Neural guided visual slam system with Laplacian of Gaussian operator

Ge Zhang¹,² | Xiaqi Yang² | Yulong Xu¹ | Yangdong Ye²

¹School of Information Engineering, Henan University of Chinese Medicine, Zhengzhou, China
²School of Information Technology, Zhengzhou University, Zhengzhou, China

Abstract

Simultaneous localization and mapping (SLAM) addresses the problem of constructing the map from noisy sensor data and tracking the robot's path within the built map. After decades of development, a lot of mature systems achieve competent results in feature-based implementations. However, there are still problems when migrating the technology to practical applications. One typical example is the accuracy and robustness of SLAM in environment with illuminance and texture variations. To this end, two modules in the existing systems are improved here namely tracking and camera relocalization. In tracking module, image pyramid is processed with Laplacian of Gaussian (LoG) operator in feature extraction for enhanced edges and details. A majority voting mechanism is proposed to dynamically evaluate and redetermine the zero-mean sum of square difference threshold according to the matching error estimation in patch search. In camera relocalization module, full convolutional neural network which focuses on certain parts of the input data is utilized in guiding for accurate output predictions. The authors implement the two modules into OpenSLAM and propose a neural guided visual SLAM system named LoG-SLAM. Experiments on publicly available datasets show that the accuracy and efficiency increase with LoG-SLAM when compared with other feature-based methods, and relocalization accuracy also improves compared with the recently proposed deep learning pipelines.

1 | INTRODUCTION

Simultaneous localization and mapping (SLAM) refers to the task of environment perception through the robot's own motion without prior knowledge, while estimating its own position and moving trajectories. It is recognized as the basic and necessary capabilities for unmanned vehicle, smart robot and drone [1]. Visual SLAM (vSLAM) is a set of algorithms to realize tracking and mapping with visual sensors only and dominates SLAM industry for economic and informative reasons. The implementation platforms of vSLAM systems are in great demand for accurate motion estimation under various illuminance and texture conditions [2]. The two factors are always changing with camera movement and time shifting in one day in real implementation scenarios of vSLAM. Therefore, the features extracted from image frames remain unstable and the position of a particular feature may change significantly in spatial space.

Variations on illuminance strength and beam angle bring grayscale intensity and shadow position changes, which affect the feature extraction and frame tracking [3]. The appearance-based tracking methods constructed on local features, which are collections of areas with obvious grayscale changes, are greatly influenced by the variations. In order to extract robust feature descriptors, SIFT [4], SURF [5] and ORB [6] adopt different mechanism to achieve certain level of tolerance on illuminance and scale variations during feature extraction. The appearance-based methods constructed on these descriptors, like PTAM [7] and ORB-SLAM [8], achieve competent performances on static and small scale scenes. However, generally they are not able to create a representation which is sufficiently illuminance invariant since the same place may change from dark to bright in grayscale with different lightening intensity and angle. Different from appearance-based methods constructed on local features, direct methods, like DTAM [9], semi-direct visual odometry (SVO) [10] and LSD-SLAM [11],
track every individual pixel in image frame and construct voxel volume to match the real environment. The precise environment perception helps achieving accurate motion estimation. But they consume significant memory with dense maps and are unable to implement in large-scale scene.

Apart from illuminance change, texture variation also affects the motion estimation and follow-on tracking. Large viewpoint changes and repetitive structures often produce dramatic spatial variation of features in closely related frames. Identical features may be placed at different coordinates in even adjacent frames. Finding the correspondence between the two frames requires feature matching and removal of influence from falsely matched features, which are called outliers [12]. The de facto standard is the random sample consensus (RANSAC) algorithm [13] and its improved versions, are frequently used in vSLAM applications to remove the outliers. But it needs to perform iterations of exponential order before finding an outlier-free minimal set under such condition. The increased time consumption is not acceptable in systems with real-time requirement. Approaches like PoseNet [14] formulate the motion estimation as a regression problem and is able to recover the camera pose in challenging scenes. However, the localization accuracy is still far below the traditional approaches in situations where approaches construct on local features perform well. An alternative solution [15] is to implement deep learning pipelines in RANSAC, which focuses on distinguishable parts of the input frames and makes the output predictions as accurate as possible [16]. But it needs depth information to train the network and the training process so that it can easily reach a local minimum due to the overlapped textures. The method cannot address the scalability issue and the performance is still far from ideal for practical vSLAM application.

To address the illuminance and texture invariance issues in vSLAM, a tracking module is proposed based on Laplacian of Gaussian (LoG) operator [17] and a relocalization module based on the deep learning model. The two modules are integrated into OpenvSLAM [18], which composes the LoG-SLAM system. In tracking module, edges and details are intensified by filtering image pyramids with LoG operator in feature extraction. The enhanced features are more robust to illuminance variation, which facilitate extracting robust features and estimating more accurate trajectories. A majority voting mechanism is proposed to redetermine the zero-mean sum of square difference (ZMSSD) threshold [19] according to the characteristics of scenes. The matching error estimation in reprojection process is set through the values of average of final accept score (AFAS), which is defined to reduce the negative effect of contrast variation brought by LoG transformation. In relocalization module, a full convolutional neural network (FCN) is implemented [20] to predict the correctly matched scene coordinates of features with learned weight and guide for accurate hypothesis selection in RANSAC. The proposal is beneficial in matching features and self-calibration of learned weights, which reduces the side effects brought by texture variation. Furthermore, the authors show experimental evidence of the superior accuracy of LoG-SLAM in motion estimation and more efficient relocalization on publicly available datasets.

The contributions of this article can be summarized as follows:

1. LoG operator is utilized in feature extraction to enhance the bold and edge detection and extracts robust features in environment with illuminance variation. AFAS is defined as the matching error estimation to optimize the patch search and a majority voting mechanism is proposed to redetermine the ZMSSD threshold for different kinds of scenes.

2. A FCN is trained to learn the sampling weights of the observations in RANSAC. The relocalization algorithm constructed on the FCN model converges faster and achieves more accurate relocalization without the help of depth information.

3. A neural guided vSLAM system is constructed based on the tracking and relocalization modules. Its performance surpasses state-of-the-art methods in public available datasets.

The rest of the article is organized as follows: Section 2 reviews the literature and recent advancements in vSLAM. Section 3 describes the tracking module, patch search and camera relocalization module in detail. Section 4 presents experiment results and Section 5 concludes the paper.

2 | LITERATURE REVIEW

Solving the appearance-based vSLAM problem consists of estimating the 6-DoF (degree of freedom) camera pose from a sequence of images in an arbitrary environment. The representation of the environment can be a database of images, a reconstructed 3D model or a trained deep neural network.

In the earliest stages, the motion estimation problem is solved by querying image database that contains millions of images with known poses. Originated from the classic computer vision technology, appearance-based image retrieval principles are used to map a query image to the nearest neighbour in a set of databases with known pose [21,22]. PTAM [7] and MonoSLAM [23] are recognized as the mature SLAM methods in the true sense, which separate the tracking and mapping threads in two parallel running modules. They achieve a certain level of robustness in scale invariance by implementing image pyramid and patch matching with fixed ZMSSD threshold. Unfortunately, illuminance consistency in motion estimation and textural influence is not presented in the above methods. The performance is limited by the accuracy of known poses and density of the keyframes. Moreover, the fixed threshold limits the application range of the method since it needs to be redetermined according to the characteristics of environment in the basis of balance between the feature quantity and quality of matching. With the development of depth camera, depth information are used in map construction with higher accuracy for better pose estimation [24,25]. Direct methods realize semi-dense mapping on CPU rather than RGB-D camera or GPU assistance [26]. They work directly on pixels rather than features or descriptors and can avoid the
corresponding artefacts, being clearly more robust to blur. In addition, their denser reconstructions are more useful for other tasks than just camera localization comparing with the sparse point map of the traditional vSLAM system. However, the performance on textured scenarios drops significantly since the semi-dense maps are not optimized. Besides the direct methods, SVO [10] is a combination of feature-based and direct methods, which estimates camera poses and its relative positions by tracking key points rather than descriptors according to their surroundings. It provides possibilities of implementation on embedded devices because of its efficiency [27]. It can achieve a tracking rate of 400 frames/s on desktop platforms, which is considered to be extremely fast. However, the method needs significant computation resources in patch search due to involvement of SURF descriptors.

