2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636548

Talk the talk and walk the walk: Dialogue-driven navigation in unknown indoor environments

Thomas Victor Ilyevsky, Jared Sigurd Johansen, and Jeffrey Mark Siskind

***Abstract*— Prior work in natural-language-driven navigation demonstrates success in systems deployed in synthetic environments or applied to large datasets, both real and synthetic. However, there is an absence of such frameworks being deployed and rigorously tested in real environments, unknown *a priori*. In this paper, we present a novel framework that uses spoken dialogue with a real person to interpret a set of navigational instructions into a plan and subsequently execute that plan in a novel, unknown, indoor environment. This framework is implemented on a real robot and its performance is evaluated in 39 trials across 3 novel test-building environments. We also demonstrate that our approach outperforms three prior vision-and-language navigation methods in this same environment.**

I. Introduction

Imagine an office environment where individuals work in separate areas to follow social-distancing guidelines. You need to give an important package to a colleague but you're unable to go in person. You summon a robot and give it navigational instructions in plain English to your colleague's office. The robot engages you in a dialogue to clarify some ambiguity in your navigational instructions and then follows the plan it infers to deliver the package. In this paper, we present a machine-learned system that makes a significant step towards this reality.

Our system consists of two main components. The first is a transformer-based network trained to interpret spoken natural language and convert it into a navigation plan that the robot can execute. Crucially, it's designed to support multiple turns in a conversation by taking the robot's current plan and a transcript of a spoken utterance as input and producing an updated plan and follow-up question (if necessary) as output. Some previous work (e.g., [1], [2]) only considers an agent receiving a single text instruction as input and producing a single plan as output. This is impractical, however, for real-life applications that depend on spoken language and speech recognition. A person might misspeak or the speech recognition could be erroneous. This necessitates support for live dialogue to rectify any potential errors or ambiguity. To train this network, we collected a novel dataset of navigational-instruction utterances, created transcript-plan pairs, and augmented them to support multi-turn dialogue.

The second component of our system is a 2D CNN-based network trained to produce navigational goals that correspond to the plan produced by the dialogue component. Navigational instructions generally consist of left or right turns in specific locations, such as "turn right at the end of the hallway." If the robot does not have a map of its environment in advance, it would be unable to directly generate an entire navigational plan to execute these navigational instructions. Therefore, the network takes the latest map (produced by SLAM) and the current step in its plan as input to produce a navigational goal as output. It is designed to produce forward goals by default until the robot arrives at the desired intersection, in which it should produce a goal corresponding to the desired direction. Additionally, it is trained to recognize when a step in the plan cannot be executed. To train this network, we collected a novel dataset of actual trajectories driven and SLAM maps produced by a robot on several floors of several buildings.

The dialogue and navigational components are combined into a system that enables dialogue-driven navigation in an end-to-end fashion in unknown, indoor environments. To test out the effectiveness of a robotic algorithm, it is important to test in the real world where it is required to operate in continuous space with noise and unanticipated conditions. To this end, we recruited volunteers to converse with the robot and provide navigational instructions to various locations in three real buildings. These volunteers were not involved in our natural-language dataset collection. These three test buildings were not part of either the dataset used to train the dialogue component or the dataset used to train the navigation component. We evaluated whether the navigational instructions were converted into a correct plan and whether the plan was correctly executed. We demonstrate our system's performance in 39 trials. We also demonstrate in Section VIII that algorithms that may work in simulation do not necessarily perform well in the real world.

Explicitly, this paper makes the following contributions:

- We provide a novel dataset of transcript-plan pairs for navigation in indoor environments. We apply a novel data augmentation method to train a transformer network to support multi-turn dialogue, allowing the robot to ask the person clarifying follow-up questions.
- We provide a novel dataset of robot trajectories paired with navigational commands in several indoor environments. We use automatic annotation and data augmentation techniques to train a 2D CNN on this data to

This work was supported, in part, by the US National Science Foundation under Grants 1522954-IIS and 1734938-IIS, and by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number D17PC00341. Any opinions, findings, views, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, official policies, or endorsements, either expressed or implied, of the sponsors. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein. *(Corresponding author: Jeffrey Mark Siskind.)*

All authors were with the School of Electrical and Computer Engineering, 465 Northwestern Avenue, Purdue University, West Lafayette, IN, 47907 USA e-mail: {tilyevsk, jjohanse, qobi}@purdue.edu.

produce navigational goals and feedback statuses that correspond to the input instructions.
- Unlike most prior work that demonstrates performance on synthetic data or their own training environments, we demonstrate performance in real indoor environments with real volunteers that are distinct from our training sets to show the generalizability of our approach.
- We train three prior vision-and-language navigation methods on our data and deploy them them on our robot in one of our test buildings. We show that our approach vastly outperforms these methods.

II. Related Work

There has been considerable prior work on vision-and-language navigation (VLN). Some of this work, [3], [4], [5], [6], [7], [8], trained and evaluated VLN models on the Room-2-Room (R2R) dataset [3]. This dataset consists of natural-language text instructions paired with corresponding trajectories in a simulated indoor environment. These trajectories are sequences of vertices in a discrete graph, where each vertex has a panorama of images to represent the view at that vertex. [9] and [10] presented similar methods to choose waypoints to reach a goal specified by natural-language instructions but in simulated outdoor environments. Our work differs significantly from all of this work in a number of key ways.

