Hindawi
Scientific Programming
Volume 2023, Article ID 9789583, 1 page
https://doi.org/10.1155/2023/9789583

Retraction

Retracted: SLAM 3D Digital Terrain Mapping with SqueezeNet Driven by Road Traffic Data

Scientific Programming

Received 8 August 2023; Accepted 8 August 2023; Published 9 August 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] L. Gu, H. Zhang, and X. Wu, "SLAM 3D Digital Terrain Mapping with SqueezeNet Driven by Road Traffic Data," *Scientific Programming*, vol. 2022, Article ID 9562527, 9 pages, 2022.

Hindawi
Scientific Programming
Volume 2022, Article ID 9562527, 9 pages
https://doi.org/10.1155/2022/9562527

Research Article

SLAM 3D Digital Terrain Mapping with SqueezeNet Driven by Road Traffic Data

Liuwan Gu,[1] **Hao Zhang**,[2] **and Xingjie Wu**[2]

[1]*School of Geographic Information and Tourism, Chuzhou University, Chuzhou, Anhui 239000, China*
[2]*College of Civil and Architecture Engineering, Chuzhou University, Chuzhou, Anhui 239000, China*

Correspondence should be addressed to Liuwan Gu; liuwan.gu@chzu.edu.cn

Received 30 December 2021; Accepted 22 February 2022; Published 24 March 2022

Academic Editor: Sheng Bin

In order to improve the efficiency of dynamic visualization of large-scale road traffic data in a web environment, this paper proposes a SLAM 3D digital terrain map visualization method using SqueezeNet in a web environment. We propose a hierarchical organization method of the road network, taking into account road attributes and drawing roads at different view heights; a multiroad merging method based on the line segment indexing feature of WebGL (web graphics library) technology, and optimizing the scene by combining view rejection and multithreading technology. The prototype system was developed and case studies were carried out using the national road network data as an example. The experimental results show that the frame rate of large-scale road traffic data visualization in the network environment is above 40 frames/second, which is 20–30 frames/second higher than that of Baidu's ECharts GL visualization method.

1. Introduction

Along with the urbanization boom, the contradiction between supply and demand between increasing urban car ownership and limited traffic resources has become more and more intense [1], and as a direct result, traffic congestion is a problem. Real-time visualization of road traffic information can clearly show the real-time traffic congestion status, reveal the trend of traffic flow changes, assist the traffic department to monitor the traffic flow information in real-time and timely release to society [2], allowing people to choose their travel time and route according to the traffic status in a more planned manner. However, the existing performance of the browser can hardly support the display of massive real-time traffic information [3]. Therefore, the key problem is how to efficiently visualize the massive amounts of real-time road traffic data in the web environment. Most of the existing web visualization technologies require the installation of browser plug-ins to achieve efficient rendering of massive data, but this poses certain difficulties in maintenance and updating later, and at the same time, browser plug-ins cannot meet users' expectations for the convenience and real-time of Web GIS [4, 5]. However, WebGL, as a plug-in-free web visualization technology, cannot directly support the high quality and efficient rendering of large-scale road traffic data.

Most domestic and foreign research on road visualization using WebGL technology reflects changes in traffic dynamics through changes in vehicle trajectory flow [6]. Public-oriented electronic products, such as Baidu Maps, Gaode Maps, and Tencent Maps, generally draw arrows on the map, using the direction, thickness, or color of the arrows to represent the direction of the trajectory and the degree of density [3], with poor visualization effects compared to animation. Data representation methods give a lower sense of stimulation to the human brain's thinking. [7] used Baidu's ECharts open-source visualization product to dynamically visualize the public transport trajectory in Chongqing; [5] visualized the vehicle trajectory by linking two and three dimensions, adding three-dimensional attribute values to the two-dimensional map to represent the speed and traffic flow, but the above method does not support the rendering of large-scale traffic data and still has shortcomings in efficiency.

In visualization systems, the use of appropriate animation and transition effects is an effective way to enhance the richness and comprehensibility of the visualization result view [8]. Streamlines and color gradients are two fundamental ways to improve visualization and have a huge impact on the visual effect and user interaction. Streamlines, which produce a flowing effect by constantly updating optical properties such as color and transparency [9], have a stronger visual impact than static lines, and color gradients can be used as an aid to increase the sensitivity of users to observe changes in traffic patterns. If a huge amount of road traffic data is rendered directly in the browser, there will be problems such as slow data acquisition, low transmission efficiency, and poor rendering effect.

Therefore, this paper proposes a lightweight visualization method for real-time road traffic data of large-scale streamlined networks. Firstly, we use the high-precision large-scale road network data as the object of study, based on which the road network data reconstruction is carried out. Based on the hierarchical organization of the road network based on road attributes, we systematically study a multiroad merging method based on the WebGL line segment indexing feature. Secondly, the scene is optimized using view rejection and multithreading techniques to design and implement animation effects such as fading and efficient real-time dynamic streamlines that support real-time changes in road congestion states. Finally, the experiment and analysis of large-scale streamline network visualization were carried out using the national road network data as an example.

2. Programme Designed

The color image obtained from the Kinect camera is extracted and matched with the previous frame, and the PnP (perspective-n-point) is solved based on the correctly matched feature information combined with the distance information of the depth image to obtain the pose transformation between the two frames [10]. The keyframes are selected by setting a reasonable range of motion, and the pose transformation information between the keyframes and the adjacent keyframes is stored. The current keyframe is loopback detected with all previous keyframes, and if loopback occurs, the two keyframes are subjected to feature extraction, feature matching, and PnP solving. Then all the positional information is globally optimized using the graph optimization tool, and the point cloud map is stitched together based on the optimized information.

