Proceedings of CONF-FMCE 2025 Symposium: Semantic Communication for Media Compression and Transmission
DOI: 10.54254/2755-2721/2025.GL25448

A Bidirectional 8-Neighborhood Border Tracking and Kalman-Based Centerline Reconstruction Method for Line-Following Robots

Tian Xia

Department of Mechanical and Aerospace Engineering, University of California, Irvine, USA
xiat14@uci.edu

Abstract. Vision-based line-following robots need highly robust and accurate centerline extraction to ensure effective autonomous navigation. Traditional methods suffer from performance degradation when encountering lane situations like crossing, “X” shaped with diagonal horizon line and etc. This paper introduces a bidirectional 8-neighborhood border tracking algorithm, enhanced with least squares border completion and Kalman-filter-based smoothing. A dual-pass strategy detects both bottom-up and top-down borders, allowing recovery of the missing line part via a combination of the top-down reconstructed centerline segment and the original bottom-up segment before the breakpoint. The top-down reconstructed centerline is filtered using a one-dimensional Kalman filter to improve smoothness and reduce noise and jitter, ensuring stability in the steering response. Experimental evaluation demonstrates a 94% success rate in reconstructing broken lanes and solving the breaks when meeting cross-type discontinuities. These results confirm the applicability of the method in real-time embedded systems for intelligent vehicle steering. Moreover, the algorithm is computationally lightweight and suitable for deployment on low-power embedded platforms.

Keywords: centerline extraction, 8-neighborhood search, Kalman filter, least squares fitting, autonomous vehicle navigation

1. Introduction

Vision-based path tracking functions as an essential component for autonomous vehicle navigation, especially in intelligent transport robots operating in factories and storage facilities. The reliability of centerline extraction directly affects the steering accuracy and stability of the robot. The traditional edge or contour-based detection faces significant applicability challenges because complex factory and storage conditions produce inconsistent lighting and poor road surfaces, and complex lane geometries that often lead to centerline breakages.

Previous studies have investigated different solutions to address these problems. The binary segmentation method known as Otsu's method has gained widespread use while researchers extend it by adding bias or local adaptive thresholds to boost edge extraction robustness in diverse lighting scenarios. The study conducted by Javed et al. demonstrated that edge detection accuracy improved

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.GL25448

by 18.5% through this approach in complex lighting conditions compared to standard global Otsu methods [1]. Liu et al. developed a geometric shape tracking algorithm that utilized vertical spatial features alongside contextual driving information to achieve lane detection accuracy above 92% in city road datasets while maintaining real-time performance at 30 FPS [2]. These detection systems face difficulties when dealing with both lane discontinuities and severe occlusion conditions. The detection of lanes has been achieved through global spline interpolation techniques as part of research studies [3]. The approach demands excessive computational resources, which makes it inappropriate for embedded and real-time robotic system implementation. Suto's method demonstrated sub-pixel accuracy in artificial datasets, yet needed more than 120 ms per frame, which exceeded the necessary latency for real-time control.

The paper presents a bidirectional border tracking system that performs bottom-up and top-down 8-neighborhood search to detect left and right track edges before calculating the midline. The least squares method reconstructs missing segments while a 1-Dimensional Kalman filter smoothens the midline after the combination process. The objective of the research focuses on developing reliable real-time line extraction techniques suitable for moderate computational real-time platforms.

2. Methods

2.1. Image binarization using Otsu thresholding with bias

The original grayscale input image from the camera is first converted into binary format (PGM) using the Otsu thresholding method. Otsu's method automatically determines the optimal threshold by maximizing the inter-class variance between foreground and background pixels, effectively separating the lane from the background. The research utilizes black lanes and a light-colored background as the working condition, where lanes appear white and backgrounds black in the binary format files. To increase robustness under different lighting conditions, a bias is added to the threshold to allow flexible tuning of image brightness, which can be directly evaluated visually by the proportion of black and white. A larger bias results in a brighter image (more white pixels), while a smaller bias creates a darker image and may introduce noise on the lane. This thresholding method ensures clearer contrast between lanes and background and provides an interface enabling real-time adjustment for image binarization based on input from external photosensitive sensors.

