Find a Way Forward: a Language-Guided Semantic Map Navigator

Zehao Wang1, Mingxiao Li2, Minye Wu1, Marie-Francine Moens2, Tinne Tuytelaars1

Abstract—In this paper, we introduce the map-language navigation task where an agent executes natural language instructions and moves to the target position based only on a given 3D semantic map. To tackle the task, we design the instruction-aware Path Proposal and Discrimination model (iPPD). Our approach leverages map information to provide instruction-aware path proposals, i.e., it selects all potential instruction-aligned candidate paths to reduce the solution space. Next, to represent the map observations along a path for a better modality alignment, a novel Path Feature Encoding scheme tailored for semantic maps is proposed. An attention-based Language Driven Discriminator is designed to evaluate path candidates and determine the best path as the final result. Our method can naturally avoid error accumulation compared with single-step greedy decision methods. Comparing to a single-step imitation learning approach, iPPD has performance gains above 17% on navigation success and 0.18 on path matching measurement nDTW in challenging unseen environments.

I. INTRODUCTION

The general-purpose robot assistant of the future assists humans with daily tasks to reduce labor overhead, e.g., as a housekeeping or indoor service robot. An essential part of the human-robot interaction involves language guided navigation, i.e., enabling the robot to execute instructions given by a human to reach a target location. This requires the robot to interpret the natural language instruction, ground them in (usually visual) observations, and move accordingly. This is a challenging task that previously has been addressed mostly using single-step greedy decision methods, such as imitation learning or reinforcement learning (e.g., [1], [2]) combined with pre-trained networks for encoding the visual [3], [4] and language inputs [5], [6]. These approaches are fully learning-based, i.e., mapping inputs to outputs without performing any geometric reasoning or relying on other types of domain knowledge. They are also ‘greedy’ in the sense that decisions are taken locally (“what is the next best move?”) rather than planning ahead, overlooking the entire trajectory. This situation reduces error tolerance and does not allow considering alternative routes.

In contrast, robot navigation has been an active area of research for decades, with well-established path planning methods on occupancy maps that have proven their value in a myriad of real-world applications. The question then naturally arises how this can be leveraged in language-guided navigation.

Some recent works for language-guided navigation include a pre-exploration phase (e.g., [7]), during which the robot explores the environment beforehand. This is introduced to tackle the domain gap by finetuning the visual encoder and has been shown to have a significant impact on the performance. Such a pre-exploration phase seems reasonable for most practical applications. Besides, in most scenarios, the layouts of the environment are mostly stationary. We usually need to carry out multiple navigation tasks in the same scene. For example, we may give orders to a housekeeping robot who always works in an indoor space, like at your home; Or the service robot of a hotel needs to deliver things to customers inside the building. In such scenarios, we can pre-scan the environments, obtain detailed semantic navigation maps, and rely on these for the actual navigation – in combination with obstacle avoidance to cope with dynamic objects. These maps may include not only the geometry of the environment, as in a traditional occupancy map, but also its semantics, such as the location of the main objects, which are helpful to ground the language instructions.

In this paper, we introduce the Map-Language Navigation task (MLN), which carries out navigation based on the 3D semantic map of environments and natural language instructions. Compared with traditional visual-language navigation (VLN) using RGBD observation during navigation, we see two main benefits of conducting navigation based on a 3D semantic map. First, the map provides global
information about the environment, which prevents the agent from being stuck in local decisions and makes it possible to plan the whole trajectory directly. Second, the maps of different environments share the same semantic definitions. This helps the agent to generalize to a previously unseen environment. To tackle this task, different from previous works [8], [9] which simply treated the map as additional visual inputs, we propose an instruction-aware Path Proposal and Discrimination model (iPPD) following planning and scoring pipeline, which is proved, in our experiments, to be a better way of leveraging map potential. Our model uses a modular design that consists of three main components. In the First module, we adopt deterministic algorithms to analyze the instructions and leverage them for path proposals. In particular, we extract key components (objects and turning actions) from the instruction and use them to reduce the solution space. For instance, in Fig. 1 candidate path by key component object 4 is filtered out since it violates key component “turn right”. Second, we introduce a semantic feature extraction module to encode the environment’s information along a path into a feature sequence. iPPD perceives the environments in an egocentric perspective for a set of key points along a path candidate. A novel feature extraction scheme is introduced to obtain permutation invariant features at the object level. We also take the low-level features of the path, e.g., moving directions and locations, into account. Third, we use a multimodal transformer encoder to fuse path features and instructions and further score paths. The final prediction is determined according to the ranking of these scores.