2.1. Feature Extraction. The main problem in building a visual odometer is how to compute the positional transformation of the camera from the image, so it is desirable to find some specific points in the image. Traditional visual SLAM algorithms usually use SIFT or SURF features, which are invariant, but their computations are all very time-consuming and cannot meet the real-time requirements of SLAM. In [11], the ORB algorithm is proposed, which is faster than SURF and SIFT and has good invariance. For this reason, this paper adopts the ORB algorithm for feature extraction.

The ORB operator consists of two parts: the key point, called "Oriented FAST (features from accelerated segment test)," is a modified FAST corner point [12]; the descriptor is called BRIEF (binary robust independent elementary features). FAST corner points are places in an image where the grey gradient varies considerably, as shown in Figure 1, where f1 is the order in the horizontal direction. First, a pixel is taken from the image and compared with the grey level of 16 pixels on a circle of radius 3, and if N consecutive points are greater or less than a set threshold, the point is considered a corner point.

To address the issue of corner point orientation, scale and rotation characteristics were added to the description. Scale invariance is achieved by constructing an image pyramid and detecting feature points for each layer of the image and then identifying the commonly detected feature points as the correct detection result. Rotation properties are described by the greyscale center of mass method, which determines the center of mass of an image from the greyscale values [13]. The image moments in an image block M are defined as

$$m_{f_1 f_2} = \sum_{x,y \in M} x^{(f_1)} y^{(f_2)} I(x, y), \tag{1}$$

where $m_{f_1 f_2}$ is the image moment, f_2 is the order in the vertical direction, and $I(x, y)$ is the greyscale value at point (x, y).

The coordinates of the centre of mass C of the image block are $((m_{10}/m_{00}), (m_{01}/m_{00}))$, and the direction vector OC is obtained by connecting the geometric centre O of the image block to the centre of mass C.

BRIEF is a binary descriptor of a vector of 0's and 1's that encodes the relationship between the sizes of two pixels near a key point. If 128 key points are taken, a 128-dimensional vector of 0's and 1's is obtained. Combining the rotational properties of the FAST corner point results in a BRIEF descriptor with rotational properties.

2.2. Feature Matching and Pn P Solving. After feature extraction of the image, a fast nearest-neighbor search algorithm is used for feature matching. A random K-D tree is built to find the corresponding point of a feature in another image, and then the Hamming distance between the BRIEF descriptors of the two feature points is calculated. The distance of the smallest matching point in the image is denoted as D. A threshold of 4 times this distance is used as a filtering criterion to filter all the matching points, and the points smaller than this distance are considered the correct matching points. Finally, the same operation is performed on the other image, and the points that are matched together are considered the final matching points.

Pn P is a method for solving 3D to 2D point pair motion [14]. If the spatial 3D coordinates of the feature points of one of the two images are known, at least three pairs of points are needed to estimate the motion of the camera. The spatial 3D coordinates of the feature points can be determined from the depth map of the camera. There are various methods for solving the Pn P problem, such as P3P, direct linear transformation, and nonlinear optimization.

