Complex & Intelligent Systems (2024) 10:5391–5409
https://doi.org/10.1007/s40747-024-01408-0

ORIGINAL ARTICLE

Lp-slam: language-perceptive RGB-D SLAM framework exploiting large language model

Weiyi Zhang[1] · Yushi Guo[1] · Liting Niu[1] · Peijun Li[1] · Zeyu Wan[1] · Fei Shao[1] · Cheng Nian[1] · Fasih Ud Din Farrukh[1] · Debing Zhang[1] · Chun Zhang[1] · Qiang Li[2] · Jianwei Zhang[2]

Received: 7 December 2023 / Accepted: 9 March 2024 / Published online: 30 April 2024

Abstract
With the development of deep learning, a higher level of perception of the environment such as the semantic level can be achieved in the simultaneous localization and mapping (SLAM) domain. However, previous works did not achieve a natural-language level of perception. Therefore, LP-SLAM (Language-Perceptive RGB-D SLAM) is proposed that leverages large language models (LLMs). The texts in the scene can be detected by scene text recognition (STR) and mapped as landmarks with a task-driven selection. A text error correction chain (TECC) is designed with a similarity classification method, a two-stage memory strategy, and a text clustering method. The proposed architecture is designed to deal with the mis-detection and mis-recognition cases of STR and to provide accurate text information to the framework. The proposed framework takes input images and generates a 3D map with sparse point cloud and task-related texts. Finally, a natural user interface (NUI) is designed based on the constructed map and LLM, which gives position instructions based on users' natural queries. The experimental results validated the proposed TECC design and the overall framework. We publish the virtual dataset with ground truth, as well as the source code for further research. https://github.com/GroupOfLPSLAM/LP_SLAM.

Keywords Simultaneous localization and mapping (SLAM) · Large language model (LLM) · ChatGPT · Natural user interface (NUI)

Introduction

Simultaneous localization and mapping (SLAM) is a perception-based problem in robotics that involves constructing a map of an unknown environment while simultaneously determining the robot's position in real-time. Traditional SLAM algorithms rely on sensors such as radars [1] and cameras [2–4] to build a geometric model of the environment and estimate the robot's pose.

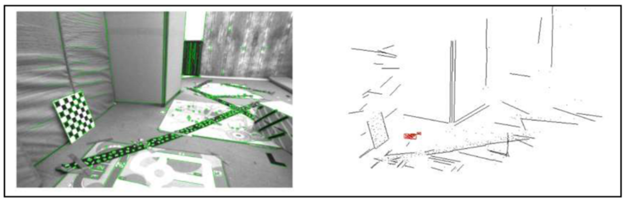
Traditional visual SLAM algorithms extract the geometry features from the scene, including feature points [5–7], feature lines [8, 9], feature planes [10, 11], or the combinations of the above. The feature points, lines, and planes can be classified into the "Geometry level" information of the scene, as shown in Fig. 1a. Emerging neural network technology [12–14] enhances SLAM systems with higher-level, semantic information, as depicted in Fig. 1b. The object detection algorithms can detect and recognize the distinguished objects in the correct classification [15, 16]. The detected objects can be leveraged in mapping and localization. In work [17], the dynamic objects are detected to help remove the dynamic features, which decreases the accuracy of the system. In work [18], the detected objects are constructed and inserted as landmarks into the map. Semantic segmentation assigns a classification label to each image pixel, offering more precise information for SLAM systems. In work [19], DS-SLAM was proposed to use semantic information for mapping, object recognition, and outlier filtering. In addition, semantic information was used to estimate the camera pose in work [20]. In another work, an object co-view was constructed with the semantic information for checking loop candidates based on the underlying geometric features during the loopback detection phase [21]. In previous works,

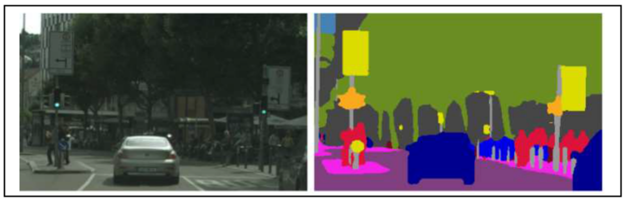
✉ Chun Zhang
zhangchun@tsinghua.edu.cn

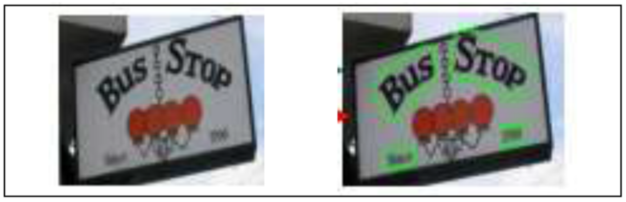
✉ Qiang Li
liqiang_hn_cn@hotmail.com

1 School of Integrated Circuits, Tsinghua University, Haidian, Beijing 100084, China

2 Group TAMS, Informatics and Natural Sciences Department of Informatics, Faculty of Mathematics, University of Hamburg, Vogt-Kölln-Straße 30 D, 22527 Hamburg, Germany


(a) Geometry level features


(b) Semantic level features



(c) Language level features

Fig. 1 Classification of features in different levels

semantics offer high-level feature information that improves localization accuracy, masks dynamic feature points, and assists in bundle adjustment (BA) and loop closure detection in SLAM.

Text is another type of complex feature that SLAM can extract, as shown in Fig. 1c. Several attempts have been made to use text for navigation and location. Rong et al. [22] built a text-assisted visual-inertial navigation system for blind people, which recognizes and tracks the pre-defined texts in a template-based method. Wang et al. [23] used SLAM to extract planar tiles, integrated multiple observations to detect text, and then fused the consecutive detections. In 2015, the same team proposed a "junction" descriptor for text spotting, integrated into real-time SLAM, which improves localization and text identification [24]. Li et al. [25] developed a text-based visual SLAM method that uses detected text as a planar feature with three parameters and illumination-invariant photometric error, resulting in more accurate 3D text maps for robotic and augmented reality applications. In [26], Tong et al. proposed a SLAM system with STR to extract the house numbers in the scene. The detected texts are labeled on the constructed map.

Current text-based SLAM systems have failed to utilize text information and only focus on the planar geometry of texts rather than their actual meaning. Thus the text information is used merely as one special geometry feature. These limitations are due to: (1) The robustness of STR falls short in providing pure and accurate information for SLAM systems. (2) The system lacks the capability to comprehend texts. Two cases of STR error often happen: mis-detection and mis-recognition. Mis-detection is the case where some text-like patterns in the scene are detected as text, thus becoming an outlier. Mis-recognition is the case where the detected texts are wrongly recognized, with spelling errors. Both two errors will severely decrease the mapping and understanding of the scene texts. With the development of LLMs models [27–29], SLAM systems can achieve a much higher understanding of the texts in the scene. The rich prior knowledge of LLMs also provides a new solution to unavoidable STR errors. In this work, LP-SLAM is proposed exploiting LLMs to deal with both challenges as mentioned before. Compared with the text-based SLAM systems above, LP-SLAM can extract the texts accurately, understand the texts, and use the text information according to the task and human queries. Therefore, the proposed LP-SLAM has a natural-language-level perception of the texts, instead of geometry-level in previous text-based works. The contributions of LP-SLAM include:

(1) A Large language model is introduced in the SLAM system first time and the proposed LP-SLAM has the ability of language perception during the construction of the map. The texts in the scene are extracted and understood to form a map with selected texts at the correct positions. A NUI is designed leveraging the LLM to bridge the users' query and the constructed maps, providing position instructions according to the specific demand.
(2) A TECC is proposed to deal with mis-detection and mis-recognizing cases, inspired by human cognition. The TECC includes three modules: (a) Similarity classification to tolerant the mis-recognition cases at the beginning of processing. (b) Two-stage memory strategy is employed to deal with the mis-detection cases, which is inspired by human's long-short memory. (c) Text-clustering to eventually solve the mis-recognition cases, where LLM is leveraged to correct the mis-recognized texts.
(3) The results based on the environments validated the proposed framework and TECC and showed that our LP-SLAM has the potential to enhance autonomous robots' ability to interact with their environment more naturally and intuitively.

Research background

Visual SLAM

Visual SLAM exclusively uses vision for external sensory perception. MonoSLAM [30] was the first real-time visual SLAM system that employed a monocular camera and Extended Kalman Filter (EKF) to estimate camera location and construct a sparse 3D map. PTAM [31] was proposed as the first SLAM system that has utilized graph optimization instead of filtering. PTAM's keyframe approach achieved superior performance with lower cost, and it introduced the concept of front-end tracking and back-end optimization to improve efficiency. ORB-SLAM [32], which has used FAST corner points and Oriented Brief (ORB) descriptors, running speed and accuracy have been improved as compared to PTAM. Additionally, ORB-SLAM incorporated Loop Closure Detection to optimize the map's pose and reduce accumulated drift errors.ORB-SLAM2 [2] extended the framework to support binocular and RGBD cameras, while also adding a pre-processing module in the tracking thread to handle more information from these devices, thus improving precision. Furthermore, it introduced full BA to rectify the map and added a localization mode. ORB-SLAM3 [33] is the latest release of the series and supports an even wider range of equipment and functions, such as pinhole, fisheye, and visual-inertial odometry. It is a multi-map system that creates a new map when VO is lost, and automatically merges with the previous map when the scene is recovered. ORB-SLAM3 has demonstrated robustness comparable to state-of-the-art systems while achieving higher precision. In recent years, SLAM systems using direct methods were also developed [34, 35] and showed advantages in efficiency. The algorithms leveraging machine learning also achieve higher accuracy. There are two ways to introduce machine learning into SLAM system. The modules such as feature extraction, feature matching, and loop closing can be replaced by machine-learning-based modules [36, 37], which improves the overall accuracy of the system. The machine learning algorithms can also be side-loaded along with SLAM to provide more information to the original system, such as object detection and semantic segmentation. Some works have used STR to extract the text information to help in localization as mentioned above. However, the previous systems could not get an understanding of the texts or only judged them as special geometry-level information. In this work, encouraged by the emerging LLMs, the potential of text information in SLAM system is explored.

Scene text recognition

The fundamental technology of our system is STR, which detects and recognizes scene texts. There are many text recognition approaches have been proposed but, Convolutional Recurrent Neural Network (CRNN) [38], which combines convolutional neural network (CNN), recurrent neural network (RNN), and connectionist temporal classification (CTC), has shown promising results in recognizing text in natural scenes. CNNs are used to extract features from the input images. The output is then fed into a bidirectional long-short-term memory (LSTM) layer to generate feature sequences and the network uses a CTC module to decode the sequence into corresponding text output. ASTER [39] is a recognition network based on the attention mechanism that can capture the spatial relationships between characters and recognize text accurately. The attention mechanism is beneficial for text recognition in natural scenes, where the text can appear at different scales, orientations, and positions in the image. Semantic reasoning network (SRN) [40] introduces a global semantic reasoning module (GSRM) to capture context, which is more robust and efficient than traditional methods. Meanwhile, SRN can be trained in an end-to-end approach and achieve state-of-the-art on multiple benchmarks.

