Robotica (2024), **42**, pp. 2619–2634
doi:10.1017/S0263574724000985

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

RGB-D visual odometry by constructing and matching features at superpixel level

Meiyi Yang[1,2], Junlin Xiong[1] and Youfu Li[2]

[1]Department of Automation, University of Science and technology of China, Hefei, China
[2]Department of Mechanical Engineering, City University of Hong Kong, Kowloon, Hongkong
Corresponding author: Junlin Xiong; Email: xiong77@ustc.edu.cn

Received: 13 March 2023; **Revised:** 14 January 2024; **Accepted:** 7 May 2024; **First published online:** 18 September 2024

Keywords: computer vision; SLAM; visual tracking; mobile robots; superpixel; robot localization

Abstract

Visual odometry (VO) is a key technology for estimating camera motion from captured images. In this paper, we propose a novel RGB-D visual odometry by constructing and matching features at the superpixel level that represents better adaptability in different environments than state-of-the-art solutions. Superpixels are content-sensitive and perform well in information aggregation. They could thus characterize the complexity of the environment. Firstly, we designed the superpixel-based feature SegPatch and its corresponding 3D representation MapPatch. By using the neighboring information, SegPatch robustly represents its distinctiveness in various environments with different texture densities. Due to the inclusion of depth measurement, the MapPatch constructs the scene structurally. Then, the distance between SegPatches is defined to characterize the regional similarity. We use the graph search method in scale space for searching and matching. As a result, the accuracy and efficiency of matching process are improved. Additionally, we minimize the reprojection error between the matched SegPatches and estimate camera poses through all these correspondences. Our proposed VO is evaluated on the TUM dataset both quantitatively and qualitatively, showing good balance to adapt to the environment under different realistic conditions.

1. Introduction

Visual odometry (VO) enables robots to perceive the surrounding environments and localize themselves and thus is essential in many industries, such as auto-driving and augmented reality. VO is generally regarded as a key component of visual-based Simultaneous Localization And Mapping (v-SLAM), which further incorporates loop closure detection and global optimization modules to alleviate the accumulated drift in a long run [1]. Compared with LiDAR- and GPS-based methods, VO estimates the trajectories of robots by analyzing the sequences of images captured. Cameras provide abundant texture information at a lower price of hardware, and the depth value of environments can be measured efficiently with the availability of RGB-D cameras, especially indoors.

In recent years, significant advancements have been made in real-time methods for SLAM and VO. Typical systems [2–6] directly operate on pixels and estimate camera poses through the correspondences between pixels. Feature-based methods [2–4] find corresponding feature points and estimate camera poses by solving PnP problem. Feature points are distinctive in texture-rich environments while their performances are generally degraded in poorer texture environments. Direct methods [5, 6], using the actual value of the measurement, are often affected by brightness changes. Deep learning-based approaches [7, 8] have shown promising results with end-to-end training, but face challenges in generalization and computational efficiency. High-level features like lines [9–11] and planes [12, 13] describe the environment structurally. Fitted by amounts of points, planes can effectively eliminate the influence of measurement noise, especially in indoor environments. Indoor environments are mostly constructed

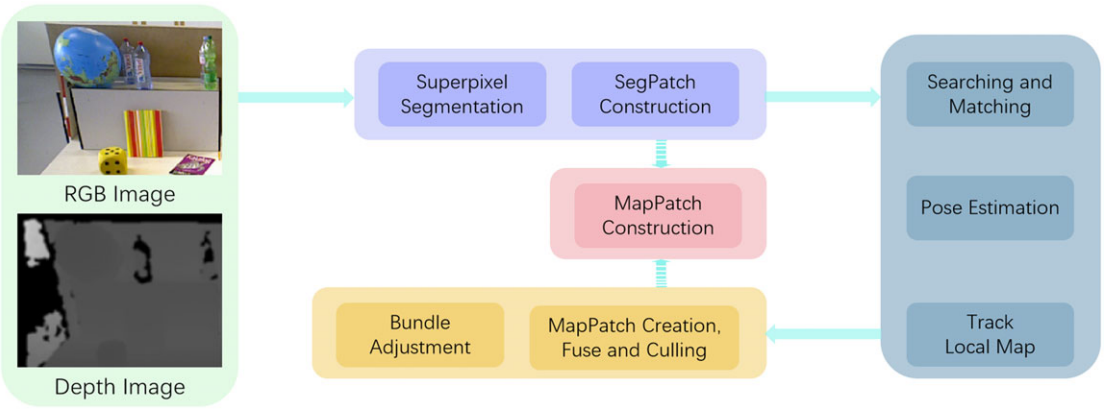

Figure 1. *Pipeline of superpixel-based visual odometry.*

by various manufactured objects and structures. Their surfaces are usually planes with limited size, contours, and edges. Planes representing indoor environment thus can be exploited to construct constraints for robust pose estimation.

In this paper, we design features at the superpixel level. A superpixel is a series of adjacent pixels with similar characteristics, including similar brightness, similar texture, and low contour energy inside the region [14]. Superpixels have good properties such as content sensitivity, high information aggregation, and edge preservation. We design SegPatch as a superpixel-based feature for scene representation instead of points and planes. SegPatch exploits the regional information of the superpixel to approximate minor details of the small structures, achieving higher computational efficiency than planes in complex environments. At the same time, SegPatch removes the spatial redundancy, making it less affected by measurement noise and more distinctive, thus offsets the lack of feature points in poor texture environments. As a result, SegPatch could robustly characterize environments with various texture densities. Correspondingly, MapPatch is designed for mapping. Structural parameters are estimated from depth measurements and updated with local maps.

We propose a superpixel-based visual odometry to achieve robust and accurate pose estimation results. Using an RGB-D camera, we decompose the RGB image into superpixels and estimate structural parameters from the depth image. SegPatches are then constructed and their corresponding MapPatches are initialized. We propose a novel matching algorithm to establish the correspondences between two consecutive frames or between frames and the local map. The similarity between SegPatches is measured by the defined distance. To improve computational efficiency, we propose a novel searching method following the idea of approximate nearest neighbor search using hierarchical navigable small world (HNSW [15]) in scale space. By constructing constraints derived from the variation in brightness, epipolar geometry, and normal consistency, the camera pose is finally estimated via all matched correspondences. The local map is established simultaneously. Figure 1 shows the framework of our proposed VO system. We provide real-world evaluation on the TUM RGB-D [16] dataset to prove the feasibility of using superpixel-based feature in VO system and show good balance of our proposed VO to adapt different environments.

In summary, the contributions of this paper are as follows:

1. We design superpixel-based feature named SegPatch. Through the approximation of small structures and the elimination of redundant information, SegPatch shows good adaptability and uniqueness in different environments.
2. We propose a novel matching algorithm for SegPatches. The efficient searching strategies and accurate similarity measurements robustly provide credible correspondences for further use.

3. We propose a superpixel-based RGB-D visual odometry and evaluate it on TUM indoor datasets, showing good adaptability in various environments and reaching comparable results with state-of-the-art solutions to VO systems.

The rest of paper is structured as follows. Section 2 provides background on related work. We first introduce the superpixel-based feature SegPatch in Section 3, then present the matching algorithm and pose estimation method in VO framework (Section 4), followed by implementations and experiments in Section 5. Section 6 concludes this paper.

2. Related works

According to different tracking and mapping method, traditional geometric-based VO can be roughly divided into feature-based and direct methods [17]. Among them, feature-based approaches are typically more robust to illumination changes than direct methods. In this paper, our proposed visual odometry is to construct and match features at superpixel level. As such, our proposed method is relevant to the existing VO methods reviewed below.

Point-feature-based methods. A number of research works using point-feature-based methods have been reported, notable examples are ORB-SLAM [2], RGBD-SLAM [18], and DVO [5]. Most of these methods have been extended to stereo or RGB-D versions such as ORB-SLAM2 [19] and DVO for RGB-D camera [20]. Point-feature-based methods generally extract and match features with strong significance and minimize geometry reprojection error. The availability of robust feature detectors and descriptors enables feature-based methods to construct correspondences even under large movement between frames [21]. Therefore, point-feature-based methods often have robust performances in rich texture scenes that involve varying illuminations, occlusions, and large viewpoint changes. However, the heavy dependence on textures impairs the robustness in poor texture environments due to the shortage of reliable feature points.