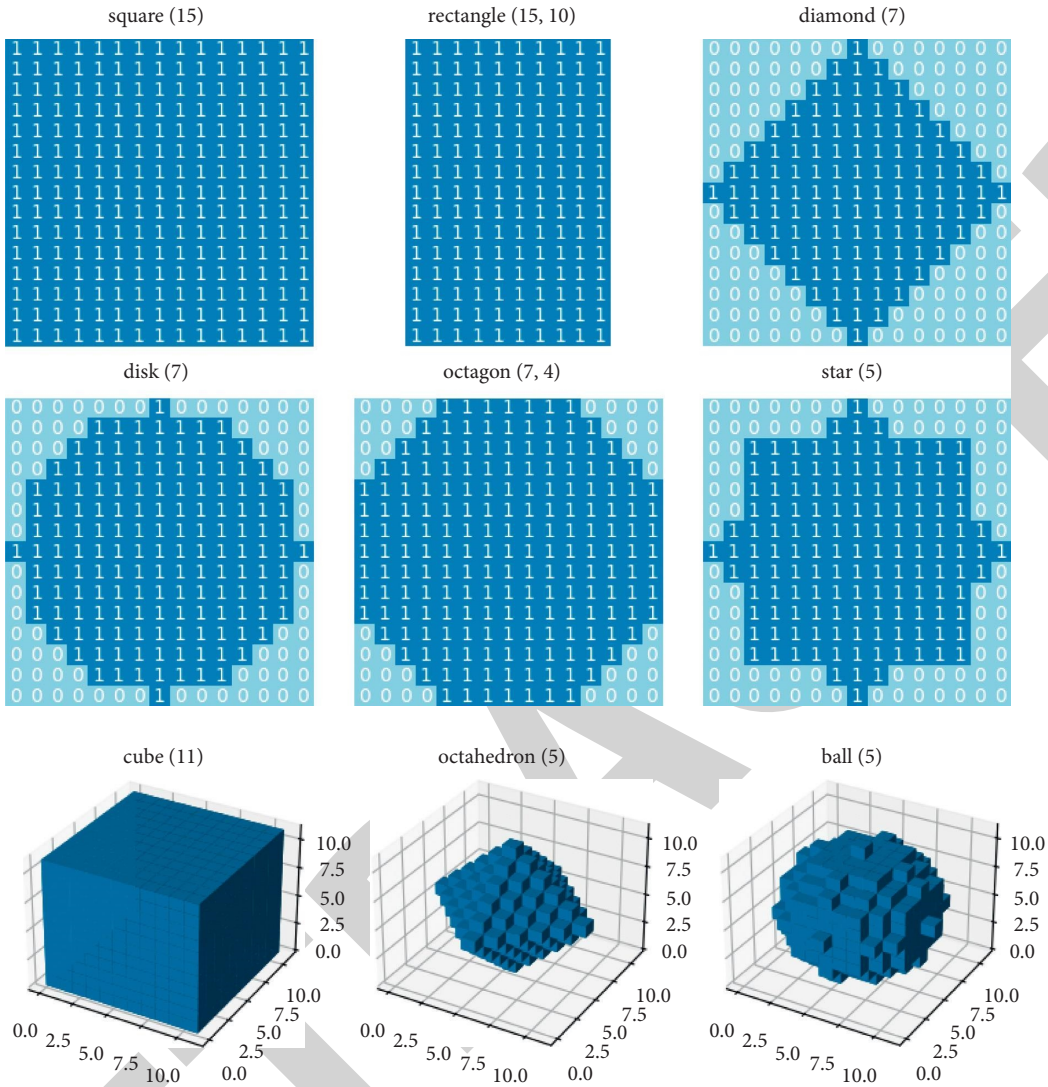

Figure 1: Diagram of FAST corner.

As shown in Figure 2, for a feature point P in space, P_1 is a reprojection point due to the presence of observation noise, there is a deviation between the observed position (pixel coordinates) $\hat{p}_2$ and the projection position p_2 calculated according to the current camera pose, which is the reprojection error, and e is the internode error.

Suppose that the rotation matrix of the spatial point $P_i = (X_i, Y_i, Z_i), i = 1, 2, \cdots, n$. The rotation matrix of the camera's locus is R and the displacement vector is t. We construct the p_n problem as a nonlinear least-squares problem defined by minimizing the reprojection error on the Lie algebra, denoting the Lie algebra of R and t by ξ, and p_i the pixel coordinates of the reprojection point $p(\xi^{\wedge})$ in this locus are (u_i, v_i), then

$$s_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = K \exp(\xi^{\wedge}) \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}, \tag{2}$$

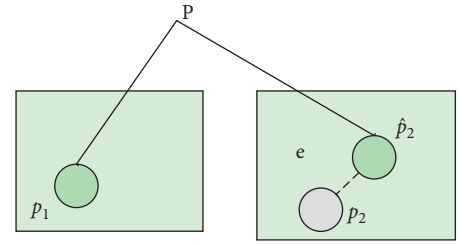

Figure 2: Error because of reprojection.

where s_i is the depth at point i; K is the conversion factor, and $\xi^{\wedge}$ is the Lie algebraic antisymmetric matrix.

If there are n pairs of matching features, the least-squares problem can be built based on the reprojection error mentioned above, as in the following equation:

$$\xi^* = \text{argmin}_{\xi} \frac{1}{2} \sum_{i=1}^{n} \left\| p_i - \frac{1}{s_i} K \exp(\xi^{\wedge}) P_i \right\|_2^2, \tag{3}$$

where ξ^* is the Lie algebraic reprojection.

This least-squares problem can be solved by the Gauss–Newton method or the Levernberg–Marquardt method. In SLAM, the process of minimizing the error by adjusting the camera poses and the coordinates of the waypoint is called BA (bundle adjustment) [15].

2.3. Loopback Detection. If the visual odometry is built considering only the motion between adjacent frames, it may lead to an accumulation of errors, and if the camera can be detected passing the same position, then constraints can be added to the estimated camera motion trajectory to make it closer to the real trajectory [16].

A convolutional neural network is a feed-forward neural network that consists mainly of convolutional and pooling layers [17]. Compared with the artificially designed image features in traditional computer vision, the image features extracted by CNN can better reflect the real characteristics of images and are much more effective than traditional computer vision methods in the field of image classification and recognition. Therefore, in this paper, CNN is used instead of the traditional bag-of-words method to perform loopback detection.

SqueezeNet was designed by UC Berkeley and Stanford researchers, not to achieve the best CNN recognition accuracy, but to simplify the complexity of the network and achieve the recognition accuracy of public networks. The core composition of SqueezeNet is the Fire module. SqueezeNet consists of 8 Fire modules from fire2 to fire9, where the structure of fire2 is shown in Figure 3, where H and W are the height and width of the feature map, respectively, and e_3 is the number of channels.

SqueezeNet reduces the number of parameters and flops in the network by means of Fire modules, each of which consists of squeeze and expand modules. The squeeze module uses e1 1×1 convolutional kernels to reduce the dimensionality of the feature map, and the feature map is $H \times W \times e_1$ after ReLu activation. The feature map is $H \times W \times e_2$ after the two taps of expand (with 1×1 and 3×3 convolutional kernels, respectively) are computed separately, and then the feature maps of the two taps are stitched together by the concat layer to generate $H \times W \times e_3$. $e_2 = 4e1$ and $e_3 = 2\ e_2 =$ Since the convolutional kernels of the fire network are identical in size, only the number of channels differs. Figure 4 shows that the fire structure is labeled only with the number of channels of the fire structure output e_3.

When the network input is a $224 \times 224 \times 3$ RGB image, the network computation is only 837 MFlops, so the network structure is more suitable for light-weight devices such as intelligent mobile robots. The network structure of SqueezeNet used in this paper is shown in Figure 4 [18].