Though STR has been largely developed in the past years, the accuracy in large-scale scenes is still not high enough for our proposed LP-SLAM. Two major errors appear frequently and severely influence further processing and understanding. Mis-detection stands for the case where some text-like objects in the scene are judged as texts, and some non-sense texts are extracted from them. Mis-recognition stands for the case where the correctly detected texts are judged into texts with spelling errors. Traditional algorithms are unable to recognize the errors due to the lack of prior knowledge of the world. However, trained by sufficient language data, LLM can correctly judge the errors and even correct them. In this work, the LLM is used as a key components to deal with the STR errors.

Large language model

A large language model (LLM) is a type of artificial intelligence (AI) system designed to process natural language and generate coherent, contextually appropriate responses to user inputs. LLMs are typically based on deep learning algorithms that are trained on massive datasets of text, allowing them to recognize patterns and relationships in language and generate highly accurate and relevant responses. One of the earliest examples of a large language model was the Statistical Language Model (SLM) [41], which was developed in the 1990s. This model was based on statistical techniques such as n-grams [42] and maximum likelihood estimation and was used for tasks such as speech recognition and machine translation. The most popular Large Language Models in use today are based on the Transformer architecture [43]. The Transformer architecture incorporates the self-attention mechanism, which allows the model to attend to various parts of the input sequence and capture complex relationships between them. The Transformer architecture offers several advantages over traditional RNNs and CNNs. It can process sequences in parallel, which makes it more efficient than RNNs and CNNs. It also enables the model to capture long-range dependencies and relationships between words and phrases, which is essential for natural language processing applications, such as language translation and language modeling.

GPT-3 (Generative Pre-trained Transformer 3) [28] is one of the largest and most powerful language models to date, with 175 billion parameters. GPT-3 was trained on a massive corpus of text data, including web pages, books, and articles, and can generate highly coherent and fluent text. Its advanced capabilities have significant implications for the field of natural language processing and offer exciting opportunities for applications in areas such as chatbots, content generation, and language translation.

LLMs have exhibited their power in natural language processing, facilitating robots in comprehending and responding to human commands with increased human-like proficiency [44]. This multifaceted nature of LLMs enables a more profound integration of linguistic comprehension and contextual understanding within the robotic domain [45]. Introduces an LLM-based approach to translate natural language commands into linear temporal logic (LTL) specifications for robot tasks [46]. Finds that when paired with LLMs to deconstruct complex natural language instructions into subgoals, their robots accomplish intricate, multi-tier tasks in real-world scenarios.

The essence of efficient task planning and scheduling lies in the ability to comprehend and interpret intricate instructions and commands, alongside the capability to strategize and allocate resources effectively. LLMs enable robotic systems to ingest and comprehend diverse sets of data, ranging from textual instructions to real-time environmental cues, and subsequently formulate optimized task plans and schedules [47]. Introduces a scene representation framework integrating contextual information into LLMs planners through VLMs, enabling robots to generate context-aware plans in real-world tasks without predefined object lists or fixed executable options [48]. Proposes a programmatic LLMs prompt structure for generating context-aware plans, demonstrating improved success rates in household and tabletop tasks across virtual and physical environments.

LLMs also have some other applications in the robotics field, such as the development of dialog agents and visual grounding [49]. Introduces a multisensory perception approach for robotic manipulation, emphasizing the coordination of visual, tactile, and auditory perception to handle complex situations, accompanied by a user-friendly mobile app based on LLMs [50]. Presents an interactive visual grounding system that effectively handles open-world scenes with ambiguous natural language instructions, through its integration of large-scale vision-language models and traditional decision-making processes. While SLAM is becoming one critical module in autonomous robots, the potential of LLMs in SLAM is not explored yet.

Proposed methods

Overall framework

The overall framework of LP-SLAM is shown in Fig. 2. LP-SLAM has 4 parts: geometry feature based SLAM, runtime text mapping, distilling, and natural user interface. Three modules leverage LLM and are marked as red.

The geometry feature based SLAM has four typical modules including tracking, local mapping, loop closing, and full BA. It estimates the pose for each frame and generates point cloud for the map. The estimated pose T_{wc} will be used to calculate the position of texts in the 3D world coordinates. In this work, ORB-SLAM3 is used as the base SLAM system. The tracking module runs in a frontend thread while the other modules run in backend threads. Thus tracking module runs in real-time and other modules do not require real-time execution.

The whole runtime text mapping part is executed along with the tracking module of the SLAM system in real-time. The input RGB image is firstly processed by the deep neural network (DNN) based scene text recognition (STR) module to extract the set of visible texts $T_i = t_{i1}, t_{i2}, ..., t_{in}$, where t_{in} denotes the $n - th$ text in the $i - th$ image. The corresponding 2D pixel position of each text is set as the center of the detection box. One text may appear in many frames and the extraction results (both content and position) differ in frames due to the limited accuracy of STR. The texts extracted from the same object in the real-world scene in different frames are called homologous texts. One real-world text object will produce many homologous texts, some of which have spelling errors (mis-recognition). The 2D pixel positions of homologous texts also differ due to the change of viewpoint and measurement error. There are mis-detection cases when some textures in the scene are similar to texts. In these cases, meaningless texts will be produced. To process the homologous texts, and deal with mis-recognition and mis-detection, TECC is proposed and used after the STR. TECC includes three modules: similarity classification, two-stage memory, and text clustering. The TECC modules will be discussed in detail in the following subsections. The TECC modules will finally generate the accurate text item for each group of homologous texts, correcting the spelling errors. The mis-detection cases will also be deleted.

The distilling part, containing two functions, is executed along with or after runtime text mapping. The text items generated by TECC are judged whether are task-relevant leveraging LLM. The texts that are not task-relevant will be discarded. Position clustering is performed on the reserved items to generate the corresponding positions in the 3D world coordinates by clustering algorithms from the stored posi-

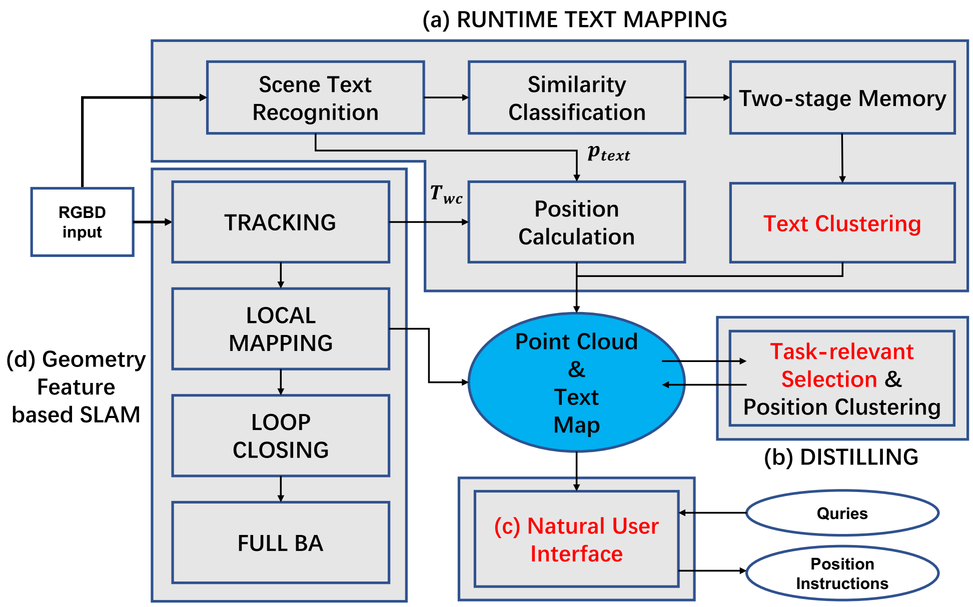


Fig. 2 Overall framework of LP-SLAM

tions. The items will be inserted in the 3D world map at the corresponding position. The point cloud & text map is thus constructed With the text items, and the geometry feature points calculated by the tracking module of the base SLAM system.

The navigation part is executed after the above two parts when the point cloud & text map is established. The query from users in natural language is input and processed by LLM to understand the users' demand. Then LLM provides position instructions according to the map and the demand, which can be used for the follow-up navigation algorithms.

Text error correction chain (TECC)

Similarity classification

Similarity Classification serves two major functions: (1) Classify the homologous texts from the same source into one group for further processing. (2) Tolerate the mis-recognition cases of STR. The STR may identify minor errors in the characters of some source text, resulting in the potential for multiple items from the same source text. In this paper, we modified the Levenshtein Distance algorithm [51] with normalization to measure the distance between texts. The Levenshtein distance algorithm employs the concept of backtracking in the comparison process, allowing for recursive operations. For the comparison between two strings A and B of lengths m and n, the matrix D[n+1][m+1] is constructed. This matrix is populated by circulating through each cell D(i,j) and calculating its corresponding value. The state transfer equation is as (1) and (2):

$$D(i,j)=\begin{cases}0, & \text{if } i=0,\ j=0\\ j, & \text{if } i=0,\ j>0\\ i, & \text{if } i>0,\ j=0\\ \text{Min}, & \text{if } i>0,\ j>0\end{cases} \tag{1}$$

$$Min = 1+\min\begin{cases}D(i-1,j)\\ D(i,j-1)\\ D(i-1,j-1)-k\end{cases} \tag{2}$$

where k=1 when the last letters of the two strings are the same else k=0. After recursive calculation, the final D(n+1,m+1) is the final edit distance length. We utilize (3) to normalize the edit distance to obtain a more accurate measure of string similarity, regardless of differences in length. This allows us to establish a uniform threshold for classification. If two strings are deemed sufficiently similar, they will be grouped into a single class for further processing by LLM.

$$Sim(A,B)=1-\frac{D(n+1,m+1)}{\max(m,n)} \tag{3}$$

The similarity classification is shown in Fig. 3a. The texts extracted by STR are first processed by the similarity classification module. This module merges homologous texts into classes, upon which subsequent memory processes depend. When a text is extracted and input to the similarity classification module, the similarities between the text and current classes are calculated. If there is one class achieving a similarity higher than a threshold, the text is merged into the class. Otherwise, one new class is created.