The features of images are computed in environment perception and the features of query images are detected at motion estimation stage. Therefore, the search on correspondence is formulated as a feature matching problem. The main techniques in solving the problem include Structure-from-Motion [28], 2D-to-3D [29], prioritized matching [30] and co-visibility filtering [31]. Recently, it has been shown that machine learning methods have great potential to tackle the problem of feature matching in appearance-based methods. RANSAC algorithm is implemented to estimate the 6-DoF camera pose from these correspondences. These methods, like PoseNet [14], estimate accurate poses and scale well with scene expansion, but fail in textured surfaces and illumination variation conditions since the feature detectors may not work due to inadequate detectable features [2, 32]. Pose regression methods train neural networks to predict the camera poses directly from input images. They vary in network architecture [33,34], pose parameterisation [14,35] or training loss [36]. They are efficient in motion estimation, but lack accuracy because of the lost information in feed-forward networks. The regression methods also estimate 2D–3D correspondences between images and environment for each pixel [37–39]. Specifically, [15] proposes a neural network for camera relocalization with a differentiable RANSAC pipeline. But their method cannot address the scalability and ambiguity issues in camera relocalization. Moreover, the network models cannot be trained in an end-to-end fashion. Although the relocalization accuracy has increased in the recent works [35, 36], their performances are still below the state-of-the-art appearance-based methods and far from ideal for practical camera relocalization.

3 | LoG-SLAM SYSTEM

The LoG-SLAM system is separated into three parts, namely tracking with LoG filter, mapping and neural guided relocalization. The tracking module estimates camera pose with input frames and selects the keyframes to be inserted into map construction. The mapping module accepts keyframes and performs local bundle adjustment (BA) [40] to reconstruct the environment according to the camera poses and positions. Occlusions or abrupt movements are inevitable in vSLAM system, which may cause tracking failure. So, to recover from the failure, a neural guided relocalization module is activated to return a 6-DoF pose of current time point. The system overview is presented as Figure 1.

In the tracking module, input frames are used to construct image pyramid and each level of the pyramid is transformed by LoG operator for enhanced edges and details. In patch search thread, ZMSSD threshold determines the number of feature point and affects the patch matching performance. AFAS is calculated to evaluate the performance and the ZMSSD threshold is redetermined according to the scene characteristics. In camera relocalization, the NG-DSAC++ framework is utilized [16] for the feature matching process. The framework predicts the sampling weights in the hypothesis selection instead of adaptively sampling of vanilla RANSAC. Further, the pipeline is migrated to the process of finding the matched feature points in camera relocalization. Since there is no depth information, LoG-SLAM approximates the scene coordinates to initialize the network and performs a three-step training in network optimization. The new method makes accurate predictions for the correct matches between two images without 3D scene models, which facilitates the recovery from tracking failure in environment with less obvious texture and changed illumination conditions.

3.1 | Tracking with LoG filter

The tracking module receives image frames from the visual sensor and maintains a real-time estimate of the camera pose relative to the constructed environment map. Features from accelerated segment test (FAST) corner detection is selected to guarantee the real-time performance on mobile devices that carry limited computation capability. The scale invariance is achieved by four-level multi-scale image pyramid structured from 640 × 480 pixels to 80 × 60 pixels video frames in local feature extraction. Shi-Tomasi value thresholding [41] guarantees the rotation consistency. The invariance to illumination and texture variations is crucial in migrating vSLAM system from

![Figure 1](image-url) Overview of tracking, mapping and relocalization modules in LoG-SLAM system
indoor environment to outdoor scenario. The motion estimation algorithm has to adapt with rapid intensity change in grayscale and various texture conditions. So, to achieve the invariance, LoG-SLAM tracks the camera with LoG filter as explained in the following sections.

Regions with grayscale intensity variation in an image can be recognized by Laplacian transform. The LoG operator filters Gaussian noise with the Gaussian filter to strengthen edges and details that have obvious grayscale variation. The filtering process not only strengthens edge and details, but also strengthens the noise. In order to detect the edges more effectively, LoG-SLAM combines Gaussian smooth operator and Laplacian sharpening algorithm. The Gaussian smooth operator is used to suppress the noise, and the Laplacian sharpening algorithm is used to enhance the edges and regions with obvious grayscale intensity variation.

As Laplacian operator may detect edges as well as isolated noise in images, it is desirable to smooth the image first by a convolution with the Gaussian kernel of width \( \sigma \). The Gaussian filter is mathematically expressed as

\[
G_\sigma(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right],
\]

where \( x \) is the distance from the origin in the horizontal axis, \( y \) is the distance from the origin in the vertical axis, and \( \sigma \) is the standard deviation of the Gaussian distribution.

Gaussian filtering on an image is denoted by the convolution between the filter and the image. The original image \( f(x * y) \) is processed by convolution operation with Gaussian kernel \( G_\sigma(x, y) \) and the Laplace calculation is performed to suppress the noise:

\[
\Delta[G_\sigma(x, y) * f(x, y)] = [\Delta G_\sigma(x, y)] * f(x, y) = \text{LoG} * f(x, y).
\]

It is possible to obtain the LoG operator \( \Delta G_\sigma(x, y) \) first and then convolve it with the image. First, the first and second derivatives of \( G_\sigma(x, y) \) are considered as

\[
\frac{\partial}{\partial x} G_\sigma(x, y) = \frac{\partial}{\partial x} e^{-(x^2+y^2)/2\sigma^2} = -\frac{x}{\sigma^2} e^{-(x^2+y^2)/2\sigma^2},
\]

and

\[
\frac{\partial^2}{\partial x^2} G_\sigma(x, y) = \frac{x^2}{\sigma^4} e^{-(x^2+y^2)/2\sigma^2} - \frac{1}{\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \frac{x^2 - \sigma^2}{\sigma^4} e^{-(x^2+y^2)/2\sigma^2},
\]

respectively.

Second, the normalizing coefficient \( 1/\sqrt{2\pi\sigma^2} \) is omitted for simplicity reason. Similarly,

\[
\frac{\partial^2}{\partial y^2} G_\sigma(x, y) = \frac{y^2 - \sigma^2}{\sigma^4} e^{-(x^2+y^2)/2\sigma^2}.
\]

is obtained. Finally, the LoG operator is obtained by

\[
\text{LoG} \triangleq \Delta G_\sigma(x, y) = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} e^{-(x^2+y^2)/2\sigma^2},
\]

where the second derivatives of Gaussian kernel and convolution operation compose LoG operator, which facilitates the blob detection. Now, it is possible to obtain kernels of any sizes by approximating the LoG expression above. The LoG operator is applied to the image and the strong zero-crossings in the image are detected and kept to suppress the weak ones, which are likely caused by noise. LoG-SLAM performs LoG transform on grayscale frames which are selected as keyframes and FAST-10 corner detection is applied on them. The operations above guarantee the feature points easily affected by illuminance variations and Gaussian noise are dropped, which facilitate the blob and edge detection. Simultaneously, the scale invariance and rotation consistency are also improved with Laplacian sharpening, which make the vSLAM system robust in different viewpoint.

The filter processed images have obvious extreme points in boundaries, which are regarded as edge response. LoG-SLAM needs to eliminate them since they affect the edge detection seriously. The same way or procedure in [42] is adopted to find whether or not principal curvature is under a certain threshold. A feature point should be reserved for further usage when below the threshold, otherwise it should be discarded.

### 3.2 | Patch search

LoG-SLAM performs image search in a fixed range around the predicted image location point to find a single map point in the current frame. The viewpoint changes between the initial observations of the patch and current camera position must be wrapped to perform the operation [7]. An \( 8 \times 8 \) pixel patch area centred by feature point on previous keyframe is built as image template and the template is back projected to current frame through projection matrix. The search area is defined as a circle which has a geometrical centre and radius of 10 pixels. Sliding matching is performed on ZMSSD with sliding matching template \( a \) and image template \( b \) where the feature point is centred. The feature point position of current keyframe is determined in this process. The ZMSSD is denoted as

\[
\text{ZMSSD} = \frac{2S_a b - S_a^2 - S_b^2}{N} + S_{qa} + S_{qb} - 2C_{ab},
\]

where \( S_a \) denotes the sum of grayscale of \( a \); \( S_b \) denotes the sum of grayscale of \( b \); \( N \) denotes template image size (\( N \) is defined as 64 in the proposed method); \( S_{qa} \) denotes the sum of square
grayscale of $a$; $S_{ab}$ denotes the sum of square grayscale of $b$; and $C_{ab}$ denotes the sum of product of corresponding grayscale of $a$ and $b$.