First, in the R2R simulator, robot position is represented as a vertex in a discrete graph and visual information, although from real images, is noise-free and deterministic at each vertex. In contrast, rather than just repeatedly outputting one of a small number of adjacent graph vertices to eventually reach a goal vertex, we address a more complex problem: controlling a physical robot in the real world with a noisy continuous position and action space and noisy continuous observations. While [8] trained in a simulated environment, it tested both in simulated environments and on a real robot in real environments. However, when the navigation graph was known *a priori*, performance in the real environment was comparable to that in simulation, but results were very poor when it was unknown and waypoints were predicted on the fly. Our system is able to successfully execute navigation instructions without a known map of the environment, just with the SLAM map that is built as the robot drives.

Second, the above prior work took single-turn text as input. Our system interacts with a person in multi-turn spoken dialogue and is trained to be robust in the face of noisy speech recognition. Such dialogue is crucial for clarifying potentially ambiguous instructions. [11] and [12] presented VLN approaches that perform continuous control, rather than waypoint selection, of an autonomous vehicle within two environments in the CARLA simulator given natural-language instructions. However, they also only considered single-turn instructions as input and the trained autonomous vehicle did not engage in multi-turn dialogue. Also, in contrast, our approach takes noisy SLAM data from a real physical robot as input. This data in noisier and less rich than the synthetically generated 3D images in the CARLA simulator. [11] only evaluated their approach in the CARLA simulator, while [12] also conducted a single experiment on real data (the KITTI dataset) and one experiment on a physical electric vehicle. We, however, rigorously demonstrate our system's performance on a real robot in real indoor environments in 39 trials.

Third, we output a feedback status along with our goal coordinates. This allows for direct feedback about whether or not the input instruction was successfully executed or whether it eventually failed (because the instruction was unachievable). The latter allows the robot to detect when an instruction it was given is incorrect.

There has been prior work, [13], [14], [15], [16], [17], that, like our work, has also focused on achieving multi-turn dialogue understanding on a physical robot. However, this work operated within smaller and simpler environments than our work; we train and evaluate our system on several large unmodified office environments. [13] proposed a method to allow a robot to learn object-related concepts through dialogue. Only a single demonstration on a real robot was done in a single room with several objects on a table. [14] demonstrated a dialogue and navigation system for a physical robot; however semantic regions in the map were provided to the system in advance and there was no full evaluation of the system's performance, only preliminary experiments in a single environment. [15] constructed a framework to control a physical robot with spoken language, but it was only tested in a simulator in which the virtual robot could execute navigation commands from a finite discrete set. [16] demonstrated their algorithm on a quadcopter, but the environment was very small with just a few objects on a green surface. The environment was only varied by placing different objects in different positions. [17] collected data from a physical robot to train their algorithm to follow natural-language instructions. However, evaluation of the approach was only performed in simulation.

There has been other prior work [18], [19], [20], [21], [22], [23], that, like ours, first converts natural-language instructions into a plan that can then be interpreted and executed. [18] and [19] modeled instruction-action pairs to convert each instruction independently into an action in a noninteractive fashion to map an instruction sequence to an action sequence. Our system interacts with a person in spoken dialogue to clarify ambiguities and update the plan in the context of the dialogue. [20] and [21] determined a set of constraints from natural-language instructions and applied those constraints to a known map to generate a navigation trajectory. [22] and [24] both depended on having topological representations of the environment. Unlike our approach, these methods would not work in an unknown environment.

Finally, some prior work only presented methods for part of the VLN task. [23] and [25] demonstrated vision-only navigation to a specified target object. [26] presented a data-driven parser for understanding navigation directions for the purpose of human-robot dialogue but did not apply it to any form of navigation, simulated or real.

III. System Architecture

Our system consists of a dialogue component and navigation component. The dialogue component facilitates multi-turn conversation with a person to produce a plan. The plan is converted into commands that the navigation component then executes. The navigation component incrementally executes each command by analyzing the SLAM map and producing goal locations for the robot to drive to and a feedback status to indicate whether to advance to the next command. A diagram of our system can be seen in Figure 1.

IV. Dialogue

The purpose of engaging in dialogue with a person is to construct a plan of how to navigate to a specific destination. If the person were to formally communicate the plan to the robot, using formalisms the robot is capable of executing, this task would be trivial. What makes this nontrivial is that the communication is done informally through spoken natural language. In order to facilitate this spoken dialogue, we collected and augmented a dataset to train a transformer-based network to take as input a *current plan* and *input transcript* of the spoken utterance, and to produce an *updated plan* and *follow-up question* as output.

