

Design of a ROS-based Autonomous Navigation Correction System for Small Vehicles based on Visual Error Correction

Hongyu Pang

Taiyuan University of Science and Technology, Taiyuan 030024, Shanxi, China

Abstract

Addressing the issues of cumulative positioning errors in traditional ROS smart cars that rely on odometry for navigation, such as trajectory deviation, path deviation, and eventual navigation failure during long-distance travel, this paper designs and implements a vision-based real-time error correction navigation system. The system utilizes ROS as its development platform, retains the original odometer track calculation function, and introduces a visual feature matching and correction module. By matching the real-time collected environmental image features with reference templates, it calculates the lateral deviation and heading deviation of the vehicle online, and dynamically corrects the accumulated drift of the odometer. The experimental results show that the uncorrected system accumulates errors continuously and deviates severely from the trajectory during long-distance driving. After incorporating visual correction, the car can identify position deviations in real-time and correct them, significantly improving positioning accuracy and trajectory conformity. This effectively solves the navigation offset problem caused by differential odometer drift. The system is lightweight and easy to deploy, suitable for long-term stable autonomous navigation of indoor ROS mobile robots.

Keywords

ROS Robot Car; Autonomous Navigation; Cumulative Error; Visual Correction; Feature Matching; Trajectory Correction.

1. Introduction

Mobile robots have been widely deployed in lightweight intelligent scenarios such as indoor inspection, intelligent warehousing, and fixed-point navigation, which requires them to possess stable autonomous navigation capabilities and high-precision trajectory tracking abilities. In low-cost ROS vehicle systems, the wheel odometer serves as the core perception unit for positioning and navigation. However, its single perception method has inherent flaws: the wheel odometer relies on iterative trajectory estimation to solve for pose, which is susceptible to factors such as ground slip, motor errors, and bumpy driving, resulting in continuous accumulated navigation deviations. This leads to trajectory drift over long distances and path deviations from the preset trajectory, severely limiting the operational accuracy of the robot. On the other hand, visual sensors possess rich environmental texture and feature information, providing absolute position observation constraints. However, their standalone use is prone to interference from lighting changes and cluttered scenes, leading to unstable perception. Therefore, neither a single odometer nor a single visual perception solution can meet the long-term, high-precision, and high-stability autonomous navigation requirements of ROS vehicles.

Multi-sensory information fusion and error correction techniques have become mainstream solutions for enhancing the robustness of mobile robot navigation. Al-Tawil et al. [1] pointed out that pure visual perception and visual SLAM methods exhibit significant vulnerabilities in scenarios with sudden changes in lighting and texture loss, and a single visual modality struggles to continuously

output stable positioning information. Shi et al.[2]reviewed the research progress of multi-sensor fusion navigation technology, pointing out that inconsistent perception timing and unreasonable adaptive weight distribution are key challenges that restrict the correction accuracy of low-cost robots. The state estimation method based on Kalman filtering [3]can mitigate the problem of robot navigation data drift to some extent, but this method highly relies on precise system motion models, and its correction effect is limited in scenarios with large parameter errors in low-cost ROS vehicles. The radar self-motion estimation method proposed by Lu et al.[4]can maintain navigation robustness in visual failure scenarios, but the deep learning model requires significant computational resources, making it difficult to achieve real-time deployment on embedded ROS terminals. Existing research has shown that the standard ROS architecture is prone to issues such as data delay and node scheduling imbalance in time-sensitive navigation correction tasks, which can further amplify navigation deviations[5-7].Meanwhile, lightweight perception and correction strategies have been validated on various embedded robot platforms, enabling efficient error correction on low-computing-power devices and adapting to the application scenarios of low-cost smart vehicles [8],[9]. To address the issues of large cumulative error in low-cost ROS vehicle odometers, poor stability in navigation with a single perception modality, and high deployment difficulty of traditional correction algorithms, this paper proposes a real-time navigation error correction method based on visual feature matching. The core contributions of this paper mainly include three aspects: (1) Error mechanism optimization: Based on the dead reckoning characteristics of differential wheeled vehicles, the generation law of cumulative odometer deviations is analyzed, and the correction timing of visual absolute constraints is clarified; (2) Lightweight visual correction mechanism: By matching environmental features, the lateral position deviation and heading angle deviation of the vehicle are solved in real-time, and the chassis control commands are dynamically corrected to achieve model-free adaptive error correction; (3) Embedded system verification: The system is built and comparative experiments are conducted based on the ROS platform. It is verified that the method proposed in this paper can effectively suppress navigation drift and significantly improve long-distance trajectory tracking accuracy and navigation stability in complex indoor scenarios.

