TextSLAM: Visual SLAM With Semantic Planar Text Features

Boying Li, Danping Zou, *Member, IEEE*, Yuan Huang, Xinghan Niu, Ling Pei, and Wenxian Yu

***Abstract*—We propose a novel visual SLAM method that integrates text objects tightly by treating them as semantic features via fully exploring their geometric and semantic prior. The text object is modeled as a texture-rich planar patch whose semantic meaning is extracted and updated on the fly for better data association. With the full exploration of locally planar characteristics and semantic meaning of text objects, the SLAM system becomes more accurate and robust even under challenging conditions such as image blurring, large viewpoint changes, and significant illumination variations (day and night). We tested our method in various scenes with the ground truth data. The results show that integrating texture features leads to a more superior SLAM system that can match images across day and night. The reconstructed semantic 3D text map could be useful for navigation and scene understanding in robotic and mixed reality applications.**

***Index Terms*—Visual SLAM, texts, semantic SLAM.**

I. Introduction

VISUAL SLAM is an important technique of ego-motion estimation and scene perception, which has been widely used in navigation for drones [1], ground vehicles, self-driving cars [2], or other applications such as Augmented and Virtual Reality (AR and VR) [3]. The typical visual SLAM algorithm extracts point features [4], [5] from images for pose estimation and mapping. Recent methods [6], [7] even directly operate on pixels. However, it is well known that incorporating high-level features like lines [8], surfaces [9] or even semantic objects [10], [11] in the visual SLAM system will lead to better performance.

One common type of high-level feature is text objects. Scene texts play a key role in identifying locations with various forms such as road marks [12], [13], building or object signs [14], [15], room names [15], [16], [17], and other textual captions [16], [17], [18]. They help us to recognize landmarks, navigate in complex environments, and guide us to the destination. Detection and recognition of scene texts from images have been developing fast [19], [20], [21], [22], [23], [24], [25], [26] because of the boom of deep neural networks and the emergence of huge text datasets such as COCO-Text [27], DOST [28], and ICDAR [29]. As extracting scene texts from images becomes easy nowadays, one question raises whether texts can be integrated into a visual SLAM system to both yield better performance and generate high-quality 3D semantic text maps that could be useful for robot navigation and scene understanding, as well as augmented reality and human-computer interaction.

Texts spotted in our daily life are mostly planar regions, at least for a single word or character if not the whole sentence. The rich texture and planar property of a text entity make the text object a good feature for tracking and localization. More importantly, the semantic messages that a text object delivers are invariant to appearance changes, hence text objects are also reliable features for matching even when the illumination or viewpoint changes significantly. Those characteristics of scene texts are certainly good for SLAM, while the key issue is how to integrate them into a visual SLAM system.

There are several attempts towards coupling SLAM with text features. A navigation system with human-computer interaction [16], [17] for blind people, assisted with text entities, is built upon the visual-inertial SLAM system shipped on Google Tango tablet. Similarly, Wang et al. proposed a method to extract text features [18], which are then used for fusion with Tango's SLAM outputs to recognize revisited places. The aforementioned works have shown the great potential of using text features with existing SLAM systems. However, they treat the SLAM system as a black box, which is unable to fully take advantage of the characteristics of scene texts that should be beneficial to SLAM.

In this paper, we present a novel SLAM system tightly coupled with semantic text features as shown in Fig. 1. Specifically, we integrate the text features into the SLAM pipeline by fully exploiting the favorable characteristics of scene texts. Geometrically, text features are treated as texture-rich planar patches used for camera pose estimation and back-end optimization to yield more accurate and robust estimation. Semantically, the meaning of those scene texts, invariant to appearance changes, are utilized for reliable place recognition and feature matching across scenes with large illumination or viewpoint changes. For the lack of SLAM benchmarks with rich texts, we collected a text-orientated dataset both indoor and outdoor with the ground truth carefully acquired for evaluation. We compare our SLAM system with state-of-the-art approaches. The results show that by tightly coupling with text objects, the SLAM system becomes

Manuscript received 16 March 2022; revised 22 August 2023; accepted 3 October 2023. Date of publication 13 October 2023; date of current version 5 December 2023. This work was supported in part by National Key R&D Program of China under Grant 2022YFB3903801, in part by the National of Science Foundation of China under Grant 62073214, and Midea Group (3D Robot Vision Project). Recommended for acceptance by P. Tan. *(Corresponding author: Danping Zou.)*

The authors are with Shanghai Key Laboratory of Navigation and Location-Based Services, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: liboying.fhyt@sjtu.edu.cn; dpzou@sjtu.edu.cn; huangyuan@sjtu.edu.cn; sjtuxhniu@sjtu.edu.cn; ling.pei@sjtu.edu.cn; wxyu@sjtu.edu.cn).

Project page: https://github.com/SJTU-ViSYS/TextSLAM

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2023.3324320, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2023.3324320

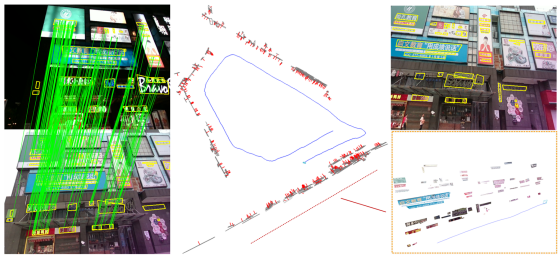


Fig. 1. TextSLAM can produce 3D text maps and match text objects correctly despite significant illumination changes. Left column: semantically matched text objects and text-guided point correspondences (green lines) between a night query image and a day image in the day-and-night test. The detected texts are shown in yellow rectangles. Middle column: 3D text maps and camera trajectory in the bird-eye view. The text objects are illustrated in gray boxes and their normal directions are shown in red. Right column: tracked texts in the image (in yellow rectangles) and the zoomed-in view of the 3D text map.

more accurate and robust, and even works well under challenging conditions such as serious illumination changes (day and night), viewpoint changes, and occlusions, where existing SLAM systems usually fail. We also compared our text-based method with the state-of-the-art visual localization methods for loop closing. The results show that our text-based method outperforms those methods in text-rich environments with a much lower computational cost.

The technical contributions of our work include:

1) A novel visual SLAM framework that integrates text features in front-end pose tracking, back-end optimization, loop closing, and mapping. To our best knowledge, this is the first work that tightly integrates scene texts into the pipeline of visual SLAM. We also contribute a dataset in various text-rich environments for evaluation.
2) We present both geometric and semantic representations of text features, as well as their observation and data association models within the SLAM pipeline.
3) A novel loop closing technique relying on the semantic meaning of text features. With the help of semantic information, reliable loop closing can be achieved even in challenging scenarios, including serious illumination changes, occlusion, and drastically varying viewpoints.

This paper is extended from our previous work [14]. The major extension is incorporating the *semantic information* into the SLAM pipeline, especially for the semantic data association and loop closure, as well as additional experiments and analysis. Specifically, the extensions include a novel semantic representation of text objects together with its update scheme (Section III-B), using the semantic information of text objects for loop closure (Section IV-E), and several improvements of the SLAM system (Section IV) such as text object culling, text selection in pose estimation, and feature sampling in the coarse-to-fine optimization. In addition, we present a challenging text-orientated dataset with ground truth. Additional tests in indoor, outdoor and day-night switching are also presented in Section V.

II. Related Work

A. Planar Features

Most scene texts can be treated as texture-rich planar features. Planar features have been studied in the visual SLAM community since the early stage. In early works [3], [34], [35], the planes in the scene were detected by RANSAC [36] among estimated 3D points and employed as novel features to replace points in the state. Since much fewer parameters are required to represent the map using planes, it reduces the computational cost significantly [37]. These works show that planar features improve both accuracy and robustness of a visual SLAM system. Existing methods require 3D information to discover the planar features, usually using a RGB-D camera [38], [39], [40]. However, this becomes difficult using only image sensors. An interesting idea [41] is to assume the region surrounding the detected feature point as a locally planar surface. The assumption seldom holds in realistic scenes, as feature points might be extracted from anywhere in the scene. By contrast, texts in realistic scenes are mostly located on planar surfaces. Unlike general planar features that usually require depth for detection [38], [39], [40], scene texts can be easily extracted by off-the-shelf text detectors [19], [20].

B. Visual SLAM With Semantics

Integrating semantic information into visual SLAM systems has been receiving increasing interest in recent years [37], [42], [43], [44], [45]. One approach is to directly fuse the 2D semantic labels with the dense 3D map from RGB-D SLAM [11], [46], [47], or a dense visual SLAM [48]. Another approach is to take semantic objects as high-level features within the SLAM pipeline [49], [50], which requires pre-scanned 3D models to precisely fit the observation on the image. Though recent methods [51], [52], [53], [54] build the 3D representation of objects online with a depth camera, it is still difficult to be generalized to unseen objects with only video cameras. Other methods seek to use 3D bounding boxes [55], [56], [57] or quadrics [58] to represent objects, but such kind of approximation suffers from loss of accuracy. In this paper, we focus on the particular semantic object, i.e., scene texts. Unlike generic objects, text objects, such as road signs, shop signs, room numbers, and commercial boards, are texture-rich planar features and contain rich semantic information about environments. Those characteristics are more favorable to visual SLAM than those of general objects.

C. Text-Aided Navigation

Scene Texts such as room numbers, road marks, route or traffic signs, and shop signs are naturally good visual landmarks to assist navigation. We summarize existing works on text-aided navigation in Table I. In the early works [30], [31], indoor text labels such as room numbers or name tags were used as guidance for a robot to navigate in the lab environments. However, text-aided navigation was still in its infancy at that time, as the technique of detection and recognition of scene texts was still under early development [59], [60]. Ranganathan et al. [12] integrated standard road marks into the pre-built GPS+IMU+camera map to estimate the vehicle's ego-motion for autonomous driving. With the prior knowledge of a comprehensive geo-tagged street-level map (e.g., GoogleMaps or OpenStreetMap) and the compass information, Radwan et al. [13] extracted text information from the street signs to assist pose estimation in a 2D map. Similarly, Wang et al. [33] used the

TABLE I
LIST OF EXISTING TEXT-AIDED NAVIGATION APPROACHES

Method	Text objects	Text extraction	Map	Task	Scene
Tomono et al. 2000 [30]	room nameplates	heuristic	office map with a corridor and doors	Robot navigation	indoor
Mata et al. 2001 [31]	room nameplates	heuristic	office map with landmark annotation	Robot navigation	indoor
Case et al. 2011 [32]	room nameplates	heuristic	laser grid-based map with text annotation	Robot navigation	indoor
Ranganathan et al. 2013 [12]	road marks	heuristic	road surface marks map	localization	outdoor
Wang et al. 2015 [18]	artificial tags	heuristic	landmarker map	loop closing	indoor
Wang et al. 2015 [33]	store signs	heuristic	floorplan with text annotation	localization	indoor
Radwan et al. 2016 [13]	store signs	heuristic	geo-tagged street-level map with text annotation	localization	outdoor
Hong et al. 2019 [15]	street&store signs	deep learning	2D imagery map	localization	indoor&outdoor
Li et al. 2019 [17]	room nameplates	heuristic	CAD model map with semantic layers	human navigation	indoor
TextSLAM (Ours)	all scene texts	deep learning	3D text map	tightly coupled SLAM	indoor&outdoor

shop names for localization by taking the building's floor plan as a prior under the assumption of Manhattan world. Following the idea of the aforementioned works, Hong et al. [15] applied neural networks to extract street and store names together with billboards in the wild for place recognition. Those works have shown the advantage of leveraging text objects in dealing with illumination and viewpoint changes for localization. It is natural to consider integrating scene texts to a SLAM system to gain better performance as well as to offer a new way for human-computer interaction.

Wang et al. [18] proposed a spatial-level feature named 'junction' for the text extraction, and then used the text objects to improve loop closing performance based on Google Tango's SLAM outputs. Case et al. [32] annotated the text labels on the existing map generated by a laser SLAM system to help robots recognize named locations and understand humans' free-text queries. Rong et al. [16] presented an assistive blind navigation system with a text spotting function based on the Tango's SLAM system. Similarly, built on the SLAM system of Tango, a mobile solution [17] of assistive navigation system combined various sources, such as text recognition results and speech-audio interaction, for blind and visually impaired people to travel indoors independently. Existing text-aided navigation systems integrate text objects loosely by regarding the SLAM system as a black box. By contrast, our proposed method integrates the text objects tightly into the SLAM system to facilitate both camera tracking and semantic mapping. Moreover, the semantic information from the established map is used for loop closing and camera localization to achieve good performance even under challenging conditions.