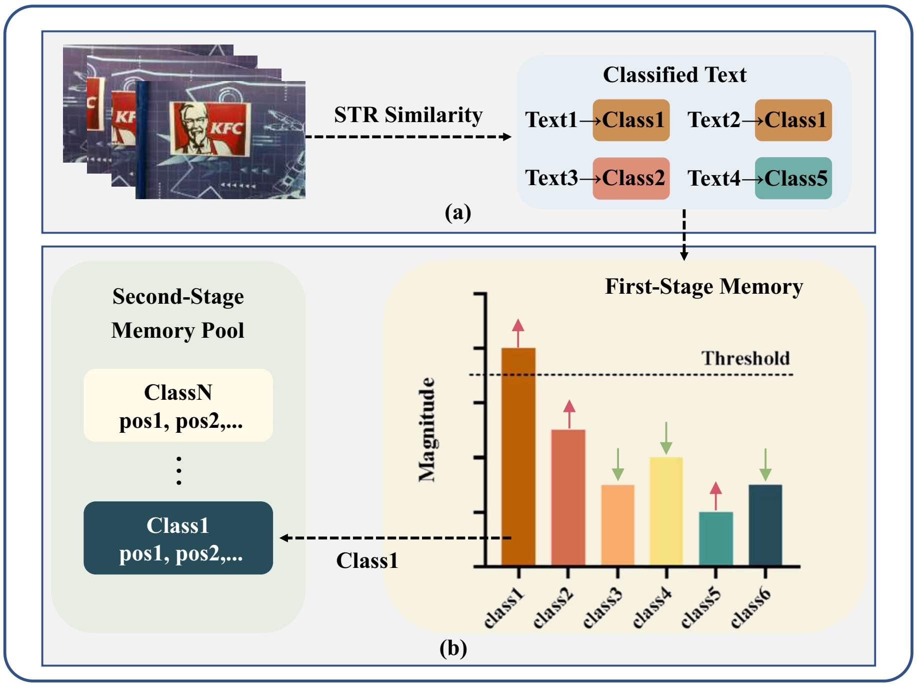


Fig. 3 Similarity classification and two-stage memory strategy

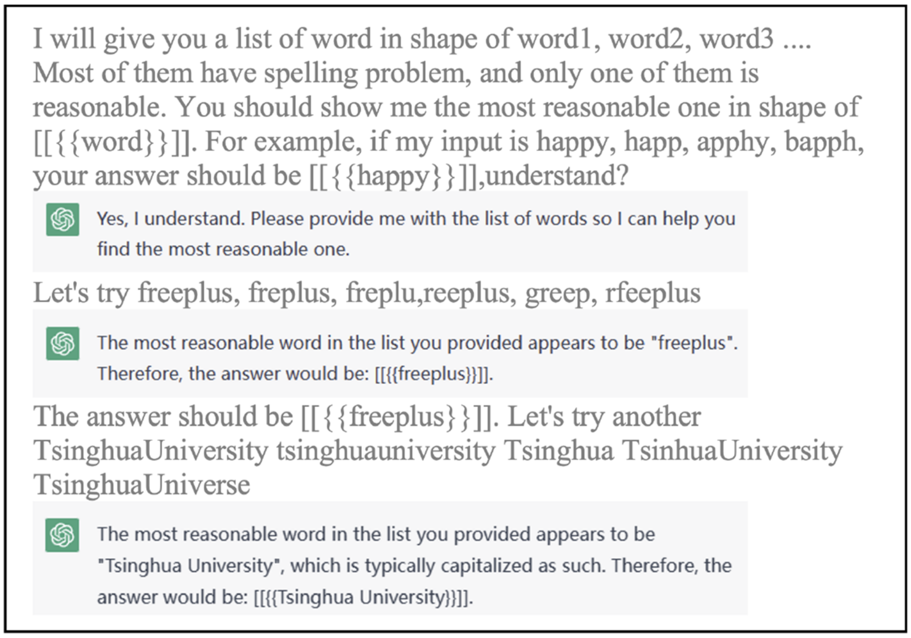
Two-stage memory strategy

While similarity classification addresses mis-recognition by accommodating spelling errors and grouping similar texts into one class, the mis-detection cases will result in meaningless classes. Two-stage memory strategy is proposed to filter out those meaningless classes, inspired by human cognition. The mis-detection happens in a low frequency, usually when one texture in the scene looks similar to text at a certain point of view. However, as the point of view changes, the mis-detection from one source will no longer happen. In contrast, texts will be constantly extracted from the real text source as the point of view changes. In human memory, objects seen only once or briefly are quickly forgotten, whereas those encountered frequently or over extended periods are remembered. Two-stage memory strategy mimics this mechanism to filter out the seldom mis-detection as shown in Fig. 3b. The first-stage memory calculates the memory magnitude of all classes. For simplification, the memory magnitude is closely related to the occurrences of the classes. At each frame, when a text is assigned to a class, the corresponding memory magnitude for the class increases. To prevent magnitude overflow, we implemented a fading-out function that simulates the human brain's forgetting mechanism. This function decreases the memory magnitude of all classes in each frame. Once the memory magnitude for a given category reaches the threshold, it is transferred into the long-term memory, which can later be used for task-relevant judgment and NUI navigation tasks. Otherwise, when the magnitude decreases to 0, it will be forgotten from first-stage memory. In the example of Figs. 2, 4 texts are detected from the current frame and are merged into three classes. Then the memory magnitude of each class changes according to the memory and forgetting mechanism. Then Class1 achieves a magnitude above the threshold and is stored in the long-term memory pool. The memory mechanism ensures that low-frequency mis-detection information is never transferred to second-stage memory, thereby eliminating the influence of irrelevant data on the system from the outset.

Text clustering

Even though similarity classification can merge homologous texts into classes, the issue of selecting the correct word in each class remains, which is referred to as the "text clustering problem". This problem is hard to deal with in previous works. The understanding capability and the prior knowledge about the world of LLM provide a new solution to the task. To effectively leverage LLM for text clustering, a prompt consisting of the mission description and example training is designed as shown in Fig. 4. Once pre-training of the prompt is completed, the LLM can be used as a text clustering tool in multiple languages. In this scenario, LLM is capable of selecting the most appropriate word in each provided class. The text clustering function is implemented as the end module of TECC to reduce the execution times of LLM, which requires high computation. The classes filtered by two-stage memory are fewer and no meaningless classes will be clustered by LLM.

Fig. 4 Pre-train of LLM for text clustering



I will give you a list of word in shape of word1, word2, word3 .... Most of them have spelling problem, and only one of them is reasonable. You should show me the most reasonable one in shape of [[{{word}}]]. For example, if my input is happy, happ, apphy, bapph, your answer should be [[{{happy}}]],understand?

Yes, I understand. Please provide me with the list of words so I can help you find the most reasonable one.

Let's try freeplus, freplus, freplu,reeplus, greep, rfeeplus

The most reasonable word in the list you provided appears to be "freeplus". Therefore, the answer would be: [[{{freeplus}}]].

The answer should be [[{{freeplus}}]]. Let's try another TsinghuaUniversity tsinghuauniversity Tsinghua TsinhuaUniversity TsinghuaUniverse

The most reasonable word in the list you provided appears to be "Tsinghua University", which is typically capitalized as such. Therefore, the answer would be: [[{{Tsinghua University}}]].

Task-relevant selection

Some texts, while meaningful, may not be relevant to the specific task at hand. For instance, when the task is navigation in a mall, the slogans are not relevant, while the shop names are relevant. The irrelevant texts consume storage space and increase the token length when LLM is used to process the texts according to human queries in the NUI. Therefore we propose a landmark judgment realized by LLM to decide whether a text extracted is task-relevant. The prompt of task-relevant selection is shown in Fig. 5. The pre-training process consists of two key components. Initially, we claim the task that LLM is expected to accomplish and subsequently, we provide relevant examples to ensure optimal performance.

Position calculation and position clustering

The position of text in world coordinates is determined by its pixel position from STR and the current camera pose in SLAM tracking part. When one point is projected onto the camera image, the point $P(x_w, y_w, z_w, 1)$ is firstly transformed into the camera coordinate:

$$P\left(x_c, y_c, z_c, 1\right) = T_{CW} \cdot P\left(x_w, y_w, z_w, 1\right) \tag{4}$$

where T_{CW} is the 4×4 transformation matrix from world coordinate to camera coordinate. Then $P(x_C, y_C, z_C, 1)$ is projected onto the image coordinates as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{Z_c} \cdot \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \overset{\text{def}}{=} \frac{1}{Z_c} \cdot F_c \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{5}$$

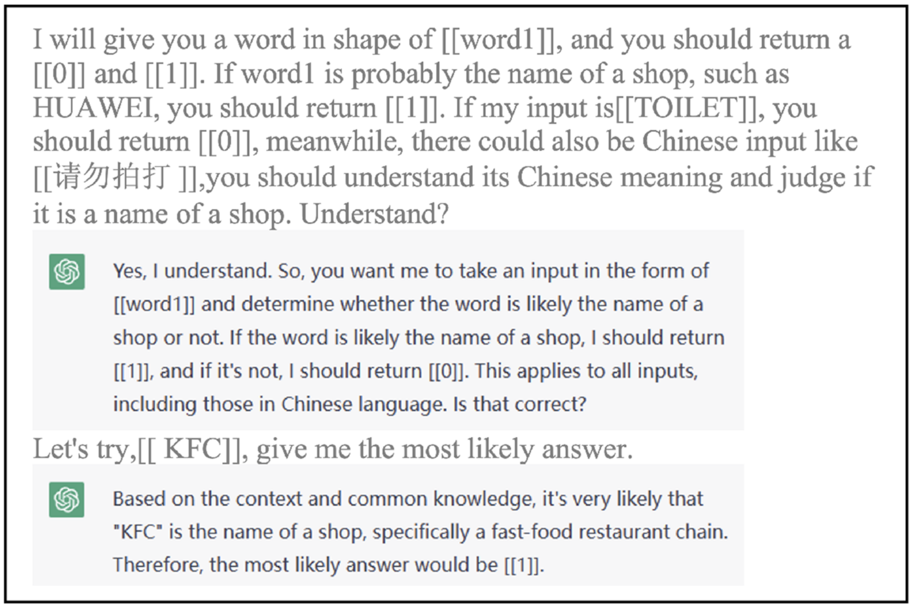
Finally, the pixel position is calculated by affine transformation based on the intrinsic of the camera:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{d_x} & 0 & u_0 \\ 0 & \frac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \overset{\text{def}}{=} I_c \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{6}$$

Inversely, the position P in real-world coordinates can be calculated from the pixel position (u,v) as:

$$P\left(x_w, y_w, z_w, 1\right) = Z_c \cdot T_{wc} \cdot F_c^{-1} I_c^{-1} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{7}$$

The pixel position of the detected text is calculated as the average of the four counters of the detection box. Each class of homologous texts has many estimated positions from different frames in different points of view. The clustering method is implemented to generate the final position of each class of homologous texts. For each class, N iterations of clustering are executed to remove the outliers. In each iteration, the average position is calculated, which is also the optimum of least squares. Then the 20% farthest points from the average position are judged as outliers and removed.


