR returned a lot of noise from its side sensors, as the walls of the corridors were under the minimum range of the LIDAR. We then took the terminal LIDAR states from each of the 3 segmented skills, and input them into DBSCAN [25] ($\epsilon = 88$ and minimum support $= 5$), which produced 5 different clusters. We learned one-class SVMs using states in each of these clusters representing end of corridor, intersection, right terminal corridor, and two left terminal corridor symbols. Example images of our learned symbols can be seen in Fig. 4a and 4c.

Using the resulting symbol sequences and language dataset, we trained our language translation model on 5000 epochs, with 50-dimensional pre-trained GloVe embedding on the encoder side, and with hidden state dimension being 256. The average exact match accuracy with cross validation was $75.71 \pm 0.37\%$. A baseline random policy has translation accuracy 0.41% with a maximum sequence length of 3 and 5 output tokens. We then brought the robot to an unseen environment, and tested three different language commands: "go to the end of the corridor and take a right," "left at the end of the hallway," and "go to the end of the corridor." In all three cases, our translation model produced valid and sufficient sequences of symbols for the given map, helping execute the instruction. We classified each point along the map sequentially to check for the termination condition of the symbols. We then planned to the points that satisfied the symbols along the map. Images of the robot performing planning for the "left at the end of the hallway" behavior along with the LIDAR states at skill terminations are in Fig. 1. These behaviors were all tested in a map unseen by the robot in training. This new corridor is of a different width than all the other corridors tested by at least 10cm. To demonstrate the stability of our learned symbols we demonstrate a longer trajectory planned and executed by the agent for the command "Drive straight until you reach a wall with an art print on it, turn left and proceed down a long hallway until you reach another wall turn left and proceed straight until you arrive at a water fountain." The videos of all these behaviors in the test environment and our data collection procedure can be found in our supplemental video.[1] The code for our methodology is also available publicly.[2]

The translation accuracy is not as high as we would like it to be as there are multiple symbols that recognize the termination for the skill of the robot has turning left. This is because we collected only 10 trajectories of the robot turning left. In half the instances the robot landed in an open corridor and in other it was in a short corridor with a wall ahead. Due to the large disparity in the depth data of the terminal states for the turn left skill, our clustering method identified two different skill terminations: one where the after turning there is wall at the end of the corridor, and another where the corridor's end cannot be seen. We believe this discrepancy caused our model to under-perform.

This creates two symbols for left that cannot be merged as the robot should see these corridors differently in the world. We were able to merge our symbols for turning right as the agent had over 20 trajectories to learn the meaning of right, and by sampling different corridors the agent was able to generalize the symbol termination across different depths. A much better method for translating from natural language to a set of learned symbols might be by conditioning the translations on the map of the world that the robot is encounters. In such cases it would be easy for the agent to establish that the current map has groundings only for certain set of symbols, allowing the left symbol's grounding to be straightforward in a given map.

V. Related Work

Instruction following is a supervised learning problem where the agent must predict a trajectory that would satisfy an input natural language command. The agent is trained using instructions paired with valid trajectories for those instructions. Semantic parsing was initially used to solve instruction following problems [32], [33]. This requires goal conditions and subgoal conditions to be pre-specified. For example, if an agent was asked to "go to the chair," the agent would need a pre-specified goal condition that returns true when the agent is next to the chair. These hand-specified goal conditions are labor intensive to design, especially on real robots. Our work, in contrast, learns symbols directly from high dimensional sensor

[1] https://vimeo.com/388650000

[2] https://github.com/nakulgopalan/change_point_detection.git

data. Moreover, these symbols are abstract representations of the sub-goals that the agent needs to achieve while planning to its goal. We learn both the underlying symbols in a trajectory demonstration and their mappings from language to allow planning in novel environments.

Some progress was made in learning symbols from data for instruction following by collecting the data for the symbols separately from the trajectories [7], [8]. MacGlashan et al. [34] learned mapping from language to a reward function learned via Inverse Reinforcement learning. However the abstractions such as objects and rooms were pre-specified in their domains, and not learned from scratch, and the reward function itself considered the permutation of these objects in the world. Matuszek et al. [35] learned mappings from sentences to attributes learned from sensor data directly. This work is the closest in spirit to what we are trying to achieve. However instead of learning object attributes we are learning abstractions for planning from scratch and grounding language to these learned abstractions.