III. SEMANTIC TEXT FEATURES

A semantic text feature is represented as a planar and bounded patch with its unique semantic meaning. We describe the geometric model (including the parameterization and observation models) of a text object, as well as the way to represent and update the semantic information in the following sections.

A. Geometric Model

1) Parameterization: Text objects are regarded as planar and bounded patches. Each text patch (enclosed by a bounding box) is anchored to a camera frame, named as the *host frame*, which is the first frame where the text object appears as shown in Fig. 2. Within the host frame, the plane where the text patch lies is given by $\boldsymbol{n}^{\mathrm{T}}\boldsymbol{p} + d = 0$, where $\boldsymbol{n} = (n_1, n_2, n_3)^{\mathrm{T}} \in \mathbb{R}^3$

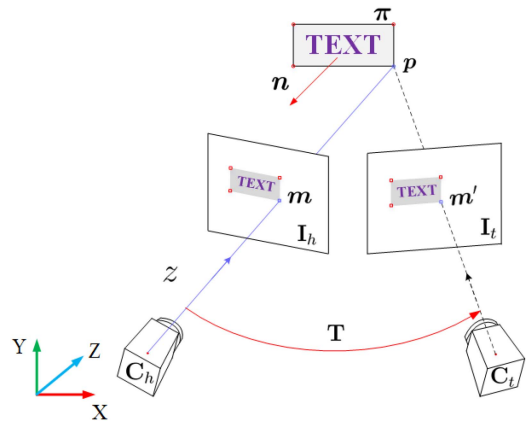


Fig. 2. Text object is compactly parameterized by $\boldsymbol{\theta}$. The inverse depth ρ of a text point $\boldsymbol{p}$ can be computed by $\rho = 1/z = \boldsymbol{\theta}^{\mathrm{T}}\tilde{\boldsymbol{m}}$ and its projection onto the target view C_t is a homography transform with respect to the relative pose $\mathbf{T}$ between the two views.

is the normal of the plane and $d \in \mathbb{R}$ is related to the distance from the plane to the origin of the host frame; $\boldsymbol{p} \in \mathbb{R}^3$ represents the 3D point on the plane.

A straightforward parameterization of a text plane could be directly using the four parameters (n_1, n_2, n_3, d) of the plane equation. However, this is an over-parameterization that leads to rank deficiency in the nonlinear least-squares optimization. We hence adopt a compact parameterization that contains only three parameters.

$$\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)^{\mathrm{T}} = -\boldsymbol{n}/d. \tag{1}$$

We'll show that this parameterization is closely related to the inverse depth of the 3D point on the text plane.

Within the host frame, each 3D point $\boldsymbol{p} \in \mathbb{R}^3$ observed on the image can be represented by its normalized image coordinates $\boldsymbol{m} = (u, v)^{\mathrm{T}}$ and its inverse depth $\rho = 1/z$. The 3D coordinates of this point are computed as $\boldsymbol{p} = (uz, vz, z)^{\mathrm{T}} = z\tilde{\boldsymbol{m}}$, where $\tilde{\boldsymbol{m}}$ represents the homogeneous coordinates of $\boldsymbol{m}$. If the 3D point locates on the text plane, we have $z \cdot \boldsymbol{n}^{\mathrm{T}}\tilde{\boldsymbol{m}} + d = 0$. The inverse depth ρ of this 3D point is then computed as

$$\rho = 1/z = -\boldsymbol{n}^{\mathrm{T}}/d\ \tilde{\boldsymbol{m}} = \boldsymbol{\theta}^{\mathrm{T}}\tilde{\boldsymbol{m}}. \tag{2}$$

That is, we can use a simple dot product to quickly infer the inverse depth of a text point from its 2D coordinates, given the text parameters $\boldsymbol{\theta}$.

On the other hand, if we have at least three points on the text patch (for example, three corners of the bounding box), with

their inverse depth value, we can immediately obtain the text parameters by solving

$$\begin{bmatrix} \tilde{\boldsymbol{m}}_1^{\mathrm{T}} \\ \vdots \\ \tilde{\boldsymbol{m}}_n^{\mathrm{T}} \end{bmatrix} \boldsymbol{\theta} = \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_n \end{bmatrix}, n \geq 3. \tag{3}$$

This allows us to quickly initialize the text parameters from the depth value of three corners of the text bounding box.

Properties such as the boundary of a text object are kept in our system. Those properties are acquired from a text detector as we'll describe later.

2) Observation: To update the parameters of a text object, an observation model should be defined. Here we choose an observation model that measures the difference between the detected text object and the projection of the estimated 3D text object on the image. In the first step, we need to project the 3D text object from the host frame onto the target image plane.

Let $\mathbf{T}_h, \mathbf{T}_t \in SE(3)$ represent the transformations of the host frame and the target frame with respect to the world frame. The relative pose between the host frame and the target frame is computed as $\mathbf{T} = \mathbf{T}_t^{-1}\mathbf{T}_h$. We let $\mathbf{R}, \boldsymbol{t}$ be the rotation and translation of $\mathbf{T}$. Given the text parameters $\boldsymbol{\theta}$ and the observed text point (with homogenous coordinates $\tilde{\boldsymbol{m}}$) in the host image, the 3D coordinates of point $\boldsymbol{p}$ in the host frame are:

$$\boldsymbol{p} = \tilde{\boldsymbol{m}}/\rho = \tilde{\boldsymbol{m}}/(\boldsymbol{\theta}^{\mathrm{T}}\tilde{\boldsymbol{m}}). \tag{4}$$

The point is then transformed into the target frame by

$$\boldsymbol{p}' = \mathbf{R}\boldsymbol{p} + \boldsymbol{t}. \tag{5}$$

Let $\tilde{\boldsymbol{m}}'$ be the homogeneous coordinates of the projected point on the target image plane. We have

$$\begin{aligned} \tilde{\boldsymbol{m}}' &\sim \rho\,\boldsymbol{p}' = \mathbf{R}\tilde{\boldsymbol{m}} + \rho\boldsymbol{t} \\ \Rightarrow \tilde{\boldsymbol{m}}' &\sim \mathbf{R}\tilde{\boldsymbol{m}} + \boldsymbol{t}\,\boldsymbol{\theta}^{\mathrm{T}}\tilde{\boldsymbol{m}}, \end{aligned} \tag{6}$$

where $\sim$ means equivalence up to a scale. Therefore, the process of projecting a 3D text object on the target image plane is a homography mapping of the text points from the host image plane to the target image plane, namely

$$\tilde{\boldsymbol{m}}' \sim \mathbf{H}\tilde{\boldsymbol{m}}, \tag{7}$$

where $\mathbf{H} = \mathbf{R} + \boldsymbol{t}\boldsymbol{\theta}^{\mathrm{T}} \in \mathbb{R}^{3\times 3}$ is a homography matrix that relies on the relative pose $\mathbf{R}, \boldsymbol{t}$ and the text parameters $\boldsymbol{\theta}$. For convenience, we write the projection process as a function:

$$\boldsymbol{m}' = \boldsymbol{h}(\boldsymbol{m}, \mathbf{T}_h, \mathbf{T}_t, \boldsymbol{\theta}), \tag{8}$$

where $\boldsymbol{m}$ denotes the observed text point on the host image plane, and $\boldsymbol{m}'$ represents the projected text point on the target image plane. $\boldsymbol{\theta}$ represents the text parameters.

We take each text region as a single patch and align it to other frames directly by minimizing the difference between them instead of detecting the word once again. Motivated by directed approaches [6], [7], our observation model computes the photometric error between the extracted text object and the projected one on the image. As we shall see in the experiments (see Fig. 9), using direct approaches will lead to better accuracy and robustness, particularly for blurry images. The biggest issue of the direct approach is handling the illumination changes. Existing work [7] adopts an affine model to address intensity changes, but it requires extra parameters involved in optimization and sophisticated photometric calibration to guarantee performance. We choose to use zero-mean normalized cross-correlation (ZNCC) as the matching cost to handle illumination changes.

Let Ω be the set of pixels within the text region, and $\boldsymbol{m} \in \Omega$ be a text pixel. The normalized intensities for text pixels are:

$$\tilde{I}(\boldsymbol{m}) = (I(\boldsymbol{m}) - \bar{I}_\Omega)/\sigma_\Omega, \tag{9}$$

where $\bar{I}_\Omega$ and σ_Ω stand for the average intensity and the standard deviation of the pixels in the text region Ω. The text patch in the host image and the predicted one in the target image (8) are then compared by :

$$ZNCC(I_h, I_t) = \sum_{\boldsymbol{m}\in\Omega} \tilde{I}_h(\boldsymbol{m})\tilde{I}_t(\boldsymbol{m}'). \tag{10}$$

The ZNCC cost is between -1 and 1. The larger ZNCC cost indicates the two patches are more similar. However, it is difficult to directly use the ZNCC cost within the optimization framework of the nonlinear least-squares problem. We hence adopt a variant form of ZNCC as the cost function

$$E(I_h, I_t) = \sum_{\boldsymbol{m}\in\Omega} (\tilde{I}_h(\boldsymbol{m}) - \tilde{I}_t(\boldsymbol{m}'))^2. \tag{11}$$

Though the cost function is similar to the SSD (Sum of Squared Difference) cost, it contains an additional normalization process to ensure the robustness to illumination changes. If we expand this cost function as :

$$\sum_{\boldsymbol{m}\in\Omega} (\tilde{I}_h(\boldsymbol{m})^2 + \tilde{I}_t(\boldsymbol{m}')^2) - 2\sum_{\boldsymbol{m}\in\Omega} \tilde{I}_h(\boldsymbol{m})\tilde{I}_t(\boldsymbol{m}'), \tag{12}$$

we can discover that minimizing this cost function is equivalent to maximizing the ZNCC cost, because $\sum \tilde{I}_h(\boldsymbol{m})^2 = N$ and $\sum \tilde{I}_t(\boldsymbol{m}')^2 = N$, where N is a constant number of pixels within the text region Ω. The photometric error of a text object π with respect to the target frame t is defined as:

$$E_{photo}^{\pi,t} = \sum_{\boldsymbol{m}\in\Omega^\pi} \phi((\tilde{I}_h(\boldsymbol{m}) - \tilde{I}_t(\boldsymbol{h}(\boldsymbol{m}, \mathbf{T}_h, \mathbf{T}_t, \boldsymbol{\theta}^\pi)))^2), \tag{13}$$

where $\phi(\cdot)$ is the Huber loss function to handle possible outliers. Here, we use Ω^π to represent the text region on the host image plane. As we'll describe later, to make the computation faster, we do not use all the pixels within the text region, instead select only some of them as the reference pixels to compute the photometric error.

B. Semantic Information Management

The semantic meanings of scene texts are valuable information for scene understanding and also benefit data association in SLAM because they are invariant to appearance changes. We represent the semantic information $\mathcal{X}$ of a text object by two parts: its meaning $\boldsymbol{s}$ and a semantic cost g^{sem}

$$\mathcal{X} = \{\boldsymbol{s}, g^{sem}\}, \tag{14}$$

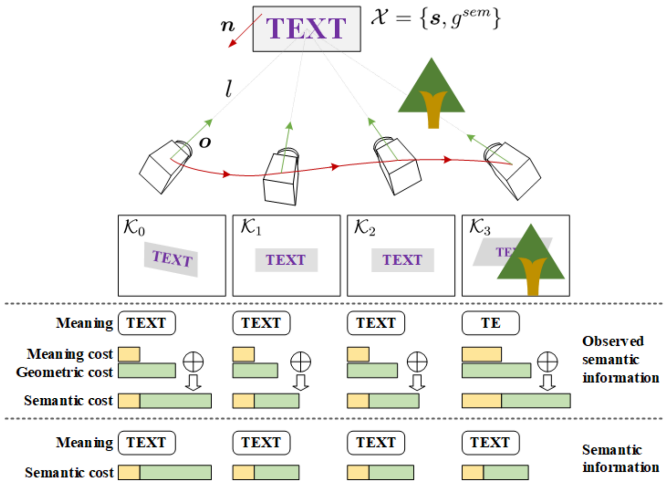


Fig. 3. Semantic information of a text object is updated whenever a new observation comes. The top row shows the text object and the camera trajectory as well as four keyframes. Note that in the fourth keyframe $\mathcal{K}_3$, the text object is occluded by the tree and is therefore partially observed. The second row shows the observed semantic information extracted from each frame, which consists of the meaning of the text object (represented by a string) and the semantic costs (including the meaning and geometric parts, the smaller the better). The third row demonstrates the semantic information of the text object is updated from the observed one at each frame. The semantic information with the smallest semantic cost is kept when updating.