I will give you a word in shape of [[word1]], and you should return a [[0]] and [[1]]. If word1 is probably the name of a shop, such as HUAWEI, you should return [[1]]. If my input is[[TOILET]], you should return [[0]], meanwhile, there could also be Chinese input like [[请勿拍打]],you should understand its Chinese meaning and judge if it is a name of a shop. Understand?

Yes, I understand. So, you want me to take an input in the form of [[word1]] and determine whether the word is likely the name of a shop or not. If the word is likely the name of a shop, I should return [[1]], and if it's not, I should return [[0]]. This applies to all inputs, including those in Chinese language. Is that correct?

Let's try,[[KFC]], give me the most likely answer.

Based on the context and common knowledge, it's very likely that "KFC" is the name of a shop, specifically a fast-food restaurant chain. Therefore, the most likely answer would be [[1]].

Fig. 5 Pre-train of LLM for task-relevant selection

Natural user interface

Different navigation algorithms are implemented on the constructed map of SLAM systems. However, the current algorithms focus on route planning given the start and end position. Compared with the explicit queries asking position of specific text, the implicit queries that only give the demand are more difficult. In this case, the framework should understand the texts and the demands, and logically bridge them. In this work, we explore the possibility of LLM to understand both the constructed map and the human queries. The natural use interface is designed to use the framework in real-time and natural language. The LLM will take in the text lists of the map, and give position instructions according to human queries in natural language. The prompt is shown in Fig. 6.

Experimental results

Real-world experiments

The real-world experiment is conducted in a mock mall environment containing shops, slogans, and public facilities shown in Fig. 7. The platform robot is a Roban child-sized humanoid robot from Leju Robotics with an Intel RealSense D435i RGBD camera. The ChatGPT−3.5 is used as LLM in this work. The functionality of the modules will be validated in real-world experiments, and the thresholds will be adjusted accordingly for the larger simulation experiments.

Scene text recognition

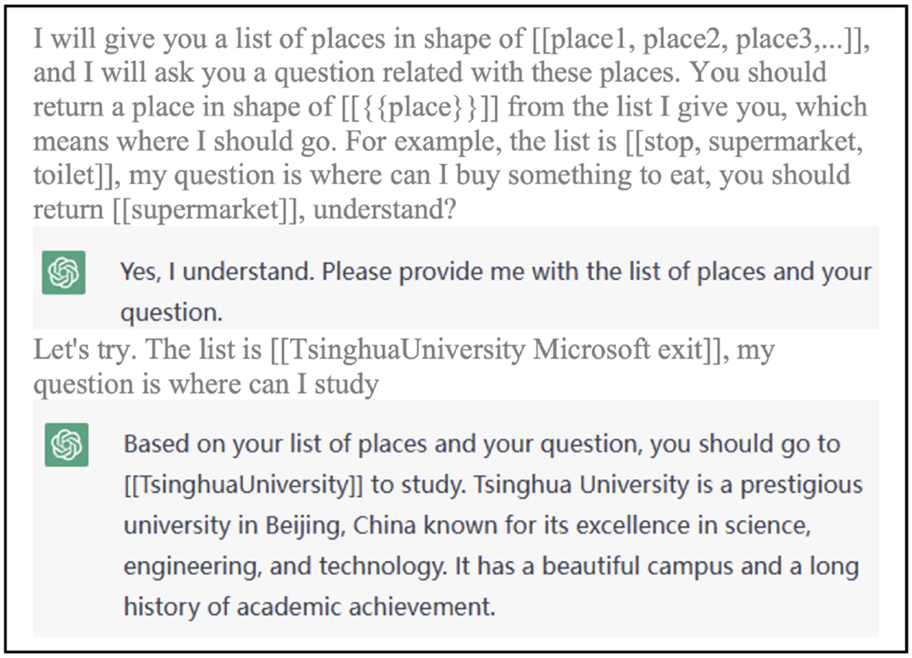
The result of scene text recognition is illustrated in Fig. 8. In Fig. 8a, the brand name “KFC” is correctly detected and recognized. The pixel position of the text is also within the threshold; thus, the detected text is surrounded by a green box and is reserved for the following processes. In Fig. 8b, three texts are detected and recognized. However, they are all surrounded by a yellow box and will be eliminated. The left upper is one part of the slogan “No Smoking”, which is not fully viewed in the current image. The cases where texts are not fully viewed will be removed. The left bottom text is one case of mis-detecting, which means that one non-text area is judged as a text area by the detecting module. The following two-stage memory strategy will solve such cases. The correctly detected text “KFC” is also eliminated because it’s near the bound of the frame. There are also mis-recognition cases. For example, “请勿拍打”(Do not beat in English) is always recognized as “ “请切拍打”(Meaningless Chinese sentence). The similar classification module and class clustering module will solve such cases.

To evaluate the performance of STR and TECC more accurately, a similarity-frequency-based accuracy calculation method is developed as (8). The accuracy is calculated as the sum of the frequency-weighted similarity between each extracted text and the ground truth text. We achieved a stable character accuracy rate of 83.59% in the STR stage.

$$\text{acc} = \sum_{i=0}^{N} \text{Sim}\left(\text{text}_i, \text{groundtruth}_i\right) \times \text{freq}_i \tag{8}$$

Similarity classification and two-stage memory

Figure 9 displays the results of the similarity classification module and two-stage memory strategy module. The extracted texts from all frames are shown in Fig. 9a, which


I will give you a list of places in shape of [[place1, place2, place3,...]], and I will ask you a question related with these places. You should return a place in shape of [[{{place}}]] from the list I give you, which means where I should go. For example, the list is [[stop, supermarket, toilet]], my question is where can I buy something to eat, you should return [[supermarket]], understand?

Yes, I understand. Please provide me with the list of places and your question.

Let's try. The list is [[TsinghuaUniversity Microsoft exit]], my question is where can I study

Based on your list of places and your question, you should go to [[TsinghuaUniversity]] to study. Tsinghua University is a prestigious university in Beijing, China known for its excellence in science, engineering, and technology. It has a beautiful campus and a long history of academic achievement.

Fig. 6 Prompt for natural user interface

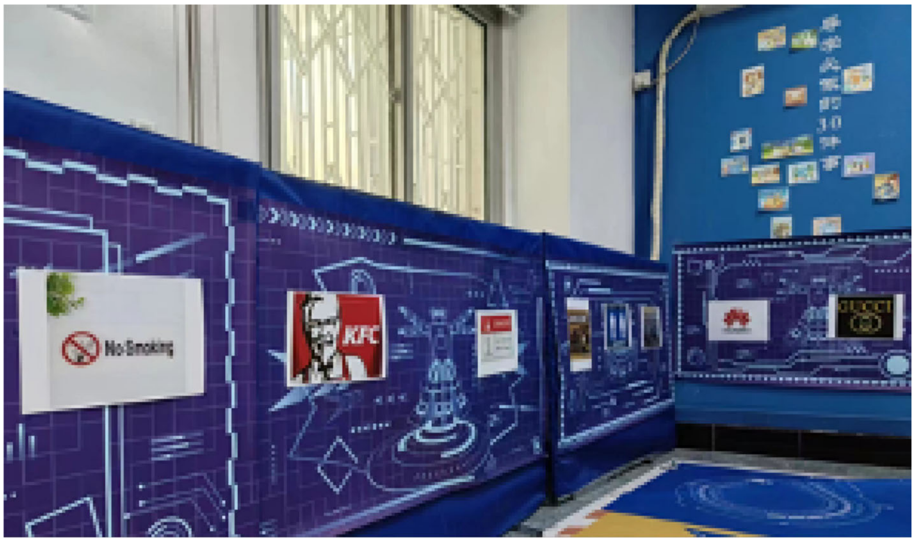


Fig. 7 Experimental environment of the mock mall

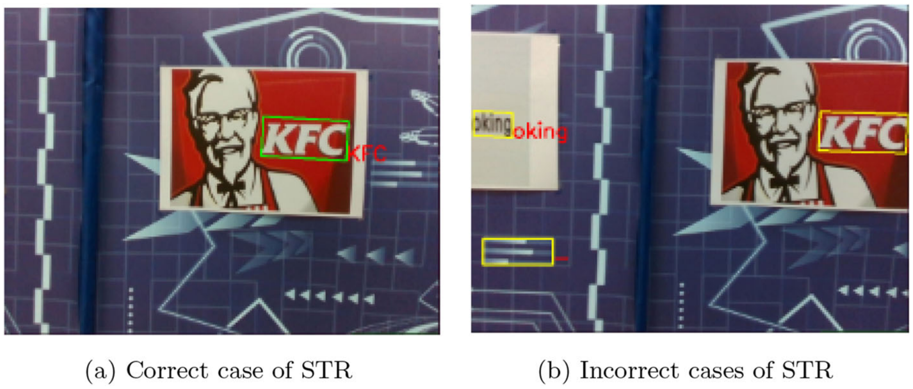


Fig. 8 Demonstration of correct and incorrect cases of STR

(a) Correct case of STR (b) Incorrect cases of STR

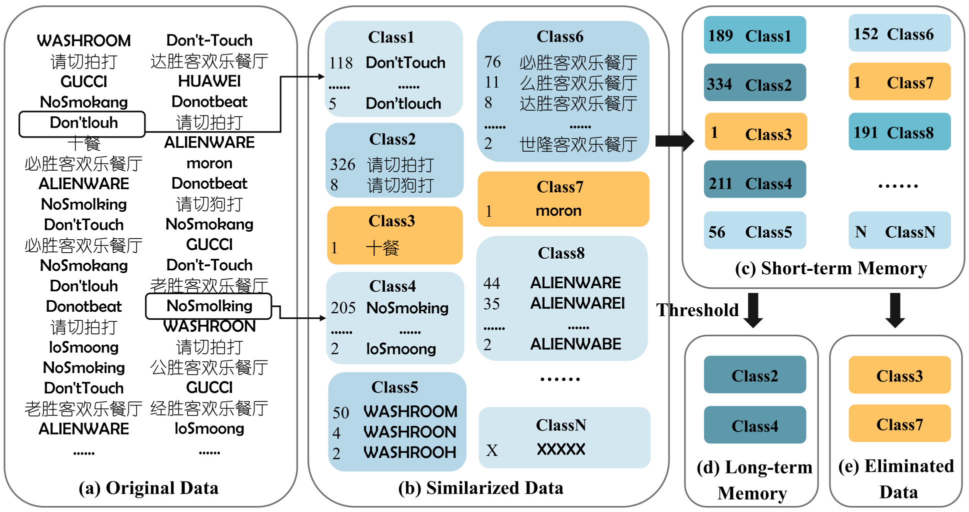


Fig. 9 Demonstration results of similarity classification and two-stage memory strategy

are disorganized and contain errors due to STR accuracy. Those texts are processed once extracted during the runtime and are listed together for result visualization when the mapping is finished. The classes of homologous texts are generated as shown in Fig. 9b. The number preceding each text indicates its frequency, which largely influences the magnitudes of the subsequent two-stage memory strategy. The magnitudes of each class are stored and adjusted in first-stage memory as shown in Fig. 9c. The darker colors represent higher magnitudes for classes in blue, while classes in yellow are occasionally extracted mis-detection cases. The classes such as class2 ("请切拍打", wrongly spelled "Don't beat") and class4 ("NoSmoking") entered the long-term memory, hile classes such as class3 ("十餐") and class7 ("moron") are eliminated from memory.