Plane and Surfel-based methods. To solve the problems of point features, planes are exploited to refine the performance in a few systems. Earlier works like [22] added planes into the extended Kalman filter state vectors. However, the growing size of the dense covariance matrix will lead to a huge computational cost, so these methods are only applicable to some small scenes. In ref. [23], a fast plane extraction and matching method for indoor mapping is proposed, and a minimal representation for infinite planes was introduced in ref. [24]. CPA-SLAM [25] modeled the environment with a global plane and used the planes for tracking and global graph optimization. Then, KDP-SLAM [26] proposed a keyframe-based approach based on dense plane extraction and matching using a projective plane association algorithm. Besides, the authors of [27, 28] used points and planes together and tried to optimize them for higher accuracy and efficiency. Structural properties indoors, such as Manhattan World Assumption, were exploited to construct relationships between planes and points [29].

Compared to extracting planes from point clouds above, surfels [30] as finite planar elements often provide more accurate surface representation, better surface fitting, and more efficient computation. DPPTAM [31] modeled the environment with 3D points in high-gradient areas and 3D piecewise planes in low-gradient areas for dense scene reconstruction. Probabilistic Surfel Map [32] was proposed to maintain a globally consistent map with both photometric and geometric uncertainties encoded. Then, UcoSLAM [33] combined the points and squared planar markers as features alleviated the relocalization problem in repetitive environments. In SP-SLAM [34], feature points and surfels were also optimized together to eliminate the effect of different texture densities on systems. MSC-VO [1] introduced structural constraints combined with estimated Manhattan axes and the reprojection errors, allowing the method to work for a wider variety of scenarios. Surfels were initialized directly from sparse planes by extracting superpixels to utilize the structural information indoors in ManhattanSLAM [35].

Learning-based methods. The great progress of deep learning stimulates the learning-based VO by predicting the relative poses between images with supervised [36] or unsupervised learning [37].

DeepVO [36] employed an RNN to predict camera pose end-to-end from input image sequence, and UndeepVO [37] concurrently trained a pose net and a depth net to form CNN-based VO systems. DVSO [38] proposed a virtual stereo term that incorporates the depth estimation from a semi-supervised network into a direct VO pipeline. D^{2}VO [7] presented a novel deep learning and direct method-based monocular visual odometry, and D3VO [8] exploited deep networks on three levels – deep depth, pose, and uncertainty estimation. However, although learning-based VO has advantages in feature extraction and robustness, it also suffers from high computational demands and generalization.

In this article, we propose a new VO method in which SegPatch is used as superpixel-based feature. SegPatch compensates for the shortcomings of points and surfels in representing various environments. Moreover, our superpixel-based method also holds potential use of learning-based methods, given its ability to extract informative and compact feature descriptors from images. A detailed explanation of superpixel-based VO is given in the following sections.

3. Concepts and model formulation

This section explains fundamental concepts that are necessary to follow the remainder of the latter. Our method is the first VO system using the superpixel-based features. Thus we design the superpixel-based feature SegPatch as basis.

3.1. Superpixel

Superpixels are sub-regions of the over-segmented image with similar information, which is sensitive to the variance of texture density. By SLIC [39] algorithm, superpixels are initialized into regular grids. Then, the pixels are clustered locally. In order to keep the central intensity gradient fluctuating within a specific range, we set upper and lower limits for merging and splitting. Iterations are repeated until every pixel is classified stably. An image I is finally decomposed into M superpixels $I = \{S_i, i = 1, \ldots, M\}$, where $\forall S_i, S_j \in I, i \neq j, S_i \bigcap S_j = \emptyset$. For a region $\Omega \in I$, the higher texture density in Ω, the more superpixels generated, and vice versa. Thus, the distribution of superpixels with different sizes reflects the complexity of the environment. Due to the intra-region similarity and inter-region dissimilarity, superpixels only maintain the major information of sub-regions, ensuring their effectiveness in various environments.

3.2. SegPatch and MapPatch

Because of the irregular and non-stable shape, the superpixel is insufficient to provide a robust image descriptor. We propose SegPatch as the superpixel-based features, extended from SuperPatch [40]. A SegPatch $\mathbf{S}_i$ is centered at the superpixel S_i and is combined with its neighboring superpixels $\{S_{i'}\}$ such that $\mathbf{S}_i = \{S_{i'} | \|c_i - c_{i'}\|_2 \leq \mathcal{R}\}$, where c_i is the spatial barycenter and $\mathcal{R}$ is the parameter based on scale factor and superpixel granularity. We use $\mathcal{R}$ to control the number of neighboring superpixels, in order that every SegPatch has the similar size topologically.

The SegPatch is then constructed as a dual descriptor $\mathbf{S}_i = \{F_{S_i}, \mathbf{X}_{S_i}\}$, including inner-feature F_{S_i} and the inter-features $\mathbf{X}_{S_i}$. Inner-feature $F_{S_i} = \{\mathbf{p}_i, \mathbf{c}_i, r_i, s_i\}$ contains the information of the central superpixel, where $\mathbf{p}_i = (u_i, v_i)^T$ is the coordinates of the barycenter, $\mathbf{c}_i = (l_i, a_i, b_i)$ is mean color in LAB color space, πr_i^2 represents size of the central superpixel, and s_i the scalar. Inter-features $\mathbf{X}_{S_i} = \{X_{S_{i'}}\}_{i'=1,2,\ldots m}$ describe the relationship between the center superpixel and the neighboring superpixels. For every neighboring element, $X_{S_{i'}} = (u_i - u_{i'}, v_i - v_{i'})$ is a vector pointing to the center superpixel, where $(u_{i'}, v_{i'})$ is the barycenter of the neighboring superpixel. Let $I(S_i)$ be the mean intensity of the superpixel, the gradient of the SegPatch

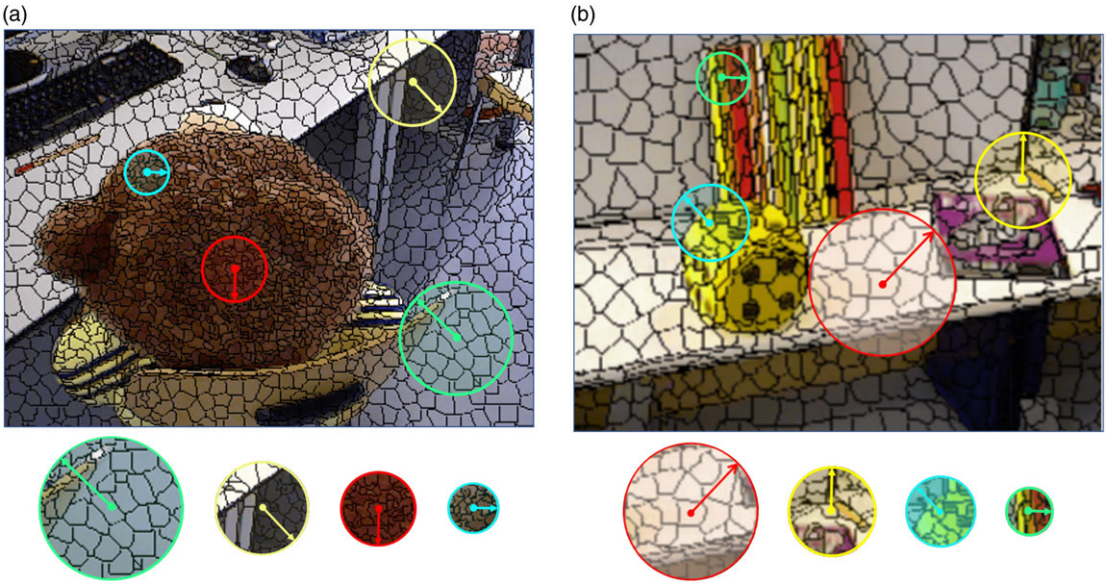

Figure 2. *SegPatch construction. Superpixels are generated into different sizes in different texture regions. The SegPatch is constructed on the neighborhood with a topological radius $\mathcal{R}$. The SegPatch is described by a dual descriptor, including inner- and inter-information of the center superpixel. The arrow shows the direction of the SegPatch.*

$$\begin{cases} g_x = \sum_{S_{i'} \in \mathbf{S}_i} (I(S_{i'}) - I(S_i)) \cdot (u_{i'} - u_i) \\ g_y = \sum_{S_{i'} \in \mathbf{S}_i} (I(S_{i'}) - I(S_i)) \cdot (v_{i'} - v_i) \end{cases} \tag{1}$$

is a holistic character to describe the direction of change in a small region.