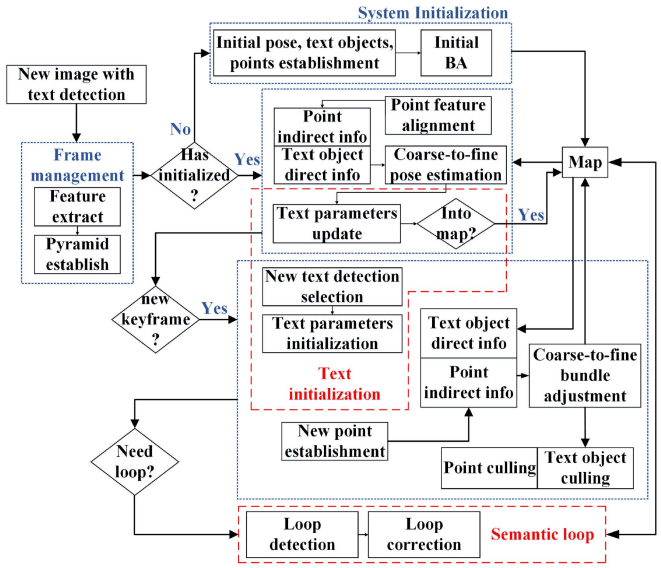


Fig. 4. Overview of our TextSLAM system.

where the text meaning $\boldsymbol{s}$ is a text string and the semantic cost g^{sem} describes the quality of the estimated text meaning. Lower semantic costs indicate better qualities. As illustrated in Fig. 3, the semantic information of a text object is initialized from the first observation and continuously updated when new observations arrive.

For the text object recognized at each frame, we extract its current semantic information, $\hat{\mathcal{X}} = \{\hat{\boldsymbol{s}}, \hat{g}^{sem}\}$. Here $\hat{\boldsymbol{s}}$ is the text strings from the recognition results of the text extractor. The semantic cost $\hat{g}^{sem}$ is defined as

$$\hat{g}^{sem} = \lambda g^{mean} + g^{geo}. \tag{15}$$

Here g^{mean} represents the meaning cost with respect to the confidence of the text extractor and g^{geo} is a cost describing if the text object is well posed towards the camera. The weight λ is used to balance two sources of information, which is set as 200 in our implementation. The smaller $\hat{g}^{sem}$ implies more reliable observed semantic information.

The meaning cost g^{mean} in (15) is set as $g^{mean} = 1 - g^{recg}$, where g^{recg} comes from the confidence of text extraction [19] and is usually located in the range of [0, 1]. The larger confidence implies a more reliable recognition result. Here we use the minus operation to keep its consistency with other components. Some cases such as image blur or occlusion (take the fourth frame $\mathcal{K}_3$ in Fig. 3 for example) will lead to a larger g^{mean}, indicating the extracted text meaning is unreliable.

The geometric cost g^{geo} in (15) is defined as $g^{geo} = l + \lambda'(1 + \boldsymbol{o}^T\boldsymbol{n}/(\|\boldsymbol{o}\|\|\boldsymbol{n}\|))$, which consists of two terms. The first term measures the distance between the text object center and the camera center. The second term measures the difference between the viewing direction $\boldsymbol{o}$ and the normal direction $\boldsymbol{n}$ of the text object as visualized in Fig. 3. The weight λ' is set to 10 in our implementation.

The semantic information of a newly detected text object is initialized as the first observation information $\mathcal{X}_0 \leftarrow \hat{\mathcal{X}}$. Whenever a new observation $\hat{\mathcal{X}} = \{\hat{\boldsymbol{s}}, \hat{g}^{sem}\}$ arrives, the semantic information of the text object is updated by

$$\mathcal{X}_k \leftarrow \underset{\mathcal{X}_{k-1}, \hat{\mathcal{X}}}{\arg\min}(g^{sem}_{k-1}, \hat{g}^{sem}). \tag{16}$$

In other words, the semantic information with the smallest semantic cost is selected. Based on this strategy, the extracted information under good conditions (legible and non-occluded text patches in the right orientation and close to the viewpoint) is preferred. Hence the semantic information will be more accurate with more high-quality observations available.

IV. TextSLAM System

Our SLAM system, TextSLAM, is built upon a point-based system and adopts the keyframe-based framework to integrate the text features tightly. The mixture of point features and text features allows our system to work properly even in the scenes without any text signs. Fig. 4 illustrates the flowchart of TextSLAM. We'll introduce the key components in the following sections.

A. Initialization of Text Objects

A text object is initialized only when it is correctly extracted from the image. Extracting scene texts from images is highly challenging until deep neural networks have been used. Recently, text detectors based on convolution neural networks [19], [20], [21], [22], [23], [24], [25], [26] have achieved impressive performances. We adopt AttentionOCR [19] as the text extractor in our implementation, which ranks at the top on ICDAR2019 Robust Reading Challenge on Arbitrary-shaped Text [29] and supports multiple languages. Some text extraction results are

shown in Fig. 19. The outputs are arbitrary-orientation quadrilaterals enclosing the text regions. Note that our system is not limited to any particular text extractors. The text extractor can be replaced if more advanced ones are available.

To initialize the parameters θ of a text object newly detected in a keyframe, we track the FAST [61] feature points within the text region by Kanade-Lucas-Tomasi [62] tracker until the next keyframe. Let $\boldsymbol{m}_i \leftrightarrow \boldsymbol{m}'_i$ be the corresponding points in both keyframes, and $\mathbf{R}, \boldsymbol{t}$ be the relative pose between the two frames. From (7), we have $\tilde{\boldsymbol{m}}'_i \times \mathbf{H}\tilde{\boldsymbol{m}}_i = 0$. By taking $\mathbf{H} = \mathbf{R} + \boldsymbol{t}\boldsymbol{\theta}^{\mathrm{T}}$, we then obtain

$$[\tilde{\boldsymbol{m}}'_i]_\times \boldsymbol{t}\tilde{\boldsymbol{m}}_i^{\mathrm{T}}\boldsymbol{\theta} = -[\tilde{\boldsymbol{m}}'_i]_\times \mathbf{R}\tilde{\boldsymbol{m}}_i, \tag{17}$$

where $\tilde{\boldsymbol{m}}_i$ and $\tilde{\boldsymbol{m}}'_i$ are the homogeneous coordinates of $\boldsymbol{m}_i$ and $\boldsymbol{m}'_i$ and $[\cdot]_\times$ denotes the skew symmetric matrix corresponding to the cross product. Note that the rank of the matrix on the left hand side is one. It requires at least three pairs of corresponding text points to solve $\boldsymbol{\theta}$.

After initialization of the parameters of a text object, we also keep the four corners of the quadrilateral indicating the text region. The newly initialized text objects are kept being updated in the following frames. They are inserted into the 3D text map whenever the two rules are met: 1) the text object has been observed in at least $n_{\min}$ (4 in our implementation) frames; 2) the text parameters converge to a relatively stable state. For the second rule, we check the normal of the text plane changes if larger than 25° in our implementation. Once the parameters of a text object have been initialized, we also keep its semantic information being updated as described in Section III-B. Several text map examples are visualized in Figs. 1 and 19.

After successful initialization, each text is aligned to other frames directly by minimizing the photometric error. Our method only requires sparse text detection to initialize texts that newly occurred and can track them reliably in the following frames without any extra detection inputs.

B. Camera Pose Estimation With Text Objects

Both points and text objects are involved in camera pose estimation. For points, as depicted in Fig. 4, it begins with the extraction of FAST points [61], followed by feature matching using BRIEF descriptors [61]. The resulting 2D-3D correspondences are used for the pose estimation. For text objects, we select text objects observed by previous 2 keyframes for pose estimation and exclude those behind the camera (at least one vertice of the text quadrilateral is behind the camera) or whose orientation is perpendicular to the current viewing direction. We also exclude those text objects whose appearance in the current frame changes much compared with that in the host frame because of occlusion. We use ZNCC for comparison and exclude those text objects with ZNCC less than 0.1 in our implementation. Camera pose estimation is done by minimizing the following cost function

$$E(\mathbf{T}_t) = E_{point}(\mathbf{T}_t) + \lambda_w E_{text}(\mathbf{T}_t), \tag{18}$$

where $\mathbf{T}_t \in SE(3)$ represents the current camera pose at frame t. The first term E_{point} represents the sum of reprojection errors of point features :

$$E_{point}(\mathbf{T}_t) = \sum_i \phi(\|\boldsymbol{m}_i - \mathcal{P}(\mathbf{T}_t, \boldsymbol{X}_i)\|^2), \tag{19}$$

where $\boldsymbol{m}_i$ is the 2D coordinates of the observed point in the image and $\mathcal{P}(\mathbf{T}_t, \boldsymbol{X}_i)$ represents the projection of the 3D point $\boldsymbol{X}_i$ onto the image plane. Here $\phi(\cdot)$ is the Huber loss function to handle outliers. The second term E_{text} contains only photometric errors of text objects, namely,

$$E_{text} = \sum_j E_{photo}^{\pi_j, t}. \tag{20}$$

Here $E_{photo}^{\pi_j,t}$ represents the photometric error of the j-th text object, which is defined in (13). Though we may use all the pixels within the text region to evaluate the photometric errors, an efficient way is to use a small part of them. Since the text region is full of textures, we adopt the FAST points [61] within the text regions as the representative pixels, then follow [7] to use an eight-pixel pattern around each representative pixel to compute the photometric errors.

The trade-off between the two terms in (18) needs to be regulated by the weight λ_w since they are in different units (position difference versus intensity difference). The weight λ_w is computed as $\lambda_w = \sigma_{rep}/\sigma_{photo}$. σ_{rep} represents the standard deviation of the reprojection error of a pair of corresponding points (in both x and y directions) and σ_{photo} represents the standard deviation of the photometric error of a text object as defined in (11). Those standard deviations can be acquired through a small set of training data (given corresponding points and text patches).

Optimization of the cost function (18) is a nonlinear least-squares problem. As the photometric cost E_{text} is highly nonlinear, it requires a good initial guess of $\mathbf{T}_t$ to avoid being trapped in a local minimum. We first use a constant velocity model to predict the camera pose and then apply a coarse-to-fine strategy for efficient optimization.

Specifically, we downsize the images by $1/2$ recursively to build an image pyramid with three levels. Both the sampled points in the text region and detected feature points outside the text regions are down-sampled to reduce the number of variables to be optimized at coarse levels. The optimization starts from the coarsest level. The result is used to initialize the optimization process at the next level until reaching the final level. To downsample the text points in the next level, we divide the bounding box of a text object into a grid. For each cell in the grid, we select the point with the largest gradient. The number of cells for sampling is set to be $N_0/4^l + 100$, where N_0 is the number of points in the original resolution. Downsampling the feature points outside text regions works in a similar way, where the whole image is divided into cells for sampling.

During coarse-to-fine optimization, those points (including the text points) with large errors are marked as outliers and discarded. For each text object, when more than 99% text points are marked as outliers, this text object is marked as an outlier at this frame.

C. Text Objects Culling

To ensure the good quality of the 3D text map, we drop those text objects from further processing which have been frequently recognized as outliers in camera pose estimation. Specifically, let $\#F_{bad}, \#F_{good}$ be the number of frames where the text object is marked and not marked as an outlier respectively. We check if the following conditions hold for the text object after finishing each bundle adjustment in our implementation:

1) the text object is marked as an inlier in at least two frames ($\#F_{good} > 2$);
2) the number of bad frames is less than the number of good frames and also less than a preset limit ($\#F_{bad} < 0.9\#F_{good}$ and $\#F_{bad} < 40$).

If one of those conditions is not met, the text object is set as 'bad object' and excluded from future processing.

D. Bundle Adjustment With Text Objects

We apply bundle adjustment from time to time in a local window of keyframes similar to existing SLAM systems [4]. The cost function of bundle adjustment also consists of the point part and the text part :

$$E(\boldsymbol{x}) = E_{point}(\boldsymbol{x}) + \lambda_w E_{text}(\boldsymbol{x}). \tag{21}$$

The cost function resembles that of camera pose estimation while involving more parameters to be optimized. The variable $\boldsymbol{x}$ includes the camera poses of keyframes in the local window, the inverse depth of point features, and the text parameters. We also adopt a coarse-to-fine method to optimize (21) as described in camera pose estimation. We perform point culling by excluding the map points identified as outliers in the bundle adjustment.

E. Loop Closing Using Scene Texts

Scene texts are reliable landmarks for place recognition because their meanings are invariant to changing illuminations or viewpoints. We present how to use scene texts to detect revisited places and also integrate them to correct the accumulated error as in our SLAM system.