2.2. Bidirectional 8-neighborhood border tracking

To detect the left and right borders of the track and prepare for fixing broken midlines, the algorithm includes two separate 8-neighborhood border searches- Bottom-Up and Top-Down. The two searches are basically the same and different in direction and the 8-Neighborhood pattern. The search consists of three parts. The first part is the Seed Points Search. The search for the left and right original seeds starts from the center of the bottom row. The search direction is from the middle to the left and right sides. The seed points are detected by identifying whether a pixel's color changes from white to black. Then the second part is Edge Tracking. Starting from these seed points, a directional search tracks black-to-white edge transitions using a predefined clockwise 8-Neighborhood pattern for the left border and a counter-clockwise pattern for the right border. As shown in Figure 1, the left border pattern starts from the bottom and rotates clockwise, while the right border uses a mirrored sequence.

DOI: 10.54254/2755-2721/2025.GL25448

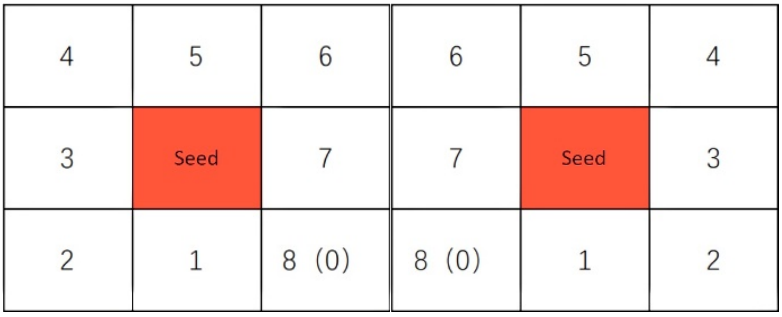


Figure 1. 8-Neighborhood pattern for left border (left) and right border (right) (photo/picture credit: original)

When the pattern detects black-to-white changes, the last black pixel is marked as the next seed, as demonstrated in Figure 2.

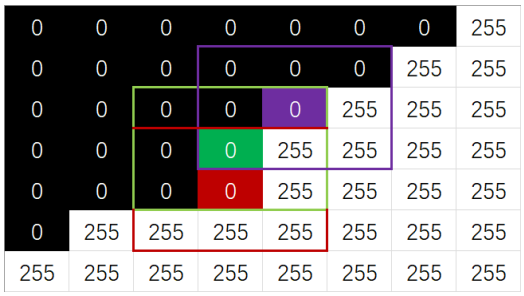


Figure 2. Example of left border search. The colored blocks are the marked seeds (photo/picture credit: original)

The final part is Recording. The x and y coordinates at each step are stored in arrays. This continues until the left and right traces intersect or reach the top.

For the Top-Down Search, the same logic is applied but starts from the top of the image. However, the directional search counter-clockwise 8-Neighborhood pattern for the left border and clockwise 8-Neighborhood pattern for the right border due to mirroring. The search in the seed pattern starts from directly above the seed ('5' in Figure 1).

2.3. Initial centerline computation

For each row y where both left and right boundary pixels are detected, the horizontal centerline is calculated as:

$$x_c(y) = \frac{x_{\text{left}}(y) + x_{\text{right}}(y)}{2} \tag{1}$$

Where, $x_c(y)$ is the x-coordinate of the centerline at row y, $x_{\text{left}}(y)$ is the x-coordinate of the detected left border at row y, $x_{\text{right}}(y)$ is the x-coordinate of the detected right border at row y.

This yields two preliminary arrays: centerline (from bottom-up) and centerline2 (from top-down).

2.4. Breakpoint detection

Two types of centerline discontinuities are detected. The Bottom Break (y_1) is detected in the centerline, scanned from bottom to top. If the change between adjacent rows exceeds 5 pixels in

DOI: 10.54254/2755-2721/2025.GL25448

horizontal shift (the threshold changes with the resolution of the image), it is flagged as a breakpoint. The Top Break (y_2) is detected in centerline2, scanned from top to bottom with the same thresholding principle applied.