The magnitude threshold influences the text extraction result significantly. If the magnitude is too high, some correct texts may be discarded. In contrast, the mis-detection cases will be added to second-stage memory if the magnitude is too low. Two metrics are designed to decide the magnitude threshold: the miss detection rate (MDR) and the false detection rate (FDR). The MDR is calculated as the ratio of wrongly or not detected classes and the number of expected classes. The FDR is calculated by dividing the number of expected classes by the number of detected classes. The analysis of the MDR and FDR across different memory thresholds is shown in Table 1. Both MDR and FDR drop to 0 in the range of 25–200, where no expected classes are missed, and no extra classes are detected. This means all the mis-detection and mis-recognition cases are discarded accurately, and all real texts are successfully extracted. The functionality of similarity classification and two-stage memory strategy is validated and it can be observed in Fig. 9.

Table 1 MDR and FDR at different thresholds

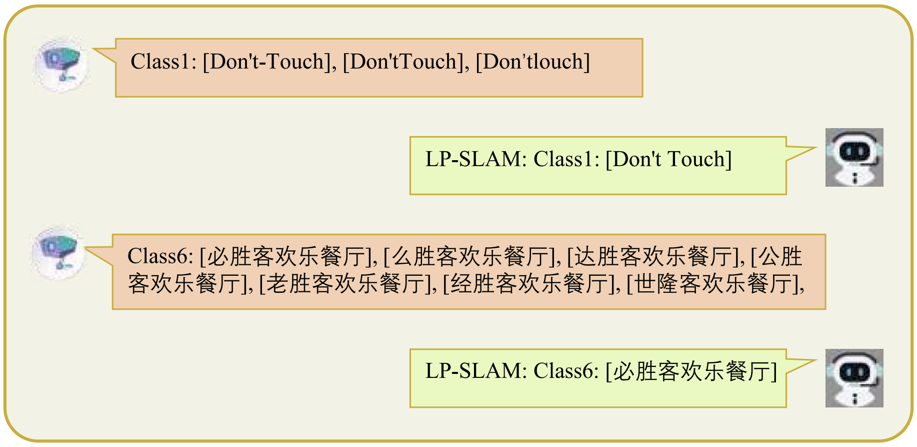
Threshold	10	20	25–200	250	300
MDR	0	0	0	8.3%	16.7%
FDR	25.0%	7.6%	0	0	0

Text clustering

Figure 10 demonstrates the effectiveness of the text clustering module with LLM, which returns the most reasonable text from a class of homologous texts. In the first dialog, we present a group of homologous texts with spelling variations, including [Don't-Touch], [Dont'tTouch], [Don'tlouch]. LLM can accurately identify the most reasonable text, which is [Don't Touch]. The next dialog demonstrates LLM's ability to handle different language texts. All the inputs are in Chinese and contain some spelling variations of "Pizza Hut". LLM can identify the correct text, despite the variations in spelling. This showcases the versatility of LLM in handling different languages and its ability to accurately cluster homologous texts.

Task-relevant selection

The task-relevant selection module in LP-SLAM identifies whether the texts are relevant to the given task. In our experiment scene, we focus on identifying whether the text stands for a shop, which is further used for customer navigation. To simulate real-world street environments, our dataset includes shop names, warning slogans, and public facilities written in both English and Chinese, which are generated by text clustering. As shown in Table 2, LP-SLAM accurately identifies all the landmarks as shops, and the irrelevant ones that are warning slogans, such as "请切拍打" (Do not beat, with one wrong word) and "HIGHTEMPERATURE". This showcases the effectiveness of the Task-relevant Selection module


Class1: [Don't-Touch], [Don'tTouch], [Don'tlouch]

LP-SLAM: Class1: [Don't Touch]

Class6: [必胜客欢乐餐厅], [么胜客欢乐餐厅], [达胜客欢乐餐厅], [公胜客欢乐餐厅], [老胜客欢乐餐厅], [经胜客欢乐餐厅], [世隆客欢乐餐厅],

LP-SLAM: Class6: [必胜客欢乐餐厅]

Fig. 10 Demonstration results of text clustering

Table 2 Task-relevant selection results in the real-world scene

Landmark
[KFC], [必胜客欢乐餐厅(pizza hut)],
[ALIENWARE], [HUAWEI], [GUCCI]
Non-landmark
[Donotbeat], [请切拍打(Do not beat, with one wrong character)], [NoSmoking], [Don't Touch],
[DANGER], [HIGHTEMPERATURE], [WASHROOM]

in accurately identifying different types of texts, despite the presence of multiple languages in the dataset.

Overall mapping

The overall mapping is illustrated in Fig. 11. The map is constructed with the recognized texts and the sparse point cloud. The blue texts are the ones judged as shops, while the pink ones are judged as others. The visualized results qualitatively validate the accuracy of text extraction, perception, and localization.

Natural language interface

The NUI of LP-SLAM gives position instructions according to the constructed map and human queries in multiple languages. Fig. 12 illustrates how the NUI works in LP-SLAM. In the first dialog, after presenting a list of landmarks in text mapping, LP-SLAM understood the user's request (I do not like pizza, where can I eat), recommended KFC, and provided its location for easy navigation.

In the second dialog, when the user asked in English "Where can I eat pizza?", LP-SLAM suggested going to "必胜客欢乐餐厅" (Pizza Hut). This showcases LP-SLAM's ability to handle queries in different languages.

The third dialog demonstrates one case in Japanese. When the user asked in Japanese, "ゲームをするためのパソコンはどこで買えますか? " (Where can I buy a computer for games?), LP-SLAM suggested "ALIENWARE" and provided its location for easy navigation.

Overall, LP-SLAM's ability to handle queries in multiple languages provides an additional advantage for users who speak different languages or are visiting foreign countries. The integration of LLM into the system further enhances its capability to handle complex queries and provide accurate recommendations.

Visualized comparison

The visualized comparison between the proposed framework and the related works is shown in Fig. 13. The TextSLAM [25] is shown as Fig. 13a, where the texts are only detected but not recognized as language information. The extracted texts are used as special geometry patterns for tracking. The TXSLAM [26] is shown as Fig. 13b, where the door numbers are detected and recognized. However, the numbers are easier compared with language characters. TXSLAM cannot correct and analyze the extracted information. There is one error "770n" in the constructed map. The proposed LP-SLAM is shown as Fig. 13c. LP-SLAM detects and recognizes the texts in the scene accurately with the help of the proposed TECC. It also has an outstanding comprehension of the extracted texts and can judge whether the texts are task-relevant. The judged texts are shown on the map in different colors. In addition, the text-map-based guidance is also firstly achieved in the related works.

Virtual experiments

Though the real-world experiments have validated the proposed method, the quantified results are lacking due to the absence of ground truth. Thus the virtual experiments are conducted for more complex environments and the quantified results. The Unreal Engine and Airsim are used to construct an environment of a mall with different shops and slogans.

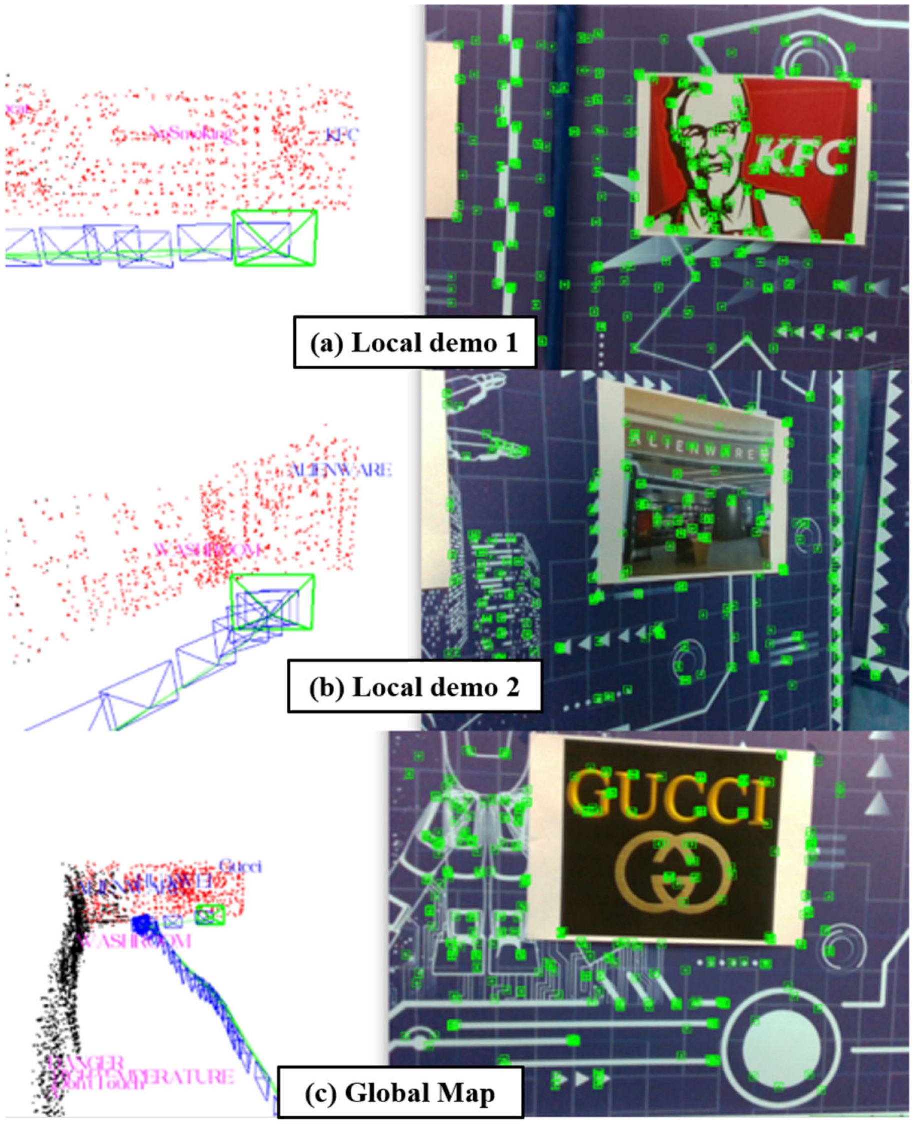
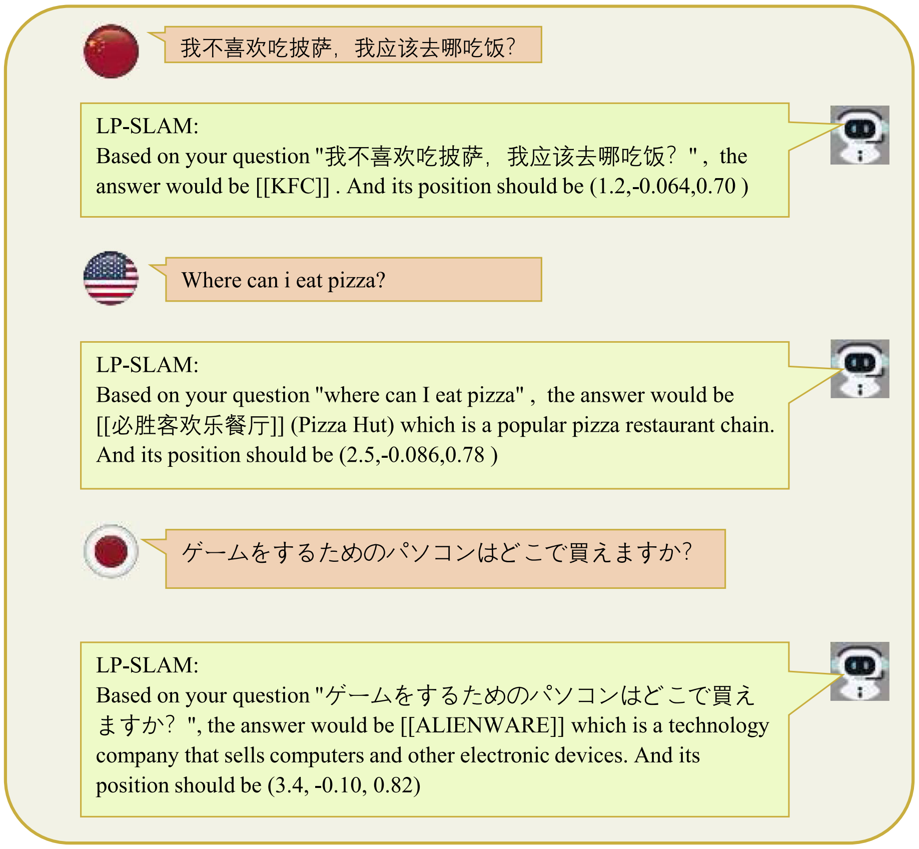