We evaluate and compare our iPPD method with traditional single-step decision methods. Comprehensive experiments show that the proposed method can leverage the maps well and can produce effective results.

II. RELATED WORK

Vision-and-Language Navigation. In the VLN [10] task, an agent needs to navigate to a goal location in a photo-realistic virtual environment following a given natural language instruction. Research on the VLN task has made significant progress in the past few years. Attention mechanisms across different modalities are widely used to learn the alignment between vision and language, hence boosting the performance [1], [7], [11]–[16]. The improvements of VLN also come from new learning approaches, such as imitation learning (IL) [17], reinforcement learning (RL) [3], [7], or the ensemble of both IL and RL [2]. Another line of work focuses on exploiting a map of the environment in navigation task [18]–[22]. Previous works mainly follow the pipeline of building the 2D map using RGB and depth images on the fly first, then integrating the built map into the model as additional visual input. Different works construct maps of different quality, which makes it hard to measure whether the performance difference comes from the navigation algorithm or the map quality. Furthermore, the 2D map inevitably drops some important environment information; for example, when two objects are stacked at one location, only one will appear.

In this work, we decouple the visual-language navigation task to map construction and map navigation. We focus on the second part and propose a language-guided map navigation task where a perfect 3D map of the environment is given and the agent can only use the language and map information, aiming to provide a base for future research to focus on the design of language-guided navigation algorithms.

Multi-modal Transformer The transformer model has achieved great success on natural language processing and vision-language tasks. [5] first pre-train the BERT model on large scale text data and achieve state of the art performances on a wide range of natural language processing (NLP) tasks. Inspired by the great success of BERT on NLP, a lot of researches extend the transformer model to process both vision and language information. [5], [23] propose a two-stream BERT model to first encode texts and images separately then fuse the two modalities via a cross modal attention layer. Another line of work introduces single stream multimodal BERT, which directly processes both vision and language information simultaneously using cross modal attention layers [24]–[26]. Different from previous works which mainly focus on studying transformer architectures that handle vision and language data, in this work we further exploit the transformer model and extend it to process language and 3D map information.

III. OVERVIEW

Given the start position $v_0$ in an environment, a natural language instruction $T$ and the voxelized 3D semantic map $M$ of the environment, our model (iPPD) aims to find a path $\hat{m}$ following the instruction. We formulate it as:

$$\hat{m} = F_{iPPD}(M, T, v_0),$$

where $F_{iPPD}(\cdot)$ is the proposed navigator. Each voxel in $M$ has three labels, i.e., navigable (binary label) and object label (from the label set $\mathcal{Q}$).

We divide the proposed pipeline into three modules, as demonstrated in Figure 2. Specifically, in the first part (§[IV-A]), we design an instruction-aware path proposing method to obtain candidate paths. In the second part (§[IV-B]), we discretize each candidate path and embed local observations into a feature sequence. iPPD extracts feature from egocentric views of the local 3D semantic map $M$ along the path. Together with low-level features, such as moving directions and locations, we construct the overall path features. Lastly, based on these path features, a language-driven discriminator (§[IV-C]) scores each candidate path, and the path with the highest score is taken as the final answer.

IV. METHOD

Exploiting a semantic map for language-guided navigation is nontrivial. Naively integrating maps into existing imitation learning approaches [1] does not have satisfactory performance, as demonstrated in experiments (§[V-D]). Therefore, we design a novel way to handle map-language navigation by path proposing and scoring.
A. Instruction-aware Candidate Path Proposing

In language-guided navigation, the agent has to navigate to the destination following the given instruction. Landmarks and moving actions mentioned in the instruction provide clues for finding the correct trajectory. We first propose an algorithm to extract key components. Then we do optimal path searching constrained by the extracted key component sequence with the A* algorithm to obtain path proposals.

