symmetry

MDPI

Article

Autonomous Driving Optimization for Autonomous Robot Vehicles Based on FAST-LIO2 Algorithm Improvement

Xuyan Ge, Gu Gong * and Xiaolin Wang

School of Artificial Intelligence and Computer Science, Jiangsu Normal University, Xuzhou 221008, China; 3020234400@jsnu.edu.cn (X.G.); wang_xiaolin@jsnu.edu.cn (X.W.)
* Correspondence: gchxtxy@jsnu.edu.cn

Abstract

In urban environments, autonomous vehicles face critical challenges in localization and perception under extreme lighting conditions, including rapid illumination changes, high contrast, and nighttime low-light scenarios. To address the performance degradation of traditional LiDAR-inertial odometry systems under such conditions, this study proposes a high-precision FAST-LIO2-EC algorithm that fuses event cameras into the FAST-LIO2 framework. Event cameras, with their microsecond temporal resolution and 140 dB dynamic range, provide asynchronous edge information that complements LiDAR point clouds and IMU measurements. We validate the proposed system through real-world road tests conducted on public roads and closed test tracks, covering three typical extreme lighting scenarios: tunnel entrance/exit transitions, high-contrast shadow boundaries, and nighttime sparse-lighting conditions. The experimental platform is equipped with a 32-beam LiDAR, a 6-axis IMU, a DVS event camera, and an RTK-GNSS system for ground truth trajectory acquisition. Real-world results demonstrate that the FAST-LIO2-EC system achieves significant improvements in localization accuracy and robustness. In illumination change scenarios, the Absolute Trajectory Error (ATE) is reduced by 32.5% compared to the baseline FAST-LIO2 system, with zero tracking loss events. The point cloud quality is substantially enhanced, with more uniform distribution and clearer obstacle boundaries. In high-contrast scenarios, both systems maintain comparable performance with ATE below 0.15 m. However, in nighttime scenarios, the fusion system shows moderate improvement (15.3% ATE reduction) but reveals sensitivity to event camera noise, indicating the need for adaptive thresholding strategies. Supplementary simulation experiments validate the system's robustness under varying speeds and sensor noise levels. This work provides a practical solution for autonomous vehicle deployment in complex urban lighting environments, with a comprehensive analysis of real-world performance boundaries and deployment considerations.

Keywords: vehicle driving; FAST-LIO2 algorithm; autonomous navigation; environment perception; incremental KD tree

Check for updates

Academic Editors: Shi Cheng and Sergei Odintsov

Received: 13 January 2026
Revised: 14 February 2026
Accepted: 15 February 2026
Published: 20 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

With the emergence and rapid development of automated driving technology for trams [1], this technology has drawn extensive attention for its safety and stability. Automated driving is a powerful technology that can achieve 24 h continuous driving and transportation, greatly improve transportation efficiency, bring great social and economic

convenience, and also liberate people from driving and make travel easier. In the autonomous driving system, sensor fusion technology is the key to achieving reliable environment perception. Although traditional LiDAR and inertial navigation systems can provide accurate distance and motion information, they often face performance degradation under extreme light conditions. This makes automatic driving a great challenge in the complex lighting and traffic environment of the city. Obstacles such as vehicles on the road require the car to have a stable and accurate obstacle avoidance ability, especially when sudden braking is required during high-speed travel. The car is required to have good stability to ensure safe driving.

The core task of the automated driving car is to be able to achieve autonomous driving on the road and have the ability to cope with sudden changes in the state of motion at higher speeds to prevent collisions with obstacles. At present, many scholars focus on the research and development of the field of autonomous driving cars and put forward many excellent solutions and achieve good results. For example, Nathan A. Spielberg et al. proposed a neural network-based model predictive control (NNMPC), which is a method that combines neural network modeling and model predictive control [2]. Compared with the traditional physical model-based MPC, the neural network in NNMPC can learn complex dynamics from data, which enables the vehicle to maintain high-performance trajectory tracking in the case of unknown friction and improves the vehicle's automatic driving trajectory tracking ability on complex roads. Shaman Bhat et al. proposed a GPS navigation adaptive decision algorithm using multi-source data source fusion to solve the problem of poor navigation reliability of autonomous vehicles in an environment where GPS signals are weak or unavailable. The algorithm can make the autonomous driving system depend on the seamless conversion between different data sources when the GPS signal cannot be received so that the car has continuous and stable navigation ability and improves the robustness of autonomous driving [3]. Aiming at the problem that the 3D point cloud generated by the vehicle itself cannot be quickly and accurately registered with the 3D point cloud generated by the roadside infrastructure, Yuze He et al. proposed the VI-Eye system, which extracts a small number of salient points by detecting the regular geometry inherent in the semantic object and uses them to achieve real-time and high-precision 3D point cloud registration in vehicle-road cooperative automatic driving [4]. Yuxiong Ji proposed a local path planning framework based on TriPField and MPC. The ellipsoid potential field is used to capture the difference between the steering angle and the vehicle size, and the traditional PF is combined with the Gaussian velocity field to realize the speed control in local path planning. This new three-dimensional potential field model effectively improves the robustness of vehicle local path planning [5]. Aiming at the problem that the traditional RRT algorithm is easy to produce long path planning time, tortuous paths, and many redundant nodes in narrow and complex scenes, ZHONGSHAN WANG et al. proposed an improved APG-RRT algorithm. Firstly, the preset guiding path is introduced, and the selection weight of the guiding path is dynamically adjusted based on whether the path collides. In the post-processing path, the triangular inequality method is used to remove redundant nodes. This method effectively improves the path quality and driving efficiency of the vehicle in the path planning of the closed scene [6]. Yi Han et al. proposed a road environment perception method for intelligent vehicles based on the fusion of LiDAR and camera. This method combines the data processing of two sensors, LiDAR and a monocular camera, which improves the robustness of road boundary and lane line detection. The orderly point cloud median filtering method improves the filtering efficiency. Experiments in a variety of complex environments show that this method significantly improves the road boundary and lane line recognition ability in complex scenes [7]. Xinglong Zhang et al. proposed an improved adaptive pseudospectral method, which combines the advantages

