Language-enhanced RNR-Map: Querying Renderable Neural Radiance Field maps with natural language (Supplementary material)

Francesco Taioli[1,*], Federico Cunico[1,*], Federico Girella[2,*], Riccardo Bologna[2,*,†], Alessandro Farinelli[2], Marco Cristani[1]

[1]University of Verona, Dept. of Engineering for Innovation Medicine

[2]University of Verona, Dept. of Computer Science

name.surname@univr.it , name.surname@studenti.univr.it[†]

1. Video examples

A video showing examples of our method can be found at https://intelligolabs.github.io/Le-RNR-Map/.

2. Obtaining 3D location given Le-RNR-Map target coordinates

To retrieve the 3D end-goal location given the (x, y) indices, expressed in Le-RNR-Map coordinate system, we use the following approach: first, we define an arbitrary rotation in the RNR-Map space, which only rotates on the axis perpendicular to the map plane by α radians. We represent this rotation with a 3×3 matrix R_{2D}. We also define a 3×1 vector t_{2D} as $\begin{bmatrix} x & 0 & y \end{bmatrix}^T$, which indicates the translation in the RNR-Map from the origin. We can then define the 4×4 roto-translation matrix as

$$R_{t_{\text{2D}}} = \begin{bmatrix} R_{\text{2D}} & t_{\text{2D}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{1}$$

where $\mathbf{0}$ indicates a 1×3 vector of zeros. Finally, we map the 2D coordinates (up to rotation uncertainty) to the 3D roto-translation matrix $R_{t_{3D}}$ as follows:

$$R_{t_{\text{3D}}} = R_{t_{\text{origin}}}^{-1} R_{t_{\text{2D}}} \tag{2}$$

where $R_{t_{\text{origin}}}$ is the roto-translation matrix of the origin in the 3D system, stored during the map generation process. $R_{t_{\text{3D}}}$ is structured as follows:

$$R_{t_{\text{3D}}} = \begin{bmatrix} R_{\text{3D}} & t_{\text{3D}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3}$$

and t_{3D} is the desired 3D pose of the agent.

3. Negative Prompts

In Tab. 1 we report the Table of the main paper, in which we add details about the corresponding negative prompts:

- *Corozal*: `"wc"`
- *Darden*: `"the floor inside the house", "the wall inside the house"`
- *Markleeville*: `"the floor inside the house"`
- *Wiconisco*: `"things", "stuff", "textures", "objects"`

Table 1: Experiments using the `val` split of Gibson tiny dataset. Note that our setup is *known* since we generate the map beforehand. Each scene has a different negative prompt.

Scene name	Negative Prompts	Success ↑	DTS (m) ↓
Corozal	✗	0.69	1.19
	✓	0.69	**0.65**
Darden	✗	0.37	4.12
	✓	**0.59**	**2.59**
Markleeville	✗	0.81	**1.38**
	✓	**0.83**	1.50
Wiconisco	✗	0.76	0.50
	✓	0.76	0.50
Average	✗	0.66	1.79
	✓	**0.72**	**1.31**

4. Object Search

In Fig. 1 we show an example of the final result of a search using Le-RNR-Map. For video examples, we refer the reader to the video attached.

5. Multi-Object Search

Le-RNR-Map can also be used to look for multiple items. When given a set Q of query prompts (e.g. `chair,`

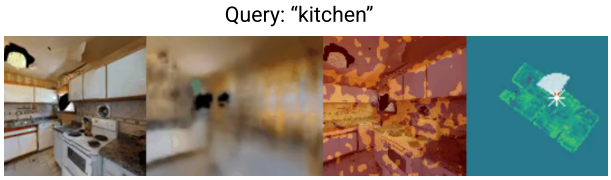

Figure 1: (a) Observation from Habitat-sim. (b) Reconstruction using latent code from Le-RNR-Map using Neural Radiance Field (c) Feature visualization of the text query using MaskCLIP. (d) Top-down view of Le-RNR-Map

`couch, cabinet`), Le-RNR-Map can be explored to look for each desired item. Given a prompt in the form of “$item_1, item_2, ..., item_n$”, we separate the single items by splitting the string on each comma. We then look for each item in sequence, following the standard procedure presented in the main paper. We refer the reader to the video attached for some examples.

6. Affordance Search

In the following, we paste the code to compute the “Affordance search” using the available OpenAI chat-completion API, selecting the `gpt-3.5-turbo` model.

```
import os
import openai
openai.api_key = "YOUR-API-KEY"

query = "Give me a location that can be relaxing
    after a long day at work"
completion = openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
    {"role": "system", "content": f"You have the
    goal to act as an affordancy query mapper. I
    enter a query, and you have to give me A
    COMMA SEPARTED LIST of possible zone, inside
    the house, that satisfy my query. The ouput
    must be a sequence of location, without
    explantion. QUERY: {query}"}
  ]
)


possible_locations = completion.choices[0].
    message
multi_object_search(possible_locations)
```

Interesting future work will be devoted to different Large Language Models and prompts, and their impact on the final results.

6.1. Examples

query = “Find me a drink to wake me up”

```
{
  "role": "assistant",
  "content": "kitchen, dining room, living room,
    office"
}
```

query = “Where can I wash my hands”

```
{
  "role": "assistant",
  "content": "bathroom, kitchen, utility room"
}
```

query = “Where can I watch the tv?”

```
{
  "role": "assistant",
  "content": "living room, bedroom, basement,
    media room"
}
```