In 3D space, we further construct the MapPatch for scene reconstruction. MapPatch is described as $\Sigma_M = [\{P_i\}_M, \mathbf{n}_M, R_M]$, maintaining the index of the corresponding SegPatches at the same time. $\{P_i\}_M$ is the set of control points in the world coordinate, where P_i is initialized from the barycenter of SegPatch $\mathbf{S}_i$, that is $P_i = K^{-1}\mathbf{p}_i$. SVD method is used to initialize the normal $\mathbf{n}_M$ from the depth data. The radius $R_M = 2r * z/(f_x + f_y)$ is also initialized so that the projection of the MapPatch can cover the corresponding superpixel in the observed frame. When a MapPatch is observed from different perspectives, the number of its control points increases, and the normal $\mathbf{n}_M$ is fine-tuned by minimizing a fitting error defined as

$$E_{\mathbf{n}_M} = \sum_i L_\delta(\mathbf{n}_M \cdot (P_i - \bar{P}_M) + b) \tag{2}$$

where $\bar{P}$ is the mean of the control points P_i and b estimates the bias. R_M^2 is always the minimal size from $\bar{P}$ to cover all the corresponding superpixels in different frames.

The SegPatch construction is indicated in Figure 2. The definitions of SegPatch and MapPatch fully exploit the sensitivity to the texture density of superpixels. SegPatch could adaptively adjust its size in areas with different texture densities, ensuring its robustness in various environments. In the poor texture region, SegPatch contains a larger region. The saliency of SegPatch is greatly improved by removing the image redundancy. By using structure parameters, MapPatch models the environment structurally. In areas with rich textures, superpixels degenerate towards individual pixels. SegPatch, as a superpixel-based feature, assures uniqueness and robustness due to the inclusion of local information.

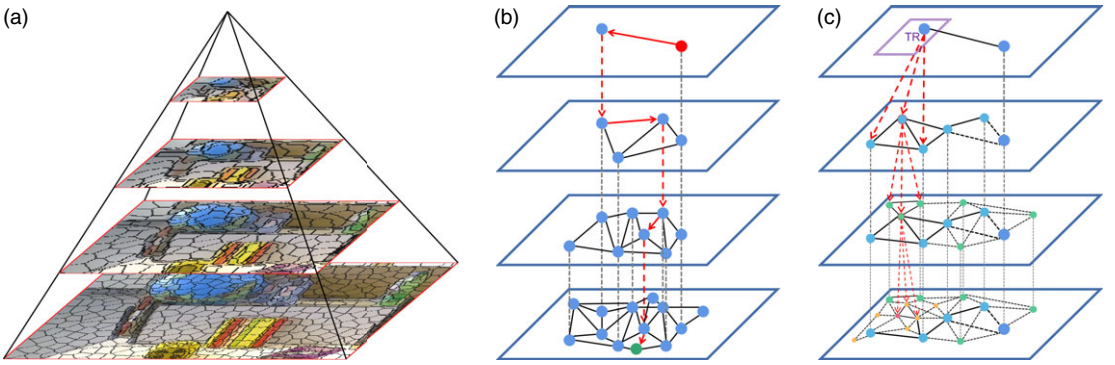

Figure 3. *Instead of extracting superpixels in different sizes, the use of Gaussian down-sampling can better reflect the distance of observation. (a) Shows the image pyramid constructed for superpixel segmentation. We introduce the idea of HNSW algorithm (b) for fast searching and matching through the image pyramid. (c) Shows the process of the proposed matching algorithm. Different from the original HNSW algorithm, the corresponding node in the next layer is not the best-matching node in this layer, but a new node split from it in the scale space.*

Drawing from the small SegPatch, the accuracy of MapPatch does not suffer from the approximation. In contrast, the accuracy of the reconstruction is guaranteed due to the fine-grained structure of the fragments. Additionally, as a regional feature, SegPatch is also not sensitive to image noise and motion blur.

3.3. Scale space and image pyramid

SegPatch needs to be constructed at different scales because searching for correspondences requires comparing images seen at different scales. The image pyramid often represents the scale space. The images are repeatedly smoothed with a Gaussian filter and then sub-sampled to achieve a higher level of pyramid. The down-sampling procedure reflects the variation of the observed distance. We extract the superpixels at each layer of the pyramid and distribute them to the superpixels at the next higher layers. Superpixels at the same level distributing to the same superpixel are brother nodes of each other and are children of the one distributed. Thus the pyramid is a fully connected hierarchical graph in the form of a multiple-tree. The construction of the image pyramid is shown in Figure 3a.

After constructing an image pyramid and extracting superpixels for multiple images of different scenes and views, we find that the number of superpixels decreases exponentially from low to high layers, and the sizes of superpixels segmented in the same scale follow the same exponential distribution. The spatial distribution of the number of superpixels between layers is also uniform. The pyramid could thus tell rich texture regions from poor texture regions.

4. Superpixel-Based visual odometry

The structure of the proposed superpixel-based visual odometry is shown in Figure 1. The system starts with RGB-D image preprocessing, in which images are segmented into superpixels, SegPatches are constructed in scale space and MapPatches are initialized. Searching and matching procedure provides correspondences between two consecutive frames or between frames and the local map. Finally, the camera pose is estimated by minimizing reprojection errors between SegPatch measurements and observations. The local map is also updated in separate thread. This section describes the key methods of the proposed VO in detail.

4.1. SegPatch comparing, searching and matching

For two SegPatches, the number of elements and geometry are generally different, so it is difficult to compare directly. To measure the similarity between two SegPatches $\mathbf{S}_i^A$ and $\mathbf{S}_j^B$ in different images I_A and I_B, the distance is defined as follows.

$$D\left(\mathbf{S}_i^A, \mathbf{S}_j^B\right) = \frac{\sum_{S_{i'}^A} \sum_{S_{j'}^B} \omega\left(S_{i'}^A, S_{j'}^B\right) d\left(F_{i'}, F_{j'}\right)}{\sum_{S_{i'}^A} \sum_{S_{j'}^B} \omega\left(S_{i'}^A, S_{j'}^B\right)} \tag{3}$$

where d is the Euclidean distance between the inner features. Let weight ω represents the relative overlapping area of neighboring superpixels $S_{i'}^A$ and $S_{j'}^B$, following Gaussian distribution symmetrically

$$\omega\left(S_{i'}^A, S_{j'}^B\right) = \exp\left(-x_{i'j'}^T x_{i'j'}/\sigma_1^2\right) \varpi\left(S_{i'}^A\right) \varpi\left(S_{j'}^B\right) \tag{4}$$

where $x_{i'j'} = (X_{S_{i'}} - X_{S_{j'}})/s_i s_j$ is the relative distance between $\mathbf{p}_{i'}$ and $\mathbf{p}_{j'}$, $\varpi(S_{i'}^A)$ weights the influence of $S_{i'}^A$ according to its distance to S_i^A as $\omega(S_{i'}^A) = \exp\left(-|X_{S_{i'}}|^2/(s_i\sigma_2)^2\right)$. $\varpi(S_{i'}^B)$ is as the same. s_i, s_j are scale coefficients, and σ_1, σ_2 are two control parameters depending on the superpixel segmentation. Depending on the superpixel deposition scale, $\sigma_1 = \frac{1}{2}\sqrt{N/M}$ for an image with N pixels decomposed into M superpixels is set as half of the average distance between superpixel barycenters. Depending on the SegPatch size, $\sigma_2 = \sqrt{2}\mathcal{R}$ is set to weight the contribution of closest superpixels.