2. System Architecture and Data Preprocessing

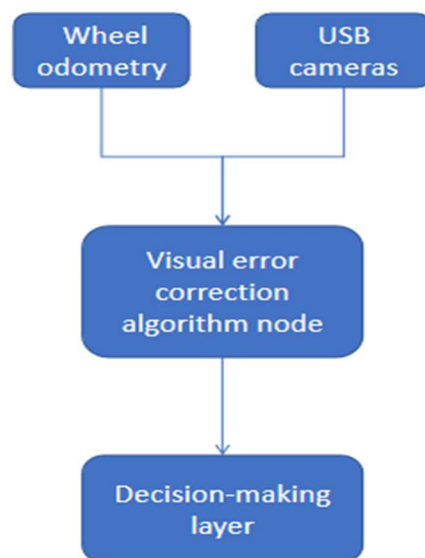


Fig. 1 Visual correction error robot framework

The system adopts a three-layer architecture of perception-correction-decision, as shown in Fig. 1. The perception layer consists of ROS driver nodes for wheel odometry and USB cameras. These

nodes collect real-time mileage and pose data of the vehicle and environmental image data, and publish the data through ROS topics, providing the original input for subsequent error resolution. The correction layer hosts the visual error correction algorithm node, which subscribes to the odometry pose data and image perception data, completes image preprocessing, environmental feature matching, and navigation deviation resolution, and achieves dynamic correction of the accumulated odometry error. The decision-making layer generates speed and steering control commands based on the precise pose information after visual correction, and issues them to the chassis motion control module to correct the vehicle trajectory and achieve stable navigation.

3. Main Algorithm Principles

3.1 Error Propagation Model of Differential Wheel Odometer

The pose of a differential wheeled mobile robot in planar motion can be described by a state vector $X_k = [x_k, y_k, \theta_k]^T$.

Let the displacement increments measured by the left and right wheel encoders within the sampling interval Δt be denoted as $\Delta s_{L,k}$ and $\Delta s_{R,k}$, respectively. Then, the displacement increment of the vehicle center and the heading angle increment can be expressed as follows:

$$\Delta s_k = \frac{\Delta s_{R,k} + \Delta s_{L,k}}{2}, \quad \Delta \theta_k = \frac{\Delta s_{R,k} - \Delta s_{L,k}}{B}$$

Where B represents the wheelbase. From this, we can derive the pose update equation based on odometer recursion:

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta s_k \cos \left(\theta_{k-1} + \frac{\Delta \theta_k}{2} \right) \\ \Delta s_k \sin \left(\theta_{k-1} + \frac{\Delta \theta_k}{2} \right) \\ \Delta \theta_k \end{bmatrix} + \mathbf{w}_k$$

In the formula, w_k represents process noise. However, due to factors such as ground slip, tire deformation, and encoder quantization error, there exists a non-zero deviation between the actual wheel speed and the theoretical value. Let the actual speeds of the left and right wheels, denoted as v_R, v_L , and their measured values, denoted as $\tilde{v}_R, \tilde{v}_L$, satisfy the following equation:

$$v_R = \alpha_R \tilde{v}_R + \beta_R, \quad v_L = \alpha_L \tilde{v}_L + \beta_L$$

Where α_R, α_L represents the scale factor error and β_R, β_L denotes the bias error. After integration, these types of errors exhibit quadratic cumulative growth over time, leading to rapid divergence of lateral and heading deviations during long-distance navigation. Assuming the heading angle obtained through odometer recursion is θ_{odo} and the true heading angle is θ_{true} , the propagation of heading angle error $\delta\theta = \theta_{odo} - \theta_{true}$ satisfies:

$$\delta\theta_k = \delta\theta_{k-1} + \frac{(\alpha_R - \alpha_L)}{B} (\tilde{v}_R - \tilde{v}_L) \Delta t + \frac{(\beta_R - \beta_L)}{B} \Delta t$$

It can be seen that even if the wheel speed difference error remains constant, the heading angle error will still accumulate linearly with the range, leading to an exponential growth in lateral position error.

