# Team 2: Landmark Recognition in Vision Language Navigation

**Qiucheng Wu**
University of Michigan
wuqiuche@umich.edu

**Naihao Deng**
University of Michigan
dnaihao@umich.edu

**Yin Lin**
University of Michigan
irenelin@umich.edu

**Abstract**

*Vision-and-Language Navigation (VLN)* task requires the agent to navigate under language instructions and vision information in unknown environments. The evaluation for end-to-end navigation agents only considers their accuracy and efficiency to arrive at navigation goals but does not be aware of the agent's ability in following the instructions. However, the agent's capacity in grounding landmarks in instructions to trajectory points in environments reveals important potential in instruction following, error correction, high-level instruction execution, and so on. In this project, we augment the Room-to-Room dataset with landmark-trajectory point mapping using "Return to" instruction. We replicate three state-of-the-art navigation agents and analyze their performance in understanding the correlation of landmarks in instructions and their references in environments.

## 1 Introduction

As one of the most critical tasks in the field of robotics, the Vision-and-Language Navigation(VLN) task requires the agent to interpret a given natural language instruction and the visual observation of the surrounding environment to navigate to the given destination. VLN task has a wide range of applications, including service robots, rescue robots, and so on.

There is a rich vein of research addresses challenges of VLN tasks, including visual-textual co-grouding (Ma et al., 2019a), synthetic data augmentation (Fried et al., 2018), and the combination of imitation learning and reinforcement learning (Wang et al., 2019). Most of the navigation agents frame the VLN task as a sequence-to-sequence problem and widely accepted metrics for navigation models include the Navigation Error(NE), Success Rate (SR), Oracle Success Rate(OSR), and Success Rate Weighted by Path Length (SPL) (Anderson et al., 2018a). However, these metrics only measure the accuracy of destination recognition and the efficiency of the navigation plan, which sets apart the roles that visual and language play in the VLN task. This observation motivates us to propose a new measurement to evaluate agents' ability to ground landmarks in natural language instructions to the object references at each trajectory point.

In this project, apart from discussing the performance of three state-of-the-art models for traditional end-to-end navigation with detailed instructions containing both **direction** and **landmark** information, we also build a *Return Room-to-Room (RR2R)* dataset to explore the model's ability in understanding the surrounding environment. Instead of providing detailed step-by-step navigation, *RR2R* dataset augments the original Room-to-Room dataset with ["Return to"] + [landmark] instructions to evaluate the model's ability to recognize landmarks mentioned in the instructions. Specifically, we aim to examine the effectiveness of the vision and language co-grounding, which is an important indication of an agent's ability to truly understand the environment.

The significance model's ability for *trajectory point-landmark* mapping lies in the following two aspects. First, the original metric fails to measure the agent's ability to follow the trajectory path specified by the natural language instruction, i.e. even the agent gets a 100% navigation accuracy, we could not tell whether the agent did a good job in following the instruction or just explore around and stop at the destination. Second, the better the *trajectory point-landmark* recognition ability of the agent, the more capable of the agent to recover from navigation errors like local loops and early stops. Therefore, we proposed *trajectory point-landmark* as an evaluation metric and compare the performance of three state-of-the-art models (Ma et al., 2019a,b; Anderson et al., 2018b) this metric.

To sum up, our contribution is twofold. First, we augment the Room-to-Room[1] dataset with landmark-trajectory point mapping and construct *RR2R* dataset. Second, we replicate three state-of-the-art models: seq2seq[2], self-monitoring agent[3], and regretful agent[4]. We compare their performance in both the original metrics and the new landmark metric.