The matching task in VO consists of matching between two consecutive frames (2d-2d), as well as matching between frames and the local map (2d-3d). PatchMatch (PM) [41] is a classical method to compute pixel-based patch correspondences between two images. As an extension of PM, SuperPatchMatch (SPM) [40] provides correspondences of irregular structures from superpixel decompositions. These two methods share one key point that good correspondences can be propagated to the adjacent patches within an image. However, it takes a long time in the propagation step to find the optimal solution.

Instead of processing images in scan order in PM and SPM, our proposed matching algorithm exploits the graph search method for fast-searching candidates. We propose a new modification of the HNSW algorithm [15] and demonstrate it to be applicable. HNSW is an approximate K-nearest neighbor search method based on navigable small-world graphs with controllable hierarchy. The image pyramid satisfies the four requirements of the implementation of the HNSW algorithm: (1) The construction of the image pyramid is a hierarchical graph; (2) The number of upward nodes (superpixels) decreases following the exponential decay; (3) The highest projection layer of a node is obtained by the exponential decay probability function based on its position and size; (4) The node could be found in all layers from the highest layer down. Furthermore, due to the assumption that the image sequences are consecutive and the motion is not so fast, we set a trust region for optimal selection. A candidate located in the trust region is with higher probability to be the best match. The search process is performed layer-by-layer from coarse to fine in both scale and position. The HNSW and our proposed method are shown in Figure 3.

The following steps are then performed to find and improve the correspondences between frames. For the sake of clarity, we assume the N-layer image pyramid with the top I_0, the current best match of a SegPatch $\mathbf{S}_i^A \in I^A$ in I_k^B, is donated as $\mathcal{M}_k^B(i)$, storing the distance and the index of their corresponding SegPatch. First of all, for every SegPatch $\mathbf{S}_i^A \in I^A$, we delineate a trust region in I_0^B according to the position and size of $\mathbf{S}_i^A$. Matching candidates are selected from the trust region. After similarity measurement and comparison, the best match $\mathcal{M}_0^B(i)$ is determined. Then, the children of $\mathcal{M}_0^B(i)$ are obtained to be the candidates of the next layer. Several other SegPatches in the trust region are also selected randomly to escape from possible local minimum. Repeat the process until the bottom of the pyramid. Instead of updating the most corresponding SegPatch when traversal processes, obtaining the best match in every layer individually assures the scale stability. Generally, due to the small motion between two consecutive frames, there will not be much change of scales between two corresponding SegPatch. But sometimes, most SegPatches scale up or down when the camera moves forwards or backends. Similarly with the

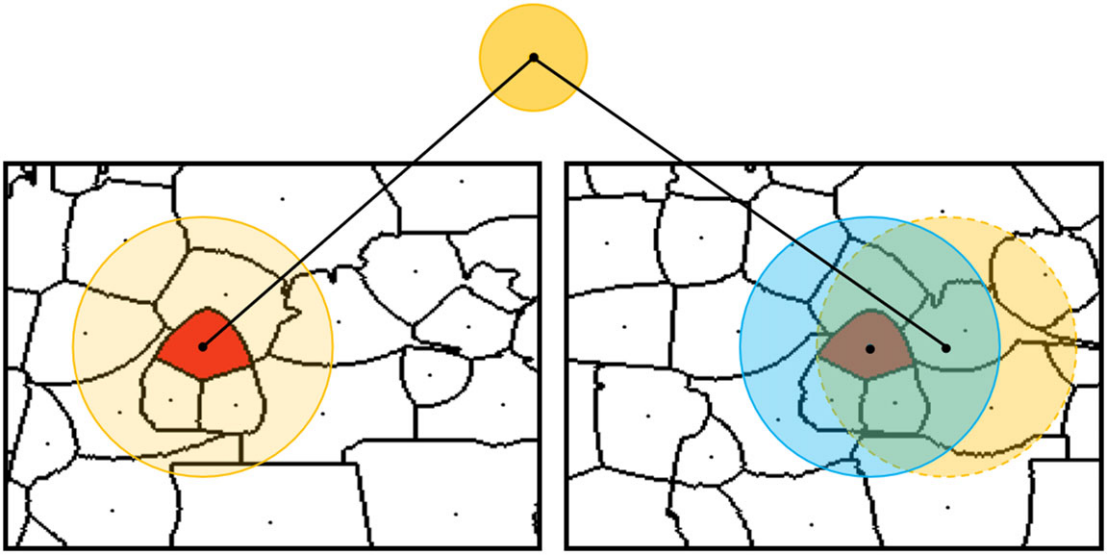

Figure 4. *Projection relationships between SegPatch and MapPatch. When a MapPatch is projected onto the image, there may be a small deviation from the correct corresponding SegPatch. Thus, the trust region is set smaller to find the best match.*

feature points, the consistency of variation for the intensity orientation is checked, and the incompatible matches will be removed in the end.

The 2d-3d matching is used for constructing a stronger connection with the local map. It is a guided match because 2d-3d matching is performed when there is already an estimated or predicted camera pose. Following the principle of geometry, the MapPatch is reprojected onto the current image. As shown in Figure 4, the projected position and size of the MapPatch may not coincide with the optimal SegPatch exactly, but near. Compared to 2d-2d matching, a smaller trust region is required here to find the nearest SegPatch. Matching candidates are also selected through an image pyramid, where the similarity between the latest corresponding SegPatch and the ones within the trust region is measured. Benefiting from the motion priors, the correspondences are constructed with high probability.

The proposed matching algorithm is well applicable to different cases. It is worth noting that our distance calculation method does not focus on the difference in size and shape between the central superpixels, but rather on the similarity in the properties and geometry inside the SegPatch. As a result, the accuracy of the similarity measure is not affected by the observed deformation from multi-view, motion blur, and variance of texture density. The graph search method based on image pyramid reduces the computational complexity. In summary, our proposed matching algorithm accurately provides robust correspondences with a small computational load. Figure 5 shows part of the matches in different intensity density, illustrating the robustness of our method.

4.2. Pose estimation

The camera pose is finally estimated through all these correspondences. Relative motion estimation is ideally performed by maximizing the overlapping area between the matched superpixels in two images.

$$\xi^* = \max_{\xi} \sum_{\substack{S_A \in \mathbf{S}_A \\ S_B \in \mathbf{S}_B}} \frac{T_\xi(S_A) \bigcap S_B}{T_\xi(S_A) \bigcup S_B} \tag{5}$$

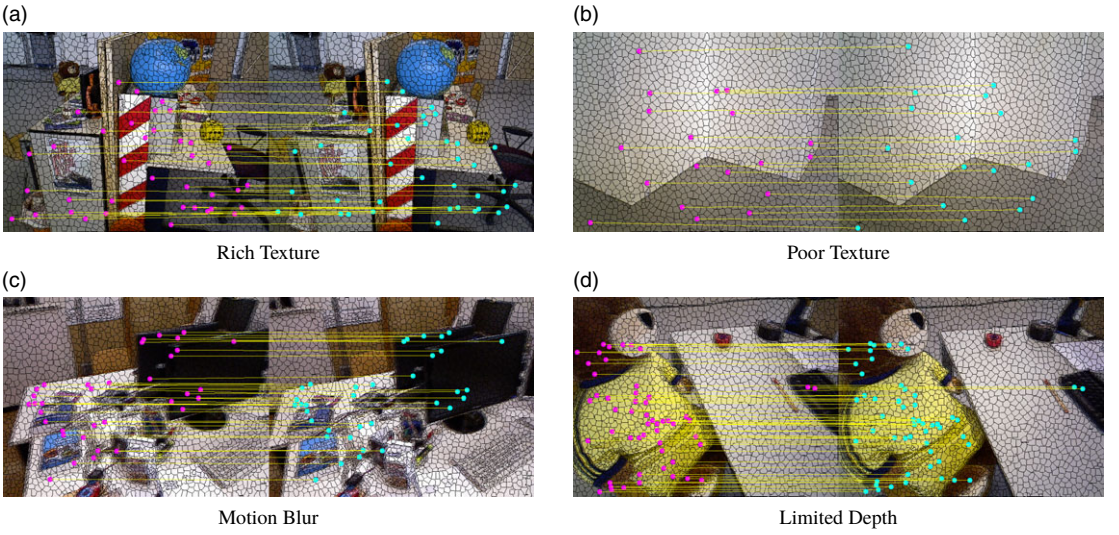

Figure 5. *Matches in different cases.*

However, due to the irregular shape of the superpixels, this computation requires the expensive count of overlapping pixels, which cancels the computational advantages of the superpixel representation. For this reason, we simplify the problem to find a suboptimal solution.