1) Detection of Loop Candidates: To detect possible loops, we need to compare the latest keyframe with old keyframes. Existing visual SLAM systems usually use the bag-of-visual-words vectors [63], [64] for comparison. The visual words, clustered from the feature descriptors, rely on the image appearances that may change drastically, leading to false or missing loop detection. By contrast, the meanings extracted from those text objects - the text strings or the real words - will not change with image appearances. Hence our idea for loop closing is to use those 'real words' instead of 'visual words' for searching the similar keyframes.

Our searching process consists of two steps. The first step is to match reliable words (have been refined by multiple covisible frames) observed in the latest keyframe to existing 3D text objects in the map. Directly matching a 3D text map is far more efficient than matching the 2D detections on the historical frames because the latter requires much more comparisons due to the repeated observations of a single word, as discussed in Section V-C2. The second step is to select loop candidates from the keyframes associated with those matched 3D text objects (note that the keyframes within the sliding window of bundle adjustment are excluded from selection).

To match a word in the query frame to a 3D text object in the map, we directly compare their meanings (text stings) $\boldsymbol{s}_i$, $\boldsymbol{s}_j$ by

$$s(\boldsymbol{s}_i, \boldsymbol{s}_j) = \frac{\max(|\boldsymbol{s}_i|, |\boldsymbol{s}_j|) - d(\boldsymbol{s}_i, \boldsymbol{s}_j)}{\max(|\boldsymbol{s}_i|, |\boldsymbol{s}_j|)} \in (0, 1], \tag{22}$$

where $|\boldsymbol{s}|$ is the length of a string $\boldsymbol{s}$ and $d(\boldsymbol{s}_i, \boldsymbol{s}_j)$ is the Levenshtein distance [65] between two strings, which measures the minimum operations changing one string $\boldsymbol{s}_i$ to the other string $\boldsymbol{s}_j$, including deletion, insertion and substitution. For example, changing 'seed' to 'seek' needs 1 operation: substituting 'd' with 'k'. So the distance is 1. The two strings are matched when the similarity score $s(\boldsymbol{s}_i, \boldsymbol{s}_j)$ is above a threshold. With such a similarity score, it allows two strings to be matched even they are not exactly the same, which may happen when the text object is partially occluded or falsely recognized. The threshold of being matched or not is selected based on the best matching result. If one text object in the query frame is exactly matched to a text object in the map, $s(\boldsymbol{s}_i, \boldsymbol{s}_j) = 1$, we require all the text objects to be exactly matched by setting the threshold to be 1. Otherwise, we set the threshold proportional to the maximum matching score $s^{\max}$ by $\max(\frac{2}{3}s^{\max}, 0.35)$ to address partial occlusion or false recognition of text objects empirically, where 0.35 served as the minimal threshold, dedicating the two texts are matched when at least one-third characters are same among the entirety. This adaptive threshold scheme increases the robustness of our system in different scenes.

The candidate keyframes (the top ten are selected) for loop closing are selected from the keyframes associated with those matched text objects where the number of matched text objects is greater than a threshold $s_{\min}$, which is set to be proportional to (60%) the minimum number of covisibile text objects in the keyframe connected to the latest frame in the covisibility graph [4], while being larger than three for outdoor scenes and two for indoor scenes in our experiments.

2) Compute the Relative Transformations: The relative transformation between the current keyframe and the loop frame is required to be estimated to close the loop. We follow [4] to compute the similarity transformation between the current keyframe and the loop keyframe and the key is to obtain point correspondences between the two frames. However, it becomes highly challenging to acquire correct point correspondences when the illumination or viewpoint varies significantly. Since text objects are matched in loop detection by their semantic meanings, they can be used as a reliable prior for searching point-level correspondences even when illumination or viewpoint changes dramatically. Specifically, we search the point correspondences based on text points within the matched text regions instead of the whole images. We find that the contrast of a text object in the image does not change as much as we expect under different illuminations (unlike the color or intensity) as shown Fig. 5. Therefore, the BRIEF descriptor [66], relying on the relative difference of a pair of pixels, works well for matching the text points within the limited regions of two matched text

Fig. 5. First two columns show the images captured at the same location in both day and night. The text sign with red rectangles is enlarged to show the dramatic appearance changes between day and night. The third column shows the text images after histogram equalization, where the top and the bottom are night and day images. Note that the contrast of the text patches does not change as much as we expect.

Fig. 6. Top row: The point correspondences by matching BRIEF descriptors. Bottom row: Result of our text-guided point matching.

signs, while it leads to a lot of false correspondences if matching is conducted on the whole images as shown in Fig. 6. The text-guided point matching is robust and accurate even across the day and night as the experimental results show (Figs. 20 and 22).

Similar to [4], after we obtain the 3D to 3D correspondences from the matched text points, we use RANSAC to compute the similarity transformation and optimize it. Next, we perform a guided search to obtain more point correspondences outside the text regions. We then optimize the similarity transformation again and accept those loop candidates with sufficient inliers.

V. Experiments

A. Data Collection

For the absence of SLAM benchmark datasets with text objects, we collected image sequences with scene texts in both indoor and outdoor scenes for evaluation. We use different devices for data collection in indoor and outdoor scenes. Our device for indoor scenes is shown in Fig. 16. It consists of an RGB camera (Intel's RS-D455) for capturing the color images and several optical markers for obtaining ground truth via a motion capture system. Our device for outdoor scenes is shown in Fig. 8. It consists of three RGB cameras recording multiple image sequences in different viewing directions simultaneously. We'll discuss how to acquire the ground truth trajectories for outdoors in the later sections. Image sequences are resized to 640×480 in all the tests.

TABLE II
Results of Indoor Tests

Seq.	ORB-SLAM		DSO		Our point-only		TextSLAM	
	APE	RPE	APE	RPE	APE	RPE	APE	RPE
Indoor_01	0.068	0.062	0.069	0.066	0.086	0.076	**0.067**	**0.055**
Indoor_02	0.092	0.075	0.070	0.060	**0.067**	0.055	0.068	**0.055**
Indoor_03	0.094	0.140	0.083	0.068	0.090	**0.060**	**0.076**	0.111
Indoor_04	0.078	0.061	0.075	0.062	**0.071**	**0.047**	0.076	0.060
Indoor_05	0.084	0.071	0.079	0.052	0.072	0.048	**0.058**	**0.037**
Indoor_06	0.089	0.070	0.074	0.054	0.084	0.059	**0.073**	**0.050**
Indoor_07	0.069	0.051	0.069	0.055	0.045	**0.031**	**0.032**	0.035
Indoor_08	0.081	0.077	0.068	0.055	0.051	0.046	**0.049**	**0.041**
Indoor_09	0.101	0.070	**0.076**	**0.055**	–	–	0.096	0.059
Indoor_10	0.075	0.062	**0.071**	**0.047**	0.094	0.065	0.076	0.053

The middle bar '–' indicates the algorithm fails to finish the whole trajectory. The bold texts indicate the best results and the underlined texts highlight the better result between TextSLAM and the point-only baseline.
Δ in the title: RPE calculates the relative poses over the distance interval Δ. The same applies in the following tables.
APE (m) and RPE (m, $\Delta = 1m$).

B. Indoor Tests

Indoor tests were conducted within a laboratory. A room with a motion capture system was used to obtain the ground truth trajectories of millimeter accuracy as shown in Fig. 16. The room was placed with random texts and those text strings were sampled from COCO-Text [27], which is a large-scale text-orientated natural image dataset, where the fonts and sizes are randomly selected.

We compare our TextSLAM system with the state-of-the-art visual SLAM systems: ORB-SLAM [63] and DSO [7], where ORB-SLAM uses point features and DSO directly operates on raw pixels. By contrast, our system uses both points and text features and is not limited to text-rich scenes - if no text has been detected, our system can use only point features. To evaluate the effectiveness of integrating text objects into the SLAM pipeline, we also present the results of our system with only point features enabled (Our point-only baseline).

1) Evaluation of Camera Trajectory: In this test, we evaluate the performance of trajectory estimation of different systems by using the relative pose error (RPE) and the absolute pose error (APE). Loop closing was disabled for all the systems. Ten indoor sequences were used for evaluation. The results are shown in Table II. We can see that our TextSLAM system performs better than our point-only baseline, demonstrating the benefit of using the high-level text features. Our system also outperforms both ORBSLAM [63] and DSO [7] in most sequences and only performs slightly worse than DSO [7] in a few sequences (Indoor_09, 10). Though the test scene with text labels is highly textured which is ideal for the point-based algorithms, our TextSLAM still performs the best among all the systems, again indicating the benefit from integrating text objects in the SLAM pipeline.

We also evaluate the robustness of the proposed method under fast camera motion. Considering that commercial cameras, such as GoPro, are prone to image blur under fast motion, we use GoPro to collect image sequences under rapid motion. The rapid motion causes severe image blur as shown in Fig. 9,

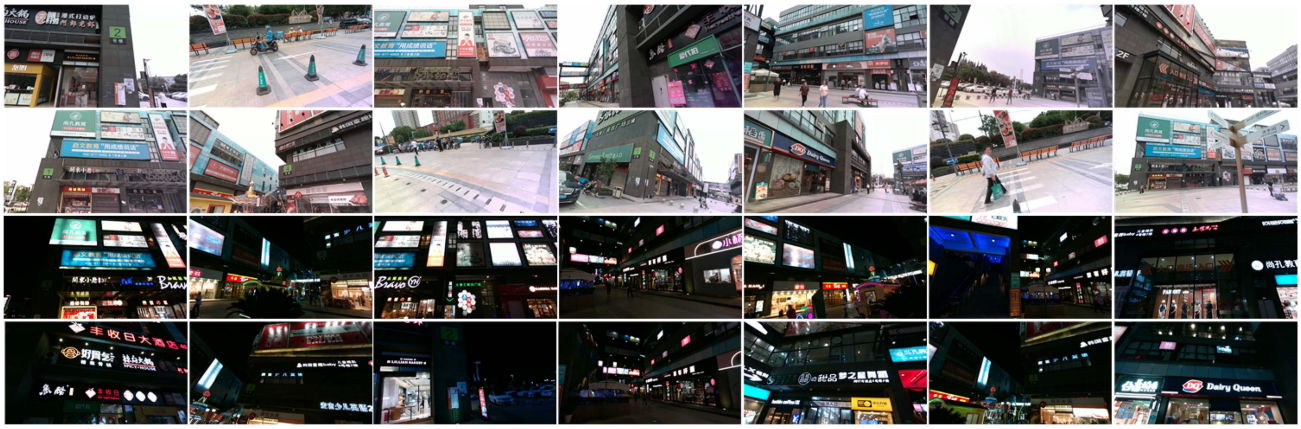


Fig. 7. Our outdoor datasets were collected in a commercial center, which is full of text signs with different sizes, fonts, and languages. The datasets consist of test sequences collected during both the day and night.

Fig. 8. Outdoor test scene is shown on the left. The data collection device equipped with three RS-D455 cameras is presented on the right.

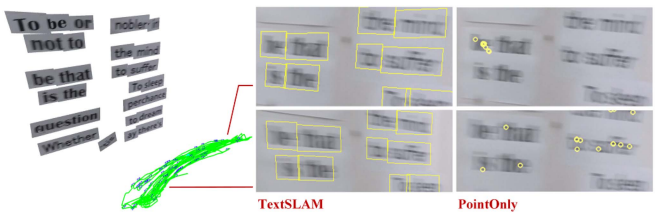


Fig. 9. TextSLAM is robust to blurry images caused by rapid camera motions. The estimated 3D text map and camera trajectory of TextSLAM are shown on the left. By contrast, the point-only method failed to track feature points on severely blurry images as shown on the right.

making our point-only baseline fail in all the cases. ORB-SLAM works properly because of its well-implemented relocalization mechanism and it simply skips bad keyframes with image blur. By contrast, no relocalization is implemented in our system. Without the relocalization mechanism, DSO also fails in one test but still performs better than our point-only baseline. By contrast, our text-based method works well in those tests. The text objects are successfully tracked as shown in Fig. 9 and the trajectories are more accurate than ORB-SLAM and DSO as shown in Table III. This is largely due to tracking the text object as a whole by directly optimizing well-designed photometric errors.

TABLE III
RESULTS OF RAPID MOTION TESTS

Seq.	ORB-SLAM		DSO		Point-only		TextSLAM	
	APE	RPE	APE	RPE	APE	RPE	APE	RPE
Rapid_01	0.061	0.122	–	–	–	–	**0.060**	**0.104**
Rapid_02	0.036	0.080	0.027	**0.041**	–	–	**0.020**	0.056
Rapid_03	0.085	0.142	0.113	**0.083**	–	–	**0.058**	0.107

The bar '–' indicates the algorithm fails to finish the whole trajectory.
APE (m) and RPE (m, $\Delta = 1m$).