The contrast of keyframe changes significantly after LoG transformation. Specific symptoms include a lot of continuous, overlapped and parallel lines after sharpening operation. The threshold of ZMSSD affects performance of tracking in LoG-SLAM system dramatically as illustrated in Table 1. A much higher ZMSSD threshold may cause a higher possibility of mismatching, resulting in unstable tracking. A lower value may reject too many matches due to inadequate feature points. In order to find a balance between the two factors, an average of final accepted score (AFAS) is defined as the quality evaluation of patch matching algorithm where smaller means better. The definition of AFAS can be expressed as

$$AFAS = \frac{1}{N} \sum_{x=1}^{N} s(x),$$  \hspace{1cm} (8)

where $s(x)$ denotes the smallest ZMSSD value in the $x$-th matching process and $N$ denotes sliding template matching counts. The LoG thresholding FAST feature points set is used to minimize re-project deviation in patch search period to obtain camera pose. The value of ZMSSD determines re-project deviation. The recommended window size of ZMSSD is 8×8 and a 9×9 window is defined as Gaussian kernel window to perform convolution operation. Extensive experiments validate that LoG-SLAM has the greatest gain under the following conditions: 9×9 window size, $\sigma = 1.5$ and kernel size $= 3$.

The AFAS value is also used to automatically redetermine the ZMSSD threshold at the beginning of every time-out period $t$. It is obvious that continuous tracking in large scale environment requires proper and ever changing ZMSSD thresholds for environments of various illuminance and texture conditions. Since the patch matching is time consuming (more than 50% of total time cost, data origin from [7]), the design ought to be simple but very efficient. At the beginning of every $t$, the patch search algorithm uses a majority voting mechanism to redetermine the ZMSSD threshold $Ts$ as

$$Ts = \begin{cases} ts_0, & \text{(default)}; \\ \frac{1}{N} \sum_{i=1}^{N} AFAS_i, & \left(0.7ts_0 \leq \frac{1}{N} \sum_{i=1}^{N} AFAS_i \leq 2ts_0\right); \\ 0.7ts_0, & \left(\frac{1}{N} \sum_{i=1}^{N} AFAS_i < 0.7ts_0\right); \\ 2ts_0, & \left(\frac{1}{N} \sum_{i=1}^{N} AFAS_i > 2ts_0\right). \\ \end{cases}$$ \hspace{1cm} (9)

where $AFAS_i$ denotes the ZMSSD value in the matching process of feature point $i$ and $N$ denotes the number of feature points. In a typical outdoor scenario of vSLAM application, the traceable feature point takes less than 25% of the total points when $Ts$ is set below 300. The tracking algorithm will be determined as a failure and keyframe insert will be terminated according to the quality evaluation. On the contrary, when $Ts$ is set above 300, mismatch will appear frequently due to the Laplacian sharpening operation. The feature points drift may cause a tracking failure. As it is found that using 300 as threshold one can easily adapt a large part of the environment in practice, so it is chosen as the default value of $Ts$ in this scenario. The threshold can be redetermined according to the characteristics of the scene through the majority voting mechanism.

### 3.3 Camera relocalization

The camera relocalization module is called in case of tracking failure when the camera encounters fast movement or motion blur, which involves 3D poses recovery from 2D images. First, the module maps 2D image patches to corresponding points in 3D scene space, which is regarded as 3D scene coordinate. Then the RANSAC algorithm is used to remove the mismatched 2D–3D correspondence and a Perspective-n-Point (PnP) solver [35] is used to estimate the 6-DoF camera pose from a set of 4 correspondences. Due to the complex and overlapped texture features of image frames, there are a lot of missed matches to be removed in adjacent image frames and RANSAC algorithm needs exponentially many iterations to converge. Inspired by scene coordinate represssion framework [43] and NG-DSAC++ pipeline [16], a neural guided relocalization module is proposed based on the deep learning pipeline.

Traditional relocalization algorithm optimizes the pose by sampling multiple minimal sets of hypotheses to create a pool $\mathcal{H}$ and selects the best one according to a scoring function, which is done through the ‘hypothesis and selection’ process of RANSAC. A pool of observations $y_j$ is selected in a round of RANSAC iteration, where $j$ belongs to the number of observations. The model $h$ is selected by randomly choosing

| Threshold (t) | AFAS | Algorithm status |
|--------------|------|------------------|
| 100          | 3146 | Unable to generate new key frame |
| 200          | 6203 | Affect generation of key frame |
| 300          | 7576 | Possible affected by similar re-project point |
| 400          | 7149 | Easy to be affected by similar re-project point |
| 500          | 9970 | High risk to be affected by similar re-project point |

Abbreviations: AFAS, average of final accepted score; ZMSSD, zero-mean sum of square difference.
a minimal set from the feature matches of them. In the implementation of camera relocalization, \( h \) describes the epipolar geometry [34] between two images and \( y \) is the feature matches between them. The feature matches set \( \mathcal{Y} \) is denoted as a collection of observations \( y \) where \( y \in \mathcal{Y} \). The final model \( \hat{h} \) is estimated by scoring its sample consensus that agrees with \( h \):

\[
\hat{h} = \arg \max s(h_j, Y),
\]

where the number of agreed scene coordinates are called the inliers count. The ‘hypothesis and selection’ process is performed until encountering a maximum score so far in the pre-defined iterations. The \( \arg \max \) mechanism grants the hypothesis \( h \) with minimum number of outliers, which are caused by noise from camera and dynamic moving objects in images, is selected for pose estimation. RANSAC needs to multiply the number of iterations to reduce the outliers when the texture variations become obvious. That means the noise reduction process consumes a lot of time and there will not be enough time for the following steps, such as bundle adjustment.

One possible solution to more accurate pose estimation is introducing deep learning models to RANSAC. But the \( \arg \max \) selection function is non-differentiable, which means the gradient of objective function cannot be back-propagated in the network [44] during training. So, softmax function is utilized to make the hypothesis selection differentiable. In that case, the discrete feature distributions are represented as probability distributions of model hypotheses. Now, it is possible to calculate the probability distributions of features and sample the model hypotheses according to the distribution. Consequently, the optimization process is able to judge whether the selected set contains the maximum number of inliers, which can be learned through the optimization of objective function.

LoG-SLAM recognizes the optimization process as a probability selection process. Now, an FCN with parameter \( w \) is utilized to guide the ‘hypothesis and selection’ process of RANSAC, which is done by optimizing the task loss function during training. In this case, camera relocalization module is able to sample observations according to the learned distribution. Specifically, the algorithm separates the optimization into two steps: (1) A prediction of hypothesis sampling is to get through the forward propagation algorithm [14] during the network training process, and the prediction is compared with ground truth to get the lost and (2) The gradient of cost function with respect to the learned weight \( w \) is calculated with back propagation algorithm [45], and the value of \( w \) is updated through gradient descent algorithm [46] with respect to the calculated gradient.

Inspired by policy gradient approaches in reinforcement learning [47] that involve the minimization of a loss function defined over a stochastic process [48], a probabilistic selection approach is utilized to model the dependency of the parameter selection process. Then the \( \arg \max \) selection is substituted to softmax probability distribution and make the RANSAC process differentiable. The potential model with the fewer outlier can be expressed as

\[
\hat{h} = h_j \sim \rho(j|\mathcal{H}) = \frac{\exp(s(h_j, \mathcal{Y}(w)))}{\sum_{k=1}^{n} \exp(s(h_k, \mathcal{Y}(w)))},
\]

where \( \rho(j|\mathcal{H}) \) is the softmax distribution of scores predicted by scoring function. In this case, the scene coordinates are converted to probability distribution of model hypotheses. Now it is possible to calculate the probability distributions of features and sample the model hypotheses according to the distribution. The best fit hypothesis \( \hat{h} \) is to be selected with higher probability. The objective function being optimized is transferred to

\[
\mathcal{L}(w) = \mathbb{E}_{j \sim \rho(j|\mathcal{H})}\left[ \ell(h_j) \right],
\]

and the training objective function can be calculated as

\[
\mathcal{L}(w) = \mathbb{E}_{\mathcal{H} \sim \rho(\mathcal{H}|w)}\mathbb{E}_{j \sim \rho(j|\mathcal{H})}\left[ \ell(h_j) \right],
\]

where we can construct \( \rho(\mathcal{H}|w) \) from discrete classified distributions \( \rho(y; w) \) on observations set \( \mathcal{Y} \) to get the training objective with two expectations:

\[
\mathcal{L}(w) = \mathbb{E}_{\mathcal{H} \sim \rho(\mathcal{H}|w)} \mathbb{E}_{j \sim \rho(j|\mathcal{H})}\left[ \ell(h_j) \right].
\]

The first is the expectation with respect to the sampling a hypothesis pools according to the probabilities predicted by the FCN and the second is the expectation with respect to the sampling of a final estimate from the pool according to the scoring function.