The image is fed into the SqueezeNet network as a 224×224 three-channel RGB image, and the 1 000-dimensional array A = (a1, a2, a3, . . ., a1 000) is extracted as the feature vector of this image. Let us assume that the feature vector of the other image is B = (b1, b2, b3, . . ., b1 000). The cosine similarity of the eigenvectors of these two

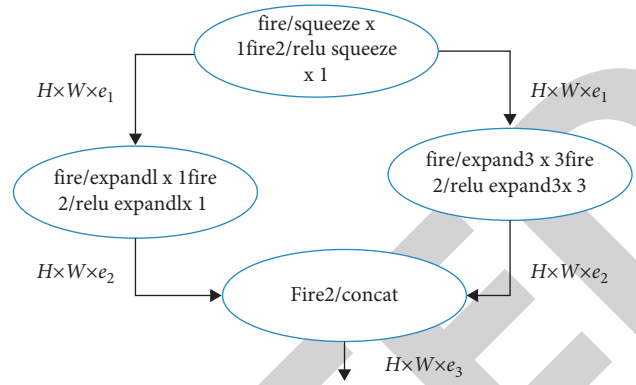

Figure 3: Structure of fire2.

images can be used to determine whether the two images are similar, as in the following equation:

$$\cos\ \theta = \frac{A \cdot B}{\|A\| \times \|B\|}. \tag{4}$$

If the cosine similarity is greater than the set threshold, then loopback is considered to have occurred. Otherwise, we skip to the next frame and compare with the feature vector of the input image again until loopback occurs.

2.4. Global Optimization. In the SLAM process, the global optimization of the back-end needs to be considered. Camera motion can be represented by a rose diagram, as shown in Figure 5, where each node represents the camera pose for each keyframe and the edges between the nodes represent the camera pose transformation between two keyframes.

The camera's pose has 6 degrees of freedom, 3 axial positions, and torques around 3 axes, denoted by $\mathbf{w}_k$, i.e., $\mathbf{w}_k = [\, x_k \ \ y_k z_k \ \ \gamma_k \ \ \kappa_k \vartheta_k \,], k = 1, 2, \cdots, j$.

The transformation relationship between the two poses is

$$\mathbf{E}_{k,j} = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{t}_{3\times1} \\ \mathbf{0}_{1\times3} & \mathbf{1} \end{bmatrix}. \tag{5}$$

The error between the two connected nodes k and j is

$$e\left(x_k, x_j, E_{k,j}\right) = x_k^* - \mathbf{E}_{k,j} x_j^*, \tag{6}$$

where x_k^* is the estimated value of node k.

Summing up all the internode errors and scalarizing them gives a total error of

$$E = \sum_{k,j\in F} e\left(x_k, x_j, E_{k,j}\right)^{\mathrm{T}} \Omega_{k,j} e\left(x_k, x_j, E_{k,j}\right), \tag{7}$$

where F is the set of nodes and $\Omega_{k,j}$ is the information matrix between x_k and x_j.

The optimized poses can be obtained by solving the least squares problem. In this paper, we use the generic graphical optimization g2o to obtain the optimised poses [19]. The types of vertices and edges are defined, information about the nodes and edges is added, and a suitable solver and iterative algorithm for the optimization algorithm are

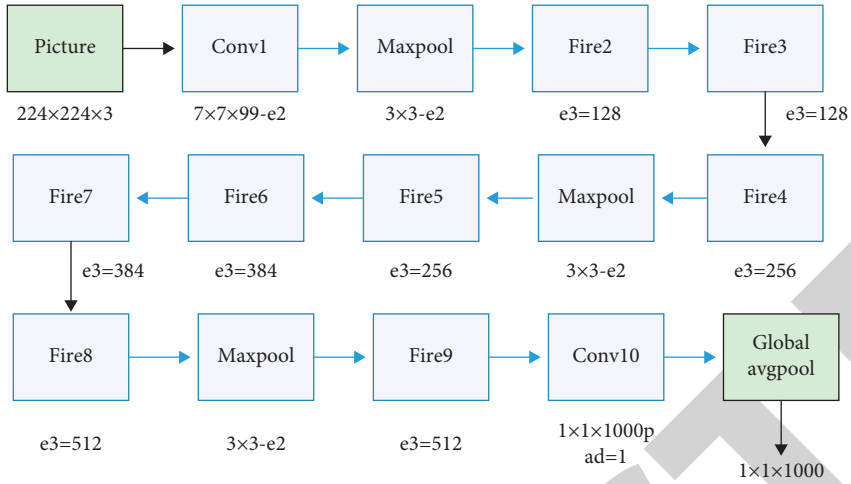

Figure 4: Network structure of SqueezeNet.

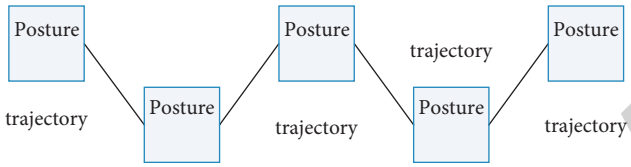

Figure 5: Pose graph.

Table 1: Datasets used in the experiment.

Serial number	Dataset	Number/frame
1	Nyuv2	782
2	TUM fr3	938
3	Turtlebot2 acquisition	968
4	TUM fr1_xyz	902

selected. g2o provides three different linear solvers: CSparse, Cholmod based Cholsky decomposition, and PCG based preconditioned conjugate gradient, and iterative algorithms including Gaussian–Newton, Levernberg–Marquardt, and Powell's dogleg.

3. Test Results and Analysis

The experiments were conducted on a Turtlebot2 robotics platform with a Kinect V1 depth camera and a NvidiaTX2 development board [20]. The algorithms were run on an Ubuntu 14.04 system, involving the ROS robotics operating system and open-source libraries such as Caffe, OpenCV, Eigen, PCL, and g2o [21]. The experimental datasets include the TUM (technical university of music) [22] dataset, the nyuv2 dataset, and the dataset collected by Turtlebot2 itself, as shown in Table 1. The TUM dataset was acquired by the Computer Vision Group of the Technical University of Munich [23] and is based on a depth camera with a high-precision motion sensor that allows the acquisition of color and depth images along with the true motion of the camera. The TUM dataset also presents a method to calculate the trajectory error between the estimated trajectory of the SLAM algorithm and the standard trajectory acquired by the high-precision motion sensor.