## 2 Problem Definition

In the VLN task, the agent begins with a natural language instruction $\bar{x} = \langle x_1, x_2, \cdots x_L \rangle$. The agent performs navigation in the environment with a set of 3D points $V$. Each trajectory point $v \in V$ is comprised of a set of 18 panoramic RGB-D images. At each time step $t$, the sensor returns a RGB image $o_t$, based on the current trajectory position $v_t$, the heading $\phi \in [0, 2\pi)$ of the agent, and the camera elevation $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. The action space contains 6 actions `left, right, up, down, forward, stop`. The status of the agent is represented by a triple $s_t = \langle v_t, \phi_t, \theta_t \rangle$. According to the current status, the agent select an action $a_t$, leading to a new status $s_{t+1} = \langle v_{t+1}, \phi_{t+1}, \theta_{t+1} \rangle$. The navigation process of an agent is sequence of status and actions $\langle s_0, a_0, s_1, a_1, \cdots, s_T, a_T \rangle$.

The augmented *RR2R* dataset aims to test the agent's ability in recognizing landmarks in the instructions. We define a set of landmarks $M = \langle m_1, m_2, \cdots, m_i \rangle$ in the environment. At the end of each instruction, the augmented instructions are "$\bar{x}+$ [Return to] + [$m_i$]", where $m_i$ is a landmark in the previous trajectory path specified by the original instruction $\bar{x}$. Each landmark is mapped to a trajectory point $v_i^{marked}$. To evaluate the agent's ability to return to the landmark, we compare the success rate(SR) and oracle success rate(OSR) of different navigation agents in returning to the landmarks after the execution of original instructions.

[1]https://bringmeaspoon.org

[2]https://github.com/peteanderson80/Matterport3DSimulator

[3]https://github.com/chihyaoma/selfmonitoring-agent

[4]https://github.com/chihyaoma/regretful-agent

## 3 Related Work

In this project, we explore the performance of three end-to-end navigation agents: seq2seq (Anderson et al., 2018b), self-monitoring agent (Ma et al., 2019a), and regretful agent (Ma et al., 2019b).

### 3.1 Seq2seq

In (Anderson et al., 2018b), the seq2seq model is proposed as the baseline for the R2R navigation task. LSTM (Hochreiter and Schmidhuber, 1997) network and ResNet-152 (He et al., 2016) for language and image encoding. Attention mechanism (Luong et al., 2015) is used to compute attentional hidden state and calculate the predictive action distribution. Seq2seq introduces two training regimes, the "teacher-forcing" and the "student forcing". Since in our *RR2R* dataset, the testing dataset has the same distribution as the training dataset. Therefore, we adopt the "teacher-forcing" training approach, which means that at each step during the training process, we choose the ground-truth target action $a_t*$ as the input.

### 3.2 Self-Monitoring

Self-monitoring agent (Ma et al., 2019a) introduces a **visual-textual co-grounding** module to locate the instruction completed in the past and use the **progress monitor** to estimate the monitoring progress. In the co-grounding model, the authors use *LSTM* to encode the current panoramic image, instruction feature, and the previously selected action for action selection. The progress monitor introduces an objective function during the training process served as a regularizer to prune unfinished trajectories during the inference. Experimental evaluations indicate that the self-monitoring agent outperforms the seq2seq baseline in both seen and unseen environments.

### 3.3 Regretful Agent

In the regretful agent, a progress monitor is introduced so that the agent's progress towards the goal is monitored. The outputs of the agent will decrease or fluctuate if the agent selects an action leading to deviation from the goal, increase if the agent moves closer to the goal. Based on the feedback provided by the progress monitor provides, the agent can select the action. The agent can regret and backtrack using a **Regretful Module** and a **Progress Marker**. **Regretful Module** examines the progress made from the last step to the current step to decide whether to take a *forward* or *rollback* action. Once the agent regrets and rolls back to the previous location, the **Progress Marker** informs whether location(s) have been visited before and rates the visited location(s) according to the agent's confidence in completing the instruction-following task (Ma et al., 2019b)

The regretful agent proposed (Ma et al., 2019b) outperforms the previous models such as self-monitoring agent (Ma et al., 2019a), Student-forcing model (Anderson et al., 2018b), and Speaker-Follower (Fried et al., 2018) on both SR and SPL metrics.

