DT-SLAM: Dynamic Thresholding Based Corner Point Extraction in SLAM System

R. WU, M. PIKE, (Member, IEEE), AND B. G. LEE, (Member, IEEE)
School of Computer Science, Faculty of Science and Engineering, The University of Nottingham Ningbo China, Ningbo 315100, China
Corresponding author: B. G. Lee (boon-giin.lee@nottingham.edu.cn)
This work was supported by the Zhejiang Provincial Natural Science Foundation of China under Grant LQ21F020024.

ABSTRACT Visual localization estimation is highly depended on the quality of video frames or captured images. Estimation quality may be affected by the poor visibility, low background texture and overexposure. Low quality frames with blurred edges and poor contrast pose tremendous difficulties for corner point detection in SLAM impacting the overall accuracy of estimation. This paper introduces DT-SLAM, a dynamic self-adaptive threshold (DSAT) approach for ORB corner point extraction in FAST to improve SLAM’s localization performance. The proposed method replaces the existing static threshold-based ORB extraction approach, enabling improved performance in complex real-world scenes. In addition, this study introduces a threshold switching mechanism (TSM) to replace the existing SLAM’s frame-level and cell-level thresholds for corner point extraction. The proposed DT-SLAM approach is validated using the TUM RGB-D and EuRoC benchmark datasets for location tracking performances. The results indicate that the proposed DT-SLAM outperforms the current state-of-the-art ORB-SLAM3, especially in challenging environments.

INDEX TERMS SLAM, corner point detection, localization, indoor-mapping, computer vision, image processing.

I. INTRODUCTION
The ability for an agent to localize and track its movement through an otherwise unmapped environment is important functionality in application areas such as autonomous robotics, self-driving cars and Unmanned Aerial Vehicles (UAVs). Often, agents will operate in environments that are either not mapped or cannot be mapped practically using conventional means, e.g., home-based environments. Additionally, agents will typically operate on constrained hardware which may lack network connectivity, requiring localization to be computed on the agent itself. Simultaneous Localization and Mapping (SLAM) is a widely used technique that allows the mapping of an unknown environment whilst tracking the position of the agent in that environment. The ubiquity of high-performance processors and high-resolution color cameras had seen the popularization of visual SLAM (vSLAM). vSLAM applies visual odometry (VO) [1], [2] to determine the position and orientation of an observer using the camera images.

The associate editor coordinating the review of this manuscript and approving it for publication was Jinjia Zhou.

Since the introduction of Parallel Tracking and Mapping (PTAM) [3] in 2007, many real-time vSLAM implementations have followed a multiple-thread strategy, whereby tracking and mapping are split into two separate tasks. Examples include, corner point-based ORB-SLAM [4], RGBD-SLAM [5], direct method-based LSD-SLAM [6] and SVO [7] which optimizes pose directly on image intensities. Among these, the third incarnation of ORB-SLAM (ORB-SLAM3) [8], which represents the current state-of-the-art, uses Oriented Fast and Rotated Brief (ORB) to detect and describe corner points as the key references for its VO. ORB itself uses the Feature from Accelerated Segment Test (FAST) corner point detector [9] and the BRIEF descriptor [10] to identify reference points. The FAST corner point detector is an image point segmentation operator which identifies pixels that have significantly different intensity values relative to their neighboring pixels. Should a pixel’s intensity value surpass that of a predetermined threshold value, then the pixel is considered by FAST to be a corner point. Therefore, the value of the predetermined threshold is an influential determinant of the quantity and quality of the identified corner points. In real-world situations, properties such as
image clarity, contrast and sharpness are affected by the dynamics of the environment, camera hardware and terrain. Motion-blur, low-texture scenes and the camera’s auto-white balance (AWB) setting can also negatively influence FASTs’ ability to accurately track corner points between consecutive frames [11], [12].

To solve this problem, we propose a novel, adaptive dynamic threshold SLAM (DT-SLAM) which utilizes a lightweight processing technique for improving corner point extraction performance in FAST. This work contributes a Laplacian-based threshold computation method for high-performance and robust corner point detection. Additionally, a novel threshold switching approach for ORB is presented, enabling the dynamic recalculation of threshold values.