This mechanism clarifies the necessity for visual correction intervention - external absolute observation constraints must be introduced before the heading deviation diverges to an unacceptable range.

3.2 Visual Feature Matching and Deviation Resolution

This method employs lightweight ORB (Oriented FAST and Rotated BRIEF) features for environmental matching, without relying on artificial landmarks or pre-built maps. The system captures a reference image I_{ref} at the starting position, extracts its keypoint set $\mathcal{F}_{\text{ref}} = \{f_i^{\text{ref}}\}_{i=1}^N$, and records the projection positions of the corresponding 3D spatial points in the vehicle coordinate system. During subsequent navigation, the current frame I_{cur} is captured in real-time, the feature point set $\mathcal{F}_{\text{cur}}$ is extracted, and brute force matching is performed with the reference frame. Mismatched pairs are eliminated through Hamming distance ratio verification.

Let the j -th pair of successfully matched feature points have normalized image coordinates $p_j^{\text{ref}} = (u_j^{\text{ref}}, v_j^{\text{ref}})^T$ in the reference frame and corresponding coordinates $p_j^{\text{cur}} = (u_j^{\text{cur}}, v_j^{\text{cur}})^T$ in the current frame. Assuming a flat ground and a fixed camera height, the planar projection transformation between the two frames can be described by the homography matrix H :

$$p^{\text{cur}} = H \cdot p^{\text{ref}}$$

However, directly solving for H involves a large amount of computation and is sensitive to matching noise. Considering that the main deviations in vehicles in this system are lateral offset and yaw rotation, and the longitudinal displacement maintains a high degree of reliability from the odometer, the deviation solution can be simplified to two-dimensional planar affine estimation.

Define the deviation vector $e = [e_x, e_y, e_\theta]^T$ of the current actual pose of the vehicle body relative to its initial pose. When visual observation is effective, select feature point pairs in the central region of the image, and use the epipolar geometry relationship to derive the closed-form solutions for lateral deviation and heading deviation:

$$e_y = \frac{1}{M} \sum_{j=1}^M \frac{f}{Z_j} (u_j^{\text{cur}} - u_j^{\text{ref}})$$

$$e_\theta = \frac{1}{M} \sum_{j=1}^M \arctan \left(\frac{v_j^{\text{cur}} - v_j^{\text{ref}}}{f} \right)$$

In the formula, f represents the camera focal length (in pixel units), Z_j denotes the scene depth corresponding to the j -th feature point, and M signifies the number of valid matching pairs. Since lateral position deviation and heading deviation manifest as horizontal translation and overall rotation in the image, respectively, the above formula achieves an approximate decoupling of the two. To ensure robustness in the case of unknown depth, median estimation of multiple pairs of feature points is adopted during actual computation to reduce the impact of outliers.

3.3 Model-agnostic Adaptive Rectification Strategy

Unlike traditional Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF), this method does not rely on an accurate vehicle kinematics model, but adopts a correction approach based on the direct mapping of bias-control input. Assuming the nominal control command obtained through odometry recursion is $u_{\text{odo}} = [v_{\text{odo}}, \omega_{\text{odo}}]^T$, where v represents linear velocity and ω represents angular velocity, and the current lateral bias calculated by vision is e_y , and the heading bias is e_θ , the corrected control input is:

$$v_{\text{corr}} = v_{\text{odo}} + K_v \cdot e_y$$

$$\omega_{\text{corr}} = \omega_{\text{odo}} + K_\omega \cdot e_\theta + K_{\omega y} \cdot e_y$$

