LSVM: A Linguistically Semantic Voxel Map Empowered by CLIP Embedding for General Purposes

[1]*Chih-Hung Tu* (涂志宏), [2,*] *Chiou-Shann Fuh* (傅楸善)

[1]Department of Electrical Engineering,
National Taiwan University, Taipei, Taiwan,
[2]Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan

r10921015@ntu.edu.tw fuh@csie.ntu.edu.tw

ABSTRACT

In recent decades, the development of robotic and intelligent systems has taken a huge step forward. Many complex tasks have already become possible by taking advantage of modern computers and deep learning. The computer now is starting to have the potential to know the meaning of the physical world. Perception is the first step in connecting to the physical world. However, the traditional intelligent system usually ignores the rich information hidden inside the surrounding, which leads to the limitation of the capability to conduct high-level tasks. Thus, we develop a system to leverage the capability of robotics in terms of its perception. The goal is to build a semantic voxel map, which contains rich information about the environment. To further improve the generality of the map, we introduce Connecting Text and Images (CLIP), to translate the real-world images into a meaningful embedding space. The embeddings will then be assigned to the voxel map as the semantic information that could later be retrieved for use. We then conduct some experiments to validate the system in a simulator, habitat-sim, and a 3D reconstructed indoor map dataset, HM3D. The results show that the system is capable to retrieve the meaning on the map.

Keywords: *Semantic map, Voxel map, CLIP, Deep Learning*

1. INTRODUCTION

In recent decades, the development of robotic and intelligent systems has taken a huge step forward. Many complex tasks have already become possible due to the power of modern computers, deep learning, and large datasets, that are unseen before. The computer now is starting to have the potential to know the meaning of the physical world. Such capabilities are critical especially for the field of robotics, which is namely the bridge between computers and the outside world. They could further be split into a few more subjects, i.e., perception, reasoning, and controlling.

Perception is the first step to connecting to the physical world. However, for a traditional intelligent computer system, the rich information hidden inside the surrounding is usually ignored and lead to the limitation of its capability to conduct a task at a higher level. For example, if a robot was asked to open a door in the living room, it can probably only, either way, open the one that is already set in front of itself by the researchers, or fail to open the correct door due to the lack of environmental information, such as the room type, the meaning of the objects. As shown in the simple example, we can know that the ability to percept and represent the outside world is crucial for a robot to enable more capabilities. A general perception system is indeed needed to empower more applications, while such a system is challenging and worth to be solved.

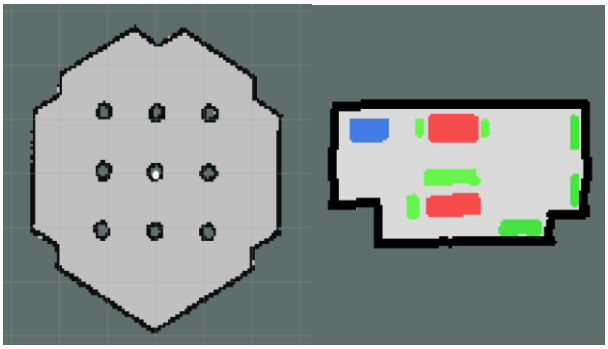
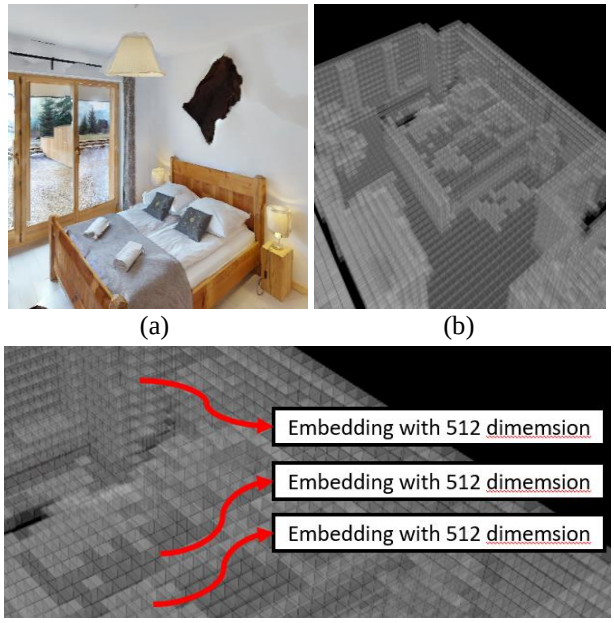


Fig. 1. The grid map (left) and the semantic map (right). The grid map represents the geometry only, but the semantic map contains the geometry and the semantic information on the map in different colors.