A. Dataset collection

To collect a dataset of navigational instructions, we recruited 27 subjects that spoke English and were familiar with three buildings on our campus. For each of four floors, for each of the three buildings (PHYS, MSEE, and EE), we described a starting location and orientation of both a robot and a passerby. We asked them to imagine they were the passerby and that the robot had posed a single query, asking for navigational instructions to a location within that building. They were allowed to provide partial instructions or indicate they did not know the location of the destination. The subjects recorded a verbal response to each query and we used Microsoft's speech-to-text engine [27] to convert the spoken responses into text. We collected a total of 8797 navigational-instruction utterances and produced their corresponding transcripts. Some navigational-instruction utterances involve using an elevator to move to another floor. In this work, we handle the dialogue component of using an elevator to move between floors, but focus on performing navigation in a single floor. Future work will address accomplishing multi-floor navigation.

We identified a set of *plan concepts* of various kinds, including directions, intersections, goals, and other that cover the vast majority of navigational instructions contained in our transcripts, so that each transcript would have a corresponding plan with a sequence of these concepts. All plan concepts, their types and definitions can be found in Table I. We posted each transcript on Amazon Mechanical Turk (AMT) and asked two workers to construct a coherent plan using these plan concepts. Workers were native English speakers who had to pass five custom qualification tests involving this task before annotating the data.

TABLE I: Possible plan concepts for plan annotation. `int-L` and `int-R` have a left or right turn, respectively. `end` refers to the end of the hallway; `elbow` refers to an elbow. The first four direction concepts refer to the direction the robot should drive in. It will continue moving in that direction until it encounters the next step in the plan. `either` refers to turning left or right at an elbow when it is not explicitly stated (e.g., "go around the corner"). `goal-F`, `goal-L`, and `goal-R` respectively refer to goals that are ahead, on the left, or on the right. □ refers to an unknown or unspecified step in the plan. `change-floor` refers to using an elevator to move between floors.

Type	Plan concepts
Intersections	`int-L`, `int-R`, `end`, `elbow`
Directions	`turn-around`, `forward`, `left`, `right`, `either`
Goals	`goal-F`, `goal-L`, `goal-R`
Other	□, `change-floor`

When the constructed plan had a □, where □ refers to an unknown or unspecified step in the plan, we asked the AMT workers to type a follow-up question that they would ask to resolve the □. Each of the 8797 transcripts had two plan annotations and (potentially) two follow-up questions. When the plans and follow-up questions were the same, we created a single sample. When the plans and follow-up questions were different, we kept both if they were both reasonable. Otherwise, we kept the most correct one. If neither were correct, we manually modified one to be correct and retained this as a sample. This determination was done by the authors. This process resulted in a total of 9818 unique samples consisting of [□] for the current plan, an input transcript, an updated plan, and a follow-up question. If the updated plan was a *complete plan* (i.e., no □ present), we used a default follow-up question of "Got it. Thanks!" Of the total number of samples, 52.3% had both annotated plans accepted and were identical to one another, 7.9% had both annotated plans accepted and were different but reasonable interpretations of the navigational instructions, 32.5% had one plan accepted (and the other discarded), 4.8% were manually modified by the authors, and 2.4% were discarded (due to incoherent speech-to-text results).

From among the 9818 samples, we found that 7467 (76.1%) had complete plans and 2351 (23.9%) had *partial plans* (i.e., a □ present). We sorted the samples with a partial plan into four categories: *empty*, *need-elevator*, *need-first*, and *need-last*, based on their plan pattern (see Table II). Although these partial plan categories occur less frequently in our dataset than samples with a complete plan, they are realistic possibilities during live spoken dialogue due to pauses in speech, poor speech recognition, or incomplete navigational instructions. Therefore, we augmented our dataset to increase the number of samples in these categories. For *empty*, we used an online tool[1] to generate random sentences that had an updated plan of [□]. These training samples were used to help train the network to handle sentences that did not

[1] https://randomwordgenerator.com/sentence.php

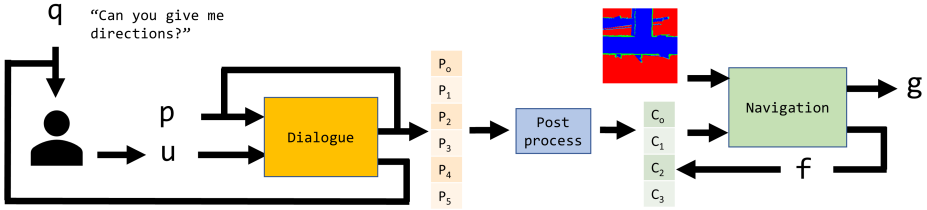


Fig. 1: System Diagram. A question, q, is posed to a person. Their utterance, u, and the current plan, p, is fed into the dialogue component, which produces an updated plan and follow-up question. Dialogue loops until a complete plan, $[p_i]$, is produced. The complete plan is converted into robot commands, $[c_i]$, which are fed into the navigation network. The navigation component produces a goal location, l, and feedback status, f, which are used to carry out all commands.

TABLE II: Number of samples for each partial plan category.