II. RELATED WORK
The performance of VO is determined by the quantity and quality of reference features. Many feature extraction methods had been applied in this field. Harris et al. [13] proposed a combined corner and edge detector to track extracted features from consecutive frames via an auto-correlation function. Bay et al. [14] proposed Speeded Up Robust Features (SURF), a scale and rotation-invariant method, utilizing a Hessian matrix to identify feature points. Similarly, Lowe [15] proposed a three stage, Scale-Invariant Features Transform (SIFT) based approach, whereby nearest-neighbour, clustering and least-squares algorithms are combined to determine feature points. The results of Lowe’s approach, however, are not stable and cannot be applied practically in time-critical applications [16].

One approach for improving vSLAM’s tracking performance is to improve the feature point extraction method. Pumarola et al. [11] proposed PL-SLAM which simultaneously extracted both lines and ORB points for localization and mapping. However, PL-SLAM which is built upon ORB-SLAM2, performed poorly in scenes with few objects. To improve tracking performance, Li et al. [17] proposed R-ORBSLAM, an evolution of ORB-SLAM2 which considered photometric error prior to feature point extraction. Fan [18] proposed a self-adaptive thresholding approach for feature point extraction based on the local variance of pixel intensities. Ma et al. [19] proposed an image entropy method using K-means clustering for computing the threshold, whereas Yang et al. [20] adjusted the threshold based on image brightness. Despite improving feature tracking across different conditions, these approaches were not practical for real-time applications due to hardware constraints. Ding et al. [21] used a KSW entropy method to calculate the FAST threshold, but was susceptible to scenes that had lots of movement. Crucially, this approach still relied on an initial static threshold which reduced the number of corner point detection [18], [22].

Sun et al. [23] proposed a pre-processing stage to remove moving objects from the tracked scene (MR-SLAM). This approach was prone to tracking failure when previously moving objects stopped moving. Huang et al. [24] presented a cross-modality method to associate local points with existing map’s 3D structure components for better mapping efficiency, but the model’s performance relied on the accuracy of existing maps. Similarly, Cho et al. [25] introduced a de-blurring and de-hazing technique method to improve the quality of frames in SLAM. Jin [26] proposed a lightweight convolution neural network-based corner point extractor instead of FAST, which improved localization accuracy. Similarly, Huang et al. [27] adopted the SuperPoint method for feature extraction, which required the repeated estimation of camera pose via deep-learning (DL-VO). However, these approaches are not suitable for use in constrained hardware applications.

III. SYSTEM OVERVIEW
In this section we provide a technical overview (see Fig. 1) of our proposed DT-SLAM implementation. DT-SLAM introduces a Dynamic, Self Adaptive Threshold (DSAT) mechanism to FAST. In doing so, it is hoped that feature point detection is more robust in challenging scenes. DSAT consists of a Dynamic Threshold Calculator (DTC) and a Threshold Switching Mechanism (TSM) as shown in Fig. 2a. DSAT accepts an input frame which is then divided into 31 × 32 uniform cells, as shown in Fig. 2b [8]. ORB extraction is then performed on each cell using FAST via a 2-step process.

First, the centre-point is identified using a 16-pixel Bresenham circle [28]. This selection process compares the pixel’s intensity to its n-nearest neighbouring pixels. These intensity difference values are then compared to an initial “hard” threshold to determine whether the centre-point is a suitable ORB candidate. If no ORB candidate is detected, this process is repeated using an alternative “soft” threshold value. ORBs are then decided from a candidate pool through the application of a non-maximal suppression (NMS) algorithm. This method replaces the static threshold value used in ORB-SLAM3 which identifies candidate ORBs through repeated trial-and-errors using RANSAC approach [29].
R. Wu et al.: DT-SLAM: Dynamic Thresholding Based Corner Point Extraction in SLAM System

IV. METHODS

A. DYNAMIC THRESHOLD CALCULATOR (DTC)

SLAM relies on matching ORB points between two consecutive frames to model the poses of the camera. In DSAT, we introduce edge detection to improve ORB point tracking between frames. Edges are comprised of consecutive points that have significantly brighter intensity relative to their neighbouring pixels. Popular edge detection methods such as Sobel operator [30], Canny [31] and Laplacian of Gaussian (LoG) [32] have been applied in many fields, including medical imaging [33], computer vision for object detection [34], and self-driving vehicles for pedestrian monitoring [35]. Existing works indicate that LoG [32], [36]–[38] yielded better edge detection performance due to its high sensitivity to fine edges whilst introducing a low computational overhead. Figure 3 illustrates the step-by-step operation of DTC upon receiving a gray-scaled frame as input source ($I_f$). A Gaussian smoothing filter $G_\sigma(x, y)$ is applied to reduce the noise interferences (shown in (1) and (2)) [37]. Edges are further refined using a Laplace operator,