In this paper, we will develop a system to leverage the capability of robotics in terms of its perception. The goal of our work is to enable the artificial system to know the meaning of the world by building a so called "semantic map", which contains the rich information about the environment ready to use by robots, in comparison to the occupancy map that only contains the geometry of the environment and could only be used for navigation in the past in figure 1.



(a) (b)

(c)

Fig. 2. (a) A typical example of the environment. (b) Corresponding voxel map. (c) Each of the voxels consists of an embedding with 512 dimensions.

However, for most semantic maps, the amount of information inside is usually limited by the categories that could be classified by the semantic segmentation. This, finally, still results in limited usage of the map and left the problem behind. To further improve the generality of the semantic map, we introduce a pretrained model, Connecting Text and Images (CLIP), to empower the capability to translate the real-world images into a meaningful embedding space. The embeddings will then be assigned to the voxel map as the semantic information that could later be retrieved for use in figure 2.

This map can search with some simple language instruction by taking advantage of CLIP. More importantly, it is no longer limited by the number of categories that can be detected by the semantic segmentation model, which is usually narrower compared with what CLIP could do. The main contributions are as follows:

1) We propose heuristic methods to update the voxel map.
2) We leverage the ability of semantic map by integrating the CLIP module to generate powerful embedding.
3) We use CLIP in a novel manner, which further expands the semantic information from image-text to the 3D world.
4) We conduct qualitative experiments to validate the proposed system.

The details of the system will be further discussed in the following sections. Section II will introduce some related works. Section III will give some of the definitions of the problem to solve. Section IV will focus on the methodology to solve the problem defined in Section III. The detail of the experiment would be discussed in Section V, and the qualitative results will be shown in Section VI. Finally, we will give a brief conclusion about our work in Section VII.

2. RELATED WORKS

This section will talk about the related works of this paper and some literature reviews on the related topics, including the CLIP, semantic segmentation, voxel map, and semantic map. We also discuss these topics in the following sections.

2.1 CLIP

CLIP is a visual/language encoder proposed by OpenAI [1]. The system contains two encoders to encode the image and text, respectively. It was trained on a huge image-text pair dataset with contrastive learning [2], which enforce both the encoder to map the images and corresponding texts to the same embedding space. Such a training process enables the model to conduct various applications and have some level of zero-shot learning capability to achieve classification over a variety of object classes.

OpenAI has already shown the capability for the model to achieve comparable performance in comparison with other classification models. They also show how general CLIP is to be directly applied to some tasks without any extra training process or finetune on the new dataset. It shows that CLIP is robust to the difference between different domains, which usually causes any serious drop in performance for traditional classification models when transferred to another dataset with a domain gap. Such property makes it a very good module for generating meaningful and general representations of images.

2.2 Semantic Segmentation

Semantic segmentation aims to assign a category for each pixel on an image, i.e., a pixel-level classification task. This is a task popular in recent years in the field of autonomous driving cars. By taking advantage of deep

learning and convolutional neural network, various works and system successfully achieve high accuracy on various benchmarks.

In recent years, semantic segmentation has gained huge success in the field of the self-driving car. It is used to detect the context of a driving scene to segment the pedestrian, cars, trucks, light signals, etc. However, most of the objects appearing in a driving scenario are of a relatively small variety. The category is not as diverse as the object in a house or interior space, so the semantic segmentation system could still handle most of the situation.

Furthermore, such a system still faces the problem of transferring to another domain. Although there are also some works trying to deal with domain adaptation issues, most of the adaptation still needs to re-train or finetune along the target dataset to get a comparable performance on the target dataset. Also, the ability of traditional semantic segmentation is still limited by the categories that could be detected by the model. Thus, a more general solution must be proposed to leverage its capability.

2.2 Voxel Map

The voxel (volume pixel) map is one kind of geometric representation in the 3D space, which could represent a 3D geometry roughly. A voxel means a unit 3D cube, while a voxel map consists of many voxels forming the shape. Similar to a 2D image or grid map, a voxel map is actually a 3D version of an image. Each of the voxels could consist of some properties, e.g., occupancy, temperature, or mass. The content that each voxel carries decides the applied task.

