Target-Driven Structured Transformer Planner for Vision-Language Navigation

Yusheng Zhao
Institute of Artificial Intelligence, Hangzhou Innovation Institute, Beihang University, Beijing, China
zhaojinyu@buaa.edu.cn

Jinyu Chen
Institute of Artificial Intelligence, Hangzhou Innovation Institute, Beihang University, Beijing, China
chenjinyu@buaa.edu.cn

Chen Gao
Institute of Artificial Intelligence, Hangzhou Innovation Institute, Beihang University, Beijing, China
gaochen.ai@gmail.com

Wenguan Wang†
ReLER, AAII
University of Technology Sydney
Sydney, Australia
wenguanwang.ai@gmail.com

Lirong Yang
Meituan Inc.
Beijing, China
yanglirong@meituan.com

Haibing Ren
Meituan Inc.
Beijing, China
renhaibing@meituan.com

Huaxia Xia
Meituan Inc.
Beijing, China
xiahua@meituan.com

Si Liu
Institute of Artificial Intelligence, State Key Laboratory of Virtual Reality Technology and Systems, SCSE, Beihang University, Beijing, China
liusi@buaa.edu.cn

ABSTRACT

Vision-language navigation is the task of directing an embodied agent to navigate in 3D scenes with natural language instructions. For the agent, inferring the long-term navigation target from visual-linguistic clues is crucial for reliable path planning, which, however, has rarely been studied before in literature. In this article, we propose a Target-Driven Structured Transformer Planner (TD-STP) for long-horizon goal-guided and room layout-aware navigation. Specifically, we devise an Imaginary Scene Tokenization mechanism for explicit estimation of the long-term target (even located in unexplored environments). In addition, we design a Structured Transformer Planner which elegantly incorporates the explored room layout into a neural attention architecture for structured and global planning. Experimental results demonstrate that our TD-STP substantially improves previous best methods’ success rate by 2% and 5% on the test set of R2R and REVERIE benchmarks, respectively. Our code is available at https://github.com/YushengZhao/TD-STP.

CCS CONCEPTS

• Computing methodologies → Computer vision tasks; Planning and scheduling; • Information systems → Multimedia information systems.

KEYWORDS

Vision-language Navigation, Target-driven Planner, Imaginary Scene Tokenization, Structured Transformer

1 INTRODUCTION

Recent years have witnessed increasing interest in the creation of an embodied agent which learns to actively solve various challenging tasks within its environment. A number of simulators [7, 29, 47] and datasets [11, 60] have been proposed, backing such tasks as navigation [4, 19], multi-agent cooperation [5, 41, 55], interactive learning [12], and visual grounding [1, 21, 37, 48].

Vision-Language Navigation (VLN), one of the most representative embodied AI tasks, poses particular challenges as it requires the agent to navigate visual environments by following linguistic instructions. Current prevalent VLN agents [9, 20, 23] are built upon a cross-modal transformer architecture which makes only use of language instructions and historical perception for decision
In addition, as the VLN agent faces structured environments, the candidate positions are countless. Second, how to make full use of the estimated long-term target to assist navigation is also an open question; prevalent methods are typically aware of present (i.e., current observation) and past (i.e., navigation history), yet paying less attention to the future. To tackle these challenges, we propose an Imaginary Scene Tokenization (IST) mechanism which enables long-term target representation and prediction, as well as accommodates target-driven planning within the prevalent Transformer-based navigation framework. IST discretizes the unexplored area into a fixed-size grid. Each grid cell is represented by a target token that captures the imagined layout of the cell. The target tokens are fed into a cross-modal transformer along with other visual-linguistic cues to model the history, present and future of the navigation. Then these tokens are used to estimate whether the navigation target is in its cell, so as to enable global planning.

In addition, as the VLN agent faces structured environments, understanding the topology of the environment is crucial for the success of navigation. However, existing methods either arrange the historical observations in a sequential manner, or adopt complicated modules (e.g., graph neural networks) for modeling environment layouts. Differently, we develop a Structured Transformer Planner (STP), where the position-embedded visual observations are used as input tokens, and the geometric relations (local connectivity) among the navigation locations are elegantly formulated as the directional attention among input tokens. With such a design, the agent is able to not only gain a comprehensive understanding of the environment layout, but also easily revisit the past visited locations.

The integration of STP and IST leads to a Target-Driven Structured Transformer Planner (TD-STP), which allows for long-horizon goal-guided and environment layout-aware navigation. Experiments on Room-to-Room (R2R) [4] and REVERIE [45] datasets show that TD-STP achieves state-of-the-art performance on the test sets. Our code is available at https://github.com/YushengZhao/TD-STP.