Category	Pattern	Original count	Augmented count
empty	[☐]	125	2125
need-elevator	[☐, `change-floor`, ...]	2100	2100
need-first	[☐, ¬`change-floor`, ...]	68	2200
need-last	[..., ☐]	58	1818

contain navigational information. The category *need-elevator* had a relatively large number of samples, so we did not perform any augmentation. For *need-first*, we used samples with complete plans, but removed certain keywords (e.g., "go straight" or "turn-around") from the beginning of the transcripts and replaced the first instruction in the plan with ☐. We performed a similar augmentation for *need-last*, but truncated text from the end of the transcript and replaced the corresponding concepts in the plan with ☐. Table II shows the total sample counts before and after augmentation.

B. Dialogue turn generation

The purpose of dialogue is to rectify any missing information (i.e., ☐) in the current plan. The data we collected only simulates the first turn in a conversation, in which the current plan is [☐], and the input transcript contains complete, partial, or no information to the destination. To facilitate dialogue, we must train the network to handle subsequent turns of conversation in which a person responds to follow-up questions produced by prior turns. This requires further augmenting the dataset to create samples in which the current plan belongs to one of the partial plan categories other than *empty*, the input transcript corresponds to the missing information in the current plan, the updated plan accurately reflects the current plan integrated with the missing information, and a valid follow-up question. To distinguish from the samples described in Section IV-A, we refer to these additional training samples as *follow-on samples*.

To generate these follow-on samples, we use custom logic to combine different samples from our dataset. We first find partial plans to serve as the current plan input in the follow-on samples. Then, based on the index of [☐] in each partial plan, we find a transcript, or piece of a transcript, in our dataset whose plan annotation would appropriately replace the [☐] in the current plan. This transcript then serves as the input transcript for the new follow-on sample. To create the updated plan for this sample, we replace the [☐] in the current plan with the plan that corresponds to the transcript used.

TABLE III: Training samples.

current plan	[☐]
input transcript	yeah, go straight and then make a right
updated plan	[`forward`, `int-R`, `right`, ☐]
follow-up question	What do I do after turning right?
current plan	[☐, `end`, `left`, `goal-R`]
input transcript	think you might have to turn around
updated plan	[`turn-around`, `end`, `left`, `goal-R`]
follow-up question	Got it. Thanks!

TABLE IV: Partial plan follow-up questions.

Category	Follow-up question
empty	Repeat original question.
need-elevator	Ask for navigational instructions to the elevator.
need-first	Ask which direction to start out going.
need-last	Ask what do to after last instruction.

Lastly, we determine the follow-up question for the follow-on sample by using the rules in Table IV depending on the partial plan category of the updated plan. If the updated plan is complete, then the follow-up question is a "thank you". Table III shows one first-turn sample and one follow-on sample.

C. Augmented dataset training and validation

With this additional augmentation, the network can be trained to support all turns in a dialogue. Given a current plan and transcript as input, it can learn to produce an updated plan and a relevant follow-up question as output. It is trained to fill in the missing plan concepts in the current plan with the information provided by the person's response in the input transcript. During training and validation, the accuracy of the updated plan output is determined by whether or not it matches the target updated plan exactly. The relevance of the follow-up question is measured by whether or not it corresponds to the partial plan category of the target updated plan based on the rules defined in Table IV. This was done computationally by comparing the text of the follow-up question with the list of questions from that partial plan category.

We used the source code at [28] to train a network based on Text Summarization with Pretrained Encoders [29] to take the current plan and input transcript as input and to produce an updated plan and follow-up question as output. For training, we specified a maximum input length of 200 tokens. We divided the subjects into five folds to perform

TABLE V: Commands produced from plan subsequences.

Plan subsequences	Command	Description
`end,left`	**end_left**	turn left at end of hallway
`end,right`	**end_right**	turn right at end of hallway
`int-L,forward`	**int-L_forward**	go forward when left available
`int-L,left`	**int-L_left**	turn left when left available
`int-R,forward`	**int-R_forward**	go forward when right available
`int-R,right`	**int-R_right**	turn right when right available
`turn-around`	**int-B_backward**	turn around when possible
`elbow,left`	**elbow_left**	turn left at elbow
`elbow,right`	**elbow_right**	turn right at elbow
`elbow,either`	**elbow_either**	go through elbow

leave-one-fold-out cross validation. The average validation accuracy was 69.9% and the average follow-up question relevance was 99.0%.

D. Spoken dialogue

When initiating a conversation, the plan is initialized to [□] and begins with the robot posing a question. It waits for a response, which is transcribed to produce a transcript. The current plan, along with this transcript, are fed into the network, which produces an updated plan and follow-up question. The robot poses the follow-up question and this process repeats itself until the updated plan has no □.

To help facilitate a more natural conversation, we have a small amount of code to address a few corner cases that may arise in spoken conversation. If a response is not heard within 5 s, the robot states this fact and repeats its question. If the robot is not able to extract any information from a response, it indicates such and may provide some information about what it does understand (e.g., directions and intersections). If the robot goes two turns without the plan changing (e.g., it fails to understand the navigational instructions or hears no response), the robot ends the conversation and carries out whatever portion of the plan is usable.