2) Evaluation of 3D Text Maps: Our TextSLAM system can directly produce a 3D text map. It would be interesting to evaluate the quality of the 3D text map. Since no other SLAM system generates text maps directly, we implement a baseline method by fitting the text planes to the 3D map points generated from ORB-SLAM and DSO within text regions using three-point RANSAC [67].

To evaluate the quality of 3D text maps, we acquire the ground truth plane equation $(\boldsymbol{n}_{gt}, d_{gt})$ by placing optical markers on the text objects as shown in Fig. 16. We use the angular error and the distance error between the estimated text plane and the ground truth for evaluation. The angular error measures the difference between the estimated normal $\boldsymbol{n}_t$ of the text plane and the ground truth $\boldsymbol{n}_{gt}$, namely $\alpha = \arccos(|\boldsymbol{n}_t^{\mathrm{T}}\boldsymbol{n}_{gt}|/\|\boldsymbol{n}_t\|\|\boldsymbol{n}_{gt}\|)$. The distance error is measured by the distance between the 3D text points and the ground truth text planes, where the four corners of the text quadrilateral are selected as the 3D text points.

The statistics distributions of angular errors and distance errors over ten sequences are shown in Figs. 11 and 12, respectively. The visual comparison of one test sequence is presented in Fig. 10. The results show that the 3D text map produced by TextSLAM is better than that of the plane fitting baselines based on either ORB-SLAM or DSO. This is largely due to the parameters of a text object being estimated as a whole in TextSLAM, while in ORB-SLAM or DSO, text points are estimated separately without considering they are on the same plane. Therefore, the fitted planes are noisy as shown in Fig. 10.

3) Evaluation of Loop Closing: Additional sequences were recorded to evaluate the loop closing performance within two

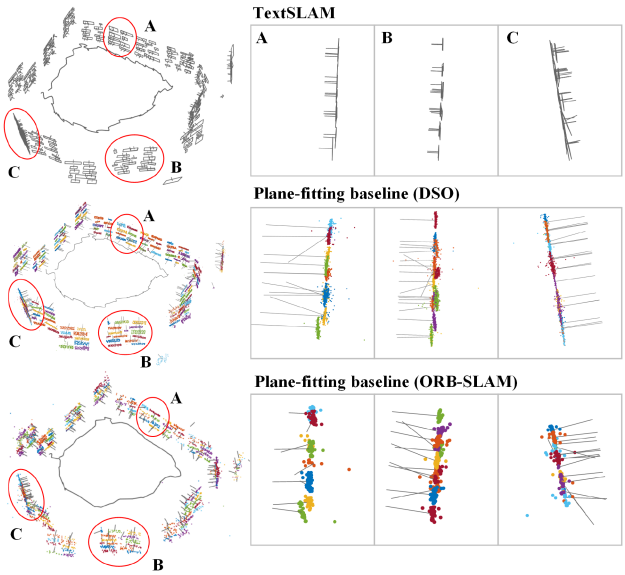


Fig. 10. Though RANSAC was adopted, plane fitting on the point clouds from ORB-SLAM still produced noisy results (as shown in the bottom row). DSO performs better because much more points were taken into the computation. By contrast, TextSLAM avoids such problems by tracking a text object as a whole via directly optimizing the photometric errors.

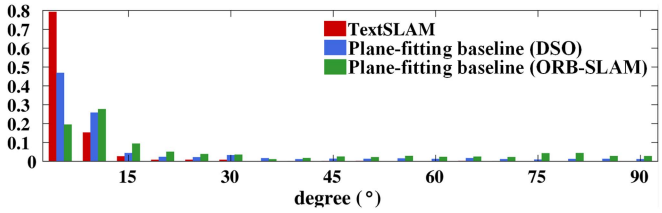


Fig. 11. Statistic distribution of the angular errors. The results of TextSLAM, plane-fitting baselines based on DSO and ORB-SLAM are illustrated in red, blue, and green respectively.

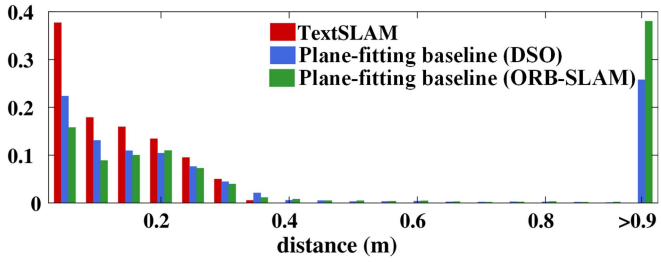


Fig. 12. Statistic distribution of the distance errors. The results of TextSLAM, plane-fitting baselines based on DSO and ORB-SLAM are illustrated in red, blue, and green respectively.

indoor scenes. The first scene is in a small room equipped with a motion capture system that provides the ground truth trajectories all the time. Some printed texts were randomly placed in the room similar to the first experiment as shown in Fig. 16. The second scene spans the whole floor of a building and contains some sparse text signs or labels as shown in Fig. 13. We started and ended recording within the same room with the motion capture system to acquire the ground truth poses in the start and end parts of a trajectory. We follow [68] to compute the positional errors using the partially available ground truth.

Fig. 13. Indoor datasets used for loop closing tests were collected within a laboratory environment where some sparse text signs are available. We collected the test sequences in day and night and also turned on and off the lights to make the tests more challenging.

TABLE IV
LOOP TESTS IN A SMALL INDOOR SCENE

Seq.	ORB-SLAM			TextSLAM		
	LOOP	APE	RPE	LOOP	APE	RPE
AIndoorLoop_01	√	**0.005**	0.046	√	0.010	**0.031**
AIndoorLoop_02	×	0.075	0.077	√	**0.018**	**0.030**
AIndoorLoop_03	×	0.076	0.090	√	**0.007**	**0.042**
AIndoorLoop_04	√	0.028	0.035	√	**0.026**	**0.027**
AIndoorLoop_05	√	**0.008**	**0.034**	√	0.017	0.040
AIndoorLoop_06	×	0.068	0.069	√	**0.010**	**0.027**
AIndoorLoop_07	×	0.083	0.071	√	**0.017**	**0.032**
AIndoorLoop_08	×	0.082	0.086	√	**0.008**	**0.026**

The tick '√' indicates a success loop closing and '×' indicates no loop has been found. The smallest errors are in bold texts.
APE (m) and RPE (m, $\Delta = 1m$).

TABLE V
LOOP TEST IN A LARGE INDOOR SCENE

Seq.	ORB-SLAM			TextSLAM		
	LOOP	APE	RPE	LOOP	APE	RPE
LIndoorLoop_01	×	0.994	0.770	√	**0.062**	**0.111**
LIndoorLoop_02	×	1.669	1.177	√	**0.057**	**0.071**
LIndoorLoop_03	×	2.102	0.595	√	**0.192**	**0.374**
LIndoorLoop_04	×	0.192	0.202	√	**0.023**	**0.075**
LIndoorLoop_05	×	0.251	0.127	√	**0.047**	**0.065**
LIndoorLoop_06	×	0.179	0.163	√	**0.032**	**0.041**
LIndoorLoop_07	×	0.206	0.328	√	**0.031**	**0.230**
LIndoorLoop_08	×	0.291	0.155	√	**0.031**	**0.042**
LIndoorLoop_09	×	0.377	0.202	√	**0.031**	**0.034**

The tick '√' indicates a success loop closing and '×' indicates no loop has been found. The smallest errors are in bold texts.
APE (m) and RPE (m, $\Delta = 1m$).

The results are shown in Tables IV and V, where the ORB-SLAM's results are also presented for comparison. We also visualize some results in Figs. 14 and 15. In those tests, ORB-SLAM failed to detect most loops, while TextSLAM can detect all the loops correctly. The reason is that the viewpoint of the current frame changes significantly from that of the loop frame (see Figs. 14 and 15), making ORB features difficult to be matched. By contrast, our TextSLAM uses the semantic meaning of those text objects to detect loop frames. The semantic meaning of a text object will keep unchanged with viewpoint changes. The results suggest that text objects can be used as reliable landmarks for loop closing even though they are distributed sparsely in the scene.

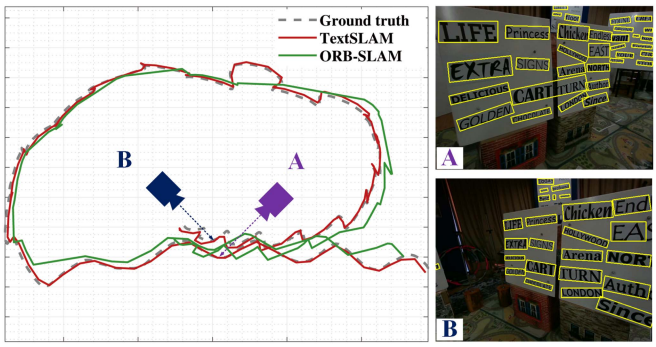


Fig. 14. Visualization of TextSLAM results in a small indoor scene. The trajectories of the two methods are shown on the left. The results of TextSLAM, ORB-SLAM, and the ground truth are illustrated in red, green, and gray respectively. The query frame B and the detected loop frame A are shown on the right, where the viewpoint changes significantly.

TABLE VI
RESULTS IN AN OUTDOOR COMMERCIAL CENTER DURING THE DAY

Seq.	ORB-SLAM			DSO		TextSLAM		
	LOOP	APE	RPE	APE	RPE	LOOP	APE	RPE
Outdoor_1	×	1.159	**0.379**	1.175	0.393	√	**0.688**	0.389
Outdoor_2	×	1.340	0.511	1.280	**0.457**	√	**0.561**	0.470
Outdoor_3	×	1.213	0.347	1.423	0.337	√	**0.807**	**0.317**
Outdoor_4	√	**0.116**	**0.108**	1.511	0.462	√	1.624	0.759
Outdoor_5	√	**0.175**	**0.094**	1.410	0.523	√	0.219	0.173
Outdoor_6	×	1.491	0.462	1.450	0.299	√	**0.412**	**0.238**
Outdoor_7	×	1.279	0.457	1.572	0.369	√	**0.563**	**0.307**
Outdoor_8	×	1.358	0.529	1.642	0.620	√	**0.446**	**0.249**

The tick '√' indicates a success loop closing and '×' indicates no loop has been found. The smallest errors are in bold texts.
APE (m) and RPE (m, $\Delta = 1m$).

C. Outdoor Tests

In this experiment, we test our TextSLAM system in a commercial plaza during the day and night. Some pictures are shown in Fig. 7. As we can see, the environment is full of text objects with various sizes, fonts, backgrounds, and languages, as well as various challenges including complex occlusions, the reflection of the glass, and moving pedestrians.

The ground truth camera trajectories are required for evaluation. One possible solution is to use the RTK GPS receiver to obtain the camera trajectories in centimeter-level accuracy. However, it is not feasible because we found that satellite signals were occluded by the surrounding buildings. Instead, we use the struct-from-motion technique to obtain the ground truth following the idea of [69], [70]. We collected a full set of image sequences to densely cover the scene and ran COLMAP [71] to obtain the camera pose for each image. After that, the camera poses obtained from COLMAP are treated as the ground truth and used to evaluate the SLAM performance. The 3D map and camera trajectories from COLMAP of the outdoor scene are visualized in Fig. 17. We selected eight sequences among the full set of image sequences for evaluation. To cover the scene more efficiently, we use three cameras with different headings to capture the images in different viewpoints as shown in Fig. 8.

Since COLMAP produces 3D structures with an unknown scale, we need to calibrate the scale by a reference distance. As shown in Fig. 8, we placed two checkerboards in the scene. Their orientation was kept the same such that the distances between corresponding points on the board are identical. The checkerboard corners can be extracted and matched, whose 3D coordinates can be estimated from the known camera poses produced by structure-from-motion. We compared the estimated distance from structure-from-motion and the measured distance by a laser rangefinder to resolve the unknown scale.

To evaluate the accuracy of the ground truth, we chose five reference points whose real-world coordinates were precisely measured by a laser rangefinder. We then placed the camera at those reference points to capture extra images and fed them into the COLMAP pipeline. Those estimated locations were aligned with the real-world coordinates to evaluate the accuracy of the ground truth approximately. We found that the average localization error is about 8.45 cm within the test area around 5500 m^2, which is sufficient for our evaluation.