## 2 RELATED WORK

The release of R2R dataset [4] stimulated the study of VLN. Various datasets have been later proposed to cover different navigation scenarios with high-level instruction [45], multilingual instruction [31], dialog-based instruction [50], and fine-grained instruction [63].

Meanwhile, numerous navigation agents have been successfully developed. Some works focus on learning better representations [24, 44, 59, 64]. For example, Wang et al. [59] propose to learn environment-agnostic representation and use multi-task learning to further enrich the representation. Some other works instead explore smarter path planning strategies [3, 26, 28, 30, 39, 53, 54, 58]. For instance, Ma et al. [39] employ a backtracking strategy that allows the agent to decide whether to continue moving forward or roll back to a previous state. To address the issue of data scarcity, several works adopt auxiliary-task learning [9, 57, 62] and data augmentation [16, 17, 25, 34, 49, 52] techniques. For example, Zhu et al. [62] use progress estimation and angle prediction as auxiliary tasks. Fried et al. [16] learn a speaker module to create instructions for unlabeled paths as extra training samples. For conducting long-term reasoning over past, structured observations, mapping based agents are built [13, 22, 53]. For instance, Wang et al. [53] store past observations in an external, graph-like memory and use a graph neural network for structured reasoning. Moreover, different
training strategies are also explored [4, 57, 58, 61, 63], including imitation learning (IL) [4], reinforcement learning (RL) [58], hybrid of IL and RL [57], and curriculum learning [61, 63]. More recently, some researchers made use of extensive, unpaired image and text data for pre-training, and then fine-tune on the limited, labeled VLN data [9, 18, 20, 40], achieving promising results.

With the success of transformer [51] in computer vision [6, 15, 35], natural language processing [14, 46] and cross-modal tasks [36], transformer-based agents have been increasingly popular in VLN task [9, 18, 20, 23, 40, 42]. A core challenge is how to incorporate the navigation history into the decision-making process under the transformer architecture. Some works equip transformer-based agents with recurrent modules. For example, Hong et al. [23] and Moudgil et al. [42] adopt a recurrent state that is updated at each time step and treat the states at different steps as the input tokens of the transformer. Though straightforward, the recurrent state inevitably loses useful information when compressing the past history into the state vector. An alternative is to keep a full sequence of navigation history. For example, Chen et al. [9] employ a hierarchical transformer to encode the full navigation history as a sequence of history tokens. Similarly, some methods adopt a memory bank to store a whole sequence of past action-observation tokens [32, 33, 43].

Our TD-STP distinguishes itself from previous models in its ability of long-term target-driven navigation planning, based on the explicit estimation of the final navigation targets and structured modeling of explored environment. Most LSTM-based methods [4, 8, 16, 53, 59, 62] and transformer-based methods [9, 23, 32, 42, 43] focus on reasoning over past observations, lacking the ability of “imagining the future”. In contrast, our agent learns to explicitly predict the long-horizon navigation target, which allows for reasoning over past and planning ahead. This idea is powerful and principled, distinctively differentiates our approach from most existing navigation agents. We notice that some previous works [22, 53] are also aware of modeling the environment layout. Compared to these works, which often use complex graph neural networks, our model naturally incorporates environment layouts into cross-modal transformer, by using local connectivity between navigation locations to guide the information flow between input tokens. A concurrent work [10] uses double cross-modal encoders for global action prediction and local action prediction, respectively. In contrast, we jointly model the environment topology and the whole action space in a single transformer, making our method elegant and flexible.

3 METHOD
3.1 Problem Setup and Overview

Problem Setup. In the VLN task, the agent is required to navigate to the target location according to a natural language instruction. We denote the textual embeddings of the instruction as \(x_0, x_1, x_2, \ldots, x_m\), where \(x_0\) is the sentence embedding and \(m\) is the length of the instruction. At each time step \(t\), the agent observes a panoramic view of the current location, consisting of 36 single views, among which the first \(k^t\) are navigable. We denote the features of these single views as \(v_{t, 1}, v_{t, 2}, \ldots, v_{t, k^t}\), \(n = 36\). In order to focus on high-level planning, in [4], the environment is assumed to be a set of discrete points and their navigability is given.