There are various works using voxels as the representation for further analysis or detection. The system can classify an object with its voxelized geometry. Zhirong Wu et al. proposed the 3D ShapeNets [3] to achieve object classification by the 3D representation. However, such a method still lacks the ability to work over a variety of domains. Also, the voxelization of geometry could be seen as a compression, which means the geometry information will be lost in the process of voxelization and leading to a drop in the accuracy of classification. For higher accuracy, one may need a voxel map with higher resolution, which, on other hand, lowers the efficiency. This issue ends up with a tradeoff between accuracy and efficiency.

2.3 Semantic Map

A semantic map usually refers to the map built with simultaneous localization and mapping (SLAM) coming along with semantic segmentation. In general, it's a map with extra information, such as the class of the object or the meaning of the space. Typically speaking, it is a map in the form of grids, voxels, or even sparse graphs, with the aforementioned information on each of the elements. This hugely expands its ability to represent a space in an informative manner. This kind of map can be an important part to achieve object searching, environment recognition, space exploration, reasoning, visual navigation, etc., and is usually used in the field of robotics, computer vision, and artificial intelligence.

There're multiple works utilizing semantic maps to achieve different tasks. Chaplot, D.S. et al. proposed a work using a semantic map to achieve object-goal navigation [4]. By building the semantic map, the system could use an agent taking the map to find a target object in an unknown environment. However, the same problem, the limitation coming from the semantic segmentation or classification model, could still be seen here. The semantic map with limited labels of classes has its drawback by its nature.

3. PROBLEM DEFINITION

We need to first define some terms in mathematical expressions for a clearer definition. In this section, we especially define the voxel map and the similarity search on the map.

3.1 Voxel Map

In this section, we will define the voxel map. Our voxel map is a set of voxels $V = \{s, c\}$, which consist of the corresponding embedding, $s \in R^n$, n=512, and the confidence state, $c \in [0,1]$, where 0 represents uninitialized. The dimension of the embedding is the same as the embedding size of CLIP. The voxel map is then defined as follows:

$$M = \{V_{ijk}, T\}$$
$$i = \{0, \ldots, w\text{-}1\}$$
$$j = \{0, \ldots, h\text{-}1\}$$
$$k = \{0, \ldots, d\text{-}1\}$$

where w, h, and d are the numbers of the voxels in axes x, y, and z, respectively. T is a 4x4 similarity transformation matrix, which transforms points from the world coordinate to the voxel coordinate. The transformation matrix defines the relationship between the voxel map and the world map. Each voxel is a unit cube in the voxel coordinate.

3.2 Similarity Search

To search the target on the semantic amp, we need to calculate the similarity between the given description and all the voxels. The similarity shows the correlation between the words and voxels. Those with higher similarities could possibly be more relative to each other. There are two popular methods to calculate the similarity between two vectors: cosine similarity or Euclidian distance. Here, we choose the cosine

similarity because it is more robust to the scale of the vector. The definition is as follows:

$$S_{ijk} = t \bullet s_{ijk} / ||t|| \bullet ||s_{ijk}||$$
$$s_{ijk} \in V_{ijk}$$

where t is the target embedding transformed from a word or a description by CLIP.

By exhaustively calculating all the similarities between the voxels and the target embedding, t, we can get a 3D heatmap. We can further emphasize some important regions by applying more operations, e.g., thresholding, normalization, or non-linear transform on the heatmap. The detail will be discussed in Section IV.

4. METHODOLOGY

The major purpose of the proposed system is to generate a voxel map with each voxel containing a semantic embedding. However, the image embedding represents the whole image, which means it is not assigned for each identity in the scene. Thus, we propose methods to properly assign the embeddings to corresponding voxels and generate the semantic voxel map with different semantic embeddings distributed over the map. The details of each process, from building a map to calculating the heatmap, will be discussed in the following sections.

4.1. System Overview

The overall system scheme is shown in figure 3. The system contains two regions: map building and searching, represented in the green and yellow blocks in figure 3, respectively.