Considering two intersecting circles with the same radius $R_\circ$ on the plane, their Intersection over Union (IoU) is a nonlinear function of radius $R_\circ$ and center distance $d_\circ$:

$$\phi_{IOU}(R_\circ, d_\circ) = \frac{\arccos\left(\frac{d_\circ}{2R_\circ}\right) R_\circ^2 - \frac{d_\circ}{2}\sqrt{R_\circ^2 - \left(\frac{d_\circ}{2}\right)^2}}{\left(\pi - \arccos\left(\frac{d_\circ}{2R_\circ}\right)\right) R_\circ^2 + \frac{d_\circ}{2}\sqrt{R_\circ^2 - \left(\frac{d_\circ}{2}\right)^2}}, \quad d_\circ \in [0, 2R_\circ)$$

Let $k = \frac{d_\circ}{2R_\circ}$, $\phi_{IOU}(R_\circ, d_\circ)$ could be written as

$$\phi_{IOU}(R_\circ, d_\circ) = \phi_{IOU}(k) = \frac{\arccos(k) - k\sqrt{1-k^2}}{\pi - \arccos(k) + k\sqrt{1-k^2}}, \quad k \in [0, 1)$$

To simplify the computation, $\phi_{IOU}(k)$ could be further approximated as a quadric function of $d_\circ$

$$\phi_{IOU\_appro}(d_\circ) = (1-k)^2 = \left(\frac{2R_\circ - d_\circ}{2R_\circ}\right)^2, \quad d_\circ \in [0, 2R_\circ)$$

Numerical results show that our proposed approximation scheme is efficient. More generally, if there are two circles with different radius $R_{\circ 1}$ and $R_{\circ 2}$, the IoU is approximated as

$$\phi_{IOU-appro}(d_\circ) = \left(\frac{R_{\circ 1} + R_{\circ 2} - d_\circ}{2 * \max(R_{\circ 1}, R_{\circ 2})}\right)^2 + c, \quad d_\circ \in [|R_{\circ 1} - R_{\circ 2}|, R_{\circ 1} + R_{\circ 2})$$

As shown in Figure 6, numerical experiments show that when the radii of two circles are similar, the accuracy of the approximation is within an acceptable range for the following use. Given that there is generally no significant scale variation between consecutive images, corresponding SegPatches tend to

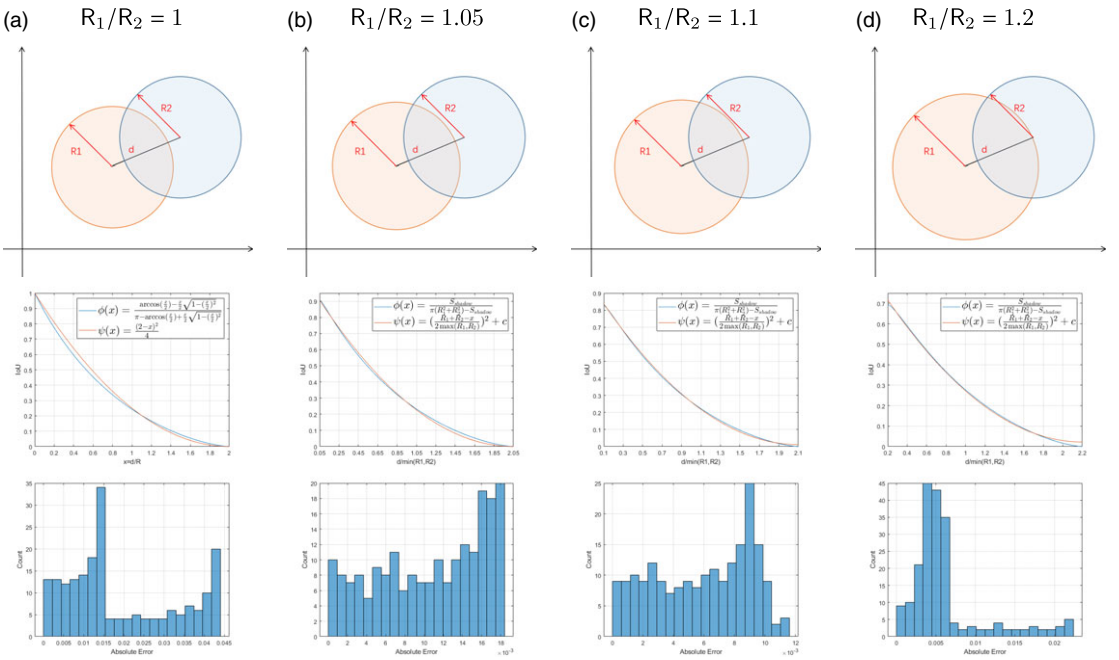

Figure 6. *Fitting function for IoU of two intersecting circles on the plane. In the second row, we show the comparison between our proposed fitting function and the original IoU curve. $\phi(x)$ shows the true IoU of the intersecting circles and $\psi(x)$ is the fitting function. The x-axis represents* $\mathrm{d}/\min(\mathrm{R}_1, \mathrm{R}_2)$*, with* $\mathrm{R}_1/\mathrm{R}_2 = 1, 1.05, 1.1, 1.2$ *for example. The third row represents the error histogram of the fitting function. Experiments show that the maximum errors of the fitting functions are 0.0438, 0.0182, 0.0114, 0.0222, the mean errors are 0.0192, 0.0107, 0.0114, 0.0062, the variance of errors are 1.9073e-4, 3.2020e-5, 1.0026e-5, 2.3721e-5 respectively. The numerical experiment demonstrated that the quadric function $\psi(x)$ could represent the IoU approximately well when two circles are of similar size.*

have similar sizes. Thus, fitting a complex nonlinear function by a simple quadric function is feasible to obtain a suboptimal IoU solution. The Equation (5) is simplified as

$$\begin{aligned}
\xi^* &= \arg\max_{\xi} \sum_{\substack{S_A \in \mathbf{S}_A \\ S_B \in \mathbf{S}_B}} \left(\frac{r_1 + r_2 - \|T_\xi(x_A) - x_B\|_2}{2\max(r1, r2)} \right)^2 \\
&\Rightarrow \arg\max_{\xi} \sum_{\substack{S_A \in \mathbf{S}_A \\ S_B \in \mathbf{S}_B}} \left(\frac{r1 + r2}{2\max(r1, r2)} - \frac{\|T_\xi(x_A) - x_B\|_2}{2\max(r1, r2)} \right)^2 \\
&\Rightarrow \arg\min_{\xi} \sum_{\substack{S_A \in \mathbf{S}_A \\ S_B \in \mathbf{S}_B}} \left(\frac{\|T_\xi(x_A) - x_B\|_2}{2\max(r1, r2)} \right)^2 \\
&\Rightarrow \arg\min_{\xi} \sum_{\substack{S_A \in \mathbf{S}_A \\ S_B \in \mathbf{S}_B}} \mu_{AB} \|T_\xi(x_A) - x_B\|_2^2
\end{aligned} \tag{6}$$

where $\mu_{AB} = 1/(2\max(r1, r2))^2$ is the weight parameter depending on the size of matched SegPatch pair. Thus, the problem is degraded approximately by minimizing the relative distance of superpixel coordinates. The weighted least square problem could be solved by Gauss-Newton or Levenberg-Marquardt algorithms.