Overview. Figure 2 provides an overview of the proposed TD-STP model. TD-STP uses a cross-modal structured transformer, similar to [9]. At time step \(t\), 5 types of tokens are sent to the transformer, i.e., the global token \(g_t^{t-1}\), the instruction tokens \(x_1, x_2, \ldots, x_m\), the target tokens \(c_1, c_2, \ldots, c_{q-1}\) (\(q\) is the number of target candidates), the history tokens \(h_{t-1}, h_{t-2}, \ldots, h_{t-k^t}\) and the view tokens \(v_{t, 1}, v_{t, 2}, \ldots, v_{t, k^t}\). Note that superscripts are used to denote the time step of a token. The instruction tokens are kept constant.
through time to reduce computation. The global token $g^{t-1}$, history tokens $h^{t-1}$, and target tokens $e^{t-1}$ are the outputs of previous time step $t - 1$. View tokens $o^t$ are newly obtained at time step $t$. The instruction tokens are the output of a BERT model [14], and the global token is initialized as the sentence embedding $g^0 = x_0$. Three other types of tokens are discussed in the following subsections.

### 3.2 Structured Transformer

To capture the structured environment layouts, our TD-STP constructs and maintains a structured representation of the explored area with the transformer architecture, which is achieved via deriving a graph from navigation history and incorporating its topology into the transformer. Concretely, at time step $t$, the model constructs a graph $S^t$, as shown in Figure 3, where the nodes represent previously visited locations and the edges represent the navigability of those locations. Thus the topology of this graph can be decomposed into two parts, i.e. the position of each node and their adjacency.

At time step $t$, the history token $h^t_i$ is constructed using panoramic view embeddings, action embeddings, temporal embeddings, and positional embedding as described below:

$$h^t_i = f_V(a^t_i, \ldots, a^t_n) + f_A(r^t) + f_T(t) + f_P(l^t),$$

(1)

where $f_V$ is a panoramic visual feature extractor (as in [9]), $r^t = (\sin \theta^t, \cos \theta^t, \sin \phi^t, \cos \phi^t)$ is the moving direction ($\theta$ and $\phi$ are heading and elevation) at time step $t$, $f_A(\cdot)$ is the action encoder consisting of a linear layer and a layer normalization, and $f_T(\cdot)$ is the temporal encoder that maps the time step integer into a feature vector. Note that an additional position encoder $f_P(\cdot)$ is utilized to incorporate the spatial location $l^t$ (e.g. the starting position has the location of (0, 0); a specific node position might have the location of (2, -9)) information of the current node, which is relative to the starting location. The position encoder $f_P(\cdot)$ consists of a linear projection and a layer normalization.

To further incorporate the adjacency information of each navigable viewpoint into the transformer, our TD-STP leverages the attention masks to control the information flow among tokens. We first define the adjacency of history tokens. At time step $t$, the input history tokens of the transformer are $h_1^{t-1}, h_2^{t-1}, \ldots, h_{2^t-1}^{t-1}$, which correspond to $t - 1$ historically visited locations $l_1, l_2, \ldots, l_{2^t-1}$. The adjacency matrix of history tokens at time step $t$ is defined as a $(t - 1) \times (t - 1)$ matrix $E$. If a navigation viewpoint $l_j$ is navigable from $l_i$, $E_{ij} = 1$, and otherwise $E_{ij} = 0$, as shown in Figure 3.

The cross-modal transformer has an attention mask matrix $M$ that controls whether one token can attend to another in the attention layer of the transformer. Formally, if the $i$-th token of the transformer input can attend the $j$-th token, $M_{ij} = 1$, and otherwise $M_{ij} = 0$. Besides, the attention mask matrix $M$ has a submatrix $M_H$, which controls whether one history token can attend to another. Thus we incorporate the adjacency information into the transformer by masking $M_H$ with the adjacency matrix $C$:

$$M_H \leftarrow M_H * C,$$

(2)

where $*$ means element-wise multiplication. In this way, two non-adjacent history tokens cannot directly affect each other during the attention computation, and the information is enforced to flow over the encoded topology. This design leads to an elegant and structured transformer-based navigator.

### 3.3 Global Action Space

The structured transformer allows the agent to have direct access to structured information of the past, which delivers a global action space, where the agent can choose from not only adjacent locations but also previously visited locations. The global action space is flexible as it allows the agent to move off from the current direction and ‘jump’ to previous locations, as shown in Figure 2.

Most transformer-based agents proposed in previous works [7, 23, 32, 42, 43] make decisions/actions from a local action space, in which the agent chooses one of the navigable single views from the current observation to walk into. For simplicity, we introduce a transformation noted as $\tau$ that maps the token to its corresponding location that is part of the action space. With this notation, the local action space at time step $t$ can be formulated as:

$$\mathcal{A}_L^t = \{\tau(\hat{o}^t_1), \tau(\hat{o}^t_2), \ldots, \tau(\hat{o}^t_k^t)\},$$

(3)

where $\hat{o}^t_i$ denotes the $i$-th view token in the transformer output, and $k^t$ is the total number of navigable single views at time step $t$.