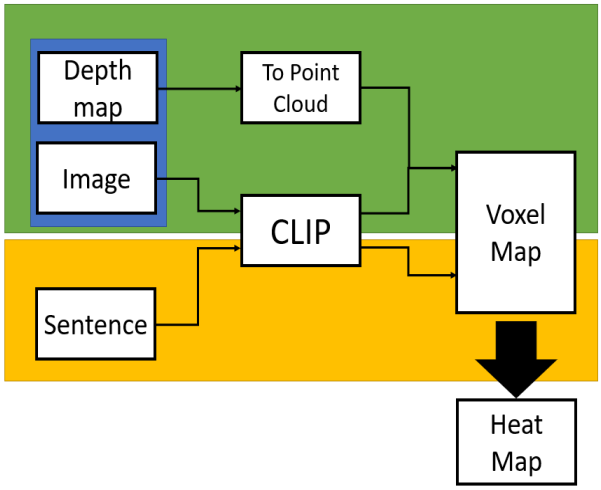


Fig. 3. The overall system schemes. The green region stands for map building and updating, while the yellow part is the searching stage.

In the map building stage, the system takes the RGB image and the depth image as the inputs. The CLIP module will later generate the embedding of the RGB image. The depth image will be transformed into point clouds. The system will take the embedding and the point clouds to either build the voxel map or update the semantic information on each of the voxels corresponding to the points in the point clouds.

As for the searching stage, the system takes a description as the input of the CLIP module and generates an embedding as the representation of the description. This embedding will be used to calculate the similarity for every single voxel on the map. Note that these two stages can be executed at the same time.

4.2. From Range Image to Point Clouds

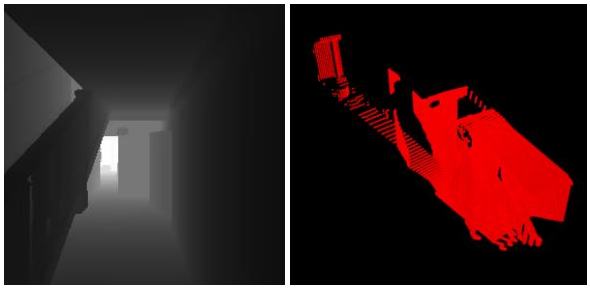
To correlate the RGB image with the voxel map, we need to know what the corresponding voxel for each pixel is so that we can assign the semantic embedding to the correct voxels to build or update the map. The

In this work, we assume perfect depth information and perfect location information, so we can transform all the pixels back to world coordinate correctly.

$$X_c = K^{-1} X_i$$
$$X_w = E^{-1} X_c$$
$$X_v = T X_w$$

where the K and E are the intrinsic matrix and the extrinsic matrix, respectively. And $X_i \in R^4$ is the position on the image coordinate with the depth as its z-axis. While $X_c \in R^4$ is the position in the camera frame, and $X_w \in R^4$ is the position in the world frame. T is the transformation from the world coordinate to the voxel coordinate.

(a)



(b) (c)

Fig. 4. This figure shows (a) the example scene, (b) the corresponding depth map, and (c) the corresponding point clouds after transformation. Note that the resolution of this figure is higher for demonstration.

By the transformation, we can cast all the pixels back to the voxel coordinate and assign each of them to a voxel for building the voxel map. An example of the transformation is shown in Figure 4.

4.3. Weighted Point Clouds

However, the content of an RGB image does not contribute to the semantic embedding evenly, i.e., only some of the regions of the image would dominate the semantic embedding. If we directly assign the embedding equivalently to all the voxels in the view, it could accidentally assign the embedding to those not related to it. In Fig. 5, the embedding of the image would be dominated by the meaning of "kitchen", however, the corridor will also be assigned with the embedding of the kitchen. To deal with this, we propose a weighting strategy for point clouds to decrease the assignment weight of non-related parts.

(a) (b)

Fig. 5. The example of an accident assignment to the corridor. (a) Image of the kitchen with the corridor captured on the right-hand side. (b) Weighted point clouds.