Some works have tried to use neural approaches for sequence-to-sequence mapping from language directly to trajectories [2], [3]. Blukis et al. [36] map language to learned locations on a simulated map using an end-to-end approach using thousands of demonstration pairs. Blukis et al. [4] extend this approach using simulation to real techniques to an image feature space that does not distinguish between real world and the simulator to learn control policies in real world. However, their methods still require thousands of demonstrations in the simulator and hundreds of demonstrations in the real world as they learn the policy and the feature space end-to-end. Our approach requires fewer demonstrations and can generalize without the need of a simulator. Designing simulators can be hard for every possible task that a user might want to teach a robot.

Other end-to-end learning methods have used reinforcement leaning [37], [38], but require millions of episodes to learn simple behaviors. There are also approaches that map language to pre-specified symbols using neural networks [5]. Mapping to symbols in these approaches allows robots to plan more robustly to sensor noise and environment stochasticity. However, the space of available symbols must be pre-defined with expert knowledge. These mappings translate natural language to the pre-specified symbol space using neural translation approaches like Seq2Seq [27]. Instead, we learn the symbols or abstraction from the trajectories and then map language to these learned symbols.

There have been many approaches to learn skills from trajectories [18], [17]. Konidaris et al. [9] learns symbols from these skills. These skills have been seen as a way to help agents learn in a reinforcement learning setting [15] or allow the robot to follow a demonstration with multiple learned skills [18]. We follow a supervised learning approach to learning skills with demonstrations similar to previously established methods [18], [17]. Our methods to convert skills to symbols and plan over them are similar to those described by Konidaris et al. [9]. We differ from the previous approaches by integrating natural language translation to allow a human to specify, using natural language, the ordering of symbolic sub-goals for planning.

VI. Conclusion

Our approach has successfully demonstrated the learning of abstract symbols from example trajectories, mapping of language to these abstract symbol sequences, and transfer of these symbols to novel environments for planning. Our approach is based on learning portable symbolic representations from demonstration trajectories, and using this representation to ground language instructions via machine translation. Our method requires very few demonstrations to learn behaviors, which is crucial in robot applications that have large state spaces, and results in learned symbols that are generalizable to unseen environments. Our experiments across the two different datasets show that our method enables real robot platforms to follow language commands with little data.

VII. Acknowledgements

We would like to thank Mike Hughes for his discussions about BNPy. This work is supported by NASA under award number NNX16AR61G, the Sloan Foundation, the US Army under award number W911NF1920145, DARPA under award numbers W911NF1820268 and W911NF-10-2-0016, the Office of Naval Resarch under award number 81825-10893, the National Science Foundation under award numbers IIS-1717569 and IIS-1637614, the US Air Force under award number FA9550-17-1-0124, NSF CAREER Award 1844960, AFOSR Young Investigator Grant agreement number FA9550-17-1-0124, and the Office of Naval Resarch under the PERISCOPE MURI Contract N00014-17-1-2699. The robot icon for Figure 2 was made by Freepik from www.flaticon.com.

References

[1] E. M. Markman, *Categorization and naming in children: Problems of induction*. MIT Press, 1991.

[2] J. Andreas and D. Klein, "Alignment-based compositional semantics for instruction following," *arXiv preprint arXiv:1508.06491*, 2015.

[3] H. Mei, M. Bansal, and M. R. Walter, "Listen, attend, and walk: Neural mapping of navigational instructions to action sequences," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[4] V. Blukis, Y. Terme, E. Niklasson, R. A. Knepper, and Y. Artzi, "Learning to map natural language instructions to physical quadcopter control using simulated flight," in *Conference on Robot Learning (CoRL)*, 2019.

[5] N. Gopalan, D. Arumugam, L. L. Wong, and S. Tellex, "Sequence-to-sequence language grounding of non-Markovian task specifications," in *Robotics: Science and Systems*, 2018.