$$G_\sigma(x, y) = \frac{1}{2\pi \sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \quad (1)$$

$$I_f = G_\sigma(x, y) \ast I \quad (2)$$

where $\ast$, $G_\sigma(x, y)$, $x$ and $y$ are the convolution operator, resulted filtered frame, and x- and y-axis of image pixel, respectively. The original Laplace operator is defined as divergence ($\nabla$) of the gradient of the pixel intensity ($\Delta P$) in a n-dimensional Euclidean space, as shown in (3) and (4),

$$\Delta P = \nabla \cdot \nabla P \quad (3)$$

$$\nabla = \left( \frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n} \right) \quad (4)$$

where $\nabla P$ is derived from $(x, y)$-coordinates as shown in (5).

$$\nabla^2 P(x, y) = \frac{\partial^2 P(x, y)}{\partial x^2} + \frac{\partial^2 P(x, y)}{\partial y^2} \quad (5)$$

Here, $\nabla P$ is computed by an approximate discrete convolution kernel of size 3 x 3 (see (6)) which generates the edge-$I_f$ image containing edges information [38]. Finally, threshold $t$ is computed as the standard deviation of the pixels intensity of edged-$I_f$ as shown in (7),

$$t = \sqrt{\frac{\sum_{w=1}^{W} \sum_{h=1}^{H} (\nabla P(x, y) - \nabla P)^2}{W \times H}} \quad (7)$$

where $\nabla P$, $W$ and $H$ are denoted as the mean pixels intensity of $\nabla P$, width and height of the $I_f$ (frame or cell), respectively.

B. THRESHOLD SWITCHING MECHANISM (TSM)

In TSM, two levels of threshold are introduced, namely hard threshold (HT) and soft threshold (ST). HT is computed by employing the DTC across all pixel intensities. If no suitable points are detected using HT, TSM is activated, with ST being calculated based on local pixel intensities. The calculated dynamic threshold value is then used by FAST for ORB points extraction in each image cell as shown in Fig. 2b. HT values are reused in subsequent frames if the number of cells with detected ORBs (SR) is over 50% (140 over 280 total cells in image pyramid level 0). Otherwise, a new HT will be computed.

C. DATASET

TUM and EuRoC are popular benchmark datasets in evaluating the performance of SLAM. DT-SLAM was evaluated using these datasets. The TUM RGB-D dataset [39] contains...
sequences of indoor videos under different environment conditions e.g., illuminance and varied scene settings, which include both static and moving object. The raw data was captured by Microsoft Kinect RGB-D sensor and the ground truth locations were generated by an external motion capture system. The EuRoC dataset [40] contains 11 stereo sequences recorded from a micro aerial vehicle (MAV). The dataset specifies two office like environments and a large industrial factory. The ground truth is measured by Leica laser tracker and Vicon motion capture system.

In this study, only RGB frames from left stereo camera are considered during the performance analysis. To avoid differences caused by testing hardware and RANSAC [29], the benchmarking of ORB-SLAM3 and DT-SLAM are performed three times under the Ubuntu version 16.04 operating system powered by Intel Xeon® Silver 4108 CPU @ 1.80 GHz and 16 GB RAM.

V. RESULTS AND DISCUSSION

Figure 4 shows the comparison of detected corner points by ORB-SLAM and DT-SLAM using two consecutive input frames obtained from the EuRoC V202 sequence. Table 1 shows that 686 and 795 corner points were extracted by ORB-SLAM3 with static threshold value of 20. DT-SLAM identifies the maximum permitted (1000) corner points with the dynamically calculated HT value of 14 and 12 on respective frames. DT-SLAM also demonstrates a 30% improvement in matching points between frames, reducing the trajectory loss rate of the overall evaluation run. Additionally, in situations where ST calculation is required, ORB point extraction increases more than five-fold in both frames as illustrated in Table 2 and Fig. 5.