1) Day Tests: In this experiment, we evaluate our methods with the image sequences collected during the day. We also present the results of ORB-SLAM and DSO for comparison. The results are shown in Table VI. TextSLAM can correctly recognize revisited places and close the loop in all test sequences, achieving the best accuracy among all the methods. ORB-SLAM fails to detect most loops because of large viewpoint changes and performs similar to DSO that has no loop closing function. To be more clear, we visualize estimated trajectories for typical sequences in Fig. 18, as well as the loop image pairs with text objects detected by TextSLAM. As shown in Fig. 18, the viewpoints change significantly between the current keyframe and the loop keyframe, making the BoW-based method (ORB-SLAM) fail to detect those loops. By contrast, the semantic message of a text object is invariant to appearance changes, hence TextSLAM is able to detect the correct loop via using this high-level information. When the viewpoint changes only slightly, e.g., Outdoor_5 in Table VI, the well-implemented ORB-SLAM can correctly close the loop and produce results as accurate as ours. Extra results of TextSLAM are presented in Fig. 19, including the reconstructed 3D text map for all scene texts existing in the test scene, as well as their 2D observations. The 3D semantic text map could have potential in multiple applications, including scene understanding, navigation, augmented reality, and human-computer interaction.

2) Day-Night Tests: Illumination change is one challenge that SLAM usually encounters. An extreme case is the day night variation. For example, we already built a map in the day, while we may want to reuse it at night. To evaluate the performance towards this change, we collect night sequences in the same path as collecting those day sequences.

To show how texts help matching scene points across day and night, we implemented the localization-only version for both the TextSLAM (TextSLAM_Loop) and ORB-SLAM systems. Based on the 3D model generated from the day sequence via each SLAM method, we test the localization performance using the night sequence. We present visual comparisons in Fig. 20. The results show that our method can correctly locate a lot of frames,

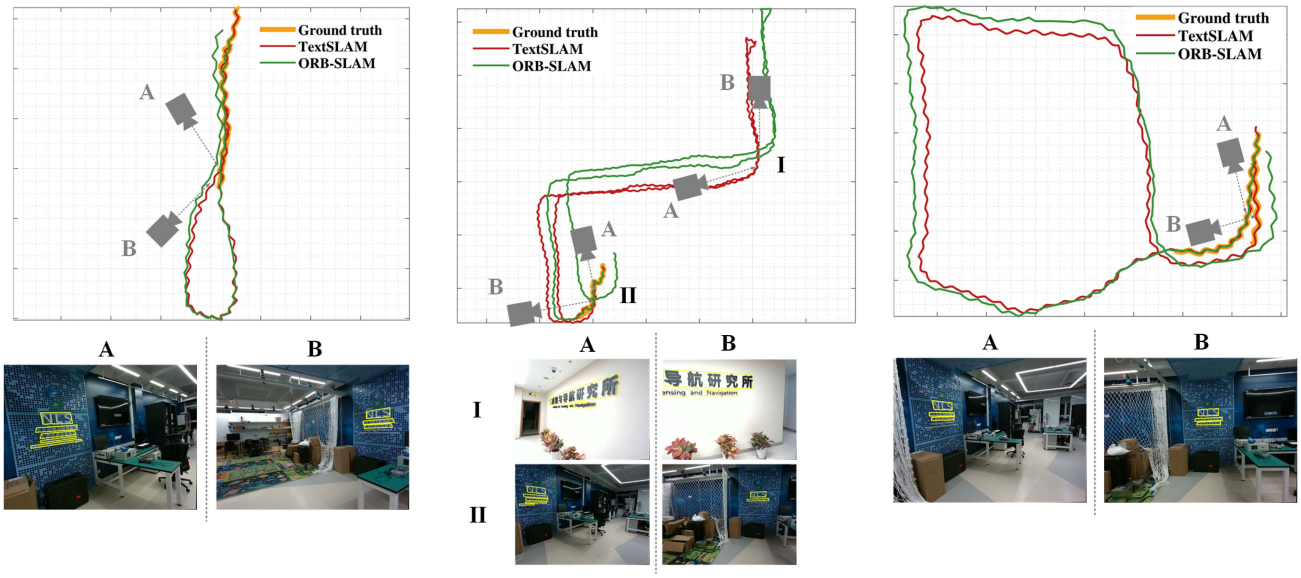


Fig. 15. Visualization of TextSLAM results in a large indoor scene. Top row: The camera trajectories of TextSLAM, ORB-SLAM, and the ground truth are visualized in red, green, and orange respectively. The ground truth (orange line) is at the start and end parts of each trajectory. Bottom row: The query frame B and the detected loop frame A in TextSLAM are visualized, where the matched text objects are highlighted in yellow boxes. We can see that loops are correctly detected in TextSLAM despite large viewpoint changes.

Fig. 16. Indoor test scene is shown on the left. The data collection device equipped with an RS-D455 camera is presented on the right.

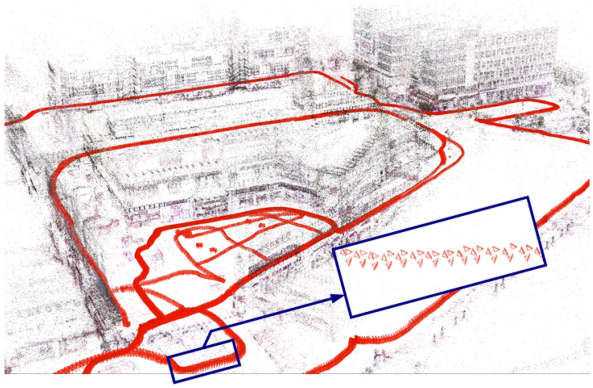


Fig. 17. Structure-from-motion model that we used as ground truth in the outdoor tests. Three surround-view cameras were used for data collection as illustrated in the enlarged area.

while ORB-SLAM works for only a few frames, implying the robustness of our method under such large illumination changes.

We also compare our method with the state-of-the-art visual localization methods in the day-night tests. Because COLMAP failed to generate the ground truth trajectory of the night sequence, we follow the image retrieval evaluation protocol to use precision and recall for comparison. Additionally, the runtime is measured to show the efficiency of the approaches. Our day-night test contains 987 day keyframes as the database and 145 night keyframes as the queries. To obtain the ground truth of the night queries, we manually label each image pair among all 143115 (987×145) pairs by checking if the two images belong to the same location.

We compared the following methods from image retrieval, place recognition, and visual localization. Some of them are the top-ranked open source implementations (3rd, 4th, 34th ranked) in the Long-Term Visual Localization benchmark [72].

- NetVLAD [73], the state-of-the-art deep learning-based image retrieval method.
- DBoW2 [64], which is widely used in existing visual SLAM systems [63].
- TextPlace [15], the place recognition method which uses 2D text extractions as the localization cue. Because TextPlace does not open its source code, we reimplement it for comparison, named as TextPlace [15]_ReImplement in results.
- NetVLAD [73]+SuperPoint [74]+SuperGlue [75] (abbr. NetVLAD+SP+SG). 3rd ranked method in [72].
- NetVLAD [73]+SuperPoint [74]+SuperGlue [75]+Patch2 Pix [76] (abbr. NetVLAD+SP+SG+PP). 4th ranked method in [72].
- NetVLAD [73]+Patch2Pix [76] (abbr. NetVLAD+PP). 34th ranked method in [72].
- NetVLAD [73]+SIFT [77] (abbr. NetVLAD+SIFT).

Here, SIFT [77], SuperPoint [74], SuperGlue [75], and Patch2Pix [76] are used to refine the retrieval results from NetVLAD. SIFT [77] is a classic feature detector and descriptor. SuperPoint [74] is a learned interest

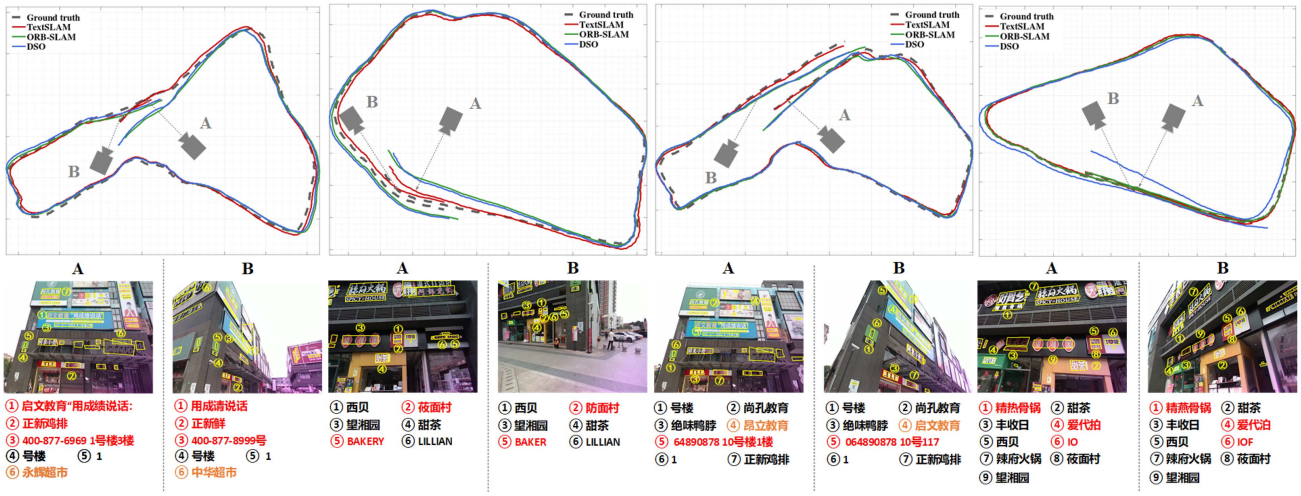


Fig. 18. Results of outdoor tests. Top row: The camera trajectories of TextSLAM, ORB-SLAM, DSO, and the ground truth are visualized in red, green, blue, and gray respectively. The loop frames detected by TextSLAM are also visualized (B represents the query frame and A is the detected loop frame). Bottom row: The semantic meanings of those matched text objects between the loop frame and the query frame are presented, where the matched pair are indicated by the same number. Our method allows those strings to be exactly matched (in black) or partially matched (in red). Some false matching results are shown in brown, which are excluded from the geometric verification during loop closing.

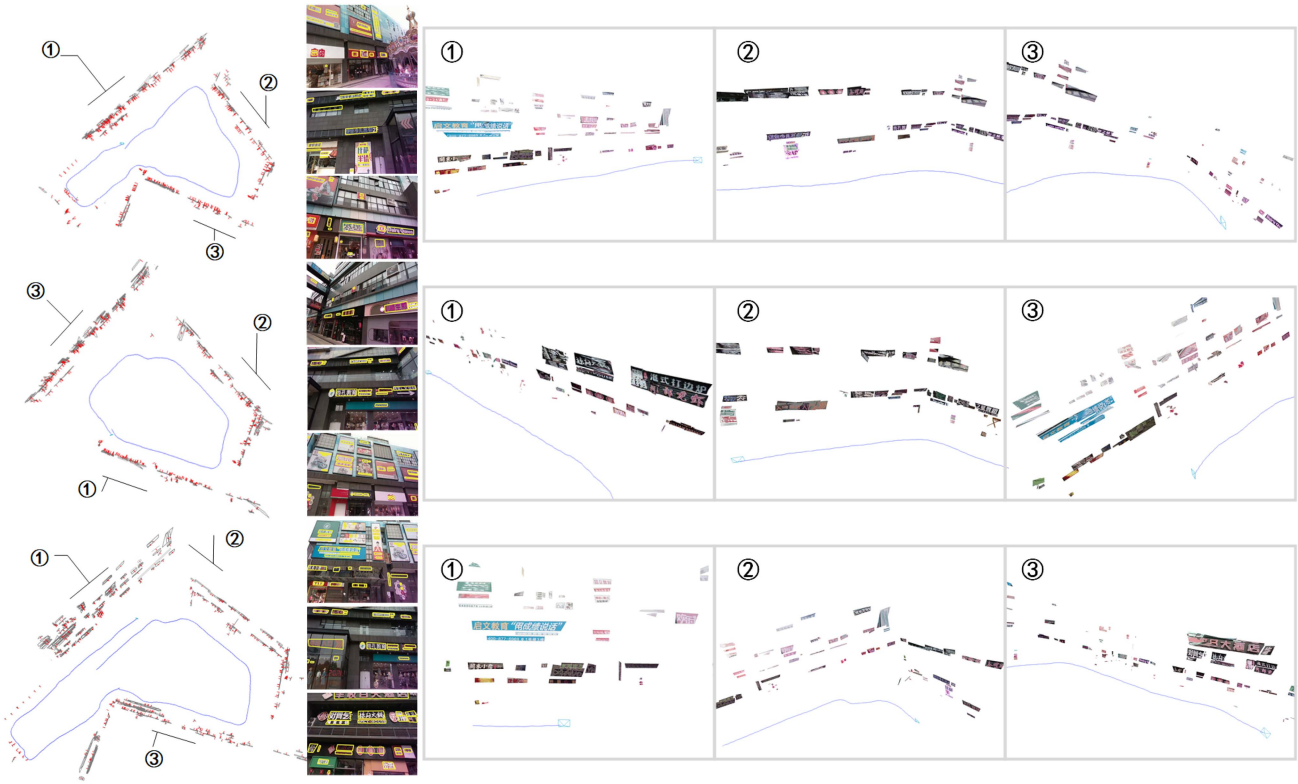


Fig. 19. Extra results of outdoor tests. First column: The full mapping and localization results are shown. Second column: The text detection results of three numbered locations are visualized. Third to fifth columns: The 3D text map in marked locations are zoomed in for more details.

point detector and descriptor. SuperGlue [75] finds pointwise correspondences using a graph neural network with an attention mechanism. Patch2Pix [76] searches correspondences in a detect-to-refine manner (patch-level to pixel match). The results from TextSLAM localization-version are displayed as 'TextSLAM_Loop'. We also present the results of TextSLAM localization-version without point matching and geometric validation as 'TextSLAM_Loop_w/o_Check'.