Then the partial derivative of task loss function with respect to weight \( w \) is calculated as

\[
\frac{\partial}{\partial w} \mathcal{L}(w) = \mathbb{E}_{\mathcal{H}}\left[ \ell(h) \frac{\partial}{\partial w} \log \rho(\mathcal{H}; w) \right],
\]

and the gradient \( \frac{\partial}{\partial w} \mathcal{L}(w) \) is approximated as

\[
\frac{1}{k} \sum_{i=1}^{k} \left[ \mathbb{E}_{j \sim \rho(j|\mathcal{H})}\left[ \ell(h_j) \right] \right] \frac{\partial}{\partial w} \log \rho(\mathcal{H}; w) + \frac{\partial}{\partial w} \mathbb{E}_{j \sim \rho(j|\mathcal{H})}\left[ \ell(h_j) \right],
\]

where the calculation of gradients requires the derivative of the task loss function \( \ell(h_j) \) since \( \mathbb{E}_{j \sim \rho(j|\mathcal{H})}\left[ \ell(h_j) \right] \) depends on parameters \( w \) via observations \( \mathcal{Y}(w) \).

The training of FCN aims at minimizing the expected loss of selected hypothesis, which is done by NG-DSAC++ using scene coordinate ground truth extracted from RGB-D training.
data. However, it is not possible to utilize this method in our camera relocalization as we do not have the in-depth information. Moreover, the training procedure may easily fail because of reaching a local minimum [15]. So, a three-step optimization training procedure to train the FCN network is proposed, which is used to predict the sampling probabilities of the scene coordinates without depth information. Each of them has an independent task loss function to be optimized.

1. Scene coordinate optimization. This step aims at optimize the distance between predicted scene coordinates and ground truth. Suppose the task loss function to be optimized is denoted as $\mathcal{L}$, we pre-train a scene coordinate regression FCN by minimizing the L1 distance between the predicted and ground truth coordinate:

$$
\mathcal{L}(\hat{y}, y^*) = \begin{cases} 
|y - \hat{y}|, |y - \hat{y}| < 10, \\
10\sqrt{|y - \hat{y}|}, otherwise
\end{cases},
$$

where $y$ denotes the predicted scene coordinate and $y^*$ denotes the corresponding ground truth during training. Since there is no depth information available, a constant depth value is assumed so that all the pixels beyond the distance of 10 meters are recognized as infinite far. The scene coordinate of a particular pixel with ground truth $h^*$ is approximated as:

$$
y^*_i \approx \hat{h}^* \begin{pmatrix} dx_i, dy_i, d, 1 \end{pmatrix}^T,
$$

where $x_i$ and $y_i$ denote the horizontal and vertical coordinates of pixel $i$, respectively, $f$ denotes the focal length and $d$ denotes a constant depth prior. This approximation ignores scene geometry completely and assumes that all the pixels have a constant distance from the camera plane. The assumption gives a rough estimation on scene coordinates in corresponding 3D space and serves as the basis for the following training steps.

The initialization training iteration runs for 300K times and we use Adam algorithm [49] as optimizer with a learning rate of $10^{-4}$ and a batch size of 1 image.

2. Reprojection error optimization. This step aims at optimizing the reprojection error between predicted scene coordinates and the ground truth. Further, the FCN is trained with task loss function similar to first step:

$$
\mathcal{L}(\hat{y}, y^*) = \begin{cases} 
|y - \hat{y}|, |y - \hat{y}| < 10, \\
\sqrt{|y - \hat{y}|}, otherwise
\end{cases},
$$

where robust distance function $|y - \hat{y}|$ is used with a threshold of 10 pixel, after which we use the square root of the reprojection error as the task loss. The optimization iteration runs for 100K times and Adam algorithm [49] is used as optimizer to substitute the traditional stochastic gradient descent (SGD) [48]. The learning rate and batch size are the same as the first step.

3. End-to-end optimization. This step combines neural guidance with vanilla RANSAC algorithm to train neural networks, which predict the matched features between two images. RANSAC includes a mechanism to randomly choose the matched feature points based on an online estimate of outlier ratio. The camera relocalization module aims at making estimation according to the predictions. The predictions are weighted probabilities parameterized by a FCN network with parameter $w$. The aim is to train the parameters $w$ such that the feature matches set with minimal outlier will be selected, which result in accurate estimate. Inspired by algorithm proposed by Brachmann et al. [15], the end-to-end training objective is defined by $\mathcal{L}(w)$ as the minimization of the expectation of task loss function.

$$
\mathcal{L}(\hat{y}, y^*) = \mathcal{L}(\theta, \theta^*) + |t - t^*|,
$$

where $\theta$ denotes the axis-angle of the camera rotation and $t$ denotes the camera translation. The algorithm measures angle $\mathcal{L}(\theta, \theta^*)$ between the estimated and ground truth rotation in degrees and distance $|t - t^*|$ between them in centimetres.

The end-to-end optimization iteration runs for 200K times using Adam algorithm [49] with a learning rate of $10^{-4}$ and a batch size of 1 image.

Upon completion of the three-step optimization, camera poses can be recovered in a local regression way. Specifically, LoG-SLAM utilizes trained FCN to predict the scene coordinates of features in the input image frames. The corresponding points can be used to estimate the camera pose through PnP algorithm. The relocalization algorithm is summarized as Algorithm 1.

### Algorithm 1 Log-Slam Camera Relocalization

**Data:** network parameter $w$; RANSAC parameters $N, K, R$, $P, Q$

**Result:** 6-DoF camera pose $\hat{h}$

1. evaluate the FCN to obtain $N$ correspondences;
2. $i = 0$;
3. while $i < K$ do
4. sample 2D-3D correspondences according to $w$;
5. generate pose hypothesis $h$ using PnP;
6. if the correspondences are inliers of $h$ then
7. $H_i \leftarrow h$;
8. $S_i \leftarrow \text{score}(h, Y)$;
9. $i = i + 1$;
10. end
11. end
12. select the final pose $\hat{h}$ according to $S_i$;
13. $j = 0$;
14. while $j < R$ do
15. sample $P$ inliers at most;
16. if found inliers quantity greater than $Q$ then
17. reevaluate the pose $\hat{h}$ according to the inliers;
18. end
19. end
20. return $\hat{h}$;
The scoring function \( \text{score}(h, y) \) is the quantity that corresponds with reprojection error less than a threshold \( \tau \):

\[
\text{score}(h, y) = \alpha \sum_{y \in Y} 1 - \text{sig}(\beta d(y, h) - \beta \tau),
\]

(21)

where \( d(y, h) \) denotes the distance between observation \( y \) and the hypothesis \( h \), \( \text{sig}() \) denotes the Sigmoid function, and \( \alpha \) and \( \beta \) controls the softness of the scoring [37]. The final camera pose \( \hat{h} \) is the one with the highest inlier count. In the actual implementation, \( \alpha = 10 \), \( \beta = 0.5 \), \( 40 \times 40 \) correspondences are sampled at each step (\( N = 1600 \)), 256 hypothesis are sampled in each image (\( K = 256 \)), \( \tau \) is set to 10 pixels, 8 refinement steps are performed (\( R = 8 \)). In an individual refinement, 100 inliers are chosen at most (\( P = 100 \)), and the refinement terminates when there is less than 50 inliers (\( Q = 50 \)).

### 3.4 Complexity analysis

Base on the LoG filtering and patch search process described above, the main threads of tracking and mapping modules of LoG-SLAM can be summarized as Algorithms 2 and 3.

The camera relocalization module is architecturally individual in vSLAM system and the network training can be done offline. So we only analyse the complexities of threads in tracking and mapping modules, which work collaboratively and asynchronously in LoG-SLAM. The thread in tracking module update camera poses based on a small part of the current coordinate of map points and the time complexity is \( O(1) \). The thread in mapping has done calculations involving current camera pose, 3D coordinate of maps and camera pose in the keyframes. Being enslaved to time complexity of global BA, the feature point counts \( P \) and keyframe count \( K \) performs \( O(PK^2) \) in time and \( O(PK) \) in space under the worst condition. The global BA algorithm only works when mapping is idle and has the property of preemptable. As a result, the time complexity of LoG-SLAM is related to the number of feature points and keyframes, and complexity is \( O(PK^2) \).