Different from the local action space, TD-STP delivers a global action space, in which the agent has direct access to history. Mathematically, the global action space is defined as follows:

$$\mathcal{A}_G^t = \{\tau(\hat{o}^t_1), \ldots, \tau(\hat{o}^t_k^t), \tau(h^t_1), \tau(h^t_2), \ldots, \tau(h^t_{2^t-1})\}.$$  

(4)

The global action space makes it possible for the agent to backtrack by choosing historically visited locations when it finds itself walking on the wrong path for several steps. Although a view token and a history token may correspond to the same navigation viewpoint, they are treated as two different actions since they contain different semantics. Specifically, the history token means backtracking, which indicates that the current path might be wrong, whereas the view token indicates that the current path matches the instruction. Therefore, with the action space expanded, the probability of each action is computed as:

$$\pi(a^t; \Theta) = \text{softmax}(\text{MLP}(\tau^{-1}(a^t) * g^t)), \ a^t \in \mathcal{A}_G^t,$$

(5)

where $\pi$ is the policy function, $\Theta$ is the parameters of the model, MLP is a multi-layer perceptron, $\tau^{-1}$ maps the action back to the corresponding token, and $g^t$ is the global token.
3.4 Imaginary Scene Tokenization Mechanism

An important part of long-term target-driven navigation is to explicitly model the possible long-term targets, which is achieved via the proposed Imaginary Scene Tokenization (IST) mechanism. Specifically, the core problem is how to model the unexplored area since the exact topology and visual information of the unexplored area are unknown. In this subsection, we elaborate on how the IST mechanism solves the problem by discretizing, imagining and refining the unexplored scene representation according to the instruction and on-the-fly collected visual clues.

As shown in Figure 4, IST first discretizes the environment into a \( d \times d \) grid, which is fixed in size and covers the navigation region. The grid has \( d^2 \) cells and the cell centers are the possible targets of navigation. The targets are spaced \( s \) meters apart, and each of them is represented by a target token. At the beginning of navigation, target tokens \( c_1^0, \ldots, c_q^0 \), \( q = d^2 \) are constructed using the positional embeddings of the targets, which is formulated as:

\[
c_i^0 = f_P(l_i) \times x_0, \quad i \in \{1, 2, \ldots, q\},
\]

where \( f_P \) is the positional encoder in Eq. 1, \( l_i \) is the spatial location of \( i \)-th target (under the same coordinate system mentioned in Eq. 1), and \( x_0 \) is the sentence embedding of the instruction.

Each target token represents an imagination of the scene layout in its cell. In the initialization, these representations might be coarse and inaccurate, but our TD-STP refines the tokens progressively during navigation. At time step \( t \), the target tokens of the previous step \( c_1^{t-1}, \ldots, c_q^{t-1} \) are sent into the transformer to update the token representation at time step \( t \) with instruction and on-the-fly collected visual clues. The refined representations are then used to predict a more precise long-term target (navigation destination). Mathematically, at time step \( t \), a multi-layer perceptron (MLP) and a softmax layer are used to obtain the probability of each target:

\[
P_{i}^{d} = \text{softmax}(\text{MLP}(c_i^{t} \ast g_i^{t})),
\]

where \( g_i^{t} \) is the global token, and \( P_{i}^{d} \) indicates the likelihood of the navigation destination being closest to the \( i \)-th target (or equivalently, in the \( i \)-th cell) at time step \( t \).

Equipping the agent with the ability to predict the long-term target leads to target-driven navigation. To better utilize this ability, we also add a positional embedding to the view tokens:

\[
s_i^{t} \leftarrow s_i^{t} + f_P(l_i^{t}), \quad i \in \{1, 2, \ldots, k^t\},
\]

where \( s_i^{t} \) is the view feature of the \( i \)-th view at time step \( t \), \( f_P \) is the positional encoder in Eq. 1, and \( l_i^{t} \) is the location that the \( i \)-th view refers to. Therefore, all the tokens that actions in the action space contain a positional embedding, and thus the decision-making process can better utilize the guidance of the predicted long-term target location.

3.5 Model Optimization

Following the common practice [9, 23, 42, 53, 57], we adopt an imitation learning loss denoted as \( L_{IL} \) and a reinforcement learning loss denoted as \( L_{RL} \), and alternate between teacher forcing (using ground truth actions) and student forcing (using actions sampled from the policy). We further consider two extra losses: the former is to boost policy training in the global action space, while the latter is to supervise long-term target prediction.

The first loss function is used when the model is trained using sampled actions. Since the action space is expanded to include previously visited locations, the model is facing the problem that newly included actions are never chosen in teacher forcing. For better convergence during student forcing, we introduce history teacher loss:

\[
L_{HT} = -\sum_{t=1}^{T} \log \pi(d_t; \Theta),
\]

where \( d_t \) is the action in the global action space that is on the ground truth trajectory and closest to the destination.