V. Navigation Commands

Once a complete plan is produced, it is post-processed to take the plan concepts and convert them into a sequence of commands for the navigation component to execute. The commands are generated from plan subsequences by pairing adjacent intersection and direction concepts in the plan, with one exception: `turn-around`, which is treated as a stand-alone command. Table V shows these conversions.

VI. Navigation

The navigation component executes each command by continuously looking at the SLAM map to determine a goal location where the robot should drive and a feedback status to indicate whether to advance to the next command. When it arrives at the intersection indicated by the command, the navigation component predicts a goal location that executes the turn indicated by the command and a feedback status of *transition* indicating it can advance to the next command. However, if the robot is in the middle of a hallway, the navigation component predicts a goal location that drives it down the hallway and a feedback status of *forward* indicating that the command has not been executed. If it reaches the end of the hallway without detecting the intersection, the navigation component stops the robot and produces a feedback status of *failure* indicating that the robot has failed to execute the command. We illustrate this in Figure 2 which depicts three SLAM maps and commands depicting different scenarios along with their corresponding feedback statuses. The robot's position and orientation is depicted by the light blue arrow and the yellow circle represents the goal location where the robot should drive.

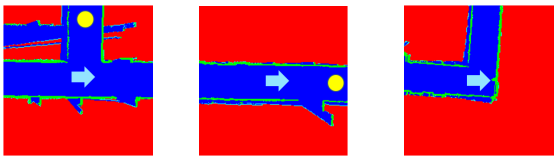
A. Network

We implement this navigation with a neural network that takes the command and SLAM map as input and predicts a goal location as (x, y) coordinates and feedback status with a 3-way classifier. The neural network consists of 4 convolutional layers that extract features from the SLAM map. These features are concatenated with a one-hot encoding of the input command which is passed through 2 fully-connected layers to create a final representation of the input. This representation is passed into two parallel layers, to predict the coordinates and feedback status, respectively.

B. Training

To train this network, we required training samples that consist of a SLAM map, input command, goal location, and feedback status. To collect the SLAM maps, we manually navigated the robot on several floors of three buildings (PHYS, MSEE, and EE), ensuring to cover every hallway. The robot was intentionally driven in the centers of hallways and turned in the centers of intersections to simulate ideal navigation. We used Google Cartographer [30] to determine the robot's positions and generate the SLAM maps as it was driving, and these were recorded at a rate of 10 Hz.

Each SLAM map is used to create multiple training samples by pairing each of the commands in Table V. Then, for each map-command pair, we determine the corresponding goal location and feedback status. To do this, we first must determine what directions are available to the robot in the SLAM map by using the robot positions that were recorded. A direction is considered available if there is a recorded position in the SLAM map in that direction within 5 m of the SLAM map's center. We also use these positions to



(a) **int-L_left** (b) **elbow_either** (c) **end_right**

Fig. 2: (a) The robot can execute the command; it produces a goal location that drives it into the hallway on its left and a feedback status of *transition*. (b) The robot is in a hallway and cannot execute the command yet; it produces a goal location that drives it further down the hallway and a feedback status of *forward*. (c) The robot has reached the end of the hallway and cannot make a right turn; it stops and produces a feedback status of *failure*.

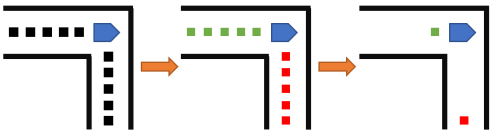


Fig. 3: Left: Black squares represent all positions within 5 m of the center of the SLAM map. Center: Green squares represent positions corresponding to *backward*. Red squares represent positions corresponding to *right*. Right: Green square is goal location for *backward* and red square is goal location for *right*. Available directions are *backward* and *right*; using Table VI, the intersection types are `int-R`, `end`, `int-B`, and `elbow`.

TABLE VI: Intersection types based on available directions.

forward	backward	left	right	Intersection types
no	no	no	yes	`int-R`
no	no	yes	no	`int-L`
no	no	yes	yes	`hallway`, `int-L`, `int-R`
no	yes	no	no	`end`, `int-B`
no	yes	no	yes	`int-R`, `end`, `int-B`, `elbow`
no	yes	yes	no	`int-L`, `end`, `int-B`, `elbow`
no	yes	yes	yes	`int-L`, `int-R`, `end`, `int-B`
yes	no	no	no	`hallway`
yes	no	no	yes	`int-R`
yes	no	yes	no	`int-L`
yes	no	yes	yes	`int-L`, `int-R`
yes	yes	no	no	`hallway`, `int-B`
yes	yes	no	yes	`int-R`, `int-B`
yes	yes	yes	no	`int-L`, `int-B`
yes	yes	yes	yes	`int-L`, `int-R`, `int-B`

determine a goal location for each available direction. This process is illustrated in Figure 3. Then we use the available directions to determine the intersection type(s) based on the correspondence indicated in Table VI.

We can now use this information to form the training samples. For each command, if the SLAM map has the intersection corresponding to it, we make a sample whose goal location corresponds to the direction specified by the command and whose feedback status is *transition*. If the SLAM map does not have the intersection, but *forward* is an available direction, we make a sample whose goal location corresponds to *forward* and whose feedback status is *forward*. If the SLAM map does not have the intersection and *forward* is not available, we make a sample whose goal location is $(0.0, 0.0)$, and whose feedback status is *failure*.