Key component extraction. The key component sequence is constructed from two aspects, i.e., object landmarks and moving actions. Since similar meaning can have various forms of expression in natural language, our key component extraction relies on part-of-speech (POS) tags [27], [28], such as “NNS” stands for plural nouns, and synsets of word tokens in each instruction. We process each instruction by Algorithm 1. A manually verified reference dictionary \( S_{obj} \) mapping the object label \( q \) to its synset is prepared beforehand. In the landmark extraction step, we identify the synset \( s \) of each noun \( w.text \) in the context of instruction \( T \) with the technique of Word Sense Disambiguation (WSD) [29]. Then we iteratively assign the closest object label \( \tilde{q} \) to each noun measured by their synset-level WuPalmer similarity score. \( ^2 \) Noun words with scores higher than a threshold \( \gamma \) are treated as key components. To keep a balance between recall and precision, we manually tune \( \gamma \) and set it to 0.85. On the other hand, the moving action is recognized by pattern matching. We formulate it as a rule-based flow in Algorithm 1, line 9-17. We consider "turn left" and "turn right" as important moving actions to constrain proposal paths. Turn left or turn right actions are identified if a sequence of tokens meet the following conditions: the first token is a verb, a 'left' or 'right' token can be found within the next \( k = 3 \) tokens and the immediate following token of 'left' or 'right' is neither a noun nor a 'turn' token.

Instruction-aware path proposals After extracting the temporally ordered key-component sequence from the instruction, we then propose candidate paths. We apply A* algorithm to plan a candidate path from a start position to a randomly sampled end position nearby the last object landmark and constrain the path to visit the key components in the extracted sequence in the temporal order. Since each object category might have multiple instances in the semantic map, this results in an average of 560 candidate paths; the worst case even has more than 80,000 paths. We set three constraints to prune the number of paths. (1) Candidates violating extracted moving actions are removed. For instance, when we process the moving action “turn left” with a following object landmark “chair”, we only keep the positions of chairs to the left of current agent orientation as the next visited points. (2) We set an upper bound on the search range towards the next landmark by the distance of five meters. We assume that landmarks that appear further than five meters are hard to follow and do not usually appear in an indoor navigation scenario. (3) We group nearby objects of the same label by density-based spatial clustering algorithm DBSCAN [30]. The landmark positions are replaced by their cluster core samples. We set the maximum distance between two samples of DBSCAN to 0.5 meters (a larger distance increases the risk of merging objects across rooms). This further merges similar paths to reduce the candidate number. The overall reduction rate is 85%. In case the model fails to extract valid key components from the instruction, we use random paths planned by A* algorithm with the same start position but different endpoints within twenty meters (maximum distance in indoor navigation dataset) as candidates. The full candidate set is denoted by \( P = \{ m_{e_n,c} | e \in V_e \} \).

B. Path Feature Encoding

Next, we propose a path feature embedding scheme and apply it to each candidate path in the semantic map \( M \). Ideally, the path feature \( \Gamma \) represents the environment context along the path and will be aligned with the language repre-
sion to verify if this path matches the given instruction. iPPD discretizes each path into multiple key points and encodes the local context of each key point along the path to form a feature sequence in temporal order. Specifically, for each keypoint feature, we introduce an object compass to perceive the local environment as illustrated in Figure 2.

The object compass \( \mathcal{C}_{\text{obj}} = \{(x_i, y_i, z_i, w_i)\} \) contains objects near the agent within a 3 meters radius. We record object position \((x_i, y_i, z_i)\) in the egocentric coordinate system centered at the agent. The object compass is treated as a 3D point cloud in a 3D egocentric view. For each point in the cloud, we embed the object’s class with a 50-dimension GLOVE word embedding [31] \( w_i \). To obtain permutation invariant representation towards points order, we use PointNet [32] to encode the object compass:

\[
\mathbf{h} = \text{PointNet}(\mathcal{C}_{\text{obj}}),
\]

where \( \mathbf{h} \in \mathbb{R}^d \) is the \( d \)-dimensional feature of the local object’s context of a keypoint.

We also incorporate low-level information, i.e., agent pose, into the feature sequence. An agent pose feature \( s \) contains location \( x \) and orientation \( \theta \) at a key point:

\[
s = \text{FFN}_{\text{ap}}(PE(x), PE(\theta)),
\]

where \([ \cdot ]\) represents concatenation, \( PE(\cdot) \) is the agent pose positional encoding function [33], and \( \text{FFN}_{\text{ap}} \) is a single-layer feed-forward network projecting agent position into the \( d \)-dimensional feature space. By gathering features of all key points, we obtain the path feature \( \Gamma = \{(\mathbf{h}_i, s_i)\} \).