The performance of proposed DT-SLAM is evaluated with the benchmark datasets based on Root Mean Squared Error (RMSE) analysis as shown in (8).

\[ RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (h(\alpha_i) - \beta_i)^2} \]  

where \( m \) is the number of keyframes in SLAM, \( \alpha_i \) and \( \beta_i \) are the generated trajectories of keyframes and their associated ground truth, respectively at \( i \)-th keyframe whereas \( h(\cdot) \) is the trajectory alignment method based on the scale and rotation consistency between \( \alpha_i \) and \( \beta_i \) [41]. Table 3 shows the RMSE for indoor localization, as tested with TUM RGB-D and EuRoC datasets. The “Fr1/desk2” sequence is a video recording of an office desk. Two frames are extracted from the sequence where “Frame A” contains multiple objects on the desk and “Frame B” contains fewer objects (Fig. 6). The corner points extracted by ORB-SLAM3 are reduced when the camera moves from Frame A (see Fig. 6a) to Frame B (see Fig. 6b). The results indicate that localization performance increases by 44.3% in “Fr1/desk2” with DT-SLAM (RMSE = 0.0571) compared to ORB-SLAM3 (RMSE = 0.1026).

The trajectory prediction performance of GMMLoc, SVO and DL-VO were also evaluation using the EuRoC dataset. GMMLoc (mean RMSE = 0.0323) and DL-VO (mean RMSE = 0.0761) outperform ORB-SLAM3 (mean RMSE = 0.1899) and DT-SLAM (mean RMSE = 0.0807). However, GMMLoc and DL-VO require an existing map of the environment in order to estimate trajectory. DT-SLAM
TABLE 3. RMSE (m) of indoor localization tested with TUM [39] and EuRoC [40] datasets.

| Sequence                  | ORB-SLAM3 | DT-SLAM | DL-VO [27] | SVO [7] | GMMLoc [24] | PL-SLAM [11] | MR-SLAM [23] |
|---------------------------|-----------|---------|------------|--------|-------------|--------------|--------------|
| TUM                       |           |         |            |        |             |              |              |
| Fr1/XYZ                   | 0.0081    | 0.0077  | -          | -      | -           | 0.0121       | -            |
| Fr1/room                  | 0.0030    | 0.0009  | -          | -      | -           | -            | -            |
| Fr1/desk                  | 0.0155    | 0.0126  | -          | -      | -           | -            | -            |
| Fr1/desk2                 | 0.1026    | 0.0571  | -          | -      | -           | -            | -            |
| Fr2/large_no_loop         | 0.0086    | 0.0074  | -          | -      | -           | -            | -            |
| Fr2/360-kidnap            | 0.0805    | 0.0382  | -          | -      | -           | 0.0392       | -            |
| Fr3/sitting_halfsp        | 0.0105    | 0.0090  | -          | -      | -           | 0.0066       | 0.0482       |
| Fr3/sitting_ar4           | 0.0054    | 0.0048  | -          | -      | -           | 0.0089       | -            |
| Fr3/walking_halfsph_1     | 0.0040    | 0.0035  | -          | -      | -           | 0.0154       | 0.0932       |
| Fr3/walking_halfsph_2     | 0.0135    | 0.0122  | -          | -      | -           | 0.0160       | 0.1252       |
| Mean                      | 0.0252    | 0.0153  | -          | -      | -           | 0.0164       | 0.0859       |
| EuRoC                     |           |         |            |        |             |              |              |
| MH01                      | 0.0126    | 0.0106  | 0.0163     | 0.17   | -           | -            | -            |
| MH02                      | 0.1316    | 0.0935  | 0.0146     | 0.27   | -           | -            | -            |
| MH03                      | 0.0452    | 0.0818  | 0.0320     | 0.43   | -           | -            | -            |
| MH04                      | 0.0552    | 0.0292  | 0.0959     | 1.36   | -           | -            | -            |
| MH05                      | 0.0250    | 0.0175  | 0.0434     | 0.51   | -           | -            | -            |
| V101                      | 0.0432    | 0.0497  | 0.0934     | 0.30   | 0.030       | -            | -            |
| V102                      | 1.3703    | 0.4539  | 0.2382     | 0.47   | 0.023       | -            | -            |
| V103                      | 0.1215    | 0.0418  | 0.1717     | X      | 0.047       | -            | -            |
| V201                      | 0.0238    | 0.0198  | 0.0415     | 0.30   | 0.018       | -            | -            |
| V202                      | 0.0491    | 0.0429  | 0.0473     | 0.47   | 0.020       | -            | -            |
| V203                      | 0.2116    | 0.0645  | X          | X      | 0.056       | -            | -            |
| Mean                      | 0.1899    | 0.0807  | 0.0761     | 0.464  | 0.0323      | -            | -            |

X: indoor trajectory lost. *: sequence with moving objects.