We train the network on these samples, using two loss functions for each of the respective outputs. For the 3-way feedback status classifier, we use weighted cross-entropy and for regressing to the (x, y) goal coordinates we use mean-squared-error. During training, we randomly apply rotation and translation (y-axis only) transformations to simulate the robot having an off-center position and orientation in a hallway, allowing the network to still predict correct goals.

C. Command Execution

To achieve autonomous navigation on the robot,we apply the trained network continuously to the current input command and SLAM map at a rate of 10 Hz. Because noisy output is possible with noisy SLAM data, we consider multiple consecutive outputs from the network to make navigation decisions. This accomplishes two crucial things: the robot will not navigate incorrectly due to a single spurious output and the robot will be able to catch an intersection if it is driving past it. Specifically, the navigation component considers the 10 most recent outputs, feedback statuses and goal locations, of the network. Among these outputs, if the most common feedback status is *forward*, the robot will drive to the goal location of the most recent output. However, if it is *transition* or *failure* the robot will stop moving and wait for 10 additional outputs to make a final decision. If the most common feedback status of these outputs is *transition*, the robot computes the aggregates the goal locations of only the outputs whose status is *transition*. The robot will then drive to the aggregated goal location. Once that navigation is complete, the robot will then take the next command as input into the network. If it is *failure*, however, the robot will stop and not attempt to execute any more commands.

VII. System Experiments

To evaluate the performance of our entire system, we tested in three buildings that were not part of our training sets.[2] These test buildings consist of multiple hallways and alcoves of varying lengths and widths, with common objects, such as water-fountains, trashcans, and chairs, found in various places (see Figure 4 for example floor plans). Although they are all office buildings, they represent a common indoor environment. Other buildings such as supermarkets, shopping malls, schools, libraries, airports, and factories also predominantly consist of grid-like hallways and intersections and contain the same aforementioned objects.

The system is deployed on a Clearpath Husky A200™ UGV robot equipped with an Open IMU UM7, Velodyne VLP-16 3D LiDAR, Blue Yeti microphone, and a System76 Laptop with two Nvidia GeForce GTX 1080 GPUs. We placed this robot in different locations on different floors of the three new buildings. In each location, it was tasked with asking a person for navigational instructions to a goal location and then following the plan it constructed. We recruited volunteers, who were not part of our training set, to engage in the dialogue. After constructing a plan, it attempted to execute the commands to reach the hallway containing the goal location. The ROS *move_base* package [31] was used for autonomous path planning, with obstacle avoidance, to driving goals produced by the navigation network. A demonstration of our system can be seen in the attached video submission.

We consider several different forms of successes. If the navigational instructions were interpreted and followed correctly, this is a complete success. If the navigational instructions were interpreted incorrectly and the robot either followed the interpretation correctly or signified an inability to do so correctly, this is a navigation success but dialogue

[2]Our dataset can be found at https://dx.doi.org/10.21227/zxk7-ca24. The software used to produce the results in this paper can be found at https://github.com/qobi/iros2021.

TABLE VII: Experimental results.

	HAMP	KNOY	ME	Total
Complete Success	5	10	6	21
Dialogue Success/Navigation Failure	2	3	5	10
Dialogue Failure/Navigation Success	4	1	3	8
Complete Failure	0	0	0	0
Total	11	14	14	39

failure. If the navigational instructions were interpreted correctly, but the navigation was unsuccessful, this is a dialogue success but navigation failure. If both were unsuccessful, it is a complete failure. Any trial that required manual intervention was considered a failure.

Table VII shows a breakdown of the trials and their corresponding category on a per-building basis. The dialogue and navigation components had high success rates of 79.4% and 74.3%, respectively. In the case of dialogue, its success rate is slightly better than the validation accuracy reported in Section IV-C. In 92.3% of the trials, the dialogue component produced plans whose first 3 concepts were correct, which means that for a strong majority of the time, the robot would have a plan that starts it going in the right direction. In these trials, the robot always succeeded at reaching the intersection corresponding to the current input command by correctly outputting the *transition* classification. However, in 9 of the 10 trials with navigation failure, the robot was unable to predict an accurate driving goal while stopped in the correct intersection, even after rotating, and indicated failure. In the other single trial, manual intervention was employed to prevent the robot from crashing while turning a corner to execute the first command and allowed the robot to successfully complete the second command autonomously. In 5 of the 39 trials, the robot's sequence of commands did not correspond to the environment either due to misinterpretation in dialogue or a mistake on the volunteer's behalf. In all 5 of these trials, the robot correctly drove forward until eventually predicting the *failure* feedback status at the end of a hallway to indicate that the command could not be executed.