## 4 Proposed Method

### 4.1 RR2R Dataset

The *RR2R* dataset used in our experiment augments from the R2R dataset[5]. R2R dataset is the first benchmark dataset for visually-grounded language navigation in real building (Anderson et al., 2018b) (Chang et al., 2017). The entire dataset contains 90 scenes, 61 for training, and validation saw 11 for validation unseen and 18 for test unseen.

We augment the dataset by adding return instructions to the dataset. These involve 3 main steps: (1) Pick proper intermediate landmarks; (2) Construct return trajectory path; (3) Modify linguistic instructions.

At first, we decide to directly pick nouns in the original instructions. However, we find many nouns cannot be selected as landmarks. Some examples are "room" and "door", which are too general for the agents to specify one. Also, we find "stair" is a bad landmark, because one stair covers a few trajectory points, and it is hard for the agents to determine one to stop. Therefore, we finally decide to manually select a few landmarks in two rooms[6].

Then, based on the landmark we selected, we start to construct a return path from the terminal point back to this landmark. Suppose the original starting point is $A$, and the original terminal point is $B$, while the landmark we selected is $C$ between $A$ and $B$. Then, the rough path will concatenate the trajectory points from $A$ to $B$ and the points from $B$ back to $C$. For the first part, we can directly use the original trajectory points. For the second part, we have to determine the corresponding trajectory point for the landmark, and we inversely enumerate the trajectory points to obtain the second half path. demonstrates a series of closed trajectory points for the landmark: the brown chairs. We finally pick the bottom right one. These images are all from the Matterport dataset.

Finally, we start to modify the linguistic instructions. We also do this manually to keep the quality of instructions. First, we add "Return to the [landmark]." at the end of the instruction sets, where "[landmark]" is the intermediate object we selected. Then,

[5]https://bringmeaspoon.org

[6]The room IDs we selected are **1LXtFkjw3qL** and **17DRP5sb8fy** in the training dataset.

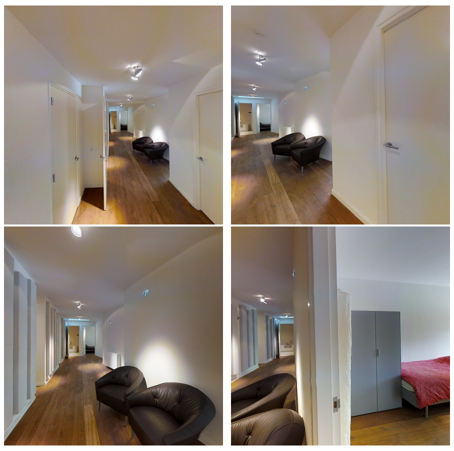


Figure 1: Four candidate trajectory points for the landmark "brown chairs." We manually check the position of the brown chairs and determine which trajectory point best describes the landmark. We finally choose the bottom right one since it is closest to the camera.

we replace the stop words with other verbs. The stop words include "stop", "wait". We replace them because we do not want the agent to stop at the previous terminal point because of these words. Finally, we add some conjunctions to formulate natural instructions. An example of conjunctions is "and", "then", etc.

We attach an example of the original instruction and modified instruction. In attachment, List 3 is the *RR2R* dataset with the returning instructions ("Return to the stone statue") corresponding to List 2.

### 4.2 Train/validation dataset split

With our *RR2R* dataset, the next step is to create the proper train/validation split. The models introduced above are proposed for the VLN tasks, so we do not expect the models to be able to roll back simultaneously. In other words, training is necessary for the models to be able to roll back even if the agents understand the intermediate landmarks. However, we only manually label 2 rooms for returning instructions, while there are 61 rooms in the originally seen dataset and 11 rooms in the original unseen validation dataset. How to train the agents with the return function while evaluating the agent's ability on normal VLN tasks is hard given that we have limited data.

We propose 3 different training datasets to help the models learn rollback instructions. These training datasets are as follows.

1. Only 2 rooms with original and return instructions

2. 2 rooms with original and return instructions, other rooms with original instructions

3. Similar to 2, but we balance the number of return instructions versus original instructions by repeating return instructions in the dataset.