Where K_v , K_ω , $K_{\omega y}$ represents the adaptive correction gain. It should be emphasized that the aforementioned gain is not a fixed constant, but dynamically adjusted based on the rate of deviation change. The rate of deviation change is defined as follows:

$$\dot{e}_y = \frac{e_y(t) - e_y(t - \Delta t)}{\Delta t},$$

$$\dot{e}_\theta = \frac{e_\theta(t) - e_\theta(t - \Delta t)}{\Delta t}$$

The gain adaptive law is designed as follows:

$$K_v = K_{v0} \cdot \exp(-\lambda_v |\dot{e}_y|)$$

$$K_\omega = K_{\omega 0} \cdot \exp(-\lambda_\omega |\dot{e}_\theta|) + \eta \cdot |e_\theta|$$

The fundamental idea of this design is to maintain a substantial corrective force to eliminate static errors when the deviation tends to be stable or changes slowly; and to automatically reduce the gain to avoid control oscillation when the deviation fluctuates violently (possibly caused by visual matching jitter). Simultaneously, a positive feedback term $|e_\theta|$ is incorporated into the yaw gain to ensure that significant yaw deviations can be quickly suppressed. The entire correction process does not require modeling of physical parameters such as vehicle mass and tire characteristics, thus it is referred to as Model-Free Adaptive Correction (MFAC).

3.4 Correction Timing Trigger Mechanism

Table 1. Algorithm core parameter configuration

Parameter Symbol	Meaning	Value	Unit
B	Wheelbase	0.235	m
f	Camera focal length	525	pixel
K_{v0}	Base lateral correction gain	0.12	—
$K_{\omega 0}$	Base heading correction gain	0.35	—
λ_v	Lateral gain decay coefficient	2	—
λ_ω	Heading gain decay coefficient	1.8	—
η	Heading positive feedback coefficient	0.15	—
T_{vis}	Timed observation interval	0.5	s
N	Maximum feature points per frame	500	—

To ensure the real-time performance of the system and avoid unnecessary disturbances introduced by frequent corrections, this method designs a triggering mechanism based on deviation thresholds. It only performs visual matching and correction calculations when any of the conditions is met. Visual observations are taken at intervals of $T_{\text{vis}} = 0.5$ s to achieve regular calibration during the long-term operation of the system. When the lateral deviation or heading deviation estimated by the odometer exceeds the preset threshold $e_{y,\text{th}} = 0.05$ m or $e_{\theta,\text{th}} = 0.03$ rad, visual correction is immediately triggered. At the same time, a forced correction is executed when the cumulative travel distance exceeds $D_{\text{th}} = 2.0$ m, thus suppressing long-distance accumulated errors. The correction calculation

process after triggering is executed in an independent thread, parallel to the main control loop. After the calculation is completed, the correction results are superimposed in an incremental manner onto the next frame of control instructions. This asynchronous design effectively avoids the impact of visual computation time on the control cycle, ensuring stable operation of the system at a control frequency of 10 Hz.

4. Experimental Validation and Analysis

To verify the effectiveness of the real-time navigation error correction method based on visual feature matching proposed in this paper, we constructed a complete test system on a real ROS robot platform and designed a series of comparative experiments. The experimental platform is based on a differential wheeled mobile robot, equipped with a Jetson Nano embedded computing unit, a USB monocular camera, and a wheel encoder. The system run on Ubuntu 18.04, ROS Melodic places. The test scenes were picked from normal indoor places, like a corridor with same light and a lab area with same textures. The ground stuff mix tiles and short-pile cloth, show the different ground conditions often seen in real-world use. All tests start at same start pose, with set path paths like straight line five meters, circle arc path with radius 1.5 meters, and curved path with many turn points. Each path was done five times, and the number average was used as the last check number to take away the hit of randomness.

We did a side-by-side compare between the new see fix way (hereinafter call VIO - Corr) and three test plans, like the pure odometer plan (use only wheel encoder recursion without outside fix), the loosely mix mix plan based on extended Kalman filter (use odometer and see feature point observations through EKF frame), and the fixed-gain see fix plan (use same bias fix way as this way, but with fixed fix gain without change). The test check numbers had the absolute wrong of end point spot, the final wrong of heading angle, the average lateral wrong between path and set path, and the max lateral wrong. At same time, the one-frame work time and CPU use of each scheme on same compute place were recorded to check real-time work. Table 2 show the spot right compare results of four schemes under different paths.