The second loss function relates to our IST mechanism. The target which is closest to the navigation destination is selected as the ground truth of target prediction. Thus a target prediction loss can be derived:

\[
L_T = -\sum_{i=1}^{T} \log P_{i}^{d},
\]

where the \( i \)-th target token is closest to the navigation destination. The total loss function can be expressed as:

\[
L = \alpha_1 L_{IL} + \alpha_2 L_{RL} + \alpha_3 L_{HT} + \alpha_4 L_T,
\]

where \( \alpha_1, \alpha_2, \alpha_3, \alpha_4 \) are set to 0.2, 0, 0, 0.1, respectively. In the student forcing iteration, the agent samples actions from the predicted probabilities. In this iteration, imitation learning loss is
Table 1: Comparison with state-of-the-art methods on the R2R dataset.

| Methods        | Validation Seen | | | Validation Unseen | | | Test Unseen |
|----------------|-----------------|-------|-------|-------------------|-------|-------|
|                | SR↑ | SPL↑ | NE↓ | OSR↑ | SR↑ | SPL↑ | NE↓ | OSR↑ | SR↑ | SPL↑ | NE↓ | OSR↑ |
| EnvDrop [49]   | 62  | 3.99 | -   | -    | 52  | 5.22 | -   | -    | 51  | 47   | 5.23 | 59   |
| OAAM [44]      | 65  | 62   | -   | 73   | 54  | 50   | -   | 61   | 53  | 50   | -    | 61   |
| AuxRN [62]     | 70  | 3.33 | 78  | -    | 55  | 5.28 | 62  | -    | 55  | 51   | 5.15 | 62   |
| SERL [56]      | 69  | 64   | 3.20 | 75  | 56  | 4.74 | 65  | -    | 53  | 49   | 5.63 | 61   |
| AP [54]        | 70  | 52   | 3.20 | 80  | 58  | 4.36 | 70  | -    | 60  | 41   | 4.33 | 71   |
| NvEM [2]       | 69  | 65   | 3.44 | -   | 60  | 4.27 | -   | -    | 58  | 54   | 4.37 | -    |
| SSM [53]       | 71  | 62   | 3.10 | 80  | 62  | 4.32 | 73  | -    | 61  | 46   | 4.57 | 70   |
| RecBERT [23]   | 72  | 68   | 2.90 | 79  | 63  | 3.93 | 69  | -    | 63  | 57   | 4.09 | -    |
| HAMT [9]       | 76  | 72   | 2.51 | 82  | 66  | 2.29 | 73  | -    | 65  | 60   | 3.93 | -    |
| TD-STP (Ours)  | 77  | 73   | 2.34 | 83  | 70  | 63   | 3.22 | 76  | 67  | 61   | 3.73 | 72   |

Table 2: Comparison with state-of-the-art methods on the REVERIE dataset.

| Methods        | Validation Unseen | | | Grounding | | | Test Unseen | | | Grounding |
|----------------|-------------------|-------|-------|-----------|-------|-------|-----------|-------|-------|
|                | SR↑ | SPL↑ | NE↓ | OSR↑ | RGS↑ | RGSPL↑ | SR↑ | SPL↑ | OSR↑ | RGS↑ | RGSPL↑ |
| Seq2Seq [4]    | 4.20 | 2.84 | 8.07 | 2.16 | 1.63 | -    | 3.99 | 3.09 | 6.88 | 2.00 | 1.58  |
| SMNA [38]      | 8.15 | 6.44 | 11.28 | 4.54 | 3.61 | -    | 5.80 | 4.53 | 8.39 | 3.10 | 2.39  |
| SIA [33]       | 31.53 | 16.28 | 44.67 | 22.41 | 11.56 | -    | 30.80 | 14.85 | 44.56 | 19.02 | 9.20  |
| RecBERT [23]   | 30.67 | 24.90 | 35.20 | 18.77 | 15.27 | -    | 29.61 | 23.99 | 32.91 | 16.50 | 13.51 |
| HAMT [9]       | 32.95 | 30.20 | 36.84 | 18.92 | 17.28 | -    | 30.40 | 26.67 | 33.41 | 14.88 | 13.08 |
| TD-STP (Ours)  | 34.88 | 27.32 | 39.48 | 21.16 | 16.56 | -    | 35.89 | 27.51 | 40.26 | 19.88 | 15.40 |

not used, and \( \alpha_1, \alpha_2, \alpha_3, \alpha_4 \) are set to 0, 1, 0.4, 0.1, respectively. More experimental results can be found at the supplementary material.