[6] D. Arumugam, S. Karamcheti, N. Gopalan, L. L. Wong, and S. Tellex, "Accurately and efficiently interpreting human-robot instructions of varying granularities," in *Robotics: Science and Systems*, 2017.

[7] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2011.

[8] T. M. Howard, S. Tellex, and N. Roy, "A natural language planner interface for mobile manipulators," in *IEEE International Conference on Robotics and Automation*, 2014.

[9] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, 2018.

[10] G. Konidaris, I. Scheidwasser, and A. Barto, "Transfer in reinforcement learning via shared features," *Journal of Machine Learning Research*, vol. 13, pp. 1333–1371, 2012.

[11] S. James, B. Rosman, and G. Konidaris, "Portable representations for high-level planning," in *Proceedings of the Thirty-Seventh International Conference on Machine Learning*, 2020.

[12] A. K. Watson, "Automated creation of labeled pointcloud datasets in support of machine-learning based perception," Naval Postgraduate School Monterey United States, Tech. Rep., 2017.

[13] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[14] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 2002, pp. 1398–1403.

[15] G. Konidaris and A. G. Barto, "Building portable options: Skill transfer in reinforcement learning." in *International Joint Conferences on Artificial Intelligence*, 2007.

[16] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Sharing clusters among related groups: Hierarchical Dirichlet processes," in *Advances in Neural Information Processing Systems 18*, 2005, pp. 1385–1392.

[17] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[18] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5239–5246.

[19] V. V. Unhelkar and J. A. Shah, "Learning models of sequential decision-making with partial specification of agent behavior," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 2522–2530.

[20] P. Ranchod, B. Rosman, and G. Konidaris, "Nonparametric Bayesian reward segmentation for skill discovery using inverse reinforcement learning," in *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 471–477.

[21] E. B. Fox, E. B. Sudderth, M. I. Jordan, A. S. Willsky *et al.*, "A sticky HDP-HMM with application to speaker diarization," *The Annals of Applied Statistics*, vol. 5, no. 2A, pp. 1020–1056, 2011.

[22] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[23] M. C. Hughes and E. B. Sudderth, "Bnpy: Reliable and scalable variational inference for bayesian nonparametric models," in *In Proceedings of the NIPS Probabilistic Programimming Workshop*, 2014, pp. 8–13.

[32] M. MacMahon, B. Stankiewicz, and B. Kuipers, "Walk the talk: Connecting language, knowledge, and action in route instructions," in *National Conference on Artificial Intelligence*, 2006.

[24] M. C. Hughes, "BNPY: Reliable and scalable variational inference for Bayesian nonparametric models," in *Proceedings of the NIPS Probabilistic Programming Workshop*, 2014.

[25] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *In Proceedings of Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[26] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems 13*, 2000, pp. 582–588.

[27] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[28] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[29] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1532–1543.

[30] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *In Robotics: Science and Systems*, 2014.

[31] M. Quigley, J. Faust, T. Foote, and J. Leibs, "ROS: an open-source robot operating system," in *In International Conference on Robotics and Automation (ICRA) Workshop on Open Source Software*, 2009.

[33] Y. Artzi and L. Zettlemoyer, "Weakly supervised learning of semantic parsers for mapping instructions to actions," in *Annual Meeting of the Association for Computational Linguistics*, 2013.

[34] J. MacGlashan, M. Babeş-Vroman, M. DesJardins, M. L. Littman, S. Muresan, S. Squire, S. Tellex, D. Arumugam, and L. Yang, "Grounding English commands to reward functions," in *Robotics: Science and Systems*, 2015.

[35] C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox, "A joint model of language and perception for grounded attribute learning," *arXiv preprint arXiv:1206.6423*, 2012.

[36] V. Blukis, N. Brukhim, A. Bennett, R. A. Knepper, and Y. Artzi, "Following high-level navigation instructions on a simulated quadcopter with imitation learning," *arXiv preprint arXiv:1806.00047*, 2018.

[37] K. M. Hermann, F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. M. Czarnecki, M. Jaderberg, D. Teplyashin *et al.*, "Grounded language learning in a simulated 3D world," *arXiv preprint arXiv:1706.06551*, 2017.

[38] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov, "Gated-attention architectures for task-oriented language grounding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.