4.3. Local mapping

The local map is updated simultaneously to provide the optimal reconstruction in the surroundings of the camera pose. Newly observed MapPatches are added. A MapPatch is initialized by one frame, but could be observed by others. Therefore, it is reprojected onto the other active frames, and 2d-3d correspondences are searched as detailed above. MapPatch observed from multi-view are fused. By gathering the information from SegPatches in different frames, MapPatch maintains the key information of control points, normal and size. As the camera moves, invisible MapPatches are removed. The local mapping is also in charge of culling redundant frames. Only frames active in the current sliding window and their corresponding MapPatches with high quality are preserved in the local map. The global map is then generated by fusing and optimizing the local map. The proposed superpixel-based visual odometry is processed with continuous pose estimation and mapping.

5. Experiments

We evaluate the accuracy and efficiency of the matching algorithm on several sequences from the publicly available TUM RGB-D dataset [16] and evaluate the localization performance against state-of-the-art techniques. Our method is practicable in these real-world image sequences under different realistic conditions like image noise, motion blur, poor focus, dynamic lighting, and other difficulties. The proposed method was implemented with *C++* code on a standard Linux computer with four cores at 2.50 GHz and 8 GB of RAM.

5.1. Matching accuracy and efficiency

For each image in a sequence, we construct a 4-layer image pyramid with a scale factor $\beta = 2$. After superpixel segmentation, SegPatch is generated from every superpixel with effective depth and convex polygon.

We evaluate the matching accuracy by the absolute distance between the centers of two matched superpixels and their average overlap in two successive images. The degree of overlap could be approximately quantified as the relative distance $\gamma = \frac{\|c_1 - c_2\|_2}{r_1}$, where πr_1^2 is the size of central superpixel and c_1 is the ideal location of the SegPatch, c_2 is barycenter of the matched SegPatch. We counted more than 16 thousand groups of 2d-2d matched superpixels in images randomly selected in different sequences. Figure 7 shows the displacement for every matched superpixel. Most matched SegPatches are offset by 10 pixels from the ground truth, 2.5 superpixels at superpixel level, shown in Figure 7a and Figure 7b, respectively. The average displacement distribution is shown in Figure 7c, demonstrating regional accuracy.

In Figure 8, we show the average relative distance with respect to the size of the trust region. Figure 8a is the direct statistical distribution for all superpixels, and Figure 8b is the relationship by Gaussian fitting. It illustrates that a balanced trust region size provides higher matching accuracy. In the areas with poor texture, a large trust region will avoid the matching algorithm falling into local optimum, while in the areas with rich texture, a small trust region will protect the accuracy of the algorithm from influence of noise. Therefore, setting the size of the trust region according to the size and scale of SegPatch helps to provide accurate correspondences efficiently.

As for computational efficiency, our approach incorporates superpixels as the fundamental processing units for input images, involving an additional step of superpixel segmentation. Considering size of the image to be MN and the number of features extracted is K, we examine two aspects: feature extraction and matching to analyze the computational complexity. On the one hand, for computational efficiency of feature extraction, we compare the time complexity of our utilized superpixel segmentation and SegPatch extraction method with traditional feature-based methods. Point-feature-based SLAM methods, such as ORB-SLAM [2], exhibit a time complexity of $O(MN)$ for feature extraction. Plane-based feature extraction methods using AHC [42] or ICP [43] algorithms have higher time complexities, such as $O((MN)^2)$

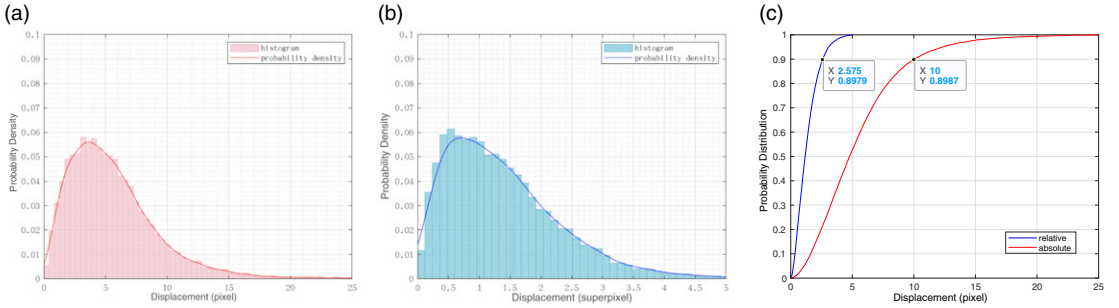

Figure 7. *Displacement for every superpixel of performing proposed matching algorithms. The absolute displacement (a) shows most of the matched superpixels are offset less than 10 pixels or say no more than 2.5 equal-size superpixels shown in (b). The average displacement distribution (c) confirms the point again.*

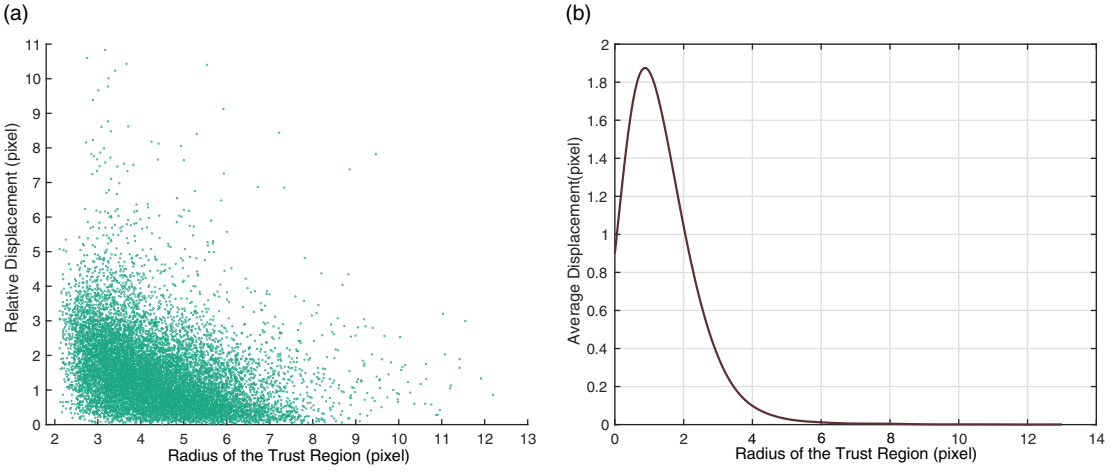

Figure 8. *Influence of radius r for relative displacement (a) in the proposed SegPatch matching algorithms. The average displacement (b) between each superpixel and one of its closest matches in the successive images.*

and $O((MN)^3)$, respectively, for plane extraction. The complexity of plane-based method will rise further in high-texture regions. Our utilized superpixel segmentation and SegPatch extraction have a time complexity of $O(MN + K)$, so no higher complexity is introduced in the feature extraction process. On the other hand, for computational efficiency of matching process, we compare the time complexity of our proposed 2d-2d matching algorithm with SPM [40]. SPM utilizes the assumption that when a patch in I_A corresponds to a patch in I_B, the adjacent patches in I_A should also match adjacent patches in I_B. All SuperPatches are processed according to a scan order to search and propagate through patch-based approximate nearest neighbor (ANN). The time complexity of SPM is $O(K^2)$. Due to the uncertainty of random assignment initialization and random search, it takes time to propagate and improve the performance. The running time grows exponentially with the number of iterations. In our algorithm, we build a trust region and search the image pyramid top-down, exploiting the consistency and continuity of the shifts in a single image. It does reduce the search time for matching and increases efficiency. Thus the computational time is reduced to $O(K \lg K)$. Therefore, considering the aforementioned aspects, the overall time complexity of our algorithm is given by $O(MN + K + K \lg K)$, indicating its superior efficiency in terms of computation time.