Our model is initialized with [9], and trained on an NVIDIA V100 GPU for 100k iterations, with a batch size of 8, a learning rate of 1e-5, and Adam optimizer [27].

4.2 Performance on R2R Dataset

**Dataset.** R2R dataset [4] is based on Matterport3D Simulator [7] and consists of 90 houses with about 10k panoramic views. R2R has about 7k trajectories, and each trajectory has 3 instructions. The dataset is divided into 4 splits: the training split, which consists of 61 houses and is used for training, the validation seen split, which is used to validate the model in houses that are seen in the training split, the validation unseen split, which consists of 11 houses that are not included in the previous two splits, and the test unseen split, which consists of 18 houses that are not part of the previous 3 splits. Among these splits, validation unseen and test unseen splits are relatively more important since they reflect the model’s ability to generalize to previously unseen environments.

**Evaluation Metrics.** We follow previous works in terms of evaluation metrics. Major evaluation metrics in R2R include the success rate (SR), which is the ratio of navigating trajectories stopping 3 meters within the ground truth target, the success weighted by path length (SPL), which is the success rate normalized by the ratio between the length of the ground-truth path and the agent’s path, the navigation error, which is the average distance between the agent’s stopping point and the ground truth target, and the oracle success rate (OSR), which is the success rate if the agent stops at the closest point to the destination in its trajectory. Among these metrics, SR and SPL are relatively more important.

**Performance.** The quantitative performance results are listed in Table 1. The results show that our proposed TD-STP achieves a consistent lead in terms of both SR and SPL. Noticeably, compared to the current SOTA [9], our TD-STP achieves a larger improvement on SR and SPL in validation unseen and test unseen splits, which shows that our model better generalizes into unseen environments. Our model outperforms the current SOTA [9] in the validation unseen and the test unseen splits by 4% and 2% respectively in terms of SR. In addition to higher SR, our model also achieves higher SPL in unseen environments, which shows that our model can achieve a better trade-off between accuracy and efficiency.

4.3 Performance on REVERIE Dataset

**Dataset.** Different from the R2R dataset, where the instructions are fine-grained, the REVERIE dataset [45] contains high-level instructions that ask the agent to find the described object. The REVERIE dataset has the same splits as the R2R dataset.

**Evaluation Metrics.** The evaluation metrics on REVERIE are similar to those in R2R, but with slight differences. The dataset has the same splits as the R2R dataset.
In Table 3, compared with baseline [9], "#1" with the ST boosts SR and SPL from [65.7%, 60.9%] to [67.7%, 62.5%] respectively. It illustrates that adding the location information and topology relation of the history viewpoints into the transformer benefits the navigation in unseen environments.

Table 3: The ablated results of the main components on the R2R dataset.

<table>
<thead>
<tr>
<th>Name</th>
<th>ST</th>
<th>GAS</th>
<th>IST</th>
<th>Validation Seen</th>
<th>Validation Unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SR↑</td>
<td>SPL↑</td>
<td>NE↓</td>
<td>OSR↑</td>
<td>SR↑</td>
</tr>
<tr>
<td>Baseline</td>
<td>75.0</td>
<td>71.7</td>
<td>2.51</td>
<td>81.9</td>
<td>65.7</td>
</tr>
<tr>
<td>#1</td>
<td>✓</td>
<td>73.7</td>
<td>70.9</td>
<td>2.71</td>
<td>79.3</td>
</tr>
<tr>
<td>#2</td>
<td>✓</td>
<td>✓</td>
<td>77.1</td>
<td>73.0</td>
<td>2.40</td>
</tr>
<tr>
<td>#3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>77.0</td>
<td>72.5</td>
</tr>
</tbody>
</table>

Table 4: The ablated studies on the grid size and spacing in IST mechanism. Adopted parameters are marked with asterisks (*).

(a) The ablation study about the grid size ($d \times d$) in IST.

<table>
<thead>
<tr>
<th>$d \times d$</th>
<th>Validation Seen</th>
<th>Validation Unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SR↑</td>
<td>SPL↑</td>
</tr>
<tr>
<td>0 × 0</td>
<td>77.1</td>
<td>73.0</td>
</tr>
<tr>
<td>3 × 3</td>
<td>77.7</td>
<td>73.0</td>
</tr>
<tr>
<td>5 × 5*</td>
<td>77.0</td>
<td>72.5</td>
</tr>
<tr>
<td>7 × 7</td>
<td>76.2</td>
<td>70.5</td>
</tr>
</tbody>
</table>

(b) The ablated study about the spacing ($s$) in IST.