FIGURE 6. Frames extracted from “Fr1/desk2” sequence of TUM dataset with detected corner points by applying ORB-SLAM3 on a frame with (a) multiple objects (Frame A, left column) and (b) a frame with less objects (Frame B, right column); DT-SLAM on (c) Frame A and (d) Frame B.

shows great stability in challenging environments. Multiple trajectories are lost at sequence “V203” which are tested with DL-VO and SVO. The same scenario is also observed at sequence “V103” with multiple trajectories lost tested with SVO. Majority of the lost trajectories is due to insufficient matching points between consecutive frames, which are caused by lack of informative ORBs being extracted. Improvements are found at sequences of “MH04”, “V102” and “V103” tested with DT-SLAM compared to ORB-SLAM3. A significant improvement can also be observed at sequence “V203”, tested with ORB-SLAM3 ($RMSE = 0.2116$) and ($RMSE = 0.0807$) as illustrated in Fig. 7 where the trajectory predicted by ORB-SLAM3 has higher variation than proposed DT-SLAM between the time frames of 1:41:45 and 1:41:50.

Similar improvements are seen with “Fr2/360-kidnap” where the mapped trajectory generated by DT-SLAM ($RMSE = 0.0382$) once again outperforms ORB-SLAM3 ($RMSE = 0.0805$) (Fig. 8). Three sequences in the TUM dataset contain moving objects, which are “Fr3/sitting_halfsp”, “Fr3/walking_halfsp” and “Fr3/ walking_halfsp”. Table 4 shows that the total number of map points and their respective mean number of matching points are higher in DT-SLAM when compared to ORB-SLAM3. The results indicate that DT-SLAM outperforms ORB-SLAM3, MR-SLAM and (largely) PL-SLAM in the tested sequences. PL-SLAM demonstrates...
TABLE 4. Comparison of total number map points and mean number of matching points for sequences with moving objects in TUM dataset.

| Sequence       | DT-SLAM | ORB-SLAM |
|----------------|---------|----------|
| Fr3/sitting_half | 3201 / 244 | 2972 / 239 |
| Fr3/walking_xyz  | 1270 / 215  | 1240 / 204  |
| Fr3/walking_halfph | 2703 / 212  | 2284 / 205  |

FIGURE 8. Plotting of trajectory tested with sequence “Fr3/360-kidnap” using ORB-SLAM3 (blue line) and proposed DT-SLAM (orange line).

TABLE 5. Comparison of mean computation time (ms/frame) of localization and mapping with proposed DT-SLAM with other state-of-the-art methods.

| Method     | Dataset | Localization (ms/frame) | Mapping (ms/frame) |
|------------|---------|-------------------------|--------------------|
| DT-SLAM    | EuRoC   | 33.72                   | 64.83              |
| ORB-SLAM3  | EuRoC   | 32.45                   | 60.08              |
| SVO [7]    | EuRoC   | 3.76                    | 1.80               |
| DL-VO [27] | EuRoC   | 23.4                    | 121.0              |
| GMMLoc [24] | EuRoC   | 34.50                   | 409.70             |
| PL-SLAM [11] | TUM     | 51.06                   | 323.85             |

VI. CONCLUSION

This study proposes DT-SLAM, a novel self-adaptive dynamic threshold-based approach for ORB feature point extraction. The main contributions of this system include a dynamic threshold algorithm and a threshold switching mechanism. We have shown that DT-SLAM outperforms the current state-of-the-art SLAM methods using standard benchmarking datasets. DT-SLAM is especially effective in challenging environments and dynamic scenes. Future work will seek to evaluate DT-SLAM in increasingly more challenging environments, including multi-room and multi-floor localization.