The major concept of the weighting strategy is to give those closer to the center the higher weight, and those farther the lower. We achieve this by first calculating the average position, p_{avg}, of the point clouds. Later, we can calculate the distance from each of the points to p_{avg} and we can weight all the points based on this distance. We can calculate the mean distance, d_{avg}, and its standard deviation, d_{std}. Then, the weight of each point:

$$w_i = 2.0 - (d_i - d_{avg}) / d_{std}$$
$$w_i = clamp(w_i, 0.0, 1.0)$$

where w_i is the final weight of point i. The clamp function trims the value of w_i to be in the interval [0,1]. This function means that the points inside a ball centered at the average position with a radius of d_{std} will have the weights of one. While those points lie outside the ball with a radius of two d_{std} will be given zero weights. Based on linear interpolation, those points lie between will be given weights between one and zero. In Figure 5(b), the points with higher weights are painted in red, while those with lower weights are in blue. One can see that to non-related part, corridor, is correctly assigned with a lower weight. With the proposed strategy, we can prevent accident assignment of the embedding to non-related voxels.

4.4. Voxelization

After we transform the points to the voxel coordinate and give each of them a weight, we need to voxelize them so that we can know which corresponding voxel is assigned in the next stage: voxelization.

Voxelization is conducted relatively simply. After we get the point clouds in the voxel coordinate, we can assign each point to the nearest voxel by a simple rounding operation over each dimension. For example, if a point is located at (0.25, 0.25, 0.25) in the voxel coordinate, then it will be assigned to voxel V_{000}. Equivalently, if a point falls inside a voxel, then we will assign this point to the voxel. If there is a point out of the range of voxels, it will not be assigned to any of them.

In Figure 6, the red dots, point clouds, are assigned to the voxels in the 3D space. Only those with any point falling in will be assigned as occupied and be visualized in Figure 6.

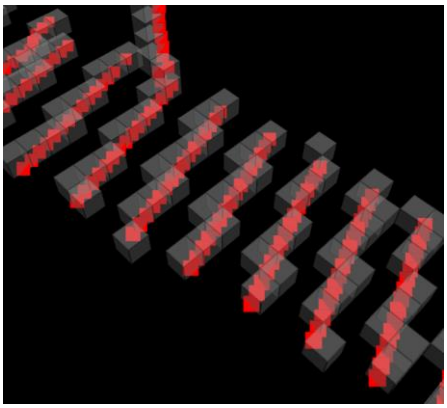


Fig. 6. The example of voxelization. The red dots are point clouds from the depth image, while the gray boxes are voxels generated from the point clouds.

4.5. Embedding Assignment

A major part of this work is the strategy to assign the embeddings to the corresponding voxels. The goal is to make the embedding distribution vary properly, rather than a directly hard assignment that makes all the embedding in the same scene identical so that we can find an object or search for certain things on the map. Thus, we propose a confidence-based embedding assignment strategy.

After the voxelization of a point, p_i, in the point clouds, we will assign the point to the corresponding voxel. Each of the voxels consists of a confidence score, c, which will be initialized as 0.0 at the beginning of the system. Later, for each assignment, the confidence will be updated with Exponential Moving Average, which, specifically, is a weighted sum of the current confidence and 1.0. The weighting factor, α, is set as 0.9 for smooth updating. The weighting factor α depends on the weight of the assigning point, p_i. The higher the weight of p_i is, the lower α is, which means that the point with higher weight updates the confidence of a voxel more. Confidence updating strategy:

$$\alpha = 1.0 - \beta w_i$$
$$c' = \alpha c + (1.0 - \alpha)$$

where c is the current confidence, and c' is the updated confidence. With this updating strategy, we can make sure that the confidence of each voxel will finally converge to 1.0 as the agent observes the same area more. This will make the embedding of each voxel more stable by updating it smoothly, and it can also prevent some unintended updates that could wash off the previously recorded embedding.

Moreover, each time the voxel is assigned, the embedding s will also be updated by a strategy similar to the confidence update. The difference is that the weighting factor, α', used to sum the current embedding, s, and the new embedding, s_{obs} , is the confidence, c', which is reversed and scaled by the weight of p_i. It means that the embedding update is controlled by both voxel confidence and point weight. Intuitively, lower confidence and a higher point weight result in a faster update:

$$\alpha' = w_i (1.0 - c')$$
$$s' = s (1.0 - \alpha') + s_{obs}\, \alpha'$$

where s' is the updated embedding of a voxel.

With our proposed mechanism, we can ensure that the embedding of each voxel will finally reach a stable state as the agent observes the region more. Moreover, this will result in an embedding that merges the information from a different perspective. Thus, we can prevent unstable semantic search results and provide a more general and averaged representation for each of the voxel.