### 4 EXPERIMENTS

In this section, evaluations are carried out to validate the improvement of LoG-SLAM in terms of motion estimation and relocalization accuracy, by means of deviation between estimated poses and ground truth against the state-of-the-art feature-based and direct vSLAM systems. Specifically, ORB-SLAM [8], LSD-SLAM [11] and the proposed method are evaluated in the motion estimation accuracy test. Besides, the tracking time consumption is compared to study the affection of dynamic threshold of ZMSSD in patch search algorithm and analyse the potential affection in accuracy of estimated poses.

ORB-SLAM is regarded as the milestone in feature-based vSLAM system and a lot of systems are constructed on its infrastructure, like OpenvSLAM and LoG-SLAM. So the comparison between ORB-SLAM and our method can be recognized as the ablation study on the affection LoG filtering and new camera relocalization module. LSD-SLAM is a typical example of direct method of vSLAM, which is usually implemented in scenario with fewer gray-scale variation. It is efficient and can build semi-dense or even dense maps. The motion estimation may benefit from the precise environment reconstruction.

The camera relocalization module is constructed on FCN, so it is needed to compare it with other deep learning pipelines. Further, in the relocalization accuracy test, evaluate PoseNet [14], DSAC [15], DSAC++ [37] and our camera relocalization algorithm are evaluated. PoseNet utilizes the CNN in camera relocalization and it is possible to test the effect of different network models. DSAC is the pioneer in the local regression method of relocalization and DASC++ is one typical example of the latest deep learning pipeline for relocalization.

The experiments are carried out on x86 architecture desktop PC with Intel Xeon E5-2620 2.4 GHz, 32 GB RAM and NVIDIA Tesla K20C GPU. As for the implementation of FCN, ResNet [50] is selected to update the network architecture of the proposed camera relocalization module.

#### 4.1 Dataset

TUM RGB-D [51], KITTI [52] and EuRoC [53] datasets are chosen to evaluate the motion estimation accuracy between LoG-SLAM and its competitors. The TUM RGB-D dataset
contains consecutive images from different scenarios with various illuminance and texture conditions. Now, several representative sequences are chosen to evaluate the performance in indoor scenes. KITTI dataset contains enormous types of sequences collected on a vehicle, which is ideal to simulate the outdoor scenes of robotics and autonomous driving. EuRoC dataset contains 3D movement in a machine hall building and office room, which is usually adopted to simulate the motion estimation for micro aerial vehicle (MAV) in 3D environment. There are obvious illuminance and texture variations in the three datasets, which is a challenge for appearance-based methods. Both of the three datasets contain accurate ground truth captured by IMU equipment and it is used to calculate the deviation between estimated poses and real trajectories of cameras. For more specified descriptions, please refer to the experiment discussion sections.

Cambridge Landmarks [35] and 7-Scenes [54] datasets are chosen to evaluate the camera relocalization accuracy between our method and competitors. There are seven scenes collected in different places, which exhibit complex textures (e.g. repeated steps in ‘Stairs’), peculiarities (e.g. reflections in ‘RedKitchen’), illuminance conditions, motion blur, flat surface and sensor noise. Each scene contains several sequences captured by different users, which are ideal for evaluating the robustness in view point variation. The corresponding ground truth camera poses were calculated using KinectFusion, which is suitable to compute the position and rotation errors. The Cambridge Landmarks dataset provides labelled image frames to train and test relocalization algorithm in the outdoor urban environment. There are significant urban clutters in the sequences, for example, pedestrians and vehicles, which is challenging for the local regression methods with various illuminance and texture conditions.

4.2 Evaluation metrics

There are two prominent methods to evaluate the accuracy of the estimated trajectory by comparing it with the ground truth provided in the benchmark. The absolute trajectory error (ATE), which is the absolute translation distance between estimated trajectory and ground truth after rigidly aligning with benchmark. ATE is well suited for measuring the performance of SLAM systems. Given the transformation \( S \) of a SLAM system, which corresponds to the least-squares solution that maps the estimated trajectory \( P_{1:n} \) onto the ground truth \( Q_{1:n} \). The ATE at time point \( i \) can be calculated as

\[
F_i := Q_i^{-1} S P_i. \tag{22}
\]

The relative pose error (RPE) is the translation drift in m/s and rotational degree in deg/s error between estimated trajectory and ground truth, which is used to measure the local accuracy of the trajectory over a fixed time interval \( \Delta \). The RPE at time point \( i \) can be calculated as

\[
E_i := (Q_i^{-1} Q_{i+\Delta})^{-1}(P_i^{-1} P_{i+\Delta}). \tag{23}
\]

Both of the two metrics can be calculated for root mean square error (RMSE), which is commonly used as the performance measurement for SLAM systems [51]. The time interval \( \Delta \) is chosen as 1 for SLAM systems that match consecutive frames, which outputs the drift per frame. For systems that contain more than one previous frame, \( \Delta \) should be chosen according to the sampling rate during sequence record. For the evaluation of SLAM system, we usually average all the possible time intervals \( \Delta \). The RMSE of ATE over all time indices of the translational component is

\[
RMSE(F_{1:n}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} ||trans(F_i)||^2}, \tag{24}
\]

where \( trans(F_i) \) refers to the translational components of the ATE \( F_i \). The RMSE of RPE over all time intervals \( \Delta \) of the translational component is

\[
RMSE(E_{1:n}) = \frac{1}{n} \sum_{\Delta=1}^{n} \sqrt{\frac{1}{m} \sum_{i=1}^{m} ||trans(E_{i})||^2}, \tag{25}
\]

where \( trans(E_i) \) refers to the translational components of the RPE \( E_i \). Note that the RPE combines rotational and translational errors into a single measure, while the ATE only considers the translational errors. As a result, the RPE value is always slightly higher than the ATE value. From a practical perspective, the ATE value has an intuitive visualization which facilitates visual inspection. So ATE is used in plotting the estimated trajectories and ground truth. Nevertheless, the two metrics are also strongly correlated and relative order remains the same independently of which measure was actually used.

4.3 Motion estimation

Six sequences from TUM RGB-D dataset are chosen to simulate the motion estimation in indoor environment. The sequences cover different texture and structure types: low texture (fr2/desk with person), non-structure with obvious texture (fr3/nostructure texture near), structure with obvious texture (fr3/structure texture near), scenarios with dynamic moving objects (fr3/sitting halfsphere, fr3/walking halfsphere) and mixed texture and structure with loop closures (fr3/long office household).

Since LoG-SLAM is constructed on OpenvSLAM, it has similar infrastructure with ORB-SLAM. The motion
estimation comparison with ORB-SLAM can be recognized as the evaluation on trajectory accuracy improvement brought by LoG filtering and improved camera relocalization. Another competitor is LSD-SLAM, which is a typical representation of direct methods. Refer to Table 2 for the detailed information of the selected sequences from TUM RGB-D dataset. The first column depicts names of the video sequences, the second and third columns depict perception time and length respectively. The fourth and fifth columns depict the average translation and rotation velocities during capture, which determine the dynamics of a particular sequence. We estimate the trajectories and calculate the RMSE of ATE and RPE. The evaluation results are summarized in the last three columns of Table 2.

The data suggest that LoG-SLAM performs consistently better in various indoor scenes of different illumination and texture conditions. We start with the sequence of ‘fr2/desk_with_person’, which contains only translational motions along the principal axes of the camera and the orientation is kept fixed in most of the time. The three algorithms perform well in this simple sequence and there is no obvious difference in three metrics. There are obvious texture variations in the sequence of ‘fr3/nostructure_texture_near’ and dynamically moving objects in the sequence of ‘fr3/structure_texture_near’, which are intended to evaluate the robustness of the algorithm. LSD-SLAM has several large drifts along the trajectory due to the lack of scene structure and closing loop motion trial. The direct method fails to generate a complete trajectory in the latter sequence. LoG-SLAM achieves acceptable RMSE with very low time consumption, which suggests that we have efficiency and accuracy superiority in the large-scale scene and comprehensive texture conditions. Sequences ‘fr3/sitting_halfsphere’ and ‘fr3/walking_halfsphere’ contain two persons walk through an office room with talk and gesture a little bit. The moving hands and other gestures randomly appears in the scene, which is challenging for motion estimation. LSD-SLAM has no advantage in the two sequences due to interference of randomly appeared moving objects and fails again in the latter one. LoG-SLAM has superior performance in RMSE of RPE because of the enhancement in edge detection. When it comes to ‘fr3/long_office_household’, the camera moves around two office desks so that the loop is closed. The direct method shows disadvantages in object recognition and achieves relatively high RMSE in this sequence. ORB-SLAM and LoG-SLAM perform roughly the same in RMSE due to the accumulated error reduced by loop closing. But LoG-SLAM is faster than traditional feature-based algorithm since most of the useless details are ignored.