REFERENCES

[1] D. Scaramuzza and F. Fraundorfer, “Visual odometry [tutorial],” IEEE Robot. Autom. Mag., vol. 18, no. 4, pp. 80–92, Dec. 2011.
[2] D. Nistér, O. Naroditsky, and J. Bergen, “Visual odometry,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), vol. 1, Jun. 2004, p. 1.
[3] G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” in Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality, Nov. 2007, pp. 225–234.
[4] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: A versatile and accurate monocular SLAM system,” IEEE Trans. Robot., vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[5] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, “3-D mapping with an RGB-D camera,” IEEE Trans. Robot., vol. 30, no. 1, pp. 177–187, Feb. 2014.
[6] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2014, pp. 834–849.
[7] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, “SVO: Semidirect visual odometry for monocular and multicamera systems,” IEEE Trans. Robot., vol. 33, no. 2, pp. 249–265, Apr. 2017.
[8] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, “ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM,” 2020, arXiv:2007.11898. Online.
[9] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in Proc. Eur. Conf. Comput. Vis. Berlin, Germany: Springer, 2006, pp. 430–443.
[10] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in Proc. Eur. Conf. Comput. Vis. Berlin, Germany: Springer, 2010, pp. 778–792.
[11] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfelipo, and F. Moreno-Noguer, “PL-SLAM: Real-time monocular visual SLAM with points and lines,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2017, pp. 4503–4508.
[12] R. Gomez-Ojeda, F. A. Moreno, D. Zaniga-Noel, D. Scaramuzza, and J. Gonzalez-Jimenez, “PL-SLAM: A stereo SLAM system through the combination of points and line segments,” IEEE Trans. Robot., vol. 35, no. 3, pp. 734–746, Jun. 2019.
[13] C. G. Harris and M. Stephens, “A combined corner and edge detector,” in Proc. Alvey Vis. Conf., 1988, vol. 15, no. 50, pp. 1–5.
[14] H. Bay, T. Tuytelaars, and V. Gool, “SURF: Speeded up robust features,” in Proc. Eur. Conf. Comput. Vis. Bremen, Germany: Springer, 2006, pp. 404–417.
[15] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” Int. J. Comput. Vis., vol. 60, no. 2, pp. 91–110, Nov. 2004.
[16] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in Proc. Int. Conf. Comput. Vis. Nov. 2011, pp. 2564–2571.
[17] C. Li, X. Zhang, and T. Cao, “SLAM with mapping based on photometric information and ORB features,” J. East China Univ. Sci. Technol., vol. 47, no. 3, pp. 331–339, 2021.
[18] G. Fan, “Research on visual SLAM algorithm of mobile robot in dynamic indoor scene,” M.S. thesis, Xi’an Univ. Technol., Xi’an, China, 2020.
[19] Y. Ma and L. Shi, “A modified multiple self-adaptive thresholds feature points extraction algorithm based on image gray clustering,” in Proc. Int. Appl. Comput. Electromagn. Soc. Symp. (ACES), 2017, pp. 1–5.
[20] H. Yang, H. Li, K. Chen, J. Li, and X. Wang, “Feature points extraction based on improved ORB-SLAM,” in Proc. Chin. Automat. Congr. (CAC), Nov. 2019, pp. 936–940.
[21] Y. Ding, J. Wang, Y. Qiu, and Y. Hai-Bo, “Fast feature detection algorithm based on self-adaptive threshold selection,” Command Control Simul., vol. 35, no. 2, pp. 47–53, 2013.
[22] X.-Z. Wu, L. Gang, W.-S. Gong, and Y. Shi, “An improved monocular ORB-SLAM method,” DEStech Trans. Comput. Sci. Eng., pp. 258–253, 2016.

[23] Y. Sun, M. Liu, and M. Q.-H. Meng, “Improving RGB-D SLAM in dynamic environments: A motion removal approach,” Robot. Autom. Syst., vol. 89, pp. 110–122, Mar. 2017.

[24] H. Huang, H. Ye, Y. Sun, and M. Liu, “GMMLOC: Structure consistent visual localization with Gaussian mixture models,” IEEE Robot. Autom. Lett., vol. 5, no. 4, pp. 5043–5050, Oct. 2020.

[25] Y. Cho and A. Kim, “Channel invariant online visibility enhancement for visual SLAM in a turbid environment,” J. Field Robot., vol. 35, no. 7, pp. 1080–1100, Oct. 2018.

[26] J. Jin, “Research on visual SLAM method based on feature points of convolutional neural network,” M.S. thesis, Hubei Univ. Technol., Wuhan, China, 2020.

[27] H. Huang, H. Ye, Y. Sun, and M. Liu, “Monocular visual odometry using learned repeatability and description,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2020, pp. 8913–8919.