<table>
<thead>
<tr>
<th>$s$ (meter)</th>
<th>Validation Seen</th>
<th>Validation Unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SR↑</td>
<td>SPL↑</td>
</tr>
<tr>
<td>4</td>
<td>77.4</td>
<td>71.1</td>
</tr>
<tr>
<td>6*</td>
<td>77.0</td>
<td>72.5</td>
</tr>
<tr>
<td>8</td>
<td>77.5</td>
<td>72.7</td>
</tr>
<tr>
<td>10</td>
<td>76.1</td>
<td>70.8</td>
</tr>
</tbody>
</table>

Global Action Space (GAS). As shown in Table 3, comparing "#2" to "#1", the GAS promotes SR from 73.7% to 77.1% in the validation seen set, and lifts SR from 67.7% to 68.5% in the validation unseen set. This demonstrates that flexibly jumping back to previously visited locations helps the agent to find the correct destination. The SPL in validation unseen split slightly drops because the backtracking of the GAS increases the trajectory length.

Imaginary Scene Tokenization (IST) Mechanism. In Table 3, comparing with "#2", "#3" with IST achieves another boost in both SR from 58.5% to 69.7% and SPL from 62.4% to 62.7% on validation unseen splits. It shows that the imagination of the target position is important for navigation in unknown environments.

Grid Size of IST. As shown in Table 4 (a), we study the grid size of IST. Note that the grid size 0 represents the model without IST. As can be seen in line 2, the model with a grid size of 3 × 3 achieves better navigation accuracy in both validation seen and unseen splits. When the grid size increases to 5 × 5, the model performs best on validation unseen split. However, when the grid size reaches 7 × 7, the SR and SPL on validation unseen split decrease, probably because too many target tokens may introduce noise into the model. We choose 5 × 5 as the setting in our final model.

Spacing Size of IST. As shown in Table 4 (b), we study the spacing $s$ of two adjacent target locations, which controls the granularity of target candidate locations. The spacing of 6 meters performs best in both SR and SPL of validation unseen split. An intuition for this is that a grid of targets that is too sparse provides little guidance for the agent, whereas a grid of targets that is too dense makes it hard for accurate target prediction.

4.4 Ablation Studies

In this subsection, a set of ablation studies are conducted to verify the effectiveness of the proposed components, as shown in Table 3. Moreover, the design of the IST is also discussed in Table 4.

Structured Transformer (ST). In Table 3, compared with baseline [9], "#1" with the ST boosts SR and SPL from [65.7%, 60.9%] to [67.7%, 62.5%] respectively. It illustrates that adding the location information and topology relation of the history viewpoints into the transformer benefits the navigation in unseen environments.

4.5 Analysis of Target-Driven Navigation

Qualitative Analysis. As shown in Figure 5, we visualize how our TD-STP model modifies the estimated targets during navigation. At the beginning, the target estimation (blue star) is coarse, and when the agent takes one or two steps, the predicted targets (the stars in green and red) are closer to the destination. This illuminates our
Figure 5: Visualization results. The circles represent navigation locations and the stars denote estimated navigation target. As the navigation proceeds (○→○→○), the predicted targets are refined (→→→) and closer to the navigation destination (☆).

Figure 6: The trend of average target estimation error during inference in three splits. As the navigation progresses, the average target estimation error decreases.

Figure 7: The SR on validation unseen split during training. Bars with different colors represent the target estimation error \(d_c\) in different intervals. As training progresses, the proportion of ill-estimated targets \((d_c \geq 6)\) decreases significantly, while the well-predicted targets \((d_c < 3)\) grows.

5 CONCLUSION

In this paper, a Target-Driven Structured Transformer Planner (TD-STP) is proposed for long-horizon goal-guided and room layout-aware navigation. TD-STP is built upon a Structured Transformer Planner (STP) with an Imaginary Scene Tokenization (IST) mechanism. Specifically, IST is for estimating the location of the final destination (typically located in the unexplored environment). By controlling information flow between input tokens (visited locations and estimated targets), STP achieves structured planning and global decision-making in an elegant and flexible manner. Extensive experiments demonstrate the superiority of our TD-STP. One limitation of this work is that TD-STP relies on the pre-defined environment graph. Thus a direction of our future effort is to incorporate SLAM technique for online map building.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China under Grant 62122010 and Grant 61876177, the Fundamental Research Funds for the Central Universities, the Key Research and Development Program of Zhejiang Province under Grant 2022C01082, and ARC DECRA DE220101390.


1 MORE ABSTRACTIONS

In this section, additional ablation studies are provided, which study the weight of history teacher loss and target prediction loss (i.e., \( \alpha_3 \) and \( \alpha_4 \)). The experiments are conducted on the validation unseen split of R2R dataset \([?]\) with Matterport3D Simulator \([?]\), and the results are listed in Table 1.