Therefore, in the following experiments, we first use the seq2seq model to determine a good train/validation dataset split. After that, we use the selected split to train all 3 models and record their rollback performance.

### 4.3 Evaluation Metrics

We are interested in the agents' ability to recognize the intermediate landmarks. We use the accuracy of the models on the dataset with return instructions to illustrate such ability, which is the validation seen dataset in our experiment. There were other metrics used by the original VLN tasks, such as path length and oracle success rate. For simplicity, we do not use these metrics in our experiment.

Meanwhile, we do not want the agents to lose the ability on original VLN tasks. Therefore, we also use the accuracy on datasets without return instructions to measure such abilities, which is our validation unseen dataset in this experiment.

Ideally, after training, the agents should have high accuracy on the validation seen dataset with return instructions, meaning the agents understand the intermediate landmarks. Also, the agents should have accuracy comparable with the original model on the validation unseen dataset without return instructions, meaning the agents do not lose the ability on original VLN tasks.

| category | seen acc (R) | unseen acc (NR) |
|---|---|---|
| (1) | 0.444 | 0.097 |
| (2) | 0.413 | 0.188 |
| (3) | 0.635 | 0.159 |
| Baseline[7] | N/A | 0.209 |

Table 1: Seq2seq performance on roll back(R) instructions in seen dataset and without roll back (NR) instructions in unseen dataset.

## 5 Experiments and Analysis

We train three models (1) seq2seq (2) Self-monitoring Agent (3) Regretful Agent on the *RR2R* dataset we proposed above. As mentioned, we first test which train/validation split leads to the best train performance on the rollback task. Then we test the rollback performance on those 3 models with the best train/validation split found.

### 5.1 Train/validation Split Result

During training, we monitor the accuracy of the model on validation dataset to determine the roll back ability and the ability to follow instruction. Specifically, there are two validation dataset: seen and unseen one. The unseen validation dataset is the same as the original R2R validation dataset. The seen one consists of return and original instructions in the 2 rooms we manually labelled above. The result is shown in Table 1. In the table, "R" means dataset with return instructions, while "NR" means dataset without return instructions.

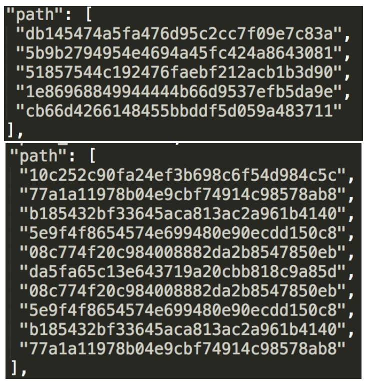
From Table 1, we observe two phenomena:

1. With the proportions of rollback instructions increasing in the training dataset, the accuracy of the agents on return instructions increases, meaning the agents learn the rollback operation with our training dataset.

2. With the rollback instructions involved in the training dataset, the accuracy of the agents on original instructions decreases, meaning the return instructions in the training dataset influence the agents' ability on following normal instructions.

[7]The baseline does not train with return instructions. Therefore, validation result on the seen dataset with return instructions is not meaningful.

We expect the first phenomenon because intuitively, training with rollback instructions will increase its ability to rolling back and return to particular landmarks. On the other hand, the rollback instruction sets are different from the original ones. Specifically, the return instruction sets add one sentence to the instructions, while they add many trajectory points to the paths, leading to a density discrepancy between the path and instruction for two sets. This may bring difficulties for model learning and lead to the second phenomenon mentioned above. Figure 2 shows the density discrepancy.



```
"path": [
 "db145474a5fa476d95c2cc7f09e7c83a",
 "5b9b2794954e4694a45fc424a8643081",
 "51857544c192476faebf212acb1b3d90",
 "1e86968849944444b66d9537efb5da9e",
 "cb66d4266148455bbddf5d059a483711"
],
```

```
"path": [
 "10c252c90fa24ef3b698c6f54d984c5c",
 "77a1a11978b04e9cbf74914c98578ab8",
 "b185432bf33645aca813ac2a961b4140",
 "5e9f4f8654574e699480e90ecdd150c8",
 "08c774f20c984008882da2b8547850eb",
 "da5fa65c13e643719a20cbb818c9a85d",
 "08c774f20c984008882da2b8547850eb",
 "5e9f4f8654574e699480e90ecdd150c8",
 "b185432bf33645aca813ac2a961b4140",
 "77a1a11978b04e9cbf74914c98578ab8"
],
```