VIII. Comparison Experiments

We compared our method to three prior VLN approaches [5], [6], [7] on the task reported in Section VII. While we evaluated using the same kind of live trials reported in Section VII, we needed a training set with the same structure as R2R, where all navigation is performed on a fixed graph with images available for all orientations at each vertex, Thus we collected an additional training set, in the style of R2R,

Fig. 4: One of the four floor plans from each building. Our training set consists of EE, MSEE, and PHYS. Our test set consists of HAMP, KNOY and ME. Green indicates hallways.

using our robot in the same three buildings (PHYS, MSEE, and EE) where we collected the training sets reported in Section IV-A and Section VI-B, by placing the robot at fixed waypoints, approximately 3 m apart, and collected 36 images at each waypoint at different camera pan and tilt positions. We then paired the natural-language utterances reported in Section IV-A with the corresponding sequences of these waypoints to create the new training set. This allowed us to construct 2,814 training samples. We trained SF [6], RCM [7] and Babywalk [5] on R2R, using the code base provided by [5], then fine tuned on this new training set. During training, the success rate was evaluated on the training set, where success is defined as the VLN agent stopping within 5 m of the target waypoint. The training success rates for SF, RCM, and Babywalk were 98.2%, 84.0%, and 99.8%, respectively.

Evaluating with live trials in novel environments in a different building, however, required us to map the small finite action space output by models trained with the prior VLN approaches to the continuous action space needed to perform our task. During the trials, upon arrival at a waypoint, the robot collected 36 images at the same pan and tilt angles described above, extracted their features as described in [5], and decided whether to drive to an adjacent waypoint or indicate completion. We determined adjacent waypoints by searching the robot's SLAM map for positions in free space that corresponded to positions in front of, behind, left of, and right of the robot's current position.

A strict evaluation of success of a purely autonomous method that cascades the prior models that output discrete navigation actions designed to navigate in a symbolic graph with our code that instead navigates to physical waypoints from those discrete actions leads to very low success rate. Thus we relaxed our success criterion to manually intervene if the robot got too near an obstacle. We moved the robot away from the obstacle to allow the trial to continue.

We conducted eight trials with each of the three methods in KNOY, where our approach had its highest success rate. We positioned the robot in the same starting location and orientation used in eight of our trials from Section VII and provided the same corresponding text instructions. A trial was concluded when the method predicted the stop condition. If the robot was in the hallway with the goal location, we considered it a success. We then manually drove the robot to the beginning of the hallway with the goal location. Using odometry, we measured the navigation error, which is the driving distance from the ending location to the beginning of the hallway with the goal location. Table VIII shows that all three of the methods failed to correctly execute the instructions specified by the input utterances in all eight of their respective trials. Table IX shows a breakdown of these trials, including the number of waypoints traversed and how many waypoints the robot successfully navigated before heading off in a wrong direction. In some, the robot would simply start driving in the opposite direction from that indicated by the instructions. In others, it would miss the turn it was supposed to take or start driving back to where it came from. Our system has a much higher success rate and

TABLE VIII: Comparison of our system with prior VLN systems. Successful trials were those where the robot reached the hallway with the goal location.

	Successful #Trials	Total #Trials	Success Rate
SF [6]	0	8	0.0%
RCM [7]	0	8	0.0%
Babywalk [5]	0	8	0.0%
Our System	10	14	**71.4%**

TABLE IX: Navigation statistics from the results reported in Table VIII. Correct Waypoints is the average number of waypoints the system correctly predicted and navigated to before heading off in a wrong direction. Total Waypoints is the average number of waypoints navigated to before predicting the stop condition. Navigation Error is the average driving distance from the ending location to the goal location.

	Correct Waypoints	Total Waypoints	Navigation Error
SF [6]	3.9	26.8	25.6 m
RCM [7]	3.5	15.8	38.9 m
Babywalk [5]	0.1	8.9	29.4 m
Our System	n/a	n/a	**3.6** m

far lower navigation error.

IX. Conclusion

We have demonstrated an end-to-end system, deployed on a real robot in a real environment, that can interpret instructions through spoken dialogue as a sequence of commands and then execute those commands. We have also shown that our approach outperforms prior VLN methods when applied to the task of a real robot understanding instructions in a real unknown environment.

References

[1] C. Matuszek, D. Fox, and K. Koscher, "Following directions using statistical machine translation," in *International Conference on Human-Robot Interaction*, 2010, pp. 251–258.

[2] J. H. Oh, A. Suppé, F. Duvallet, A. Boularias, L. Navarro-Serment, M. Hebert, A. Stentz, J. Vinokurov, O. Romero, C. Lebiere, *et al.*, "Toward mobile robots reasoning like humans," in *Conference on Artificial Intelligence*, 2015, pp. 1371–1379.

[3] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. Van Den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Computer Vision and Pattern Recognition*, 2018, pp. 3674–3683.

[4] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer, "Vision-and-dialog navigation," in *Conference on Robot Learning*, 2020, pp. 394–406.

[5] H. Zhu, Wang and-Hu, J. Chen, Z. Deng, V. Jain, E. Ie, and F. Sha, "Babywalk: Going farther in vision-and-language navigation by taking baby steps," in *Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 2539–2556.