These two points (y_1 and y_2) define a "hole" in the centerline that requires reconstruction.

2.5. Least squares fitting of border lines

To reconstruct the missing part, the algorithm performs linear regression on 10 rows of left or right boundary points just above y_2. The least squares method solves for the best-fit line:

$$x = ky + b \tag{2}$$

Where, x is the predicted x-coordinate of the border point, y is the vertical image row index, k is the slope of the fitted line, b is the y-intercept of the line.

Independently for both left and right borders. This process reduces the influence of noise and provides a relatively stable geometric extension of the border paths downward toward y_1.

2.6. Border completion and centerline patching

Once slope k and intercept b are determined from both sides, the algorithm extrapolates left and right borders from y_2 to y_1 using the fitted slopes and intercepts. The centerline in this region is recalculated row by row as the average of the reconstructed left and right border positions.

However, due to noise from border fitting and pixel jitter with the movement of the robot, the raw extrapolated centerline may appear jittery. To mitigate these effects, a 1D Kalman filter is applied:

State Definition

The Kalman filter models two state variables: x: the horizontal position of the centerline. v: the rate of change of the centerline (slope)

State transition model:

$$\begin{bmatrix} x_k \\ v_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ v_{k-1} \end{bmatrix} \tag{3}$$

Measurement model:

$$z_k = x_k + w_k \tag{4}$$

Where, z_k is the observed centerline position at time step k, w_k is the measurement noise, modeled as zero-mean Gaussian noise.

Kalman Update Equations

Prediction gain:

$$K_x = \frac{P_x}{P_x + R} \tag{5}$$

$$K_v = \frac{P_v}{P_v + R} \tag{6}$$

Where, K_x, K_v is Kalman gains for position and velocity, respectively, P_x, P_v are prior error covariances of position and velocity, R is measurement noise variance.

DOI: 10.54254/2755-2721/2025.GL25448

State update:

$$x_k = x_k + K_x\left(z_k - x_k\right) \tag{7}$$

$$v_k = v_k + K_v \cdot \left(\left(z_k - x_k\right) - v_k\right) \tag{8}$$

Where, x_k , v_k are updated estimates for position and velocity, $(z_k - x_k)$ is the innovation (difference between observation and prediction).

Covariance update:

$$P_x = (1 - K_x)P_x + Q_x \tag{9}$$

$$P_v = (1 - K_v)P_v + Q_v \tag{10}$$

Where, Q_x , Q_v are process noise covariances for position and velocity.

Parameter Settings (from implementation in the situation of this research):

These values were empirically tuned for a balance between responsiveness and stability. The detailed configuration is provided in Table 1.

Table 1. Parameter settings and benchmark value

Parameter	Meaning	Value
P_x	Initial position covariance	0.3
P_v	Initial slope covariance	0.3
Q_x	Process noise for position	0.05
Q_v	Process noise for slope	0.05
R	Measurement noise variance	4.0

The output of the filter is a noise-reduced centerline, which is suitable for visual tracking and stable steering control.

2.7. Final centerline merging and blending

Finally, the algorithm constructs a unified centerline: For rows below y_1, the original bottom-up centerline is used. For rows above y_1 and below y_2, the Kalman-filtered centerline from top down is applied. For rows above y_2, the original top-down centerline is used.

As shown in Figure 3, these three sections are color-coded for visualization.