Table I. *Computational cost and accuracy for matching process.*

	Time Cost(s/frame)	Accuracy(%)
SPM(Matlab)	5968.1	91.7
SPM(*C*++)	1055.7	80.7
ours	1.0408	93.3

Table II. *RGB-D odometry benchmark ATE RMSE (m).*

Sequence	ours	ORB-SLAM [19]	SP-SLAM [34]	DVO-SLAM [5]	PP-SLAM [28]	Elastic Fusion [44]	D2VO [7]	DeepVO-Adapt [45]	SAVO [46]
fr1/desk	0.025	**0.016**	0.051	0.021	0.026	-	0.186	-	-
fr1/room	0.064	0.047	-	**0.043**	-	0.068	0.285	-	–
fr2/desk	0.039	**0.009**	0.042	0.035	0.025	0.071	-	0.158	0.402
fr3/str_notex_far	0.039	0.142	0.029	0.105	0.089	**0.027**	-	0.104	0.216
fr3/str_notex_near	0.024	0.033	0.0305	**0.012**	0.024	0.113	-	0.207	0.204
Average (rich texture)	0.042	**0.024**	0.047	0.033	0.0255	0.053	0.236	0.158	0.402
Average (poor texture)	0.032	0.0875	**0.0298**	0.0585	0.0565	0.070	-	0.156	0.21
Average (All)	**0.0384**	0.0409	0.0401	0.0498	0.041	0.0598	0.236	0.156	0.274

Here, we exclusively concentrate on the comparison of runtime efficiency for methods at superpixel level. The computational cost and the accuracy comparison results with SPM are shown in Table I, where the accuracy is measured by the percentage of matched features within 2.5 equal-size superpixels compared to the ground truth. Computational time and accuracy are given for each frame in a single thread, with the same initial region size and ruler used for superpixel extraction. SPM results are obtained with $k = 20$ ANN and $R = 50$ neighboring pixels. We show the result of the open-source Matlab version and rewrite it in C++ format for fair comparison. In our method, the matching candidates are selected from the image pyramid within the trust region. And in practice, we do not perform matching on every superpixel in the image. To efficiently triangulate and obtain MapPatches after obtaining correspondences, only SegPatches with valid depth measurements are operated. Multithreaded implementations have also been used to shorten computation time (around 0.02s in practice) to ensure real-time operation of the system. Experiments show that our proposed matching algorithm is efficient and accurate.

5.2. Pose estimation accuracy

We choose some representative sequences from the public TUM-RGBD datasets to evaluate our system. `fr1/desk`, `fr1/room`, `fr2/desk` represent the indoor environment with rich texture intensity, while `fr3/str_notex_far` and `fr3/str_notex_near` represent poor texture scenes indoors. We follow the same evaluation protocol as in ref. [16] by computing the maximum drift (deviation from ground-truth pose) across the whole sequence in translation and rotation (represented in quaternion) of camera motion for all trajectories. We compared the estimated camera pose with ground-truth trajectories by computing relative trajectory root-mean-square errors (RMSE) and absolute trajectory (AT) RMSE in various challenging scenes. After projecting the local MapPatches onto the current frame by the estimation from 2d-2d matching results, the pose estimation could limit the relative translation error to around 2.16 cm and rotation error below 1.05 degrees. For most sequences, we achieve comparable results with most mainstream VO algorithms, meaning that our method can maintain correct camera trajectories in challenging scenes. Table II shows the results. It is worth noting that although our method may not be

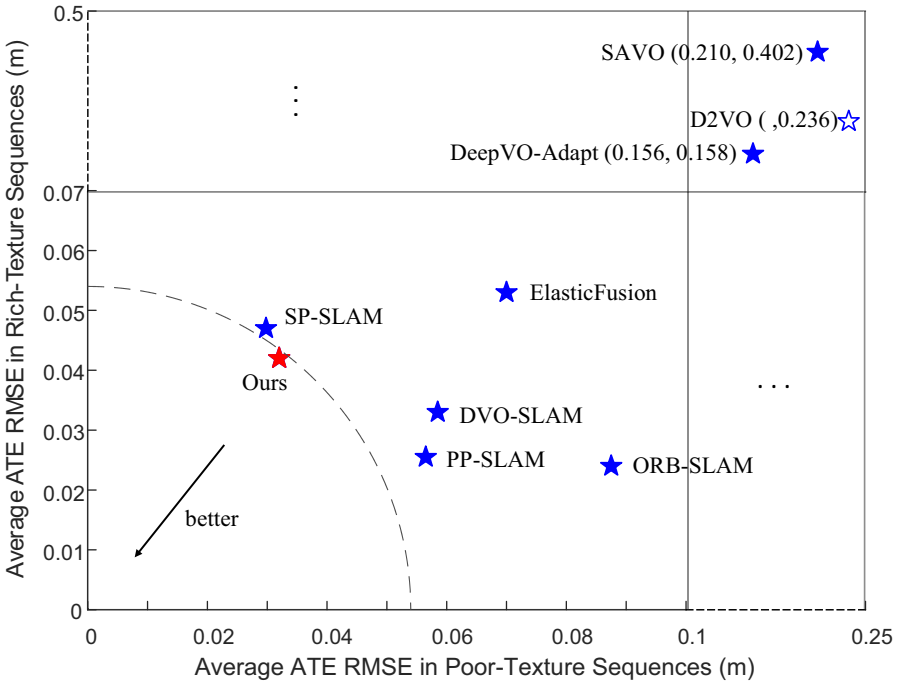

Figure 9. *Compared with other typical VO systems, our visual odometry achieves a balanced effect in environments with different texture density.*

the best under one condition of single texture density, it shows good balance to adapt to the environment shown in Figure 9. Additional work, such as global bundle adjustment and loop closure, is left for future work to become more competitive.

6. Conclusion

We have presented a novel attempt at superpixel-based visual odometry for motion estimation. The design of the superpixel-based feature SegPatch is essential to scene representation. The searching and matching algorithm could construct reliable correspondences of SegPatch in high efficiency. The proposed visual odometry operates robustly under various conditions such as different texture densities, image noise, and motion blur. We have provided an evaluation of the quality of matching and pose estimation on a variety of real-world datasets to show how our method works and demonstrate the utility of SLAM.

Author contributions. All authors contributed to the study conception, design, and implementation. Material preparation, data collection, and analysis were performed by Meiyi Yang. The original draft of the manuscript was written by Meiyi Yang and reviewed and edited by Junlin Xiong and Youfu Li. All authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Financial support. This work was supported by National Natural Science Foundation of China under Grant 62273320 and 62173286.

Competing interests. The authors declare no competing interests exist.

Ethical approval. Not applicable.

References

[1] J. P. Company-Corcoles, E. Garcia-Fidalgo and A. Ortiz, "MSC-VO: Exploiting Manhattan and structural constraints for visual odometry," *IEEE Robot Autom Lett* **7**(2), 2803–2810 (2022).

[2] R. Mur-Artal, J. M. Montiel and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans Robot* **31**(5), 1147–1163 (2015).

[3] H. Liu, C. Li, G. Chen, G. Zhang, M. Kaess and H. Bao, "Robust keyframe-based dense SLAM with an RGB-D camera" (2017). arXiv preprint arXiv: 1711.05166, 2017.

[4] Y.-T. Wang and G.-Y. Lin, "Improvement of speeded-up robust features for robot visual simultaneous localization and mapping," *Robotica* **32**(4), 533–549 (2014).

[5] C. Kerl, J. Sturm and D. Cremers, "Robust Odometry Estimation for RGB-D Cameras," **In:** *IEEE International Conference on Robotics and Automation 2013*, (IEEE, 2013) pp. 3748–3754.

[6] J. Engel, V. Koltun and D. Cremers, "Direct sparse odometry," *IEEE Trans Pattern Anal* **40**(3), 611–625 (2017).

[7] Q. Jia, Y. Pu, J. Chen, J. Cheng, C. Liao and X. Yang, "D2VO: Monocular Deep Direct Visual Odometry," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (IEEE, 2020) pp. 10158–10165.

[8] N. Yang, L. v. Stumberg, R. Wang and D. Cremers, "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry," **In:** *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (IEEE, 2020) pp. 1278–1289.