3.1. Comparison with the Algorithm for Loopback-Free Detection. Dataset acquired indoors using Turtlebot2, consisting of 968 frames of color images with a resolution of 640×480 pixels and corresponding depth images, acquired at a rate of 30 frames/s.

The results of the building estimation with and without loopback detection are shown in Figure 6. When the algorithm without loopback detection is used to build the map, the camera trajectory drifts and fails to close due to the accumulation of errors. In contrast, when this algorithm is used, the motion constraint is added to the loopback due to the successful detection of the loopback, resulting in a closed motion trajectory [24].

Figure 7 shows the point cloud map obtained when using the loopback detection algorithm and the loopback detection algorithm in this paper. The point cloud maps largely overlap at the locations where loops occur, indicating that the estimated trajectory is in error from the true trajectory. This shows that this algorithm can successfully detect loops and add constraints to the global trajectory optimization, making the estimated motion and the built map closer to the real situation.

3.2. Compare and Contrast with the Bag-of-Words Method. The algorithm to determine whether a loopback has occurred may be different from the fact that there are roughly 4 different scenarios, as shown in Table 2.

Accuracy T_a and recall T_r are used to measure the goodness of the loopback detection results, as in equations (8) and (9).

$$T_a = \frac{T_1}{T_1 + T_2}, \tag{8}$$

$$T_r = \frac{T_1}{T_1 + T_3}, \tag{9}$$

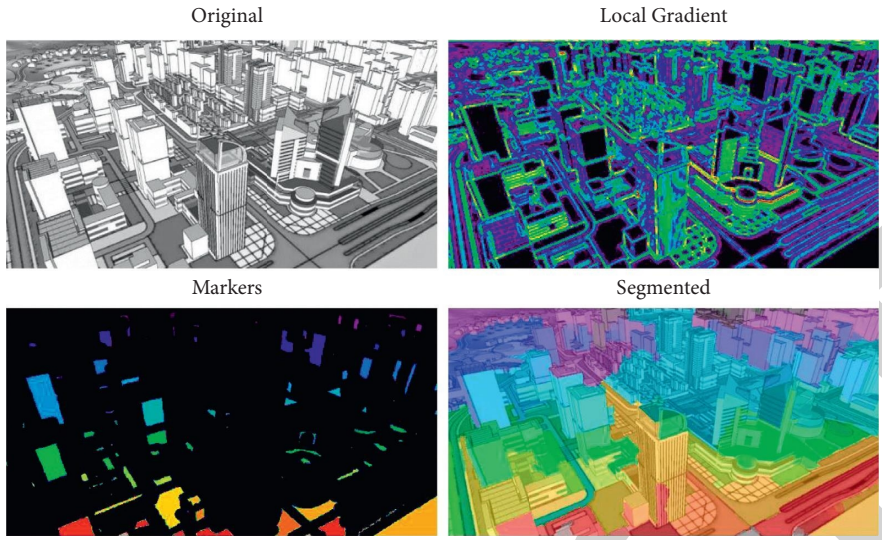

Figure 6: Comparison of building estimates.

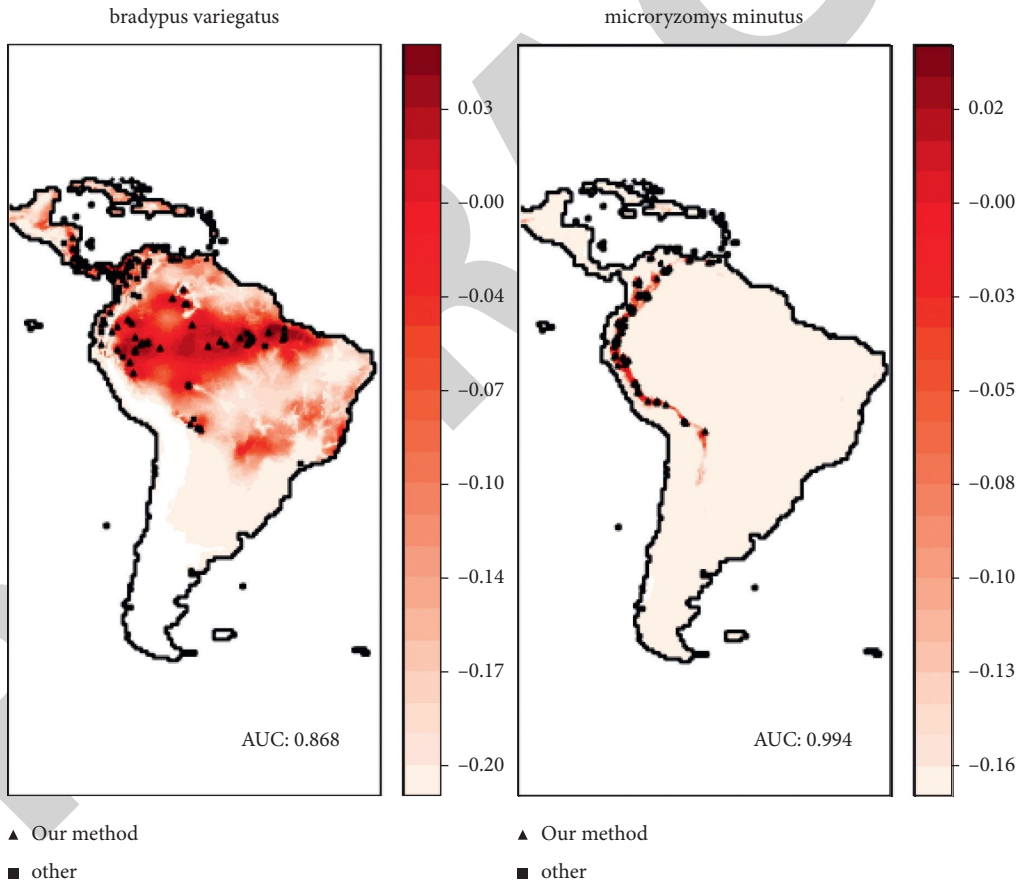

Figure 7: Comparison of point cloud plots (red circles are where loops appear).