DOI: 10.54254/2755-2721/2025.GL25448

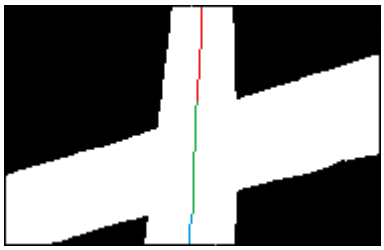


Figure 3. Visualization of centerlines extracted bottom-up part (blue), the top-down part (green), and the reconstructed part (red) (photo/picture credit: original)

3. Results

The output binary image clearly shows the final extracted centerline. Visual inspection confirms that the final after-merged centerline closely follows the real midline in various lane conditions, even when the original image has broken parts like crossings. This result is illustrated in Figure 4, which shows both the original grayscale input (left) and the binarized image with the reconstructed centerline overlaid (right).

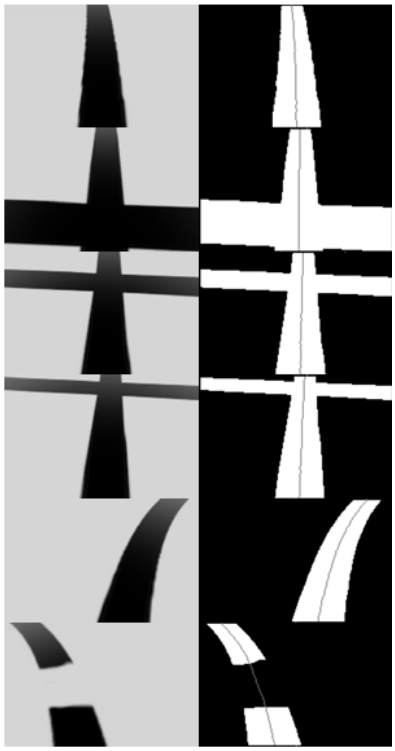


Figure 4. Original image (left) and binary image with final centerline overlaid (right). Note the accurate reconstruction across the broken region (photo/picture credit: original)

Moreover, across 100 test cases with real-world road images, the breakpoint detection and repair module achieved: 94% success rate in centerline restoration at detected breakpoints. The 94 % recovery rate was computed as the proportion of test cases where the reconstructed centerline successfully bridged the detected discontinuities with less than 5-pixel deviation. Average error reduction of 4.2 pixels at the point of discontinuity compared to the baseline (single direction

DOI: 10.54254/2755-2721/2025.GL25448

tracking). Table 2 summarizes the comparison of performance metrics between the proposed dual-directional Kalman method and traditional single-direction tracking across 100 real-world test cases.

Table 2. Comparison of centerline recovery accuracy, positional error, and smoothness before and after Kalman-based smoothing (image resolution: 188×120)

Metric	Single Direction	Dual Direction + Kalman
Breakpoint Recovery Rate	48%	94%
Mean Positional Error (px)	16.3	2.1
Centerline Smoothness (std)	6.7	1.6

For the Runtime Performance, the algorithm is implemented in C and C++ and was tested on an embedded Linux platform (e.g., RDK X5) based on CPU. It achieved frame rates of up to 45 FPS with image resolution 188×120, meeting real-time processing requirements for smart vehicles and robots.

4. Discussion

The bidirectional 8-neighborhood border tracking and Kalman-based centerline reconstruction method provides multiple advantages over standard line-following algorithms. Javed et al. apply global Otsu thresholding with Hough transforms for lane extraction, yet this approach fails when dealing with complex light conditions and broken lanes caused by road situations like crossings or lane surface abrasions [1]. Liu et al. developed a strong model that incorporates vertical spatial features together with contextual information to enhance stability in urban settings [2]. Their framework lacks specific solutions for handling lane discontinuity problems. Suto deployed spline-based lane tracking on embedded systems, but the approach remains computationally expensive for real-time processing on low-power platforms [3].

The bidirectional border tracking approach provides better resistance against occlusion and shadow, and partial wear situations than single-directional border tracking methods. The method performs least-squares extrapolation to generate reliable geometric approximations of absent lane sections. The method has proven successful in related fields such as railway detection, where Zheng et al. achieved more than 90% reconstruction accuracy through LSD and least squares fitting for occluded rail segments in complex scenes [4]. The 1-D Kalman filter produces smooth results while reducing jitter, which is important for reliable steering control. The Kalman filter outperforms traditional smoothing methods like moving average filters because it provides adaptive noise reduction and enhanced dynamic response to sudden changes or broken lines [5].