[6] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell, "Speaker-follower models for vision-and-language navigation," in *Conference on Neural Information Processing Systems*, 2018, pp. 3318–3329.

[7] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Computer Vision and Pattern Recognition*, 2019, pp. 6629–6638.

[8] P. Anderson, A. Shrivastava, J. Truong, A. Majumdar, D. Parikh, D. Batra, and S. Lee, "Sim-to-real transfer for vision-and-language navigation," in *Conference on Robot Learning*, 2020, pp. 1–11.

[9] H. Chen, A. Suhr, D. Misra, N. Snavely, and Y. Artzi, "Touchdown: Natural language navigation and spatial reasoning in visual street environments," in *Computer Vision and Pattern Recognition*, 2019, pp. 12 538–12 547.

[10] H. de Vries, K. Shuster, D. Batra, D. Parikh, J. Weston, and D. Kiela, "Talk the walk: Navigating New York City through grounded dialogue," *arXiv:1807.03367*, 2018.

[11] J. Roh, C. Paxton, A. Pronobis, A. Farhadi, and D. Fox, "Conditional driving from natural language instructions," in *Conference on Robot Learning*, 2020, pp. 540–551.

[12] N. Sriram, T. Maniar, J. Kalyanasundaram, V. Gandhi, B. Bhowmick, and K. M. Krishna, "Talk to the vehicle: Language conditioned autonomous navigation of self driving cars," in *International Conference on Intelligent Robots and Systems*, 2019, pp. 5284–5290.

[13] J. Thomason, A. Padmakumar, J. Sinapov, N. Walker, Y. Jiang, H. Yedidsion, J. Hart, P. Stone, and R. Mooney, "Jointly improving parsing and perception for natural language commands through human-robot dialog," *Journal of Artificial Intelligence Research*, vol. 67, pp. 327–374, 2020.

[14] C. Theobalt, J. Bos, T. Chapman, A. Espinosa-Romero, M. Fraser, G. Hayes, E. Klein, T. Oka, and R. Reeve, "Talking to Godot: Dialogue with a mobile robot," in *International Conference on Intelligent Robots and Systems*, 2002, pp. 1338–1343.

[15] S. Lukin, F. Gervits, C. Hayes, P. Moolchandani, A. Leuski, J. G. Rogers III, C. S. Amaro, M. Marge, C. Voss, and D. Traum, "Scoutbot: A dialogue system for collaborative navigation," in *Annual Meeting of the Association for Computational Linguistics*, 2018, pp. 93–98.

[16] 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*, 2020, pp. 1415–1438.

[17] S. Banerjee, J. Thomason, and J. J. Corso, "The RobotSlang benchmark: Dialog-guided robot localization and navigation," in *Conference on Robot Learning*, 2020, pp. 1–10.

[18] D. L. Chen and R. J. Mooney, "Learning to interpret natural language navigation instructions from observations," in *Conference on Artificial Intelligence*, 2011, p. 859–865.

[19] H. Mei, M. Bansal, and M. Walter, "Listen, attend, and walk: Neural mapping of navigational instructions to action sequences," in *Conference on Artificial Intelligence*, 2016, p. 2772–2778.

[20] T. M. Howard, S. Tellex, and N. Roy, "A natural language planner interface for mobile manipulators," in *International Conference on Robotics and Automation*, 2014, pp. 6652–6659.

[21] Z. Hu, J. Pan, T. Fan, R. Yang, and D. Manocha, "Safe navigation with human instructions in complex scenes," *Robotics and Automation Letters*, vol. 4, no. 2, pp. 753–760, 2019.

[22] G. Sepulveda, J. C. Niebles, and A. Soto, "A deep learning based behavioral approach to indoor autonomous navigation," in *International Conference on Robotics and Automation*, 2018, pp. 4646–4653.

[23] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihls, M. Yatskar, and A. Farhadi, "RoboTHOR: An open simulation-to-real embodied AI platform," in *Computer Vision and Pattern Recognition*, 2020, pp. 3164–3174.

[24] X. Zang, A. Pokle, M. Vázquez, K. Chen, J. C. Niebles, A. Soto, and S. Savarese, "Translating navigation instructions in natural language to a high-level plan for behavioral robot navigation," *arXiv:1810.00663*, 2018.

[25] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *International Conference on Robotics and Automation*, 2017, pp. 3357–3364.

[26] R. Meena, G. Skantze, and J. Gustafson, "A data-driven approach to understanding spoken route directions in human-robot dialogue," in *Conference of the International Speech Communication Association*, 2012, pp. 226–229.

[27] Microsoft, "Azure speech-to-text," 2020, retrieved April 1, 2020 from https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/.

[28] C. Schäfer and D. Tran, "Headliner," 2020, retrieved March 14, 2020 from https://github.com/as-ideas/headliner.

[29] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," in *Empirical Methods in Natural Language Processing*, 2019, pp. 3730–3740.

[30] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LiDAR SLAM," in *International Conference on Robotics and Automation*, 2016, pp. 1271–1278.

[31] E. Marder-Eppstein, "move_base," 2018, retrieved Sept 1, 2018 from http://wiki.ros.org/move_base.