The deep learning-based methods ran on the NVIDIA RTX A6000 with the Intel i9-10980XE CPU of 128-GB RAM. Other methods ran in a single thread on an Intel Core i7- 9700 K desktop computer with 32-GB RAM. For all compared methods,

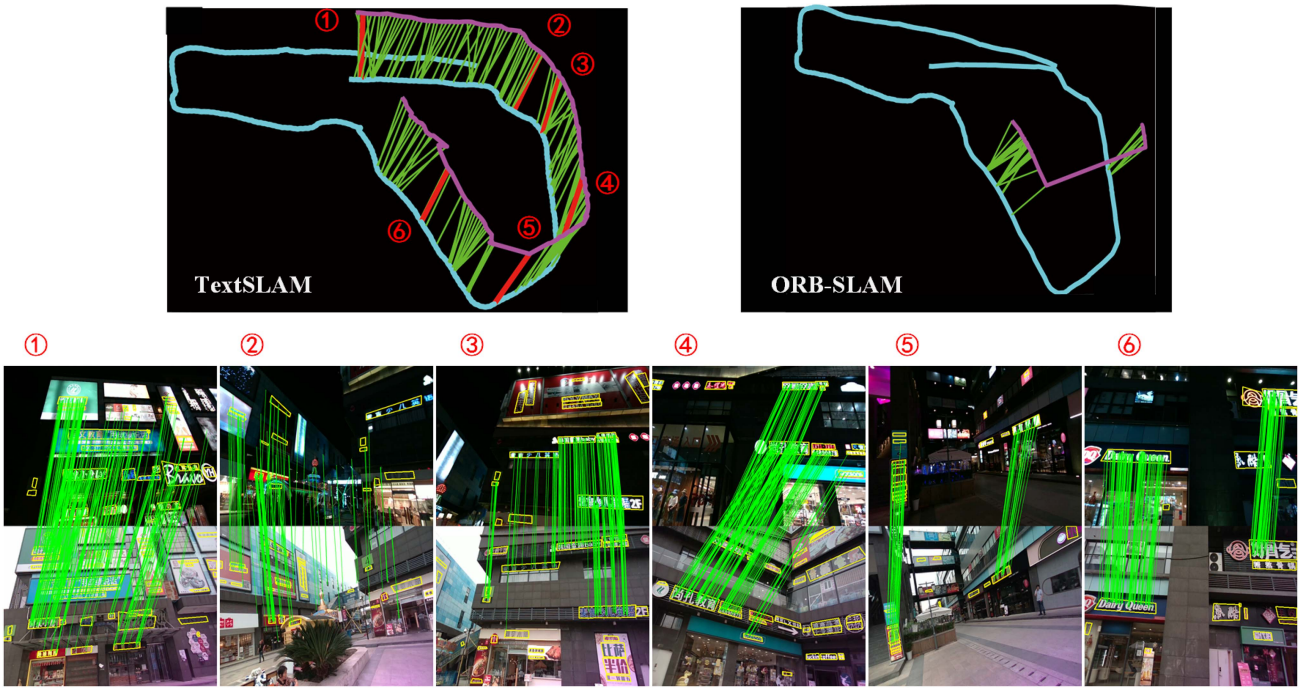


Fig. 20. Top row: The blue trajectories are estimated by TextSLAM and ORB-SLAM respectively using day sequences, while the magenta trajectories are the localization results by registering the night images to the 3D map built during the day. We shift the night trajectories and connect the loop frames by green lines for better illustration. Bottom row: We also visualized the matched points by TextSLAM in six different places. We can see that text-guided point matching correctly matched most of the text points despite large illumination changes.

TABLE VII
AVERAGE RUNTIMES OF LOCALIZATION METHODS. (S)

Methods	Text extraction	Image retrieval	Point matching	Geometric check
NetVLAD [74]	–	0.055 ± 0.001	–	–
NetVLAD+PP	–	0.055 ± 0.001	1.427 ± 0.157	7.999 ± 0.508
NetVLAD+SP+SG	–	0.055 ± 0.001	1.625 ± 0.034	5.992 ± 0.683
NetVLAD+SP+SG+PP	–	0.055 ± 0.001	2.183 ± 0.168	6.332 ± 0.674
TextPlace [15]*	0.521 ± 0.212	0.358 ± 0.077	–	–
TextSLAM_Loop	0.521 ± 0.212	$\mathbf{0.005 \pm 0.002}$	$\mathbf{0.037 \pm 0.011}$	$\mathbf{0.083 \pm 0.018}$

The abbreviations 'PP', 'SP', 'SG','TextPlace [15]' represent Patch2Pix [77], SuperPoint [75], and SuperGlue [76], TextPlace [15]_ReImplement, respectively.*
*The second column **Text extraction** represents the runtime per frame, including the text detection, and recognition for day and night sequences.*

the top 10 candidates are retrieved and validated in further steps. For the methods that output point correspondence results, we adopt a geometric check (fundamental matrix estimation by RANSAC [67]) to reject outliers, and use the inlier number to rank the candidates from the previous step. The re-ranked results are taken as the final retrieval result. For the TextSLAM pipeline, we use PnP to reject the outliers. The precision and recall curves are displayed in Fig. 23. To compare the efficiency of the approaches fairly, we randomly select 10 night queries to run each method repeatedly. The average 10-queries running time results are shown in Table VII. Several examples are shown in Figs. 21 and 22.

The results show that TextSLAM (with and without geometric validation) outperforms the state-of-the-art NetVLAD-based methods on this day-night test, while the latter was particularly trained to address significant illumination changes. Interestingly, another text-based approach (Textplace [15]) also performs better than NetVLAD-based methods. It implies that feature matching can greatly benefit from text semantics in text-rich environments. TextSLAM performs better than Textplace [15] because Textplace [15] uses 2D-2D text matching for image retrieval which is prone to false text detection and recognition. By contrast, TextSLAM matches the 2D text objects directly with the 3D text map where the semantics of 3D text objects are derived from multiple observations and hence are more reliable. The 2D-3D matching strategy also makes TextSLAM highly efficient as shown in Table VII because much fewer text objects in the 3D map are required to be matched.

D. Runtime Analysis

We selected four typical sequences from above experiments to test the runtime of TextSLAM. We ran TextSLAM in a single thread on an Intel Core i7-9700 K desktop computer with 32-GB RAM. The text extractor [19] using a neural network ran on the NVIDIA GeForce RTX 2080 Ti. The runtimes of major components in the system are present in Table VIII. We can see that text extraction requires much more time than other front-end components (point extraction and pose estimation). It is therefore the bottleneck of TextSLAM's efficiency. But this problem can be solved if a highly efficient text extractor appears.

We also present the running time of our text-based loop closing in Table IX, which includes the average single-threaded runtime for loop detection (Section IV-E1), relative transformations calculation (Section IV-E2), and loop correction.

Fig. 21. Image retrieval results of DBoW2, NetVLAD, TextPlace_ReImplement and TextSLAM_Loop_w/o_Check, respectively. The correct and wrong results are shown in green and red boxes, respectively.

Fig. 22. Point correspondence results of TextSLAM_Loop, NetVLAD+SP+SG+PP, NetVLAD+SP+SG, NetVLAD+PP, and NetVLAD+SIFT, respectively. We can see that TextSLAM obtains correct point correspondence based on the invariant semantic text meaning. The deep-learning-based methods also work well during day-night change, as shown in the second and third columns. One failure case of the deep-learning-based method is the fourth column, where the similar blue boards (in red boxes) confuse the approach. SIFT also failed because of two similar boards (in red boxes) in different locations.

VI. Limitation

Here, we discuss the limitations of our method. First, TextSLAM relies on the text objects in the scene. When no text object exists, TextSLAM will switch into the point-only mode. So no high-level information can be used for improving SLAM in this case. Fortunately, many daily scenes, such as city commercial plazas or shopping malls, are full of text objects. Second, TextSLAM still relies on the texture in the environment and easily fails in the scene with little textures, similar to other visual SLAM methods. Third, the text detector still does not work perfectly. False detection and recognition still happen frequently. For example, some windows and bench legs are recognized as text strings. Moreover, these text extraction methods require

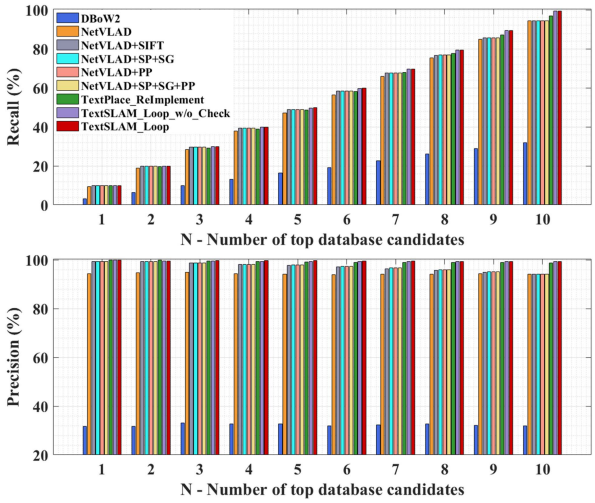


Fig. 23. Recall and precision of the top-10 results. We normalize the recalls by dividing their maximum possible value: 10/{average number of ground-truth pairs per query} for better visualization.

TABLE VIII
RUNTIME ANALYSIS OF OUR METHOD. (S)

Seq.(#Text)	Text extraction	Point extraction	Pose estimation	LocalBA
LIndoorLoop_07 (2)	0.227 ± 0.037	0.008 ± 0.002	0.004 ± 0.003	0.489 ± 0.142
LIndoorLoop_01 (3)	0.245 ± 0.048	0.008 ± 0.002	0.008 ± 0.009	0.660 ± 0.366
Indoor_07 (22)	0.573 ± 0.106	0.008 ± 0.002	0.064 ± 0.019	2.788 ± 0.691
Outdoor_1 (25)	0.547 ± 0.195	0.010 ± 0.002	0.068 ± 0.042	2.438 ± 1.226

*#**Text** in the first column means the average observed text object number per frame. The second column **Text extraction** represents the runtime per frame, including the text detection, recognition as well as text representative pixel extraction.*

TABLE IX
RUNTIME ANALYSIS OF THE LOOP OF OUR METHOD

Seq.	Loop detection	Sim3 calculation	Loop correction
LIndoorLoop_07	0.002s	0.095s	23.761s
LIndoorLoop_01[1]	0.009s	0.195s	33.626s
LIndoorLoop_01[2]	0.008s	0.127s	64.380s
Outdoor_1	0.008s	0.360s	85.855s

The superscript [1] and [2] indicate the first loop and the second loop of LIndoorLoop_01, respectively.

lots of training data that needs great effort for labeling. Finally, our TextSLAM cannot run in real-time currently because of the time-consuming operations on text extraction and the back-end optimization. Nevertheless, the system can be further optimized for efficiency.

VII. CONCLUSION

In this paper, we fully explore the text objects both geometrically and semantically and propose a novel visual SLAM approach tightly coupled with the semantic text features, where text objects are regarded as local planar patches with rich textual and semantic meaning. We tested our method in various indoor and outdoor environments involving challenges such as fast camera motions, viewpoint changes, and day-night illumination shifts. The results show that with the help of scene texts, our method outperforms the state-of-the-art methods including SLAM and visual localization in terms of accuracy and robustness, indicating the benefits of integrating semantic text objects into visual SLAM. The 3D text map produced by our system can serve as an important medium to bridge humans and robots. We hope our work could inspire more explorations to the semantic texts in various applications in robotics, navigation, human-computer interaction, AR and VR, etc.