[9] F. Taubner, F. Tschopp, T. Novkovic, R. Siegwart and F. Furrer, "LCD-Line Clustering and Description for Place Recognition," **In:** *2020 International Conference on 3D Vision (3DV)*, (IEEE, 2020) pp. 908–917.

[10] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu and F. Moreno-Noguer, "PL-SLAM: Real-time Monocular Visual SLAM with Points and Lines," **In:** *Proceedings - IEEE International Conference on Robotics and Automation*, (IEEE, 2017) pp. 4503–4508.

[11] L. Zhao, S. Huang, L. Yan and G. Dissanayake, "A new feature parametrization for monocular SLAM using line features," *Robotica* **33**(3), 513–536 (2015).

[12] S. Yang, Y. Song, M. Kaess and S. Scherer, "Pop-up SLAM: Semantic Monocular Plane SLAM for Low-texture Environments," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2016*, (IEEE, 2016) pp. 1222–1229.

[13] S. Yang and S. Scherer, "Monocular object and plane SLAM in structured environments," *IEEE Robot Autom Lett* **4**(4), 3145–3152 (2019).

[14] X. Ren and J. Malik, "Learning a Classification Model for Segmentation," **In:** *Proceedings of the IEEE International Conference on Computer Vision 2003*, (IEEE, 2003) pp.10–17.

[15] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Trans Pattern Anal* **42**(4), 824–836 (2020).

[16] J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2012*, (IEEE, 2012) pp. 573–580.

[17] N. Yang, R. Wang and D. Cremers, "Feature-based or direct: An evaluation of monocular visual odometry," 1–12 (2017), arXiv preprint arXiv: 1705.04300, 2017.

[18] F. Endres, J. Hess, J. Sturm, D. Cremers and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Trans Robot* **30**(1), 177–187 (2014).

[19] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans Robot* **33**(5), 1255–1262 (2017).

[20] C. Kerl, J. Sturm and D. Cremers, "Dense Visual SLAM for RGB-D Cameras," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems 2013*, (IEEE, 2013) pp. 2100–2106.

[21] A. Pretto, E. Menegatti, M. Bennewitz, W. Burgard and E. Pagello, "A Visual Odometry Framework Robust to Motion Blur," **In:** *IEEE International Conference on Robotics and Automation 2009*, (IEEE, 2009) pp. 2250–2257.

[22] J. Weingarten and R. Siegwart, "EKF-based 3D SLAM for Structured Environment Reconstruction," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems 2005*, (IEEE, 2005) pp. 3834–3839.

[23] T.-K. Lee, S. Lim, S. Lee, S. An and S.-Y. Oh, "Indoor Mapping Using Planes Extracted from Noisy RGB-D Sensors," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems 2012*, (IEEE, 2012) pp. 1727–1733.

[24] M. Kaess, "Simultaneous Localization and Mapping with Infinite Planes," **In:** *IEEE International Conference on Robotics and Automation (ICRA) 2015*, (IEEE, 2015) pp. 4605–4611.

[25] L. Ma, C. Kerl, J. Stückler and D. Cremers, " CPA-SLAM: Consistent Plane-Model Alignment for Direct RGB-D SLAM," **In:** *IEEE International Conference on Robotics and Automation (ICRA) 2016*, (IEEE, 2016) pp. 1285–1291.

[26] M. Hsiao, E. Westman, G. Zhang and M. Kaess, "Keyframe-based Dense Planar SLAM," **In:** *IEEE International Conference on Robotics and Automation (ICRA) 2017*, (IEEE, 2017) pp. 5110–5117.

[27] Y. Taguchi, Y.-D. Jian, S. Ramalingam and C. Feng, "Point-Plane SLAM for Hand-Held 3D Sensors," **In:** *IEEE International Conference on Robotics and Automation 2013*, (IEEE, 2013) pp. 5182–5189.

[28] X. Zhang, W. Wang, X. Qi, Z. Liao and R. Wei, "Point-plane SLAM using supposed planes for indoor environments," *Sensors* **19**(17), 3795 (2019).

[29] Y. Li, R. Yunus, N. Brasch, N. Navab and F. Tombari, "RGB-D SLAM with Structural Regularities," **In:** *IEEE International Conference on Robotics and Automation (ICRA) 2021*, (IEEE, 2021) pp. 11581–11587.

[30] H. Pfister, M. Zwicker, J. van Baar and M. Gross, "Surfels: Surface Elements as Rendering Primitives," **In:** *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, (ACM Press/Addison-Wesley Publishing Co, 2000) pp. 335–342.
[31] A. Concha and J. Civera, "DPPTAM: Dense Piecewise Planar Tracking and Mapping from A Monocular Sequence," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2015*, (IEEE, 2015) pp. 5686–5693.
[32] Z. Yan, M. Ye and L. Ren, "Dense visual SLAM with probabilistic surfel map," *IEEE Trans Vis Comput Gr* **23**(11), 2389–2398 (2017).
[33] R. Muñoz-Salinas and R. Medina-Carnicer, "UcoSLAM: Simultaneous localization and mapping by fusion of Keypoints and squared planar markers," *Pattern Recognit* **101**, 107193 (2020).
[34] H. M. Cho, H. Jo and E. Kim, "SP-SLAM: Surfel-point simultaneous localization and mapping," *IEEE/ASME Trans Mechatron* **27**(5), 2568–2579 (2021).
[35] R. Yunus, Y. Li and F. Tombari, "ManhattanSLAM: Robust Planar Tracking and Mapping Leveraging Mixture of Manhattan Frames," **In:** *IEEE International Conference on Robotics and Automation (ICRA) 2021*, (IEEE, 2021) pp. 6687–6693.
[36] S. Wang, R. Clark, H. Wen and N. Trigoni, "DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks," **In:** *IEEE international conference on robotics and automation (ICRA) 2017*, (IEEE, 2017) pp. 2043–2050.
[37] R. Li, S. Wang, Z. Long and D. Gu, "UnDeepVO: Monocular Visual Odometry Through Unsupervised Deep Learning," **In:** *IEEE International Conference on Robotics and Automation (ICRA) 2018*, (IEEE, 2018) pp. 7286–7291.
[38] N. Yang, R. Wang, J. Stuckler and D. Cremers, "Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry," **In:** *Proceedings of the European Conference on Computer Vision (ECCV)*, (Springer, 2018) pp. 835–852.
[39] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans Pattern Anal* **34**(11), 2274–2281 (2012).
[40] R. Giraud, V.-T. Ta, A. Bugeau, P. Coupe and N. Papadakis, "SuperPatchMatch: An algorithm for robust correspondences using superpixel patches," *IEEE Trans Image Process* **26**(8), 4068–4078 (2017).
[41] C. Barnes, E. Shechtman, A. Finkelstein and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Trans Graphic* **28**(3), 24 (2009).
[42] C. Feng, Y. Taguchi and V. R. Kamat, "Fast Plane Extraction in Organized Point Clouds Using Agglomerative Hierarchical Clustering," **In:** *IEEE International Conference on Robotics and Automation (ICRA) 2014*, (IEEE, 2014) pp. 6218–6225.
[43] P. J. Besl and N. D. McKay, "Method for Registration of 3-d Shapes," **In:** *Sensor Fusion IV: Control Paradigms and Data Structures 1992*. vol. 1611 (Spie, 1992) pp. 586–606.
[44] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison and S. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," *Int J Robot Res* **35**(14), 1697–1716 (2016).
[45] S. Li, X. Wu, Y. Cao and H. Zha, "Generalizing to the Open World: Deep Visual Odometry With Online Adaptation," **In:** *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2021*, (IEEE, 2021) pp. 13184–13193.
[46] S. Li, F. Xue, X. Wang, Z. Yan and H. Zha, "Sequential Adversarial Learning for Self-Supervised Deep Visual Odometry," **In:** *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) 2019*, (IEEE, 2019) pp. 2851–2860.

Cite this article: M. Yang, J. Xiong and Y. Li (2024). "RGB-D visual odometry by constructing and matching features at superpixel level", Robotica **42**, 2619–2634. https://doi.org/10.1017/S0263574724000985