4.6. Searching on Semantic Map

Finally, we need to search the similar embeddings among all the voxels inside the semantic voxel map to generate the heat map as the distribution of the likelihood that a target could possibly appear in a different region in the environment.

After we get a target embedding encoded to form a description, e.g., “The bedroom with one large red bed inside.” We can use this target embedding to search through the voxel map and calculate a similar score for each of the voxels by the cosine similarity in Section III-II. The similarity score calculated in this stage serves as the raw score, which we can further emphasize or process with some non-linear functions and normalization.

The first operation to do is normalization since the similarity is not promised to be uniform in its scale for different cases. For example, searching a yellow ball among a bunch of balls in different colors might result in a similarity distribution with small variation since all the embeddings are close to each other, while searching a cup among different types of dishes might result in a similarity distribution with a larger variation because a cup is more obvious to be different from other dishes. To make the distribution remain similar regardless of different situations, we will normalize the distribution of the similarity score to a distribution with its mean equal to 0 and its standard deviation to be 1.

Finally, we will apply the Softmax function to all similarity scores to get the final probability distribution over the voxel map. Softmax is a generalized logistic function in various fields, e.g., probabilistic, machine learning, deep learning, and so on. It can normalize a K-dimensional vector from real numbers R^K to a probabilistic distribution $(0, 1)^K$.

5. EXPERIMENT SETUP

Here, we will give descriptions of experiment details, including the simulation environment, dataset, specification, and the process of building the semantic voxel map.

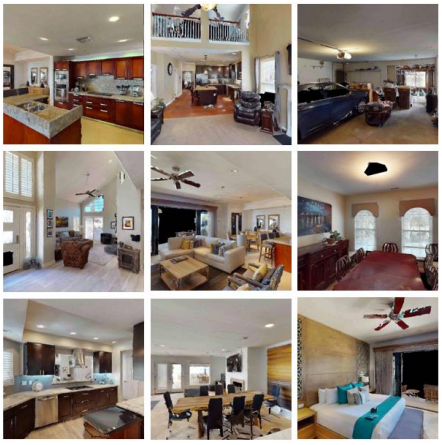


Fig. 7. Some typical examples of the Habitat-Matterport 3D Research Dataset (HM3D) with various indoor environments in different styles and types.

5.1 Simulation Environment

We simulate the environment in habitat-sim [5][6], a simulator developed by META for indoor environment simulation. This simulator was designed to simulate an

agent with configurable sensors, e.g., RGB camera, depth sensor, and configurable action of the agent, e.g., go forward, turn right, and left, and so on. Also, the agent can interact with the interactable objects in the simulation environment.