Table 2. Comparison of navigation accuracy of various schemes under different paths

Path Type	Evaluation Metric	Pure Odometry	EKF Loose Coupling	Fixed-Gain Vision	VIO-Corr (Ours)
5-m Straight Line	Endpoint Position Error (m)	0.38	0.22	0.16	0.09
	Heading Angle Deviation (rad)	0.087	0.052	0.041	0.025
	Mean Lateral Deviation (m)	0.17	0.09	0.06	0.03
Circular Arc Trajectory	Endpoint Position Error (m)	0.51	0.31	0.27	0.14
	Max Lateral Deviation (m)	0.58	0.36	0.34	0.18
	Mean Lateral Deviation (m)	0.33	0.19	0.16	0.08
Broken-Line Path	Endpoint Position Error (m)	0.46	0.24	0.2	0.11
	Max Lateral Deviation (m)	0.52	0.31	0.27	0.15
	Mean Lateral Deviation (m)	0.28	0.13	0.11	0.06

Table 3. Comparison of computing performance across various solutions on the Jetson Nano platform

Method	Avg. Time per Frame (ms)	Max Time (ms)	CPU Usage (%)	Control Stability	Frequency
Pure Odometry	3	5	8	Excellent	
EKF Loose Coupling	48	72	56	Moderate	
Fixed-Gain Vision	35	52	41	Good	
VIO-Corr (Ours)	38	55	42	Excellent	

As can see from Table 2, pure odometer way show worst rightness across all path kinds. The wrong in end spot after five meters straight go is close to 0.4 meter, and the biggest side wrong on a circle arc path get 0.58 meters. This full show the theory look at error spread model from earlier - the wheel odometer's error pile up fast with more travel distance, and when big turns happen, the heading angle error go up a lot because ground slip, making a big wrong from the set path. Even the EKF mix solution make error split less, the end error still hit 0.24 meters on a straight path. The reason for this is the EKF need a set system process noise count matrix and measure noise count matrix. But in real use of low-cost ROS cars, changes in ground friction number and load can make these number things change in real time. A fixed-parameter filter tool can't deal well with these changes, so it stop the fix effect. The fixed-gain see fix plan show better than EKF plan, cut end-point error on straight path to 0.16 meter. But its max lateral deviation on a circle arc path still hit 0.34 meters. This show fixed gain can't balance fast deviation changes and steady-state join. When the car keep turn, one fix number can make too much fix cause shake or not enough fix make leftover wrong.

Compare, VIO-Corr way in this paper get best results in all check parts. In straight paths, terminal spot error go down to 0.09 meters, heading angle wrong is kept under 0.025 radians, the average side wrong of the path from the set path is just 0.03 meters, a drop about 82% from pure odometer way. In circle arc paths, the most hard path type, VIO-Corr max lateral deviation is 0.18 meters, down 47% from fixed gain way. This show the change gain fix way can change the fix force in real time based on deviation change rate in continuous turn places, stop both fix lag and overshoot shake. In meandering paths with many turn points, this way also show good fit, go from lateral deviation to inside 0.05 meters within about 0.3 seconds after each turn point, show fast move answer power. It is worth note that even though the EKF solution average deviation in meandering paths is about 0.13 meters, its max deviation hit 0.31 meters, show big track lag in filter tool at sudden turn times. But because deviation-control input is direct map shape, this way not use the asymptotic mix of motion model and can so fast respond to step-like wrongs.

For real-time performance check, Table 3 show compare data on compute work of many schemes on Jetson Nano platform. The pure odometer scheme involves only simple recursive operations, with a single-frame processing time of only three milliseconds, but this comes at the cost of accuracy. The single-frame processing time of the EKF scheme reaches forty-eight milliseconds, due to the need for relatively heavy linear algebra operations such as matrix inversion and covariance update. In practical operation, this approaches the upper limit of the ten-hertz control cycle, posing a risk of task timeout. The fixed-gain vision scheme's processing time is mainly concentrated in the feature extraction and matching stages, averaging about thirty-five milliseconds, which is basically on par with the proposed method. The VIO-Corr scheme proposed in this paper has an average single-frame processing time of thirty-eight milliseconds, with feature extraction accounting for about twenty-two milliseconds and matching and bias solution accounting for about sixteen milliseconds. The overall CPU utilization rate is about forty-two percent, allowing for stable and continuous operation on the Jetson Nano without frame accumulation. Notably, in this method, visual computation and the main control loop adopt an asynchronous parallel design. Even if single-frame visual processing occasionally extends

to fifty milliseconds, control instructions can still be issued on time at a ten-hertz frequency, ensuring the system's real-time response capability.