Fig. 11 Overall mapping result of the real-world scene

The unmanned aerial vehicle (UAV) is used to explore the scene recording the RGB-D frames. The overall scene is shown in Fig. 14, which contains 28 shops and random non-shop slogans. The UAV is used to explore the whole environment and the ground-truth trajectory is recorded. Three goals are to be reached by the LP-SLAM: 1. Constructing the map of the whole environment; 2. Extracting the task-relevant texts in the scene and mapping them onto the corresponding position; 3. Understanding the texts and providing instructions for human users.

Scene and text mapping

The overall mapping is shown in Fig. 15a, where the sparse point cloud map is constructed. The red points are currently visible, and the black ones are invisible. The blue rectangles are the keyframes created during the map construction, which form one trajectory of the UAV. The zoomed-in patches are shown in 15b and c, where the texts are detected and inserted into the map at the correct position. The task-relevant judgment result is also illustrated in 15d, where blue texts are the ones judged as shop names while green ones are not. The proposed system successfully judged whether the texts were shop landmarks or just slogans. During the whole mapping process, 24,017 texts containing 10,239 different spellings are extracted from the scene in total, most of which are nonsense mis-detection and wrongly-spelling mis-recognition cases. The similarity classification module classifies the texts into 6077 classes, which is a reasonable number to be processed further. The two-stage strategy further filtered out the outliers and only 54 classes were finally stored in the second-stage memory pool. Finally, the 23 texts are accurately judged as task-relevant and saved in the map. Five shops are not judged into the map, influenced mostly by the logos and tex-



我不喜欢吃披萨，我应该去哪吃饭？

LP-SLAM:
Based on your question "我不喜欢吃披萨，我应该去哪吃饭？"，the answer would be [[KFC]] . And its position should be (1.2,-0.064,0.70)

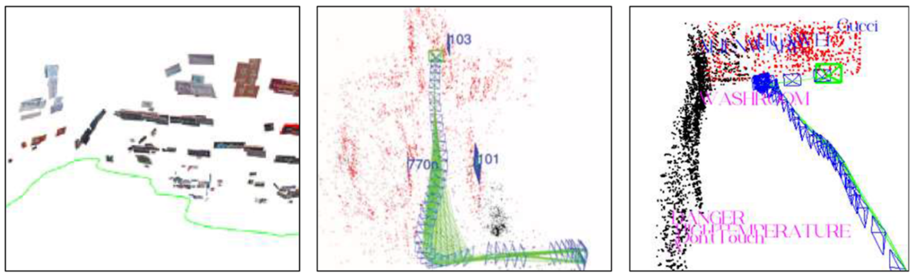
Where can i eat pizza?

LP-SLAM:
Based on your question "where can I eat pizza" , the answer would be [[必胜客欢乐餐厅]] (Pizza Hut) which is a popular pizza restaurant chain. And its position should be (2.5,-0.086,0.78)

ゲームをするためのパソコンはどこで買えますか?

LP-SLAM:
Based on your question "ゲームをするためのパソコンはどこで買えますか？ ", the answer would be [[ALIENWARE]] which is a technology company that sells computers and other electronic devices. And its position should be (3.4, -0.10, 0.82)

Fig. 12 Results of responding to natural language queries



(a) TextSLAM (b) TXSLAM (c) LP-SLAM

Fig. 13 Visualized comparison between different algorithms

tures near the text. In the meantime, though toilet is not a shop, it's important for navigation in a mall. Thus it's also reserved as "卫生间" (Chinese). The processing validates the functionality to deal with tremendous mis-detection and mis-recognition.

The trajectory estimation accuracy is shown in Fig. 16. The estimation error results from the light change, the lack of distinguished geometry features. The position accuracy of the texts is shown in Table 3. We recorded the estimated positions of the texts and the UAV position when each text was inserted into the map. All the positions of the extracted texts are calculated accurately within a reasonable error. The text position error and the trajectory error are calculated accordingly. The text position error is defined as the distance between the estimated text position and the true text position. For each text, the trajectory error is the distance between the estimated UAV position and the true UAV position. The trajectory error comes from the base SLAM system and has a direct impact on text position error. The text position error comes from two reasons: 1. the accumulated error of the basic geometry-feature-based SLAM system; 2. the error between the estimated center of the texts and the true center, which is caused by the STR algorithm; 3. the calculation error from 2D frame to 3D world coordinates.

Response to natural language queries

To validate the ability of the understanding of the constructed map, different queries in natural language are given and the system provides the positions to meet the requirements

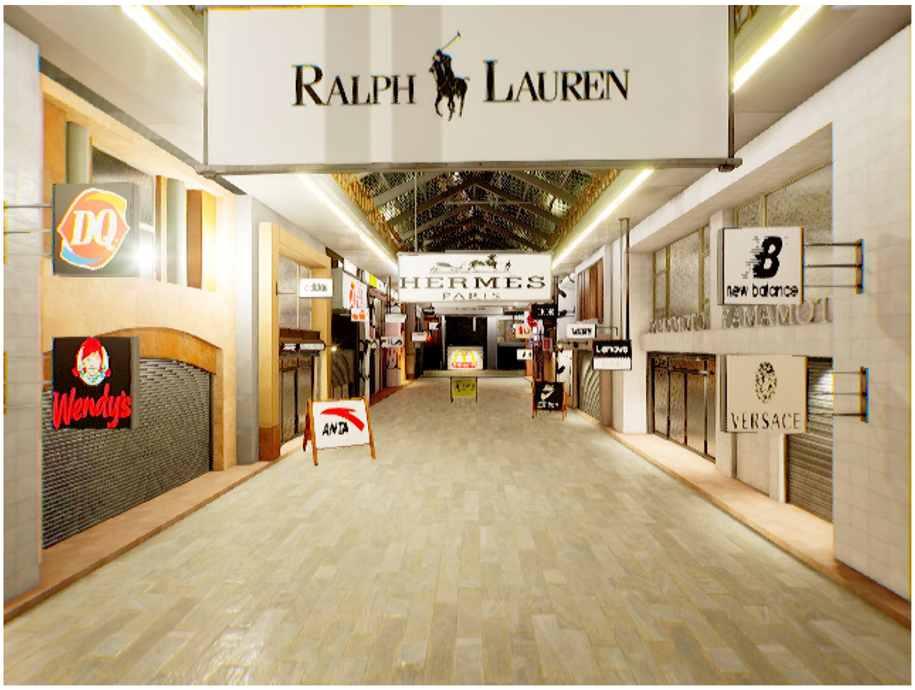


Fig. 14 The virtual scene

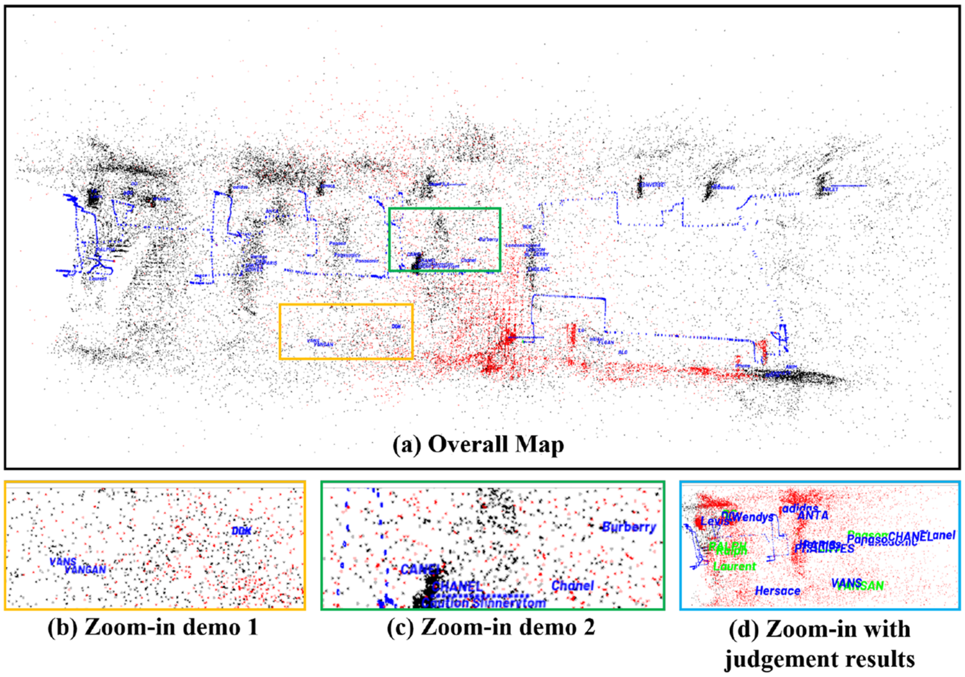


Fig. 15 The constructed map of the simulated scene

accordingly. The experiment results are shown in Table 4. We did three types of experiments in Chinese and English. The tasks include 1. Simple questions about the position of a specific shop (Q1–Q5); 2. Queries based on a specific demand(Q6–Q14), where one or two positions are involved; 3. Distance and semantic-based queries (Q15–Q16). The results show that the system can understand both human requirements and the constructed map, thus further providing reasonable suggestions.