We evaluate the three methods in outdoor scenario with six sequences from the KITTI visual odometry benchmark: residential area with low dynamic factors (KITTI00), road and traffic scenario with low dynamic factors (KITTI02), city area with obvious shadows (KITTI07), long road parking scenario

| Sequence               | Duration | Length   | Avg. trans vel. (m/s) | Avg. rotat vel. (deg/s) | Algorithm     | RMSE of ATE (m) | RPE (m/s) | RPE [deg/s] |
|------------------------|----------|----------|-----------------------|-------------------------|---------------|-----------------|------------|-------------|
| fr2/desk_with_person   | 69.15    | 18.88    | 0.193                 | 6.338                   | ORB-SLAM      | 0.169           | 0.102     | 1.5         |
|                        |          |          |                       |                         | LSD-SLAM      | 0.394           | 0.287     | 3.6         |
|                        |          |          |                       |                         | LoG-SLAM      | 0.114           | 0.115     | 1.3         |
|                        |          |          |                       |                         | ORB-SLAM      | 0.345           | 0.379     | 0.8         |
| fr3/long_office_household| 87.1     | 16.12    | 0.225                 | 12.053                  | LSD-SLAM      | 1.385           | 1.490     | 0.7         |
|                        |          |          |                       |                         | LoG-SLAM      | 0.235           | 0.278     | 0.9         |
|                        |          |          |                       |                         | ORB-SLAM      | 1.467           | 1.064     | 2.7         |
| fr3/nostructure_texture_near| 56.48   | 13.46    | 0.242                 | 7.43                    | LSD-SLAM      | 7.451           | 5.050     | 5.6         |
|                        |          |          |                       |                         | LoG-SLAM      | 1.382           | 1.057     | 2.4         |
|                        |          |          |                       |                         | ORB-SLAM      | 1.627           | 1.135     | 2.7         |
| fr3/sitting_halfsphere | 37.15    | 6.503    | 0.18                  | 19.094                  | LSD-SLAM      | 5.877           | 4.457     | 9.1         |
|                        |          |          |                       |                         | LoG-SLAM      | 1.533           | 1.104     | 2.5         |
|                        |          |          |                       |                         | ORB-SLAM      | 1.580           | 1.041     | 1.9         |
| fr3/structure_texture_near | 36.91   | 5.05     | 0.141                 | 7.677                   | LSD-SLAM      | -               | -         | -           |
|                        |          |          |                       |                         | LoG-SLAM      | 1.416           | 1.034     | 2.6         |
|                        |          |          |                       |                         | ORB-SLAM      | 1.425           | 1.541     | 2.5         |
| fr3/walking_halfsphere  | 35.81    | 7.69     | 0.221                 | 18.267                  | LSD-SLAM      | -               | -         | -           |
|                        |          |          |                       |                         | LoG-SLAM      | 1.336           | 1.438     | 2.1         |

Note: The bold numbers indicate the best result of an algorithm in an individual sequence.

Abbreviations: ATE, absolute trajectory error; RMSE, root mean square error; RPE, relative pose error.
Table 3 summarizes the ATE and RPE of the estimated trajectories from the three algorithms in visual odometry benchmark of KITTI dataset. The results suggest that LoG-SLAM has advantages in scenes with obvious illuminance variation and provides robustness in motion blur. The three algorithms have similar performances in the ‘KITTI00’ and ‘KITTI02’ sequences, which contain low dynamic factors. But when it comes to medium difficulty level scenes, the fast motion brings less detectable texture details. LSD-SLAM encounters several tracking failures and performs poorly. ORB-SLAM needs more time to initialize and the performance is acceptable. LoG-SLAM also experiences tracking failure, but it recovers quickly and generates an accurate trajectory comparing to the ground truth.

Figure 2 illustrates the estimated trajectories and ground truth comparison in an intuitive way. LSD-SLAM experiences several tracking failure and fails to generate enough poses to be corrected in scale. But, the proposed method is compared only with ORB-SLAM, as both have the similar infrastructure, to evaluate the contribution of LoG filtering and camera relocalization. Our advantage is more obvious in scenes with more vehicle turning and stopping, which has large chance of encountering overlapped textures and shadows of low illuminance condition. The side effect of illuminance variation is reduced and it is possible to relocalize the camera poses in less time during tracking failure. The trajectories estimated by LoG-SLAM are more accurate and there are only fewer chances in generating trajectories far from the ground truth.

In order to evaluate the visual localization and environment reconstruction in 3D space, our system is tested in EuRoC dataset. The EuRoC dataset contains two batches of videos, which are collected in an industrial environment and a room equipped with a motion capture system, respectively. Six video sequences are chosen, ranging from slow flights under good visual conditions to dynamic flights with motion blur and poor illumination. The three systems are thoroughly tested and various texture conditions are also simulated.

The sequences in EuRoC are classified as easy, medium and difficult based on MAV’s speed, illumination and scene texture. Experiment is not carried out on three sequences of Vicon Room 2 as part of the frames are missing in monocular capture. Table 4 shows the absolute translation RMSE of three algorithms. ORB-SLAM achieves a localization precision of a few centimetres in stereo mode, but significantly larger RMSE in monocular mode. LSD-SLAM fails in 2/3 of the sequences since the textures are not obvious. Due to the extra time for bundle adjustment, trajectories estimated by LoG-SLAM have comparatively lower RMSE, especially in sequences with illumination variations.

### Table 3 Evaluation of motion estimation for selected sequences from KITTI dataset

| Sequence | Algorithm | ATE (m) | RPE (m/s) | RPE (deg/s) |
|----------|-----------|---------|-----------|-------------|
| KITTI00  | ORB-SLAM  | 0.172   | 0.234     | 0.9         |
|          | LSD-SLAM  | 0.471   | 0.357     | 1.0         |
|          | LoG-SLAM  | 0.233   | **0.104** | **0.7**     |
| KITTI02  | ORB-SLAM  | 0.357   | 0.201     | 0.7         |
|          | LSD-SLAM  | 0.684   | 0.347     | 1.5         |
|          | LoG-SLAM  | **0.288** | 0.117   | **0.6**     |
| KITTI07  | ORB-SLAM  | 1.625   | 2.369     | 3.5         |
|          | LSD-SLAM  | 4.321   | 5.137     | 8.1         |
|          | LoG-SLAM  | **1.316** | **1.844** | **2.7**     |
| KITTI08  | ORB-SLAM  | 0.264   | 0.257     | 1.2         |
|          | LSD-SLAM  | 0.798   | 0.388     | 1.2         |
|          | LoG-SLAM  | **0.233** | **0.135** | **1.1**     |
| KITTI09  | ORB-SLAM  | 1.032   | 0.987     | 5.9         |
|          | LSD-SLAM  | 3.013   | 2.321     | 9.2         |
|          | LoG-SLAM  | **0.716** | 1.021   | **4.6**     |
| KITTI10  | ORB-SLAM  | 6.103   | 5.141     | 7.2         |
|          | LSD-SLAM  | –       | –         | –           |
|          | LoG-SLAM  | **5.022** | **4.174** | **6.6**     |

Note: The bold numbers indicate the best result of an algorithm in an individual sequence.
Abbreviations: ATE, absolute trajectory error; RMSE, root mean square error; RPE, relative pose error.
and severe motion blur. Figure 3 shows examples of computed poses compared to ground truth in four medium to difficult scenarios from EuRoC dataset. EuRoC dataset contains sufficient motion in the beginning of each capture to aid the initialization of monocular SLAM system. For this reason, the two systems have successful initialization. But the trajectories matches to ground truth better in scenarios with dark illumination and fast motion as illustrated in Figure 3(c). The green trajectory has obvious drift in some part of the ground truth while the blue one fits better.

4.4 Tracking time consumption

The tracking time of keyframes are collected to test the contribution of dynamic threshold in reducing the time consumption of patch search in four sequences of TUM RGB-D dataset. ORB-SLAM uses a fixed ZMSSD threshold during patch search and it can be recognized as a baseline for evaluation on efficiency. LSD-SLAM does not provide keyframes during tracking and is not involved in this test. Since the threshold is supposed to set according to the characteristics of scene, the sequences are selected in such a way that both of the two systems initialize successfully to make the experiment fair. The time consumption in tracking of keyframes with ORB-SLAM and LoG-SLAM is illustrated in Figure 4.