Figure 2: Example of density discrepancy between return and original instruction. Adding one sentence "Return to the painting" leads to many new trajectory points on path.

Therefore, we adopt the third training dataset to ensure the model learns the return functions without losing too much ability on normal navigation.

### 5.2 Performance of Models on RR2R Dataset

We trained the previous 3 models with the dataset in the last section, and we record their performances on both datasets with return instructions and the original unseen validation dataset. Table 2 demonstrates the results.

From Table 2, we observe that as the current state-of-the-art, the regretful agent

| model | acc(R) | acc(NR) |
|---|---|---|
| seq2seq | 0.635 | 0.159 |
| self-monitoring | 0.508 | 0.127 |
| regretful-agent | 0.436 | 0.459 |
| Baseline[8] | N/A | 0.209 |

Table 2: 3 models performance on datasets with (R) /without (NR) roll back instructions.

does best on the original unseen validation dataset, which is expected. However, we have two other interesting findings. First, we observe that all of these 3 models are not able to fully recognized the intermediate landmarks, and specifically, the state-of-the-art model does worse in recognizing intermediate landmarks, leading to a worse accuracy on the seen validation dataset with rollback instructions. Second, the self-monitoring agent has a very low unseen validation accuracy. We will discuss these two observations below.

For the first observation, we take a further look at the path generated by each model. We observed that for the seq2seq model, the model tends to stop at the intermediate landmark rather than following the instructions and return in the end. On the other hand, the other two models seem to be able to return based on the path they output on the validation seen dataset. We analyze this phenomenon, and we believe this is because of the property of the seq2seq model. Namely, since the seq2seq model processes the instruction entirely at the encoder stage, it might weight more on the beginning point and the ending point, leading to the agent terminates earlier. List 1 gives an example of early terminate behavior for the seq2seq agent.

The second phenomenon is out of our expectations. We check the original Github, finding other researchers also have difficulties replicate this result[9]. The problem might be related to tuning the model with proper parameters. We are working on a solution. [10]

[10]https://github.com/chihyaoma/selfmonitoring-agent/issues/11

Listing 1: Early terminate behavior for seq2seq agent. We expect the agents to go pass the point **77a1a11978b04e9cbf74914c98578ab8** and roll back to this position, but the agent directly stops here.

```
{"instr_id": "322801_1",
 "trajectory": [
 ["6800f98e9e67463e9928a4253253bc2f",
      2.617993877991494, 0.0],
 ["10c252c90fa24ef3b698c6f54d984c5c",
      2.617993877991494, 0.0],
 ["10c252c90fa24ef3b698c6f54d984c5c",
      2.0943951023931953, 0.0],
 ["10c252c90fa24ef3b698c6f54d984c5c",
      1.5707963267948966, 0.0],
 ["77a1a11978b04e9cbf74914c98578ab8",
      1.5707963267948966, 0.0]]}
```

## 6 Conclusion and Future Work

Vision-and-Language Navigation (VLN) is one of the critical tasks in the field of robotics. Current evaluation metrics for end-to-end models lack a measure of the agents' ability in grounding intermediate landmarks in instructions to their references in environments. We augment the Room-to-Room dataset with "Return to + [landmark]" instructions and replicate three state-of-the-art navigation agents: seq2seq, self-monitoring agent and the regretful agent. We evaluate the navigation success rate and oracle success rate of these agents in returning to intermediate landmarks.