Table 2: Possible results of loopback testing.

Detection result	In fact, there is a loopback	In fact, there is no loopback
It is a loop	True positive	False positive
Not a loop	False negative	True negative

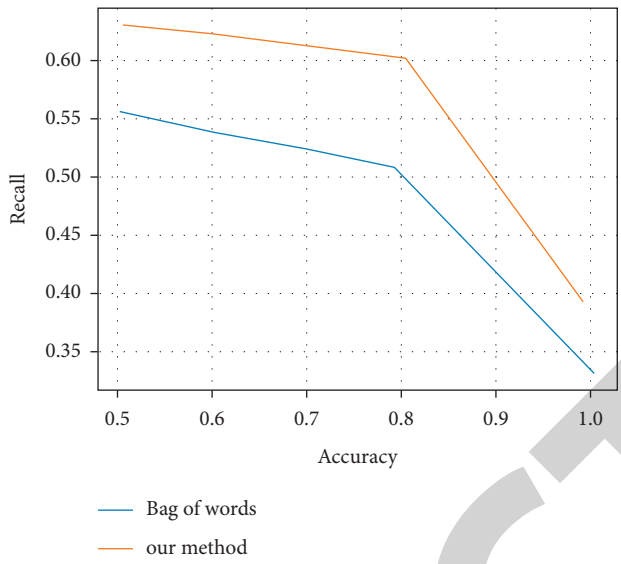

Figure 8: Accuracy and recall curves of this algorithm compared with the bag-of-words method.

Table 3: Comparison of our algorithm with the bag-of-words method.

Algorithm	Recall rate with an accuracy of 1	Time taken for similarity calculation of two pictures/S
Word bag method	0.33	0.50
Paper algorithm	0.40	0.13
Difference between the two	0.07	0.37
Percentage%	21	74

where T_1, T_2, and T_3 are the number of true positives, false positives, and false negatives, respectively.

The loopback detection algorithm and bag-of-words method proposed in this paper were tested on the new college dataset and the self-made dataset, respectively, and the experimental results of the accuracy and recall comparison were obtained as shown in Figure 8, and the specific data are shown in Table 3.

As can be seen from Figure 8, the recall rate of this algorithm is significantly higher than that of the bag-of-words method for the same accuracy. Since accuracy is much more important than recall in loopback detection algorithms, a false positive result will have a very bad impact on the final map, so it is important to ensure that the accuracy is 1. The recall of this algorithm is 0.4 when the accuracy is 1, while the recall of the bag-of-words method is 0.33. The algorithm improves by about 21% compared to the bag-of-words method, and the time taken to calculate the similarity of the two images is reduced by 74%. The time taken to calculate the similarity between the two images was reduced by 74%.

3.3. Accuracy of Build. The Computer Vision Group at the Technical University of Munich, Germany, provides a measure of trajectory error and absolute trajectory error (ATE) in the TUM dataset [25–28]. ATE is obtained by calculating the root mean square error of the estimated trajectory and the standard trajectory. Assuming that the standard trajectory of the camera is $\mathbf{X} = \{X_1, X_2, \cdots, X_n\}$ and the estimated trajectory of the camera is $\widehat{X} = \{\widehat{X}_1, \widehat{X}_2, \ldots, \widehat{X}_n\}$, the root mean square of the absolute trajectory error is

$$A(\widehat{X}, X) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\text{trans}\left(\widehat{X}_i\right) - \text{trans}\left(X_i\right)\right]^2}, \tag{10}$$

where trans(-) is the function to obtain the gesture translation vector.

Figure 9 shows the estimated trajectory of the algorithm and the standard trajectory obtained by the high-precision motion sensor. Due to the high acquisition frequency of the high-precision motion sensors used in the TUM dataset, the trajectory information obtained is much richer.

Table 4 compares the mapping accuracy of the algorithms. The total acquisition time of the TUM fr3_stf dataset used is 31.55 s, the total length of the standard trajectory is 5.884 m, and the average camera movement speed is 0.193 m/s. The root mean square error of the absolute trajectory estimated by the algorithm is 0.050 0 m. The same mapping was performed using RGB-D SLAM [23], and the root mean square error of the absolute trajectory is 0.049 0 m. The RMS value of the absolute trajectory error is 0.049 0 m. From Figure 9(a), it can be seen that there is no loopback in TUM fr3_stf, and the error of this algorithm is similar to that of RGB-DSLAM.