The figure suggests that LoG-SLAM consumes less time on tracking in one particular keyframe comparing with ORB-SLAM. A possible reason is the time spent by patch search contributes to large part of time consumption in tracking module, and the time is reduced by the improved ZMSSD threshold redetermination. The redetermined sliding window by majority voting mechanism contributes to the patch matching algorithm, and reduces the overall time consumption. Although ORB-SLAM does not need to perform patch search, it is still slower than LoG-SLAM because construction of ORB descriptors is complex. Combined with the above two factors, we only save 0.01–0.05 ms on tracking in one particular keyframe comparing with ORB-SLAM. But the total reduction is significant since there are usually 5000–10,000 keyframes in one typical motion estimation process of vSLAM application. The total time consumption is able to reduce by 50–150 ms, which can achieve higher frame rate in identical environment or extend to larger scale environment with consistent frame rate.

4.5 Camera relocalization

Cambridge Landmarks and 7-Scenes dataset are selected to do an in-depth study on the relocalization accuracy. Since deep learning models are implemented in this process, the

| Sequence   | Duration (s) Length (m) | Avg. vel. (m/s) angular vel. (rad/s) | Note                              | ORB-SLAM | LSD-SLAM | LoG-SLAM |
|------------|-------------------------|-------------------------------------|-----------------------------------|----------|----------|----------|
| MH_01 easy | 80.6/182                | 0.44/0.22                           | Good texture, bright scene        | 0.035    | -        | 0.041    |
| MH_02 easy | 73.5/150                | 0.49/0.21                           | Good texture, bright scene        | 0.018    | -        | 0.026    |
| MH_03 medium | 130.9/132            | 0.99/0.29                           | Fast motion, bright scene         | 0.028    | -        | 0.21     |
| MH_04 difficult | 91.7/99             | 0.93/0.24                           | Fast motion, dark scene           | 0.119    | -        | 0.087    |
| MH_05 difficult | 97.6/111            | 0.88/0.21                           | Fast motion, dark scene           | 0.060    | -        | 0.057    |
| VR_01 easy | 58.6/144                | 0.41/0.28                           | Slow motion, bright scene         | 0.035    | 0.066    | 0.031    |
| VR_02 medium | 75.9/83.5            | 0.91/0.56                           | Fast motion, bright scene         | 0.020    | 0.074    | 0.018    |
| VR_03 difficult | 79.0/105            | 0.75/0.62                           | Fast motion, motion blur          | 0.048    | 0.089    | 0.031    |

Abbreviation: RMSE, root mean square error.

FIGURE 3 The comparison of estimated trajectories and ground truth in four typical sequences of EuRoC dataset. Blue lines indicate the estimated trajectories of ORB-SLAM and green ones indicate the results of LoG-SLAM. The black dashed lines indicate the ground truth.
FIGURE 4  Time consumption comparison of tracking module in four sequences from TUM RGB-D dataset. In an individual subfigure, the horizontal axis denotes the number of keyframes and the vertical one denotes the time consumption. The red lines indicate time consumption of ORB-SLAM and blue ones indicate LoG-SLAM.
competitors are the representative deep learning pipelines: PoseNet, DSAC and DSAC++. All scenes are used for comprehensive training of FCN and separated relocalization is performed in individual sequences.

Part of the data are extracted directly from their corresponding articles. The first column in Table 5 depicts the sequence name, second one depicts the spatial area of the capture, which is a measurement of environment scale. The third and fourth columns depict training and testing numbers of image. The remaining columns summarize the relocalization accuracy of our method and competitors.

The data suggest that the three step training process of LoG-SLAM can reduce the overfit phenomenon significantly, which improves the relocalization accuracy accordingly. In the sequences of ‘Chess’, ‘Pumpkin’ and ‘RedKitchen’, the fast motion brings difficulties in feature recognition. LoG-SLAM performs the best among the four algorithms. An exception is discovered in scene ‘Heads’, which contains only two sequences. DSAC++ has the highest relocalization accuracy since they correct the drawbacks of rotation and scale tolerance. LoG-SLAM focuses on the illuminance variation and texture overlap and the accuracy fluctuates significantly. The performance of LoG-SLAM also fluctuates dramatically in the ‘Office’ scene, where there are 10 sequences with different angles. But the overall result is acceptable as variations in orientations are challenging for feature based methods. PoseNet acquires the highest accuracy in scene ‘Stairs’ with the help of depth information. But the method constructs very dense maps, which limit the migration to larger environment.

Figure 5 shows the cumulative histogram of errors for two scenes in Cambridge Landmarks dataset. The curves suggest that our probabilistic camera relocalization consistently has better or equivalent performance without the involvement of depth information. The weights predicted by FCN put lower probability in the recurring area, like blue sky, and the relocalization algorithm can ignore the irrelevant texture information. A cleaner representation through focused feature

| Sequence | Spatial extent (m²) | Frames | PoseNet | DSAC+ | LoG-SLAM |
|----------|--------------------|--------|---------|--------|----------|
| Chess    | 3                  | 4K 2K  | 85.8%   | 94.7%  | 93.8%    |
| Fire     | 4                  | 2K 2K  | 71.2%   | 75.3%  | 75.6%    |
| Heads    | 2                  | 1K 1K  | 81.1%   | 71.9%  | 88.6%    |
| Office   | 5.5                | 6K 4K  | 76%     | 69.2%  | 75.4%    |
| Pumpkin  | 6                  | 4K 2K  | 63.9%   | 50.3%  | 55.9%    |
| RedKitchen | 6                | 7K 5K  | 54.8%   | 46.2%  | 50.7%    |
| Stairs   | 5                  | 2K 1K  | 37%     | 5.3%   | 19.7%    |
| Average  | -                  | -      | 67.1%   | 60.1%  | 60.4%    |

Note: The bold numbers indicate the best result of an algorithm in an individual sequence.
matching results in better accuracy during relocalization. Only ‘King’s College’ and ‘St Mary’s Church’ sequences are
compared since the authors of PoseNet do not provide results for the other five sequences officially.

5 | CONCLUSION

Focusing on illuminance and texture invariance issues in large-scale vSLAM application, a vSLAM system is proposed
with LoG operator: LoG-SLAM. In the tracking module of LoG-SLAM, the FAST-10 algorithm is selected to extract
features and to construct image pyramid. Each level of the image pyramid is processed with LoG operator for
strengthened edges and details. AFAS is defined to automatically redetermine the threshold of ZMSSD according to
the scene characteristics, which accelerates the tracking process of vSLAM in various applications for real-time
performance. Moreover, LoG-SLAM performs Gaussian convolution on video frames with Gaussian Laplacian kernel
and improves the patch matching algorithm. In order to recover the camera pose during a tracking failure, a camera
relocalization module is proposed based on deep learning model. The overall experiment shows the proposed method
has better performance under various surface and illuminance conditions. Individual experiment shows the relocalization
accuracy surpasses other sparse feature-based and learning-based methods.

ACKNOWLEDGEMENTS

This work was supported by the National Key Research and Development Program of China under Grant No.
2018YFB1201403, National Natural Science Foundation of China under Grant No. 61772475, Henan Science and Tech-
nology Research under Grant No. 172102210011.