The present method needs enough lane contrast to function properly, yet it shows reduced performance when dealing with local glare and reflection. Future research should investigate the combination of external sensors, like radar or photodetectors. Research on agricultural navigation systems shows that multi-sensor processing enhances detection reliability when objects are occluded. Han et al. developed a LiDAR-camera fusion system that used top-down visual analysis to boost obstacle avoidance accuracy in unstructured environments [6]. Meanwhile, recent advances in multi-camera fusion lane detection show promising performance in complex urban environments with frequent occlusions and missing markings, which shows that multi-camera fusion can also be an improved plan [7]. For improvement in the algorithm, multi-segment 8-neighborhood approaches can also enhance centerline reconstruction when dealing with local glare and reflection. The C-based implementation works efficiently on embedded hardware, achieving real-time performance at 45

DOI: 10.54254/2755-2721/2025.GL25448

FPS. Systems similar to those implemented on Raspberry Pi have shown frame rates of around 38 FPS using handcrafted edge detection pipelines, but often suffer from decreased robustness when handling broken lane segments [8]. The bidirectional method, by contrast, maintains both frame rate and recovery accuracy through dual-directional search and Kalman-based correction.

5. Conclusion

The proposed pipeline uses a bidirectional 8-neighborhood border tracking and centerline reconstruction method, which combines 8-neighborhood search with least squares fitting and Kalman filtering. The proposed method solves the problems of track line breaks and noise, which frequently appear in real-world vision-based line-following robots. The bottom-up and top-down border extraction method enhances robustness when dealing with broken sections. The least squares fitting method precisely rebuilds missing parts of the track, and the Kalman filter minimizes jitter to generate a smooth final centerline. The method demonstrates a 94% recovery rate at breakpoints and shows a substantial reduction in centerline variance. The research demonstrates its high suitability for embedded system real-time applications.

Future improvements to this algorithm include using external photosensitive sensors to detect abnormal brightness points and dynamically adjusting Kalman filter parameters for better robustness under different conditions, and implementing a multi-segment 8-neighborhood to handle more broken centerline segments when local glare and reflection occur. The method shows promise for application in urban small autonomous driving scenarios where road markings are partially occluded or worn out, like an autonomous road sweeper.

References

[1] Javed, M. A., Sabri, M. S., & Jian, X. (2023). Lane line detection and object scene segmentation using Otsu thresholding and the fast Hough transform for intelligent vehicles in complex road conditions. Electronics, 12(5), 1079.

[2] Liu, W., Yan, F., Zhang, J., & Deng, T. (2021). A robust lane detection model using vertical spatial features and contextual driving information. Sensors, 21(3), 708.

[3] Suto, J. (2021). Real-time lane line tracking algorithm to mini vehicles. Transport and Telecommunication Journal, 22(4), 461–470.

[4] Zheng, Y.-S., Jin, Y.-W., & Dong, Y. (2021). Rail detection based on LSD and the least square curve fitting. Machine Intelligence Research, 18(1), 85–96.

[5] DSP StackExchange. (2014, March 10). When is a Kalman filter different from a moving average? StackExchange DSP. https: //dsp.stackexchange.com/questions/18892/when-is-a-kalman-filter-different-from-a-moving-average

[6] Han, C., Wu, W., Luo, X., & Li, J. (2023). Visual navigation and obstacle avoidance control for agricultural robots via LiDAR and camera. Remote Sensing, 15(22), 5402.

[7] Xiong, H., Wang, X., Liu, Y., & Wang, S. (2020). Fast and robust approaches for lane detection using multi-camera fusion in complex scenes. IET Intelligent Transport Systems, 14(12), 1551–1559.

[8] Huang, G., Wang, Y., & Lin, H. (2024). Real-time lane detection using Raspberry Pi for an autonomous vehicle application. ARPN Journal of Engineering and Applied Sciences, 19(3), 314–320.