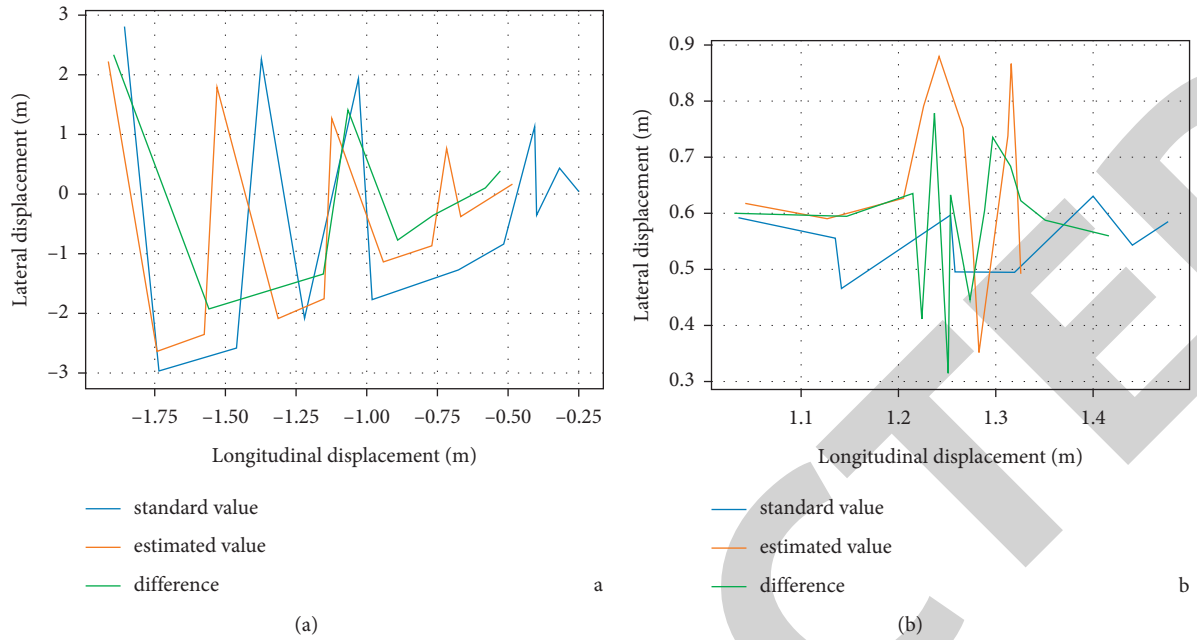

Figure 9: Standard trajectory, estimated trajectory, and trajectory error. (a) TUM fr3_stf dataset. (b) TUM fr3_xyz dataset.

Table 4: Comparison of mapping errors.

Data set	Algorithm error in this paper/M	Literature algorithm error/M	Difference%
TUM fr3_stf	0.005	0.049	2
TUM fr3_xyz	0.010	0.013	29

The data collection time of TUM fr3_xyz used is 30.09 s, the total length of the standard trajectory is 7.112 m, and the average speed of camera movement is 0.244 m/s. The root mean square (RMS) of the absolute trajectory error is 0.010 4 m. The RMS of the absolute trajectory error is 0.013 5 m when the same RGB-D SLAM is used to build the map. From Figure 9(b), we can see that there are more loops in TUM fr3_xyz, and the error in building the map is reduced by 29% compared to RGB-D SLAM.

4. Conclusions

This paper presents a lightweight visualization method for SLAM 3D digital terrain maps using SqueezeNet in a web environment. This paper proposes a multiroad merging method based on the line indexing feature of WebGL (web graphics library) technology and optimizes the scene by combining the view rejection and multithreading technology and designing and implementing the animation effects such as fading and dynamic streamlines that support the real-time change of road congestion state. Finally, the prototype system was developed, and case experiments and analyses were carried out with national road network data. The experimental results show that, compared with RGB-D SLAM, the algorithm described in this paper is more suitable for the dynamic visualization of large-scale road traffic data in a network environment with a 29% reduction in the error of map construction.

Data Availability

The data underlying the results presented in the study are available within the manuscript.

Conflicts of Interest

The authors declare that there are no conflicts of interest with this study. All authors have seen the manuscript and approved it for publication.

Acknowledgments

This work was sponsored in part by the Provincial Natural Science Foundation Project of Anhui(Research on BIM + GIS 3D Scene Modeling and Integration Method) (KJ2021A1082); the Anhui Construction Science and Technology Plan Project (Construction and Key Technology Research of Intelligent Fire Fighting Platform based on BIM + GIS, 2021-YF11); the Key Projects of Chuzhou Science and Technology Plan(Research on Key Technologies and Demonstration Application of Intelligent Fire Protection based on BIM, 2020ZN004).

References

[1] Y. Motoya, T. Toshio, and Y. Keiichi, "A 3-D model for optimal alignment search system OF highway design BY extended digital mapping data," *Doboku Gakkai Ronbunshu D*, vol. 64, no. 4, pp. 580–585, 2009.

[2] W. E. Featherstone and J. F. Kirby, "The reduction of aliasing in gravity anomalies and geoid heights using digital terrain data," *Geophysical Journal of the Royal Astronomical Society*, vol. 141, no. 1, pp. 204–212, 2010.

[3] C. d. S. Chagas, E. I. Fernandes Filho, C. A. O. Vieira, C. E. G. R. Schaefer, and W. d. Carvalho Júnior, "Atributos topográficos e dados do Landsat7 no mapeamento digital de solos com uso de redes neurais," *Pesquisa Agropecuária Brasileira*, vol. 45, no. 5, pp. 497–507, 2010.

[4] J. Deng, X. Huang, Q. Feng, X. Ma, and T. Liang, "Toward improved daily cloud-free fractional snow cover mapping with multi-source remote sensing data in China," *Remote Sensing*, vol. 7, no. 6, pp. 6986–7006, 2015.

[5] B. Direito, C. Teixeira, B. Ribeiro, M. Castelo-Branco, F. Sales, and A. Dourado, "Modeling epileptic brain states using EEG spectral analysis and topographic mapping," *Journal of Neuroscience Methods*, vol. 210, no. 2, pp. 220–229, 2012.