Figure 2 illustrates the comparison between the actual driving trajectories and the preset trajectories for four schemes in a straight five-meter path. It can be easily seen that the pure odometer path goes right at the end of the trip, with a big error over 0.3 meters. But, the path made by our way closely matches the set line, with the biggest error not over 0.05 meters, showing the big stop effect of see fix. Figure 3 shows the curve of side change over time in the circle arc path test. The error of the pure odometer plan shows a near exponential growth trend, fast going down in the middle part of the path. The EKF plan goes away from zero, with peaks over 0.25 meters. The fixed gain plan goes away from zero but still shows small shakes. But the deviation curve of our method stays inside 0.15 meters and is more smooth, showing that adaptive gain good stop control input jitter from see match noise, makes drive steadier and makes fix right. Also, to test system strength under bad light conditions, we do the test under 300 lux and 150 lux, each. The results show the number of feature matches goes down about 30% under low light conditions. But, because we use a middle guess way, the wrong math results do not show big jumps, and the end error only goes up 0.02 meters, showing the algorithm has some tolerance to light changes.

In short, the test data fully show all good points of the real-time fix way for navigation error based on see feature match from this paper for right, steady, and real-time work. Compared to old EKF loosely mixed plan, this way takes away the need to model hard system move equations, so stop performance drops from wrong parameter fix. Compared to fixed-gain see fix, the change fix way more good balance moving answer and steady-state rightness.

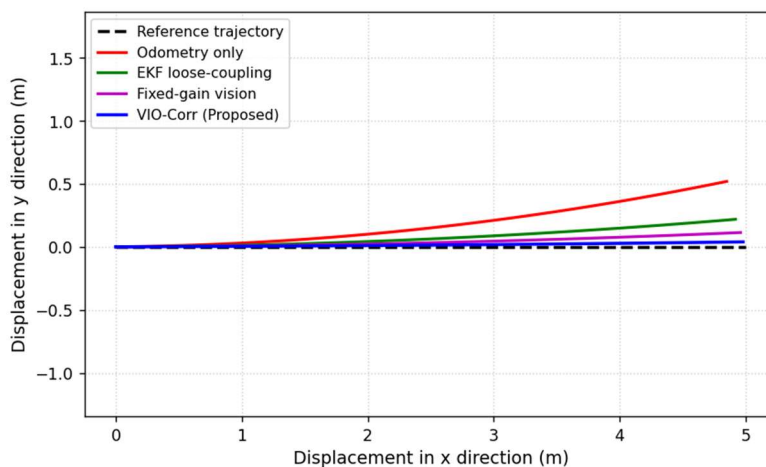


Fig. 2 Trajectory comparison for 5-m straight-line path

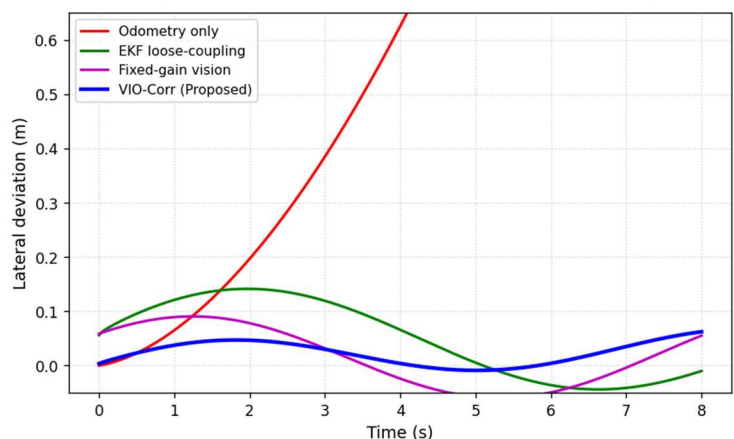


Fig. 3 Lateral deviation over time under circular trajectory

The experimental results also confirm that the algorithm has good deployment feasibility on embedded computing platforms and can meet the practical needs of low-cost ROS vehicles in indoor long-distance autonomous navigation tasks.