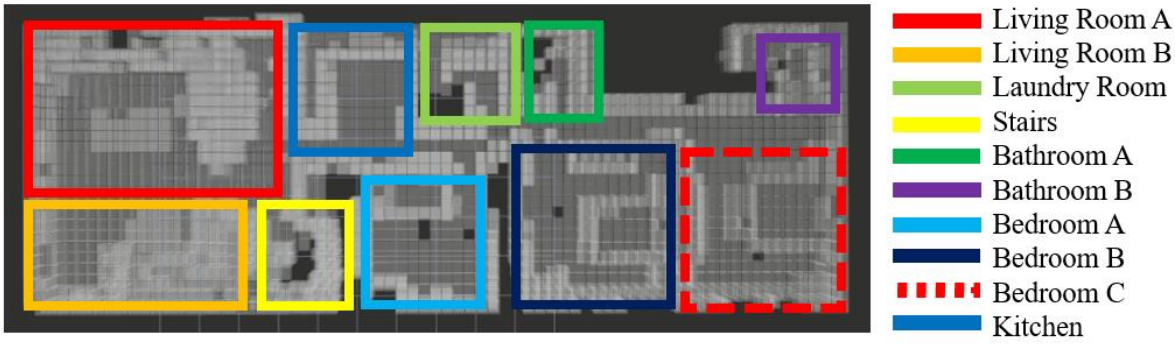


Fig. 8. The experimental scene and its corresponding types of spaces**.**

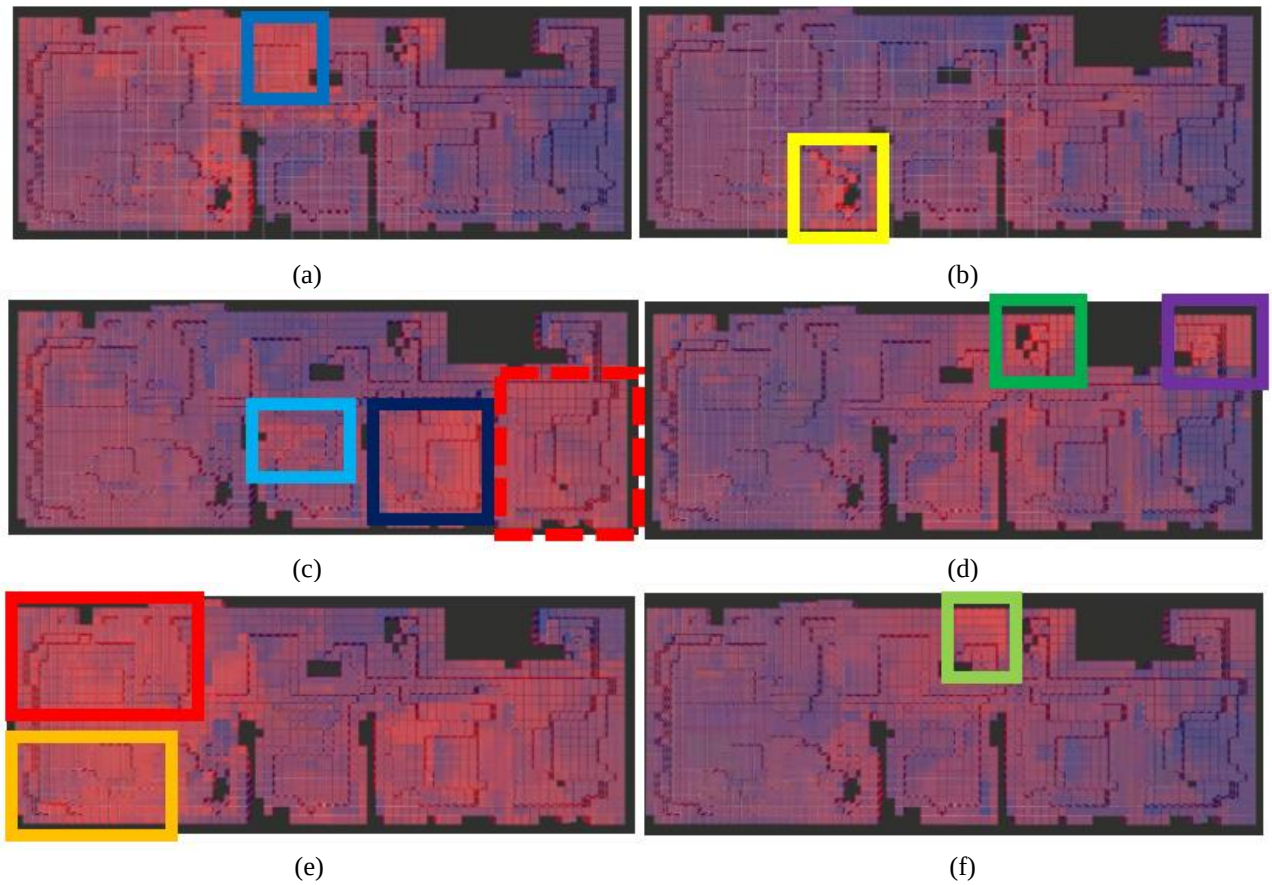


Fig. 9. The resultant heat map for different target embeddings. The target embeddings are (a) kitchen, (b) stairs, (c), bedroom, (d) bathroom, (e) living room, and (f) laundry room, respectively. The region with higher probability is shown in red. The bounding boxes show the ground truth, and their colors correspond to those in Fig. 8.

5.2 Indoor Dataset

The dataset is Habitat-Matterport 3D Research Dataset (HM3D) [7] provided by Matterport: large indoor 3D environment dataset containing 1,000 high-resolution 3D reconstructions of different buildings from the real world, including residential spaces, civic spaces, and others. We especially evaluate the system on some of the maps for validation. There are some typical examples of the scenes of HM3D in Figure 7.