[6] M. Trojnowicz, "Local modelling of the disturbing potential with the use of quasigeoid heights, grvimetric data and digital terrain model," *Geodezja I Kartografia*, vol. 51, pp. 35–43, 2002.

[7] I. A. Sadisun, J. A. Telaumbanua, and R. D. Kartiko, "Weight of evidence method for landslide susceptibility mapping in sigi biromaru, central sulawesi," *IOP Conference Series: Earth and Environmental Science*, vol. 830, no. 1, Article ID 012029, 2021.

[8] Z. Li, L. Chen, C. Liu et al., "Animated 3D human avatars from a single image with GAN-based texture inference," *Computers & Graphics*, vol. 95, pp. 81–91, 2021.

[9] T. Toutin, "Three-dimensional topographic mapping with ASTER stereo data in rugged topography," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 10, pp. 2241–2247, 2002.

[10] Y. Yang, H. U. Nan, and L. Guo, "On surveying and mapping technology for 3D digital topographic map," *Bulletin of Surveying and Mapping*, vol. 85, no. 1, pp. 75–77, 2011.

[11] M. Peng, W. Wan, and K. Wu, "Topographic mapping capbility analysis of Chang'e-3 Navcam stereo images and three-dimensional terrain reconstruction for mission operations," *Journal of Remote Sensing*, vol. 18, no. 5, pp. 995–1002, 2014.

[12] M. Tatsuro, O. Kiyoshi, and F. Masaaki, "Traffic performance of a bulldozer running on a weak terrain - vehicle model test," *Doboku Gakkai Ronbunshu*, vol. 1988, no. 397, pp. 151–157, 2010.

[13] C. Hu, Y. H. Zhou, C. J. Zhao, and Z. G. Pan, "Hydraulic engineering topographic mapping and modeling based on three dimensional laser scanning technology," *Applied Mechanics and Materials*, vol. 744-746, pp. 1695–1700, 2015.

[14] J. A. Beraldin, J. Wang, and Y. Shen, "Automatic 3D high-fidelity traffic interchange modeling using 2D road GIS data," *International Society for Optics and Photonics*, vol. 7864, Article ID 78640U, 2011.

[15] J. S. Ronning, G. V. Ganerod, and E. Dalsegg, "Resistivity mapping as a tool for identification and characterisation of weakness zones in crystalline bedrock: definition and testing of an interpretational model," *Bulletin of Engineering Geology and the Environment*, vol. 73, no. 4, pp. 1225–1244, 2014.

[16] R. Branko and P. Jennifer, "Autonomous exploration and mapping with RFS occupancy-grid SLAM," *Entropy*, vol. 20, no. 6, p. 456, 2018.

[17] C. H. Cao, Y. N. Tang, and D. Y. Huang, "IIBE: an improved identity-based encryption algorithm for WSN security," *Security and Communication Networks*, vol. 2021, 2021.

[18] D. Wu, C. Zhang, L. Ji, R. Ran, H. Wu, and Y. Xu, "Forest fire recognition based on feature extraction from multi-view images," *Traitement du Signal*, vol. 38, no. 3, pp. 775–783, 2021.

[19] T. Xie, C. Zhang, and Y. Xu, "Collaborative parameter update based on average variance reduction of historical gradients," *Journal of Electronics and Information Technology*, vol. 43, no. 4, pp. 956–964, 2021.

[20] H. Li, D. Zeng, and L. Chen, "Immune multipath reliable transmission with fault tolerance in wireless sensor networks," in *Proceedings of the International Conference on Bio-Inspired Computing: Theories and Applications*, pp. 513–517, Springer, Singapore, 2016.

[21] C. Fürst, H. König, and K. Pietzsch, "Pimp your landscape - a generic approach for integrating regional stakeholder needs into land use planning," *Ecology and Society*, vol. 15, no. 3, pp. 299–305, 2010.

[22] F. Oniga and S. Nedevschi, "Processing dense stereo data using elevation maps: road surface, traffic isle, and obstacle detection," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 3, pp. 1172–1182, 2010.

[23] F. Qiu, M. Abdelsalam, and P. Thakkar, "Spectral analysis of ASTER data covering part of the Neoproterozoic Allaqi-Heiani suture, Southern Egypt," *Journal of African Earth Sciences*, vol. 44, no. 2, pp. 169–180, 2006.

[24] A. Sujiwo, T. Ando, E. Takeuchi, and Y. Ninomiya, "Monocular vision-based localization using ORB-SLAM with LIDAR-aided mapping in real-world robot challenge," *Journal of Robotics and Mechatronics*, vol. 28, pp. 479–490, 2016.

[25] H. Saybasili, A. Z. Faranesh, C. E. Saikus, C. Ozturk, R. J. Lederman, and M. A. Guttman, "Interventional MRI using multiple 3D angiography roadmaps with real-time imaging," *Journal of Magnetic Resonance Imaging*, vol. 31, no. 4, pp. 1015–1019, 2010.

[26] F. B. P. Fiorini, "Visual slam - mobile robot localization with environment mapping," *IFAC Proceedings Volumes*, vol. 39, no. 15, pp. 286–291, 2006.

[27] S.-Y. Kim and K.-W. Lee, "Development of mobile 3D terrain viewer with texture mapping of satellite images," *KOREAN JOURNAL OF REMOTE SENSING*, vol. 22, no. 5, pp. 351–356, 2006.

[28] J. Clemens, T. Kluth, and T. Reineking, "β -SLAM: simultaneous localization and grid mapping with beta distributions," *Information Fusion*, vol. 52, pp. 62–75, 2019.