REFERENCES

1. Gai, J., et al.: A review of visual inertial odometry from filtering and
optimisation perspectives. Adv. Robot. 29(20), 1289–1301 (2015)
2. Sattler, T., et al.: Benchmarking 6D of outdoor visual localization in
changing conditions. In: Proceedings of IEEE Conference on Computer
Vision and Pattern Recognition, pp. 8601–8610 (2018)
3. Khalid, Y., Bab-Hashash, A., Reza, H.: An overview to visual odometry and
visual slam: Applications to mobile robotics. Intell. Ind. Syst. 1(4),
289–311 (2015)
4. Lowe, D.G.: Distinctive image features from scale-invariant keypoints.
Int. J. Comput. Vis. 60(2), 91–110 (2004)
5. Herbert, B., et al.: Speeded-up robust features (surf). Comput. Vis. Image
Und. 110(3), 346–359 (2008)
6. Rublee, E., et al.: An efficient alternative to sift or surf. In: Proceedings
of International Conference on Computer Vision, pp. 2564–2571 (2011)
7. Murray, D.K.G.: Parallel tracking and mapping for small area workspaces.
In: Proceedings of IEEE and ACM International Symposium on Mixed
and Augmented Reality, pp. 225–234 (2007)
8. Mur-Artal, R., Juan, D.: Tardós. Orb-slam2: An open-source slam system
for monocular, stereo, and rgb-d cameras. IEEE Trans. Robot. 33(5),
1255–1262 (2017)
9. Newcombe, R.A., Lovegrove, S.J., Drum, J.D.A.: Dense tracking and
mapping in real-time. In: Proceedings of IEEE International Conference
on Computer Vision, pp. 2320–2327 (2011)
10. Christian, F., et al.: Svo: Fast semi-direct monocular visual odometry. In:
Proceedings of IEEE International Conference on Robotics and Auto-
mation, pp. 15–22 (2014)
11. Engel, J., Thomas, S., Daniel, C.: Lsd-slam: Large-scale direct monocular
slam. In: Proceedings of European Conference on Computer Vision, pp.
834–849 (2014)
12. Mur-Artal, R., Juan, D.T.: Fast relocalisation and loop closing in key-
frame-based slam. In: Proceedings of IEEE International Conference on
Robotics and Automation, pp. 846–853 (2014)
13. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for
model fitting with applications to image analysis and automated cartog-
raphy. Commun. ACM. 24(6), 381–395 (1981)
14. Kendall, A., Grimes, M., Cipolla, R.: PoseNet: A convolutional
network for real-time 6-dof camera relocalization. In: Proceedings of
IEEE International Conference on Computer Vision, pp. 2938–2946
(2015)
15. Brachmann, E., et al.: DSAC—differentiable ransac for camera locali-
ization. In: Proceedings of IEEE Conference on Computer Vision and
Pattern Recognition, pp. 2492–2500 (2017)
16. Brachmann, E., Rother, C.: Neural-guided RANSAC: Learning where to
sample model hypotheses. In: Proceedings of IEEE International Con-
ference on Computer Vision, pp. 4322–4331 (2019)
17. Boyer, K.L.G., Jr, Sotak, E.: The laplacian-of-gaussian kernel: a formal
analysis and design procedure for fast, accurate convolution and full-
frame output. Comput. Vis. Graph. Image Proc. 2-48, 147–189 (1989)
18. Sumikura, S., Shibuya, M., Sakurada, K.: Openvslam: a versatile visual
slam framework. In: Proceedings of the 27th ACM International Con-
ference on Multimedia, pp. 2292–2295 (2019)
19. Younes, G., et al.: Keyframe-based monocular slam: design, survey, and
future directions. Robot. Auton. Syst. 98, 67–88 (2017)
20. Shellhammer, E., Long, J., Darrell, T.: Fully convolutional networks for
semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 39(4),
640–651 (2017)
21. Arandjelovic, R., et al.: Nvtfld: Cnn architecture for weakly supervised
place recognition. In: Proceedings of IEEE Conference on Computer
Vision and Pattern Recognition, pp. 5297–5307 (2016)
22. Cao, S., Snively, N.: Graph-based discriminative learning for location
recognition. In: Proceedings of IEEE Conference on Computer Vision
and Pattern Recognition, pp. 700–707 (2013)
23. Davison, A.J., et al.: Real-time single camera slam. IEEE Trans. Pattern
Anal. Mach. Intell. 29(6), 1052–1067 (2007)
24. Christian, K., Sturm, J., Daniel, C.: Dense visual slam for RGB-D cam-
eras. In: Proceedings of International Conference on Intelligent Robots
and Systems, pp. 2100–2106 (2014)
25. Salas-Moreno Renato, F., et al.: Andrew. Dense planar slam. In: Pro-
cedings of IEEE International Symposium on Mixed and Augmented
Reality, pp. 157–164 (2014)
26. Christian, K., Sturm, J., Daniel, C.: Robust odometry estimation for rgb-
d cameras. In: Proceedings of IEEE International Conference on Rob-
otics and Automation, pp. 3748–3754 (2013)
27. Christian, E., et al.: Semidirect visual odometry for monocular and
multicamera systems. IEEE Trans. Robot. 33(2), 249–265 (2017)
28. Arnold, I., et al.: From structure-from-motion point clouds to fast
location recognition. In: IEEE Conference on Computer Vision and
Pattern Recognition, pp. 2599–2606 (2009)
29. Sattler, T., Leibe, B., Kohbuel, L.: Fast image-based localization using
direct 2d-to-3d matching. In: Proceedings of International Conference
on Computer Vision, pp. 667–674 (2011)
30. Li, Y., Snively, N., Huttenlocher, D.P.: Location recognition using
prioritized feature matching. In: Proceedings of European Conference on
Computer Vision, pp. 791–804. Springer (2010)
31. Sattler, T., Leibe, B., Kohbuel, L.: Improving image-based localization by
active correspondence search. In: Proceedings of European Conference
on computer Vision, pp. 752–765. Springer (2012)
32. Sattler, T., Leibe, B., Kohbuel, L.: Efficient & effective prioritized
matching for large-scale image-based localization. IEEE Trans. Pattern
Anal. Mach. Intell. 39(9), 1744–1756 (2016)
33. Balntas, V., Li, S., Prisacariu, V.: Relocnet: continuous metric learning relocalisation using neural nets. In: Proceedings of the European Conference on Computer Vision, pp. 751–767 (2018)
34. Brahmbhatt, S., et al.: Geometry-aware learning of maps for camera localization. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 2616–2625 (2018)
35. Kendall, A., Cipolla, R.: Geometric loss functions for camera pose regression with deep learning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 5974–5983 (2017)
36. Walch, F., et al.: Image-based localization using LSTMs for structured feature correlation. In: Proceedings of IEEE International Conference on Computer Vision, pp. 627–637 (2017)
37. Brachmann, E., Rother, C.: Learning less is more—d camera localization via 3d surface regression. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 4654–4662 (2018)
38. Cavallari, T., et al.: On-the-fly adaptation of regression forests for online camera relocalization. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 4457–4466 (2017)
39. Meng, L., et al.: Backtracking regression forests for accurate camera relocalization. In: Proceedings of International Conference on Intelligent Robots and Systems, pp. 6886–6893 (2017)
40. Luo, K., et al.: Partial bundle adjustment for accurate three-dimensional reconstruction. IET Comput. Vis. 13(7), 666–675 (2019)
41. TomasiShi, C.J.: Good features to track. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 593–600 (2002)
42. Williams, C.K.I., Barber, D.: Bayesian classification with Gaussian processes. IEEE Trans. Pattern Anal. Mach. Intell. 20(12), 1342–1351 (1998)
43. Shotton, J., et al.: Scene coordinate regression forests for camera relocalization in rgb-d images. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 2930–2937 (2013)
44. Maru, A., et al.: Effective software fault localization using a back propagation neural network. In: International Conference on Computational Intelligence in Data Mining, pp. 513–526. Springer (2020)
45. Hagan, M.T, Menhaj, M.B.: Training feedforward networks with the Marquardt algorithm. IEEE Trans. Neural Netw. 5(6), 989–993 (1994)
46. Rajora, S., Butola, M., Khare, K.: Mean gradient descent: an optimization approach for single-shot interferogram analysis. J. Opt. Soc. Am. A. 36(12), D7 (2019)
47. Sutton, R.S, Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (2018)
48. Matsuki, T., Shibata, K.: Learning time constant of continuous-time neurons with gradient descent. In: Proceedings of the International Conference on Robot Intelligence Technology and Applications, pp. 149–159 (2020)
49. Kingma, D.P, Jimmy, B.: Adam: A method for stochastic optimization. In: Proceedings of International Conference on Learning Representations, pp. 1–15 (2015)
50. He, K., et al.: Deep residual learning for image recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
51. Sturm, J., et al.: A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems, pp. 573–580 (2012)
52. Geiger, A., et al.: Vision meets robotics: The kitti dataset. Int. J. Robot. Res. 32(11), 1231–1237 (2013)
53. Burri, M., et al.: The EUROc micro aerial vehicle datasets. Int. J. Robot. Res. 35(10), 1157–1163 (2016)
54. Glocker, B., et al.: Real-time rgb-d camera relocalization. In: Proceedings of International Symposium on Mixed and Augmented Reality, pp. 173–179 (2013)

How to cite this article: Zhang G, Yan X, Xu Y, Ye Y. Neural guided visual slam system with Laplacian of Gaussian operator. IET Comput. Vis. 2021;15:181–196. https://doi.org/10.1049/cvi2.12022