Table 1: Ablation study about different values of \( \alpha_3 \) and \( \alpha_4 \). The adopted values are marked with asterisks.

(a) The results in terms of SR

<table>
<thead>
<tr>
<th>( \alpha_3 )</th>
<th>( \alpha_4 )</th>
<th>( \alpha_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>68.3</td>
<td>68.4</td>
</tr>
<tr>
<td>0.4</td>
<td>69.3</td>
<td>69.4</td>
</tr>
<tr>
<td>0.6</td>
<td>69.2</td>
<td>69.6</td>
</tr>
</tbody>
</table>

(b) The results in terms of SPL

<table>
<thead>
<tr>
<th>( \alpha_3 )</th>
<th>( \alpha_4 )</th>
<th>( \alpha_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>61.6</td>
<td>61.7</td>
</tr>
<tr>
<td>0.4</td>
<td>62.2</td>
<td>62.3</td>
</tr>
<tr>
<td>0.6</td>
<td>61.9</td>
<td>61.9</td>
</tr>
</tbody>
</table>

As can be seen from the results, when \( \alpha_3 \) is set to 0.4, the model achieves best performance with respect to SR and SPL. When \( \alpha_3 \) is too low, the backtracking process is not well-supervised, which hinders the global decision making. On the other hand, when \( \alpha_3 \) is too high, it breaks the balance between global decision making and other navigation processes, which results in lower SR and SPL. Similarly, when \( \alpha_4 \) is set to 0.1, the model achieves the best performance. When \( \alpha_4 \) is too low, the target prediction process lacks proper supervision and therefore the performance is relatively undesirable compared to a higher \( \alpha_4 \). However, when the loss weight is too large, the model fails to achieve a balance between target prediction and current action selection, which leads to inferior performance.

2 MORE VISUALIZATIONS

In this section, additional visualization results are provided, as is shown in Figure 1 and Figure 2. We compare the results of ours and those of HAMT \([?]\) on the validation unseen split of R2R dataset \([?]\). The qualitative results demonstrate that the proposed TD-STP achieves better results with target-driven planning and structured modeling of the environment.

![Figure 1](image1.png)

**Figure 1**: An example from the R2R validation unseen split. The panoramic views are displayed, and the red arrows denote the direction taken by the agents. We compare the navigation of our agent (left) and that of HAMT (right). The key difference between the two agents is the second step, which is highlighted. The result shows that our target-driven agent ends up in the right place and the HAMT agent ends up in the bedroom, which is far from the ground truth.

![Figure 2](image2.png)

Figure 1 compares our model with HAMT \([?]\) on a challenging example. Note that the key difference is the second step. With the instruction “through the hallway”, two possible directions can be observed: one taken by our TD-STP, the other taken by HAMT. Different from HAMT, which only considers the history information without modeling the future, our proposed TD-STP navigates with the guidance from the predicted target. Specifically, the proposed model is able to infer from the instruction and partially observed visual information the likely navigation destination, which is probably in the living room with a couch. In addition to statistical priors
of typical room layouts, visual clues in the second step also provide some information. From the left hallway taken by HAMT, a nightstand can be vaguely seen, which offers a hint of a bedroom. By contrast, the correct direction that our agent selects leads to the edge of a couch, which is likely to be the navigation destination. The proposed TD-STP is able to infer from these visual-linguistic clues and estimate a likely target, which helps guide the navigation.

**Instruction:** Go through the doorway, past the dining table, and through the doorway into the large lobby area, waiting here.

![Figure 2: An example from the R2R validation unseen split.](image)

The comparison of TD-STP (ours) and HAMT offers another evidence of the importance of target-driven ability in navigation. In the third step, where the two agents differ, our agent predicts the navigation target (the lobby area) and heads to the direction of the target. By contrast, although the HAMT achieves good modeling of the history, it fails to look forward to the navigation future and walks in the wrong direction.

Figure 2 offers another comparison of TD-STP and HAMT. The key difference is the third step, when the agent is supposed to “go past the dining table”. Here, two possible ways of passing the dining table are available, and our agent selects the correct one which leads to the lobby area while the HAMT agent heads to the wrong direction, goes off the path, and loses direction. This example offers yet another evidence that being aware of the long-term target is crucial to navigation. At the third step, the lobby area is clearly in sight, but the HAMT agent fails to choose this direction. One explanation for this is that walking past the long edge of the dining table is statistically more common in the dataset and the agent adopts this prior. However, putting the navigation task aside, predicting the likely destination at the third step is relatively easy, and our agent is ready to utilize this predicted destination to guide navigation. Thus, the target-driven agent again outperforms the HAMT agent.