[28] J. Bresenham, “A linear algorithm for incremental digital display of circular arcs,” Commun. ACM, vol. 20, no. 2, pp. 100–106, Feb. 1977.

[29] R. Hartley and A. Zisserman, “Estimation-2D projective transformations,” in Multiple View Geometry, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2003, pp. 87–90.

[30] K. K. Pinglu, “Visual perception by a computer,” in Automatic Interpretation and Classification of Images, A. Grasselli, Ed. New York, NY, USA: Academic, 1969, pp. 277–284.

[31] L. Ding and A. Goshtasby, “On the Canny edge detector,” Pattern Recognit., vol. 34, no. 3, pp. 721–725, Mar. 2001.

[32] D. Marr and E. Hildreth, “Theory of edge detection,” Proc. Roy. Soc. London B, Biol. Sci., vol. 207, pp. 187–217, Feb. 1980.

[33] Z. Yu-Qian, G. Wei-Hua, C. Zhen-Cheng, T. Jing-Tian, and L. Ling-Yun, “Medical images edge detection based on mathematical morphology,” in Proc. IEEE Eng. Med. Biol. 27th Annua. Conf., Jan. 2005, pp. 6492–6495.

[34] K. S. Fu and J. K. Mui, “A survey on image segmentation,” Pattern Recognit., vol. 13, no. 1, pp. 3–16, 1981.

[35] W. Farag and Z. Saleh, “An advanced road-lanes finding scheme for self-driving cars,” in Proc. 2nd Smart Cities Symp. (SCS), 2019, pp. 1–6.

[36] N. Chandrakar and D. Bhonsle, “Study and comparison of various image edge detection techniques,” Int. J. Manage., IT Eng., vol. 2, no. 5, pp. 499–509, 2012.

[37] G. T. Shrivakshan and C. Chandrasekar, “A comparison of various edge detection techniques used in image processing,” Int. J. Comput. Sci. Issues, vol. 9, no. 5, pp. 269–276, 2012.

[38] X. Wang, “Laplacian operator-based edge detectors,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 5, pp. 886–890, May 2007.

[39] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Oct. 2012, pp. 573–580.

[40] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, “The EuRoC micro aerial vehicle datasets,” Int. J. Robot. Res., vol. 35, no. 10, pp. 1157–1163, Sep. 2016.

[41] M. Grupp. (2017). EVO: Python Package for the Evaluation of Odometry and SLAM. [Online]. Available: https://github.com/MichaelGrupp/evo

M. PIKE (Member, IEEE) was born in U.K., in 1990. He received the M.Eng. degree in computing from Swansea University, Swansea, U.K., in 2012, and the Ph.D. degree in computer science from the University of Nottingham, Nottingham, U.K., in 2017.

Since 2018, he has been working as an Assistant Professor with the University of Nottingham Ningbo China (UNNC), Ningbo, China. Previously, he worked as a User-Experience Researcher with Huawei. His research interests include human–computer interaction and human factors and has published significantly in these fields.

B. G. LEE (Member, IEEE) received the B.I.T. degree in information technology from Multimedia University, Malacca, Malaysia, in 2007, the M.S. degree in electronic engineering from Dongseo University, Busan, South Korea, and the Ph.D. degree in electronic engineering from Pukyong National University, Busan.

From 2012 to 2013, he was a Researcher with the Ubiquitous Sensor Network Laboratory, Pukyong National University. Then, from 2013 to 2014, he worked as a Software Engineer with Changsung Ace Company Ltd., Seoul, South Korea. After this, from 2014 to 2019, he was employed as an Assistant Professor with the Department of Electronic Engineering, Keimyung University, South Korea. Since 2019, he has been an Assistant Professor with the School of Computer Science, The University of Nottingham Ningbo China. He has published more than 20 SCI/SCIE(Q1-rank) journals. He has served as a reviewer for more than ten journals and holds two patents. His research interests include human–computer interaction, deep learning, ubiquitous healthcare, computer vision, mobile application programming, and sensors networks. He was a recipient of the IEEE SENSORS Journal Best Paper Award, in 2019.

R. WU received the B.E. degree in computer science and technology from Ningbo University, Ningbo, China, in 2020. He is currently pursuing the Ph.D. degree with the School of Computer Science, The University of Nottingham Ningbo China. His research interests include computer vision, deep learning, and human–computer interaction.