ACKNOWLEDGMENTS

The authors would like to thank Shanghai Alpha square for the support of data collection.

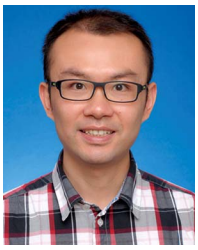
REFERENCES

[1] L. Heng et al., "Autonomous visual mapping and exploration with a micro aerial vehicle," *J. Field Robot.*, vol. 31, no. 4, pp. 654–675, 2014.
[2] H. Lategahn, A. Geiger, and B. Kitt, "Visual slam for autonomous ground vehicles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 1732–1737.
[3] D. Chekhlov, A. P. Gee, A. Calway, and W. Mayol-Cuevas, "Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam," in *Proc. IEEE ACM Int. Symp. Mixed Augmented Reality*, 2007, pp. 1–4.
[4] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[5] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2003, Art. no. 1403.
[6] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 15–22.
[7] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.
[8] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu, "StructSLAM: Visual SLAM with building structure lines," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1364–1375, Apr. 2015.
[9] A. J. Trevor, J. G. Rogers, and H. I. Christensen, "Planar surface slam with 3D and 2D sensors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 3041–3048.
[10] K.-N. Lianos, J. L. Schonberger, M. Pollefeys, and T. Sattler, "VSO: Visual semantic odometry," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 234–250.
[11] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 4628–4635.
[12] A. Ranganathan, D. Ilstrup, and T. Wu, "Light-weight localization for vehicles using road markings," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 921–927.
[13] N. Radwan, G. D. Tipaldi, L. Spinello, and W. Burgard, "Do you see the bakery? Leveraging geo-referenced texts for global localization in public maps," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 4837–4842.
[14] B. Li, D. Zou, D. Sartori, L. Pei, and W. Yu, "TextSLAM: Visual SLAM with planar text features," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2102–2108.
[15] Z. Hong, Y. Petillot, D. Lane, Y. Miao, and S. Wang, "TextPlace: Visual place recognition and topological localization through reading scene texts," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 2861–2870.
[16] X. Rong, B. Li, J. P. Muñoz, J. Xiao, A. Arditi, and Y. Tian, "Guided text spotting for assistive blind navigation in unfamiliar indoor environments," in *Proc. Int. Symp. Vis. Comput.*, Springer, 2016, pp. 11–22.
[17] B. Li et al., "Vision-based mobile indoor assistive navigation aid for blind people," *IEEE Trans. Mobile Comput.*, vol. 18, no. 3, pp. 702–714, Mar. 2019.
[18] H.-C. Wang et al., "Bridging text spotting and SLAM with junction features," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 3701–3708.
[19] J. Zhang, W. Wang, D. Huang, Q. Liu, and Y. Wang, "A feasible framework for arbitrary-shaped scene text recognition," 2019, *arXiv: 1912.04561*.
[20] X. Zhou et al., "EAST: An efficient and accurate scene text detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2642–2651.

[21] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai, "Mask TextSpotter: An end-to-end trainable neural network for spotting text with arbitrary shapes," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 67–83.
[22] H. Wang et al., "All you need is boundary: Toward arbitrary-shaped text spotting," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 12160–12167.
[23] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-time scene text detection with differentiable binarization," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 11474–11481.
[24] M. He et al., "MOST: A multi-oriented scene text detector with localization refinement," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8809–8818.
[25] Y. Zhu, C. Yao, and X. Bai, "Scene text detection and recognition: Recent advances and future trends," *Front. Comput. Sci.*, vol. 10, no. 1, pp. 19–36, 2016.
[26] X.-C. Yin, Z.-Y. Zuo, S. Tian, and C.-L. Liu, "Text detection, tracking and recognition in video: A comprehensive survey," *IEEE Trans. Image Process.*, vol. 25, no. 6, pp. 2752–2773, Jun. 2016.
[27] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie, "COCO-text: Dataset and benchmark for text detection and recognition in natural images," 2016, *arXiv:1601.07140*.
[28] M. Iwamura, T. Matsuda, N. Morimoto, H. Sato, Y. Ikeda, and K. Kise, "Downtown Osaka scene text dataset," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2016, pp. 440–455.
[29] C. K. Chng et al., "ICDAR2019 robust reading challenge on arbitrary-shaped text-RRC-ArT," in *Proc. Int. Conf. Document Anal. Recognit.*, 2019, pp. 1571–1576.
[30] M. Tomono and S. Yuta, "Mobile robot navigation in indoor environments using object and character recognition," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2000, pp. 313–320.
[31] M. Mata, J. M. Armingol, A. de la Escalera, and M. A. Salichs, "A visual landmark recognition system for topological navigation of mobile robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2001, pp. 1124–1129.
[32] C. Case, B. Suresh, A. Coates, and A. Y. Ng, "Autonomous sign reading for semantic mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3297–3303.
[33] S. Wang, S. Fidler, and R. Urtasun, "Lost shopping! Monocular localization in large indoor spaces," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2695–2703.
[34] A. P. Gee, D. Chekhlov, W. W. Mayol-Cuevas, and A. Calway, "Discovering planes and collapsing the state space in visual slam," in *Proc. Brit. Mach. Vis. Conf.*, 2007, pp. 1–10.
[35] A. P. Gee, D. Chekhlov, A. Calway, and W. Mayol-Cuevas, "Discovering higher level structure in visual SLAM," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 980–990, Oct. 2008.
[36] M. Y. Yang and W. Förstner, "Plane detection in point cloud data," in *Proc. Int. Conf. Mach. Control Guid.*, 2010, pp. 95–104.
[37] A. J. Davison and J. Ortiz, "FutureMapping 2: Gaussian belief propagation for spatial AI," 2019, *arXiv: 1910.14139*.
[38] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 573–580.
[39] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1285–1291.
[40] P. Kim, B. Coltin, and H. Jin Kim, "Linear RGB-D SLAM for planar environments," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 333–348.
[41] N. Molton, A. J. Davison, and I. D. Reid, "Locally planar patch features for real-time structure from motion," in *Proc. Brit. Mach. Vis. Conf.*, 2004, pp. 1–10.
[42] M. Sualeh and G.-W. Kim, "Simultaneous localization and mapping in the epoch of semantics: A survey," *Int. J. Control Automat. Syst.*, vol. 17, no. 3, pp. 729–742, 2019.
[43] A. J. Davison, "FutureMapping: The computational structure of spatial AI systems," 2018, *arXiv: 1803.11288*.
[44] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An open-source library for real-time metric-semantic localization and mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1689–1696.
[45] Y. Chang, Y. Tian, J. P. How, and L. Carlone, "Kimera-Multi: A system for distributed multi-robot metric-semantic simultaneous localization and mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11210–11218.
[46] M. Runz, M. Buffier, and L. Agapito, "MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *Proc. IEEE ACM Int. Symp. Mixed Augmented Reality*, 2018, pp. 10–20.
[47] M. Grinvald et al., "Volumetric instance-aware semantic mapping and 3D object discovery," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 3037–3044, Jul. 2019.
[48] S. Zhi, M. Bloesch, S. Leutenegger, and A. J. Davison, "SceneCode: Monocular dense semantic reconstruction using learned encoded scene representations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11776–11785.
[49] D. Gálvez-López, M. Salas, J. D. Tardós, and J. Montiel, "Real-time monocular object SLAM," *Robot. Auton. Syst.*, vol. 75, pp. 435–449, 2016.
[50] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM : Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1352–1359.
[51] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level SLAM," in *Proc. Int. Conf. 3D Vis.*, 2018, pp. 32–41.
[52] T. Laidlow and A. J. Davison, "Simultaneous localisation and mapping with quadric surfaces," 2022, *arXiv:2203.08040*.
[53] K. Mazur, E. Sucar, and A. J. Davison, "Feature-realistic neural fusion for real-time, open set scene understanding," 2022, *arXiv:2210.03043*.
[54] B. Xu, A. J. Davison, and S. Leutenegger, "Learning to complete object shapes for object-level mapping in dynamic scenes," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 2257–2264.
[55] S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D object slam," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 925–938, Aug. 2019.
[56] S. Yang and S. Schere, "Monocular object and plane SLAM in structured environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3145–3152, Oct. 2019.
[57] J. Dong, X. Fei, and S. Soatto, "Visual-inertial-semantic scene representation for 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 960–970.
[58] L. Nicholson, M. Milford, and N. Sünderhauf, "QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented slam," *IEEE Robot. Automat. Lett.*, vol. 4, no. 1, pp. 1–8, Jan. 2019.
[59] D. Létourneau, F. Michaud, and J.-M. Valin, "Autonomous mobile robot that can read," *EURASIP J. Adv. Signal Process.*, vol. 2004, no. 17, pp. 1–13, 2004.
[60] X. Liu and J. Samarabandu, "An edge-based text region extraction algorithm for indoor mobile robot navigation," in *Proc. IEEE Int. Conf. Mechatronics Automat.*, 2005, pp. 701–706.
[61] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.
[62] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1994, pp. 593–600.
[63] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.
[64] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.
[65] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys. Doklady*, Soviet Union, vol. 10, no. 8, pp. 707–710, 1966.
[66] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2010, pp. 778–792.
[67] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[68] D. Zou, Y. Wu, L. Pei, H. Ling, and W. Yu, "StructVIO: Visual-inertial odometry with structural regularity of man-made environments," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 999–1013, Aug. 2019.
[69] T. Sattler et al., "Are large-scale 3D models really necessary for accurate visual localization?," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6175–6184.
[70] T. Sattler et al., "Benchmarking 6DOF outdoor visual localization in changing conditions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8601–8610.
[71] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4104–4113.
[72] C. V. R. G. at Chalmers University of Technology, "Long-term visual localization benchmark," 2019. [Online]. Available: https://www.visuallocalization.net/benchmark/

[73] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5297–5307.
[74] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-supervised interest point detection and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 224–236.
[75] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning feature matching with graph neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4938–4947.
[76] Q. Zhou, T. Sattler, and L. Leal-Taixe, "Patch2Pix: Epipolar-guided pixel-level correspondences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4669–4678.
[77] P. C. Ng and S. Henikoff, "SIFT: Predicting amino acid changes that affect protein function," *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3812–3814, 2003.

Boying Li received the BSc degree in automation engineering from Northwestern Polytechnical University, Xi'an, China, in 2016. She is currently working toward the PhD degree with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China. Her research interests include 3D vision and visual SLAM.



Danping Zou (Member, IEEE) is an associate professor with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University. He is now leading the Vision and Intelligence SYStem (ViSYS) Group, Institute for Sensing and Navigation. His research mainly focuses on 3D vision and autonomous systems.

Yuan Huang received the BSc degree in communication engineering from Huazhong University of Science and Technology, Wuhan, China, in 2019. She is currently working toward the MEng degree with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China. Her research interests include computer vision and structure-from-motion.

Xinghan Niu received the BE degree in information engineering from Shanghai Jiao Tong University, Shanghai, China, in 2020. He is currently working toward the master degree with the Department of Electronic Engineering, Shanghai Jiao Tong University. His research interests include depth map prediction and machine learning.

Ling Pei received the PhD degree from Southeast University, Nanjing, China, in 2007. From 2007 to 2013, he was a specialist research scientist with the Finnish Geospatial Research Institute, Masala, Finland. He is currently a professor with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. He has authored or coauthored more than 90 scientific articles. He is an inventor of 24 patents and pending patents. His main research is in the areas of indoor/outdoor seamless positioning, ubiquitous computing, wireless positioning, bio-inspired navigation, context-aware applications, location-based services, and navigation of unmanned systems. He was a recipient of Shanghai Pujiang Talent in 2014.

Wenxian Yu received the BS, MS, and PhD degrees from the National University of Defense Technology, Changsha, China, in 1985, 1988, and 1993, respectively. From 1996 to 2008, he was a professor with the College of Electronic Science and Engineering, National University of Defense Technology, where he served as the deputy head of the College and the assistant director of the National Key Laboratory of Automatic Target Recognition. From 2009 to 2011, he was the executive dean with the School of Electronic, Information, and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. He is currently a Yangtze River Scholar distinguished professor and the head of the research part at the School of Electronic, Information, and Electrical Engineering, Shanghai Jiao Tong University. His research interests include remote sensing information processing, automatic target recognition, and multisensor data fusion. He has published more than 200 research papers in these areas.