Computational complexity analysis

The proposed framework leverages multiple neural-network-based modules. PP-OCRv2 [52] is implemented as the STR module in our experiment. The size of the implemented PaddleOCR model is 155.1M. The text clustering, task-relevant selection, and natural user interface are executed by ChatGPT 3.5. The size of ChatGPT 3.5 is approximately 175B. The other non-neural-network-based modules including sim-

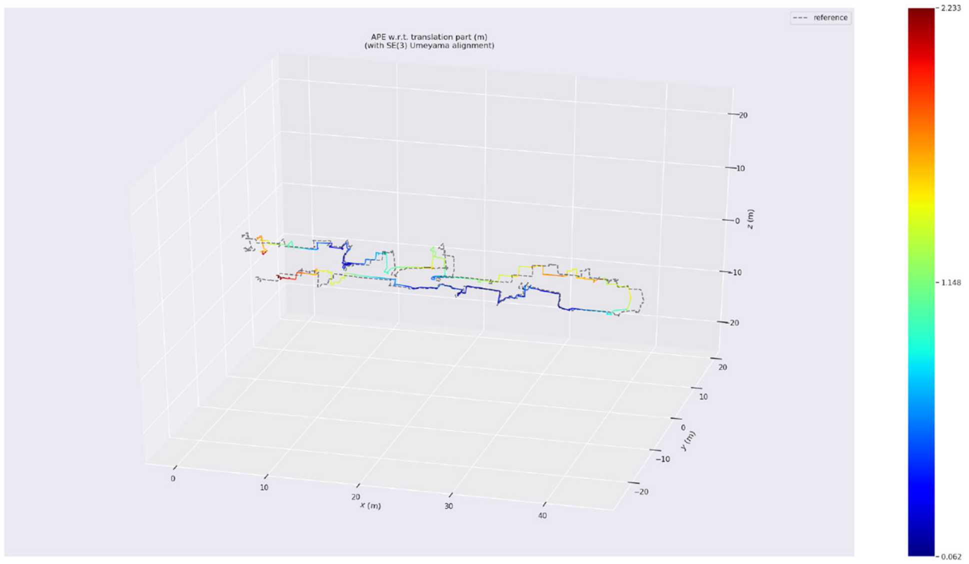


Fig. 16 The estimated trajectory error

ilarity classification, two-stage memory, position calculation, and position clustering have a very low complexity and account for a very small proportion of the total time.

The STR is implemented offline and the ChatGPT is executed online via APIs from OpenAI. Thus the execution speed of the STR module is restricted by the local hardware and the LLM-based modules are restricted by the response time of ChatGPT. The STR module and the three LLM-based modules are executed in two individual threads along with the tracking thread of SLAM. When a new frame arrives, the tracking thread and runtime text mapping thread process the image in parallel. The runtime mapping thread that contains the STR module costs more time than the tracking thread and thus becomes the bottleneck of the two threads. The distilling thread further processes the texts in a background thread that is decoupled with the tracking thread. Therefore the distilling thread will not influence the frame rate. The natural user interface is executed after the whole mapping process and also does not influence the frame rate. In our experiment, the platform is one desktop computer with Intel i7 CPU and Nvidia 3080ti GPU and the measured frame rate is 25fps, which meets the requirement of real-time.

Flexibility of the framework

The implemented STR network and LLM networks in the experiment are replaceable. There are two major considerations when selecting the models. Firstly, the model should be selected by the functionality. Different STR models are trained for various languages and various environments such as cities or villages. The further fine-tuned LLMs are more suitable for specific tasks in addition to the mall navigation tasks. The different LLMs are also more capable in different languages. For example, WenXinYiYan has better accuracy in a Chinese environment. Secondly, the model size should be considered according to the target platform. The trade-off between execution time and accuracy should be designed based on specific tasks. The base SLAM algorithm is also replaceable in consideration of accuracy and computation complexity.

Conclusion

In this work, LP-SLAM is proposed, one natural-language-level perception SLAM system leveraging LLMs, specifically ChatGPT. LP-SLAM can extract natural-language-level information from the world, judge the information, and store the important information. The human cognition-inspired techniques including similarity classification and long-short-term memory are designed to achieve better robustness against mis-detecting and mis-recognition. LP-SLAM is also capable of providing simple navigation guidance according to the query of users in different languages, showing great potential in real applications. While LP-SLAM illustrates how LLMs improve the perception of

Table 3 Position accuracy

	Detected texts	Whether is shop	Estimated text position	True text position	Estimated camera position	True camera position	Text error	Camera error
1	特步	Y	(1.34, −3.13, 3.75)	(3.75, 1.01, −1.83)	(0.42, −3.54, 3.76)	(3.71, 0.08, −2.23)	1.6	3.81
2	Wendys	Y	(5.16, −2.42, 3.79)	(3.43, 4.24, −1.4)	(0.87, −4.21, 3.45)	(3.4, 0.46, −2.84)	0.73	2.8
3	DQ	Y	(5, −4.36, 3.71)	(3.67, 4.22, −2.89)	(1.28, −3.71, 1.09)	(1.26, 1.04, −2.5)	2.82	2.02
4	Levis	Y	(2.56, −4.25, 3.52)	(3.45, 1.92, −2.83)	(−0.2, −2.96, 3.4)	(3.39, −0.49, −1.67)	2.73	4.35
5	ANTA	Y	(12.11, −1.07, 3.2)	(3.04, 10.81, 0.05)	(3.75, −3.23, 0.84)	(1.4, 3.09, −2.07)	2.23	2.7
6	Adidas	Y	(12.51, −4.72, 3.81)	(3.57, 10.35, −3.18)	(8.27, −2.76, 1.85)	(2.04, 6.78, −1.5)	2.45	3.58
7	FILA	Y	(24.52, −3.09, 4.35)	(3.68, 20.28, −1.91)	(17.06, −6.15, −0.77)	(−0.35, 14.57, −4.31)	0.95	2.63
8	CHANEL	Y	(23.61, −5.43, −0.16)	(−0.05, 22.2, −4.25)	(3.7, −2.72, 3.36)	(3.43, 3.19, −1.51)	3.48	4.37
9	Burberry	Y	(29.65, −5.68, −0.09)	(0.23, 26.8, −4.43)	(17.2, −3.06, −0.73)	(−0.35, 14.7, −1.8)	2.9	1.81
10	McDonald’s	Y	(39.25, −3.73, 3.52)	(3.65, 35.02, −2.25)	(37.24, −3.68, 2.46)	(2.77, 33.49, −2.24)	1.13	1.05
11	Animate	Y	(41.66, −3.67, −6.27)	(−3.63, 38.13, −3.39)	(40.32, −4.4, 2.46)	(3.09, 36.58, −2.92)	0.37	1.16
12	iPhone	Y	(39.87, −4.44, −5.36)	(−4.09, 37.25, −2.76)	(41.69, −3.64, −3.73)	(−2.63, 38.79, −2.23)	1.59	3.39
13	LG	Y	(34.43, −4.15, −4.74)	(−2.84, 30.46, −2.9)	(23.1, −5.11, 3.52)	(3.4, 20.1, −3.36)	0.65	0.7
14	Intel	Y	(33.19, −2.54, −4.31)	(−3.23, 30.56, −1.15)	(26.5, −4.95, −0.84)	(−0.17, 23.41, −3.48)	2.57	1.66
15	必胜客	Y	(26.43, −4.9, −6.05)	(−3.22, 25.75, −3.12)	(29.88, −3.81, −5.29)	(−3.18, 28.43, −2.3)	2.58	3.34
16	Dior	Y	(22.86, −4.77, −3.92)	(−2.99, 20.6, −3.29)	(18.33, −2.98, −0.75)	(−0.35, 15.72, −1.69)	1.62	1.83
17	Nike	Y	(18.54, −1.42, −6.47)	(−3.04, 18.31, −0.24)	(22.45, −2.56, −6.11)	(−2.91, 21.81, −1.29)	2.33	1.24
18	Vans	Y	(17.4, −3.47, −5.1)	(−3.31, 15.48, −2.13)	(10.99, −4.14, −0.33)	(0.45, 9.47, −2.85)	0.98	0.85
19	Lenovo	Y	(13.42, −2.96, −7.75)	(−3.61, 13.11, −1.76)	(20.56, −1.24, −6.61)	(−3.34, 19.98, −0.1)	0.52	0.5
20	Versace	Y	(5.79, −2.36, −8.43)	(−3.3, 6.47, −0.89)	(9.56, −3.14, −7.58)	(−2.86, 10.12, −1.72)	0.94	1.29
21	New Balance	Y	(5.69, −3.92, −8.43)	(−3.36, 6.52, −2.39)	(8.83, −3.57, −7.87)	(−3.03, 9.46, −2.13)	2.33	1.71
22	HERMES	Y	(14.22, −4.96, −0.24)	(0.04, 12.03, −3.54)	(9.03, −2.58, 1.85)	(2.02, 7.42, −1.33)	2.31	3.77
23	Panasonic	Y	(21.18, −6.41, −0.61)	(−0.08, 16.99, −4.22)	(9.33, −1.93, 1.07)	(1.09, 7.71, −0.79)	2.92	4.04
24	卫生间	N	(43.85, −5.26, 3.05)	(3.2, 39.23, −3.65)	(41.06, −4.05, 1.71)	(2.04, 36.92, −2.52)	1.97	0.5