5.3 Experiment Specification

The specification of the simulated agent is a robot with an RGBD camera. The resolution of the RGB

image is 512 x 512 pixels, later resized to 224 x 224 pixels as the input of the CLIP module. The resolution of the depth image is 50 x 50 pixels, which can be used for SLAM to build the voxel map. We assume ideal depth sensing and localization to achieve the perfect SLAM. This decouples the problem to focus on how to merge the semantic information from CLIP with the voxel map.

5.4 Building the Voxel Map

To build the semantic voxel map, we need to sample all the regions over the 3D map. However, we currently manually control the simulated robot in the house all around to sample the environment to build up and update the semantic voxel map. This process is manually done to ensure the quality of samples. If we use an automatic controller to scan all over the map, the result might be affected by the sampling efficiency and performance of the automatic controller. For example, unbalanced sample numbers or places that could never be visited. While controlling by a human could promise a consistent result.

6. EXPERIMENTAL RESULTS

In Figure 8, experimental environment contains various rooms with different looks. We show the ground truth types of rooms with our common knowledge and try to translate the type of the room into embedding, e.g., feed the word “bedroom” to the CLIP and get a corresponding embedding of the word. Moreover, we search through the semantic voxel map with our proposed method in Figure 9. We transform each of the room types into its embedding and search for each of them. In Figure 9, the highlight parts correctly show the region that could possibly appear to be the target type.

Although there might be some confusing parts, for example, the other region to be highlighted wrongly in Figure 9(a)(e). For Figure 9(a), the stairs were confused as a kitchen, while for Figure 9(e), bedrooms B and C are confused as living rooms, which is explainable since a bedroom and a living room could sometimes be similar to each other.

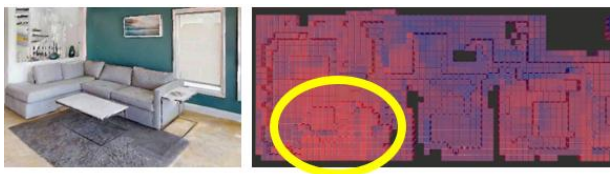


Fig. 10. An example of finding “sofa” in the semantic voxel map. The living room B is highlighted, and the sofa does exist in living room B.

Figure 10 shows an example of finding the “sofa”. The living room with a sofa is highlighted while the one without any sofa is darker. This example shows the ability of the proposed map to achieve general object search via the semantic voxel map.

7. CONCLUSIONS

In this work, we proposed a novel method to take advantage of the state-of-the-art visual-language model, CLIP, and integrate it into our system to leverage the ability of the semantic voxel map. The semantic voxel map generated with our proposed system is more general in terms of flexibility and has more potential in the future for high-level tasks related to linguistic commands.

We show the detail of our proposed method in detail and validate the system with quantitative experiments, which shows that our system can locate the target room or object in the semantic voxel map and distinguish between some delicate differences in descriptions. How to further increase the precision and distinguishability remain as the future works.

REFERENCES

[1] A. Radford, et al. "Learning Transferable Visual Models from Natural Language Supervision," *Proceedings of International Conference on Machine Learning*, Virtual, pp. 1-48, 2021.

[2] P. H. Le-Khac, G. Healy and A. F. Smeaton, "Contrastive Representation Learning: A Framework and Review," in IEEE Access, vol. 8, pp. 193907-193934, 2020, doi: 10.1109/ACCESS.2020.3031549.

[3] Z. R. Wu et al., "3D ShapeNets: A Deep Representation for Volumetric Shapes," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, pp. 1912-1920, 2015, doi: 10.1109/CVPR.2015.7298801.

[4] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, “Object Goal Navigation Using Goal-Oriented Semantic Exploration,” *Proceedings of Neural Information Processing Systems*, Virtual, pp. 1-11, 2020.

[5] M. Savva, et al. "Habitat: A Platform for Embodied AI Research," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Seoul, Korea, pp. 1-17, 2019.

[6] A. Szot, et al., "Habitat 2.0: Training Home Assistants to Rearrange Their Habitat," *Advances in Neural Information Processing Systems*, Vol. 34, pp. 1-32, 2021.

[7] S. K. Ramakrishnan, et al., "Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI," arXiv:2109.08238, 2021.