C. Language-driven Discriminator

We deploy a transformer-based model [34] as Language-driven Discriminator \( F_{iJ} \) for scoring paths. The natural language instruction \( T \) is tokenized and embedded by GLOVE [31] embeddings \( \psi(\cdot) \). We project the word embeddings to feature vectors using a single-layer feed-forward network \( \text{FFN}_{\text{lang}}(\cdot) \):

\[
\mathcal{T} = \text{FFN}_{\text{lang}}(\psi(T)),
\]

where \( \mathcal{T} \) is the representation of word sequence.

The \( F_{iJ} \) has \( N_t \) encoder layers as shown in Figure 3. It takes a concatenated sequence of \( \mathcal{T} \) and \( \Gamma \) as inputs and predicts a score for the corresponding path. A learnable special token embedding \([CLS]\) is put at the front of the input sequence to aggregate context information. The Language-driven Discriminator is formulated as

\[
\hat{y} = F_{iJ}([CLS], \mathcal{T}, \Gamma)]
\]

where \( \hat{y} \) is the prediction score projected from the \([CLS]\) token embedding. We pick the highest score path as the predicted path \( \hat{m} \) in inference.

During training, the ground truth score of each candidate path is set to be the combination of normalized Dynamic-Time Warping (nDTW) [35] and Euclidean distance between candidate endpoint \( v_e \) and golden target point \( v_{gt} \). Specifically, nDTW, as a path shape similarity metric, indicates how well the candidate path matches the ground truth path. The target score \( y \) is the linear combination of them, formulated as:

\[
y = \lambda \cdot \text{nDTW}(\hat{m}_{x,e}, m_{gt}) + (1 - \lambda) \cdot D(v_e, v_{gt})
\]

where \( \lambda \) is a coefficient to balance these two components and \( m_{gt} \) is the ground truth trajectory annotated in the dataset. In addition, we apply the Masked Language Prediction task [5] to enhance the model’s context-awareness. We randomly mask 15% of the tokens in the instruction and train the model to predict the masked tokens during training. The total training loss \( \mathcal{L}_{\text{total}} \) is formulated as:

\[
\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{MSE}},
\]

where \( \mathcal{L}_{\text{MLM}} \) is the Masked Language Prediction loss, and \( \mathcal{L}_{\text{MSE}} \) is the mean square error between \( y \) and \( \hat{y} \).

V. EXPERIMENTS

A. Dataset

We extend the existing VLNCE [1] dataset for this map-based language-guided navigation task and augment the dataset with our constructed 3D semantic maps but drop the RGBD visual input. We follow the original dataset splits and rename the splits as train, evaluation seen, and evaluation unseen, respectively. The environments in the evaluation seen set have been observed in the training phase, but the instructions are not, while both environment and instruction are not observed during training for episodes in the evaluation unseen set.

Algorithm 1: Key component extraction

| Input: \( T, \gamma, k \) // text, score thr, phrase limit |
| Output: \( Y \) // key component sequence |
| Data: \( S_{\text{obj}} \) // mapper from object to synset |

1. \( T \leftarrow \text{POSTagging}(T) \)
2. Set \( Y \) to empty list
3. foreach \( w \) in \( T \) do
   \[ // \text{Landmark extraction step} \]
   if \( w.\text{tag} \) is in \([\text{NN, NNS}]\) then
     \[ s = \text{WSD}(w.\text{text}, T) \]
     \[ \hat{q} = \arg \max_{q \in Q} \text{WuPalmer}(s, S_{\text{obj}}(q)) \]
     if WuPalmer\((s, S_{\text{obj}}(\hat{q})) > \gamma \) then
       \[ Y.\text{append}(\hat{q}) \]
   \[ // \text{Moving action recognition step} \]
   else if \( w.\text{tag} \) is in \([\text{VB, VBZ, VBP}]\) then
     \[ i \leftarrow w.\text{index} \]
     foreach \( j \) in \([i + 1, i + k]\) do
       if \( T[j] \) in \([\text{left}, \text{right}]\) then
         \[ Y.\text{append}(T[j]) \]
       else if \( T[j + 1].\text{tag} \) not in \([\text{NN, NNS}]\) then
         \[ Y.\text{append}(T[j]) \]
       \[ \text{Exit the loop} \]
\[ \]
B. Implementation Details

Semantic Map Generation. Since the instruction annotations are from VLNCE [1](version 1.2), we construct semantic maps on its corresponding ninety Matterport environments [36]. We first interpolate the semantic mesh of Matterport environments to construct fine-grained semantic point clouds. The 3D semantic maps $M$ are constructed by voxelizing semantic point clouds with a resolution of 0.05 meters. The semantic label of each voxel is the most frequent instance id and instance category among incoming points. Each instance category $q$ is one of the forty object categories $Q$. The navigable areas are constructed from NavMesh of Matterport [36]. We add a binary value for each voxel, indicating whether it is navigable or not during path planning. 