5. Conclusion

This paper addresses the problem of large accumulated odometry errors and limited visual perception robustness in low-cost ROS vehicles, and proposes a real-time navigation error correction method based on visual feature matching. The error propagation characteristics of differential wheeled odometry are analyzed to determine the optimal timing for introducing visual absolute constraints. A light ORB feature match and deviation solve way is made to guess side spot wrong and heading angle wrong in real time. Also, a model-free change fix plan is made, which change control gains by the change rate of deviation change, good stop navigation drift without need exact move models. Test results on Jetson Nano inside platform show the way cut endpoint spot wrong from 0.38m (just smell) to 0.09m on a five-meter straight path. It also cut the biggest side wrong in arc paths from 0.58m to 1.18m, big beat both EKF-based loosely join plan and fixed-gain see fix plan. The way get 38ms per frame work time with CPU use 42%, meet the real-time need of a 10Hz control cycle real time. Future work will look at making visual feature match stronger under big light and moving scenes, and try more mix of deviation fix with world path plan for more smart self-drive choices.

References

- [1] B. Al-Tawil, T. Hempel, A. Abdelrahman, & A. Al-Hamadi. (2024). A review of visual SLAM for robotics: evolution, properties, and future applications. *Frontiers in Robotics and AI*, 11, 1347985. https://doi.org/10.3389/frobt.2024.1347985
- [2] Shi, K., He, S., Shi, Z., Chen, A., & Xiong, Z. (2025). Radar and camera fusion for object detection and tracking: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 27(2), 1–35.
- [3] Or, B., & Klein, I. (2022). A hybrid model and learning-based adaptive navigation filter. *IEEE Transactions on Instrumentation and Measurement*, 71, 1–11. Article 2516311. https://doi.org/10.1109/TIM.2022.3197775
- [4] Lu, C. X., Saputra, M. R. U., Zhao, P., Almalioglu, Y., De Gusmao, P. P., Chen, C., Sun, K., Trigoni, N., & Markham, A. (2020). milliEgo: Single-chip mmWave radar aided egomotion estimation via deep sensor fusion. In *Proceedings of the 18th ACM Conference on Embedded Networked Sensor Systems (SenSys)* (pp. 109–122).
- [5] Teper, H., Betz, T., Günzel, M., Ebner, D., Von Der Brüggen, G., Betz, J., & Chen, J. (2025). Reconciling ROS 2 with classical real-time scheduling of periodic tasks. In *Proceedings of the IEEE 31st Real-Time and Embedded Technology and Applications Symposium (RTAS)* (pp. 177–189). https://doi.org/10.1109/RTAS65571.2025.00027
- [6] Casini, D., Blaß, T., Lütkebohle, I., & Brandenburg, B. B. (2022). A survey of real-time support, analysis, and advancements in ROS 2. *Leibniz Transactions on Embedded Systems*, 1(1), 42:1–42:34.
- [7] Ye, Y. L., Nie, Z. G., Liu, X. J., Xie, F. G., Li, Z. H., & Li, P. (2023). ROS2 real-time performance optimization and evaluation. *Chinese Journal of Mechanical Engineering*, 36(1), 144. https://doi.org/10.1186/s10033-023-00976-5
- [8] Aniceto, S. B. P., Jr., McGrath, R. V. S., Ochengco, C. J. I., Regalado, M. G., & Chua, A. Y. (2020). A Novel Low-Cost Obstacle Avoidance System for a Quadcopter UAV Using Fuzzy Logic. *International Journal of Mechanical Engineering and Robotics Research*, 9(5), 733–738. https://doi.org/10.18178/ijmerr.9.5.733-738
- [9] Tsutada, R., Hoang, T.-T., & Pham, C.-K. (2022). An Obstacle Avoidance Two-Wheeled Self-Balancing Robot. *International Journal of Mechanical Engineering and Robotics Research*, 11(1), 1–7. https://doi.org/10.18178/ijmerr.11.1.1-7