Table 4 Queries and answers of the nature user interface in simulated scene

	Questions	Answers	
1	特步店铺在什么位置？ (Where is XTEP, a Chinese sports brand)	特步 (XTEP)	(1.34, −3.13, 3.75)
2	斐乐店铺在什么位置？ (Where is FILA, a Chinese sports brand)	FILA	(24.52, −3.09, 4.35)
3	香奈儿店铺在什么位置? (Where is the Channel)	CHANEL	(23.61, −5.43, −0.16)
4	Where is Pizza and More?	必胜客 (Pizza and More)	(26.43, −4.9, −6.05)
5	Where can I buy an iPhone?	iPhone	(39.87, −4.44, −5.36)
6	Where can I buy a necklace?	Dior	(22.86, −4.77, −3.92)
7	Where can I buy a ring?	CHANEL	(23.61, −5.43, −0.16)
8	我想吃披萨，请问可以去哪里? (I'd like to have pizza, where can I go?)	必胜客 (Pizza and More)	(26.43, −4.9, −6.05)
9	我想买一双运动鞋，请问可以去哪里？ (I'd like to buy sport shoes, where can I go?)	Adidas	(12.62, −4.66, 3.88)
10	I want to go to Dior and Channel, can you design a route for me?	Dior CHANNEL	(22.86, −4.77, −3.92) (23.61, −5.43, −0.16)
11	我的手机需要维修，请问我应该去哪里？ (I need to fix my phone, where can I go?)	Panasonic	(19.8, −5.83, −0.19)
12	我想先买一双运动鞋，然后去吃饭，但我不喜欢吃披萨，请帮我设计一条路线 (I'd like to buy sport shoes first, then I will go for a meal. But I do not like pizza. Please design a route for me.)	Nike McDonald's	(18.54, −1.42, −6.47) (39.25, −3.73, 3.52)
13	I want to go to the toilet first, then I would like to buy a bag for my wife, can you design a route for me?	卫生间 (Toilet) Burberry	(43.85, −5.26, 3.05) (29.65, −5.68, −0.09)
14	I am now at (0, 0, 0), please help me find the nearest sport shop	特步 (XTEP)	(1.34, −3.13, 3.75)
15	我现在在(0, 0, 0)，帮我找到离我最近的餐厅 (I am at (0, 0, 0), please help me find the nearest restaurant.)	必胜客 (Pizza and More)	(26.43, −4.9, −6.05)

SLAM to brand new natural language level, there are still further works to be researched. For instance, topics including how language information can help in the accuracy and efficiency of the SLAM threads are potential fields. In addition, there are further works to be explored: 1. The instruction texts such as signpost can provide more information, which can be used to enrich the ability of the SLAM system. 2. The extracted texts may help to improve the mapping accuracy.

Data availability The source code and the experiment dataset are available at: https://github.com/GroupOfLPSLAM/LP_SLAM. Other experimental data are available on request from the corresponding author.

Declarations

Conflict of interest All authors have no conflicts of interest to declare that are relevant to the content of this article.

Ethical approval All authors declare that this work does not include humans and animals, as well as did not collect data from human subjects.

References

1. Chen K, Lopez BT, Agha-mohammadi A-A, Mehta A (2022) Direct lidar odometry: Fast localization with dense point clouds. IEEE Robot Autom Lett 7(2):2000–2007
2. Mur-Artal R, Tardós JD (2017) Orb-slam2: an open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Trans Robot 33(5):1255–1262
3. Labbe M, Michaud F (2013) Appearance-based loop closure detection for online large-scale and long-term operation. IEEE Trans Robot 29(3):734–745
4. Endres F, Hess J, Sturm J, Cremers D, Burgard W (2013) 3-d mapping with an rgb-d camera. IEEE Trans Robot 30(1):177–187
5. Harris C, Stephens M et al (1988) A combined corner and edge detector. In: Alvey Vision Conference, vol 15, pp 10–5244
6. Shi J et al (1994) Good features to track. In: 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp 593–600
7. Rublee E, Rabaud V, Konolige K, Bradski G (2011) Orb: an efficient alternative to sift or surf. In: 2011 International Conference on Computer Vision, pp 2564–2571
8. Hirose K, Saito H (2012) Fast line description for line-based slam. In: BMVC, pp 1–11
9. Zuo X, Xie X, Liu Y, Huang G (2017) Robust visual slam with point and line features. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 1775–1782
10. Yang S, Song Y, Kaess M, Scherer S (2016) Pop-up slam: semantic monocular plane slam for low-texture environments. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 1222–1229
11. Yang S, Scherer S (2019) Monocular object and plane slam in structured environments. IEEE Robot Autom Lett 4(4):3145–3152
12. Liu B, Zhao W, Sun Q (2017) Study of object detection based on faster r-cnn. In: 2017 Chinese Automation Congress (CAC), pp 6233–6236
13. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp 2961–2969
14. Sünderhauf N, Pham TT, Latif Y, Milford M, Reid I (2017) Meaningful maps with object-oriented semantic mapping. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5079–5085
15. Liu M, Zhu M, White M, Li Y, Kalenichenko D (2019) Looking fast and slow: memory-guided mobile video object detection. arXiv preprint arXiv:1903.10172
16. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2018) Mobilenetv2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4510–4520
17. Wu W, Guo L, Gao H, You Z, Liu Y, Chen Z (2022) Yolo-slam: A semantic slam system towards dynamic environment with geometric constraint. Neural Comput Appl 1–16
18. Yang S, Scherer S (2019) Cubeslam: monocular 3-d object slam. IEEE Trans Robot 35(4):925–938
19. Yu C, Liu Z, Liu X-J, Xie F, Yang Y, Wei Q, Fei Q (2018) Ds-slam: a semantic visual slam towards dynamic environments. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 1168–1174
20. Bao SY, Bagra M, Chao Y-W, Savarese S (2012) Semantic structure from motion with points, regions, and objects. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp 2703–2710
21. Qian Z, Fu J, Xiao J (2022) Towards accurate loop closure detection in semantic slam with 3d semantic covisibility graphs. IEEE Robot Autom Lett 7(2):2455–2462
22. Rong X, Li B, Munoz JP, Xiao J, Arditi A, Tian Y (2016) Guided text spotting for assistive blind navigation in unfamiliar indoor environments. In: Advances in Visual Computing: 12th International Symposium, ISVC 2016, Las Vegas, NV, USA, December 12-14, 2016, Proceedings, Part II 12, Springer, pp 11–22
23. Wang H-C, Landa Y, Fallon M, Teller S (2014) Spatially prioritized and persistent text detection and decoding. In: camera-based document analysis and recognition: 5th International Workshop, CBDAR 2013, Washington, DC, USA, August 23, 2013, Revised Selected Papers 5, Springer, pp 3–17
24. Wang H-C, Finn C, Paull L, Kaess M, Rosenholtz R, Teller S, Leonard J (2015) Bridging text spotting and slam with junction features. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 3701–3708
25. Li B, Zou D, Sartori D, Pei L, Yu W (2020) Textslam: visual slam with planar text features. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp 2102–2108
26. Tong Q, Ma L, Chen K, Liu J, Qian J, Zhang J (2022) Txslam: a monocular semantic slam tightly coupled with planar text features. In: 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp 728–733
27. Scao TL, Fan A, Akiki C, Pavlick E, Ilić S, Hesslow D, Castagné R. Luccioni AS, Yvon F, Gallé M et al (2022) Bloom: a 176b-parameter open-access multilingual language model. arXiv preprint arXiv:2211.05100
28. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al (2020) Language models are few-shot learners. Adv Neural Inf Process Syst 33:1877–1901
29. Vemprala S, Bonatti R, Bucker A, Kapoor A (2023) Chatgpt for robotics: design principles and model abilities. Microsoft Auton Syst Robot Res 2:20
30. Davison AJ, Reid ID, Molton ND, Stasse O (2007) Monoslam: Real-time single camera slam. IEEE Trans Pattern Anal Mach Intell 29(6):1052–1067
31. Klein G, Murray D (2007) Parallel tracking and mapping for small ar workspaces. In: 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp 225–234
32. Mur-Artal R, Montiel JMM, Tardos JD (2015) Orb-slam: a versatile and accurate monocular slam system. IEEE Trans Robot 31(5):1147–1163
33. Campos C, Elvira R, Rodríguez JJG, Montiel JM, Tardós JD (2021) Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam. IEEE Trans Robot 37(6):1874–1890
34. Engel J, Schöps T, Cremers D (2014) Lsd-slam: Large-scale direct monocular slam. In: European Conference on Computer Vision, Springer, pp 834–849
35. Zubizarreta J, Aguinaga I, Montiel JMM (2020) Direct sparse mapping. IEEE Trans Robot 36(4):1363–1370
36. DeTone D, Malisiewicz T, Rabinovich A (2018) Superpoint: Self-supervised interest point detection and description. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp 224–236
37. Sarlin P-E, DeTone D, Malisiewicz T, Rabinovich A (2020) Superglue: learning feature matching with graph neural networks. In:

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 4938–4947
38. Shi B, Bai X, Yao C (2016) An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Trans Pattern Anal Mach Intell 39(11):2298–2304
39. Shi B, Yang M, Wang X, Lyu P, Yao C, Bai X (2018) Aster: an attentional scene text recognizer with flexible rectification. IEEE Trans Pattern Anal Mach Intell 41(9):2035–2048
40. Yu D, Li X, Zhang C, Liu T, Han J, Liu J, Ding E (2020) Towards accurate scene text recognition with semantic reasoning networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 12113–12122
41. Bahl LR, Brown PF, Souza PV, Mercer RL (1989) A tree-based statistical language model for natural language speech recognition. IEEE Trans Acoust Speech Signal Process 37(7):1001–1008
42. Brown PF, Della Pietra VJ, Desouza PV, Lai JC, Mercer RL (1992) Class-based n-gram models of natural language. Comput Linguist 18(4):467–480
43. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst 30
44. Tellex S, Gopalan N, Kress-Gazit H, Matuszek C (2020) Robots that use language. Ann Rev Control Robot Auton Syst 3:25–55
45. Pan J, Chou G, Berenson D (2023) Data-efficient learning of natural language to linear temporal logic translators for robot task specification. arXiv preprint arXiv:2303.08006
46. Mees O, Borja-Diaz J, Burgard W (2023) Grounding language with visual affordances over unstructured data. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp 11576–11582
47. Chen B, Xia F, Ichter B, Rao K, Gopalakrishnan K, Ryoo MS, Stone A, Kappler D (2023) Open-vocabulary queryable scene representations for real world planning. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp 11509–11522
48. Singh I, Blukis V, Mousavian A, Goyal A, Xu D, Tremblay J, Fox D, Thomason J, Garg A (2023) Progprompt: generating situated robot task plans using large language models. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp 11523–11530
49. Wang W, Li X, Dong Y, Xie J, Guo D, Liu H (2023) Natural language instruction understanding for robotic manipulation: a multisensory perception approach. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp 9800–9806
50. Mo Y, Zhang H, Kong T (2023) Towards open-world interactive disambiguation for robotic grasping. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp 8061–8067
51. Levenshtein VI et al (1966) Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady, vol 10, Soviet Union, pp 707–710
52. Du Y, Li C, Guo R, Cui C, Liu W, Zhou J, Lu B, Yang Y, Liu Q, Hu X et al (2021) Pp-ocrv2: Bag of tricks for ultra lightweight ocr system. arXiv preprint arXiv:2109.03144