Model setup. Our method has three modules, the path proposal generator, the path feature encoder, and the path discriminator. The path proposal generator is a fully CPU-intensive module that, in parallel, extracts candidate paths and point-wise object compasses. Path features are encoded by Feed-forward networks and PointNet as aforementioned. The results are batched and sent to a 12-head 6-layer transformer model for scoring. We optimize our model using AdamW [37] optimizer with a learning rate of $1e^{-4}$ and train it on a single NVIDIA-P100 GPU for 12 hours.

C. Evaluation Metrics

We evaluate results using the standard evaluation metrics in visual navigation and visual-language navigation tasks [10], [38], [39]: trajectory length (TL), navigation error - average distance to goal in meters (NE), normalized dynamic-time warping (nDTW), oracle success (OS) which measures the percentage of predicted trajectories that pass the target point, success rate (SR) and success weighted by the normalized inverse of the path length (SPL). We choose nDTW and SR as our primary metrics, as they cover two important aspects of the navigation task: (a) predicted and ground truth path similarity, and (b) accuracy of reaching the target location.

D. Main Results

We implement three baseline models to benchmark this new map-language navigation task. The first is the sequence-to-sequence model [1], [2] which is widely used in sequential decision-making tasks such as VLN. It jointly encodes map observation at each point and instruction into one feature vector and uses recurrent neural network GRU [40] for action sequence prediction. Cross Attention Model (CMA) is another model that has proved successful in VLN task. The model conducts cross attention between language and observation at each time step to learn better multimodal alignment. We follow the previous work [1] and train both sequence-to-sequence and cross-attention models using imitation learning. The last is our proposed iPPD model, which proposes potential instruction-aligned paths in 3D maps and uses transformer networks to do multimodal fusion and scoring. Table I shows iPPD outperforms the other two baselines by a large margin. Specifically, in the challenging unseen environment, the iPPD surpasses the Seq2Seq and CMA models by 17% and 20% in SR. Our model can also conduct better instruction-aligned navigation, which is proved by the better performance in nDTW – iPPD and it achieves around 0.19 higher score than both Seq2Seq and CMA.

E. Are both path proposal and discriminator helpful?

This experiment studies the effectiveness of our instruction constrained path proposal algorithm and transformer-based path discriminator. To this end, we first compare the results of random path selection from the map with random path selection from the candidate path pool proposed by our path proposal algorithm. We named these two different algorithms random path and iPPD w/o path discriminator, respectively. We then further compare the performance of our iPPD with the iPPD w/o path discriminator algorithm. The results are presented in Table III.

In the first experiment, we conducted both random path and iPPD w/o path discriminator algorithms. For random path, we leverage the training dataset statistic that path lengths roughly follow a normal distribution with a mean of 8.89 meters and a standard deviation of 2.67. The lengths of randomly sampled paths follow this length distribution. As for the iPPD w/o path discriminator, we uniformly sample one path from the candidate path pool. A path is considered a success if it ends within three meters from the goal position. Table III illustrates that random path performs at least 15% worse on both seen and unseen sets. Especially the path matching score nDTW drops significantly - more than 0.35 on both seen and unseen set. It indicates that our instruction-aware proposal strategy can effectively reduce the solution space, which further benefits the path discriminator.

In the second experiment, we compare the iPPD w/o path discriminator with our iPPD model. From Table III, we could observe that although iPPD w/o path discriminator has
In this paper, we propose the map-language navigation task where an agent needs to navigate to an instruction-specified destination on a given 3D semantic map. The task is benchmarked with two commonly used language navigation models: Seq2Seq and CMA. The results of the two baseline models reveal the difficulties of the task. They also verify our assumption that the semantic map can benefit agents’ generalization in unseen environments. We further introduce the iPPD model following a new proposal and discriminating pipeline to handle the task. Extensive experimental results show the superiority and effectiveness of our proposed rough and fine-grained two-step navigation algorithm. Furthermore, our algorithm allows parallel training and inference, which could bring much convenience in real applications. We hope our work can inspire future research to focus on designing language-aware algorithms that can better leverage the 3D semantic map information.

**ACKNOWLEDGMENT**

This project is supported by EWI Flanders AI Research program and KULeuven C1 project Macchina.