From the evaluation result, we come to the following main conclusions:

1. Although we have the density discrepancy problem between instructions and trajectory points, it is still possible to train an agent with roll-back modified path and linguistic instructions. In these three models, they are all be able to stop at the intermediate landmarks we introduced in the dataset.

2. Seq2seq agent sometimes does not return to the landmarks; it stops at the landmarks instead. This is possible due to the structure of seq2seq model, which deals with the linguistic instruction completely ahead of vision and actions.

3. The regretful agent and self-monitoring agent have lower accuracy than

seq2seq, while theoretically, they should be better since they utilize the progress monitor to ground instructions with the visual environment. Although this might be caused by the seq2seq early terminate behavior, it is still possible for these 2 agents to increase their rollback abilities.

Also, we realize that there are some drawbacks to the current experimental design. With the limited data, there are several extra variables between the return dataset and the original dataset. First, the augmented dataset only consists of 2 rooms, while the training dataset consists of 61 rooms, meaning 59 rooms are unattended in the validation seen dataset. Second, the return instructions are all in the seen environment, while the original instructions are all in the unseen environment, meaning we cannot directly compare the accuracy between these two datasets. And finally, it would be better if we have a test dataset, rather than using a validation dataset to demonstrate its performance.

The key problem we faced is still the amount of work for manually labeled data. We are trying automatic augment data generator, e.g. using the object detector to find the correct trajectory points for landmarks, and generating the roll-back linguistic instructions as well. With the automatic augment dataset, the future work is to systematically test the ability of roll-back in different models.

## References

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. 2018a. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018b. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3d: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, pages 3314–3325.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. 2019a. Self-monitoring navigation agent via auxiliary progress estimation. *arXiv preprint arXiv:1901.03035*.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. 2019b. The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6732–6740.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6629–6638.

## 7 Attachment

Listing 2: Example of the original R2R dataset

```
{
    "distance": 7.54,
    "scan": "1LXtFkjw3qL",
    "path_id": 7238,
    "path": [
    "a3fe827d49db4f7caa076c313434d418",
    "e0f025c0baa94beba57ea499e3d846c5",
    "6ab39d830183407cb7ec17206889000d",
    "74d5b2290be74a1ba052b6fe2320e064",
    "e8e2d73795e54b6db89cd32745e79fb9"
  ],
  "heading": 1.739,
  "instructions": [
    "Go to the bottom of the stairs and
        turn left. Walk along the orange
        wall, past the stone statue, and
        go through the doorway. Turn left
        and stop at the top of the steps
        leading down. ",
    "Go down the stairs, and take a left.
        Go around the dining table and
        exit through the door on the left.
         Go down the three stairs and stop
         at the bottom of the stairs. ",
    "Go downstairs. U turn left. Go
        straight and then turn left. Wait
        near the double white doors. "
  ]
},
```

Listing 3: Example of the RR2R dataset

```
{
  "distance": 7.54,
  "scan": "1LXtFkjw3qL",
  "path_id": 723800,
  "path": [
   "a3fe827d49db4f7caa076c313434d418",
   "e0f025c0baa94beba57ea499e3d846c5",
   "6ab39d830183407cb7ec17206889000d",
   "74d5b2290be74a1ba052b6fe2320e064",
   "e8e2d73795e54b6db89cd32745e79fb9",
   "74d5b2290be74a1ba052b6fe2320e064",
   "6ab39d830183407cb7ec17206889000d"
  ],
  "heading": 1.739,
  "instructions": [
   "Go to the bottom of the stairs and turn
       left. Walk along the orange wall,
       past the stone statue, and go through
        the doorway. Turn left and go to the
        top of the steps leading down.
       Return to the stone statue. ",
   "Go to the bottom of the stairs and turn
       left. Walk along the orange wall,
       past the stone statue, and go through
        the doorway. Turn left and go to the
        top of the steps leading down.
       Return to the stone statue. ",
   "Go to the bottom of the stairs and turn
       left. Walk along the orange wall,
       past the stone statue, and go through
        the doorway. Turn left and go to the
        top of the steps leading down.
       Return to the stone statue. "
  ]
 },
```