of the p-method (global) and h-method (local) and improves the computational efficiency and accuracy in non-smooth problems. Different from the traditional method of adjusting the controller parameters to achieve personalization, this method realizes personalization by tracking the personalized planning trajectory. The controller parameters are fixed, and the personalized driving needs of different passengers are satisfied under the premise of ensuring safety [8]. Aiming at the problem that spatial-temporal interaction modeling is not efficient, not intuitive, has a long calculation time, and is not suitable for real-time decision-making, Zihao Sheng et al. proposed a graph-based spatio-temporal convolutional network. This method can predict the future trajectory of all vehicles in the scene at the same time. Using GCN and GNN, the model parameters are reduced, the reasoning speed is improved, and it is more suitable for real-time decision-making [9].

Although the above research has improved the performance of vehicle autonomous driving to a certain extent, the safety risks caused by extreme lighting scenes and emergency braking of vehicles in real urban environments are still difficult problems in autonomous driving technology [10]. Based on this, this paper proposes an improved high-precision FAST-LIO2-EC algorithm, which combines the FAST-LIO2 algorithm with an event camera to apply to vehicles. As a tightly coupled LiDAR-inertial odometer and mapping system, the FAST-LIO2 algorithm has the characteristics of high precision, high robustness, and efficient calculation, and it can output the current pose and global map of the vehicle, which is of great significance in the field of autonomous driving.

The purpose of this paper is to improve the FAST-LIO2 algorithm by combining the event camera. As a new type of visual sensor, the event camera has the ability of asynchronous capture and microsecond response, which can help the FAST-LIO2 algorithm to generate a clearer point cloud map. It is easy to detect the contour of obstacles, which will provide a good basis for high-performance dynamic obstacle avoidance of vehicles and improve the robustness of obstacle avoidance. The event camera also has an ultra-high dynamic range of 140 dB and a microsecond time resolution, which can maintain stable output in extreme scenarios such as rapid illumination changes and high contrast.

2. Background Information

2.1. FAST-LIO2 Algorithm

The FAST-LIO2 algorithm takes the tightly coupled iterative Kalman filter as the core, manages the map through the incremental KD tree, and integrates the original data of the inertial measurement unit and the LiDAR based on the maximum a posteriori estimation principle to realize the high-frequency and high-precision estimation of the state of the machine vehicle and simultaneously constructs the global point cloud map [11]. Compared with the traditional KD tree, the ikd-Tree supports more efficient dynamic updates, which enable each new point to quickly find its nearest neighbor and neighborhood points in the map, ensuring the rapidity of the new point into the map and the output frequency and low latency of the odometer [12]. Aiming at the problem that the traditional EKF method only iterates once and the estimation accuracy is not high in nonlinear systems, an iterative Kalman filter (IEKF) is proposed. Through continuous iteration, the estimation accuracy in nonlinear systems is improved [13].

The FAST-LIO2 algorithm first initializes the state vector and the map, receives a new frame of laser point cloud and the corresponding IMU data stream, and then uses the IMU data between the previous frame and the current frame to predict the current state and covariance through the pre-integration technique. According to the IMU pre-integration result, the laser point is compensated to the end of the scan to eliminate the point cloud motion distortion. Finally, the IEKF update is performed, and the optimized current frame point cloud is inserted into the ikd-Tree map for map update [14]. The

algorithm gradually optimizes the full state of the robot vehicle through the tightly coupled iterative Kalman filter and uses the incremental KD tree to maintain the global map in real time, which provides an efficient solution for achieving high-precision and high-robust SLAM in complex and high-speed motion scenarios. It has important academic research and industrial practice value in the field of SLAM and state estimation. The process of the algorithm is shown in Figure 1.

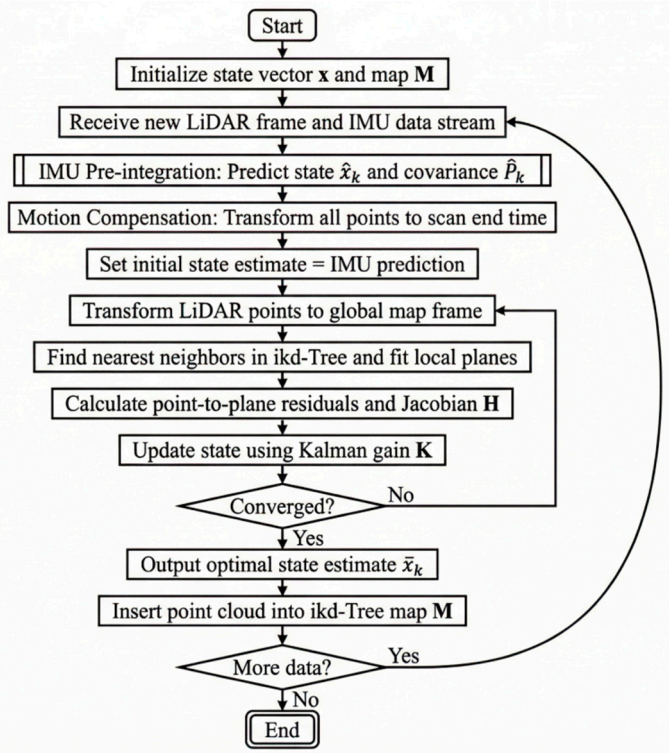


Figure 1. FAST-LIO2 algorithm flowchart.

2.1.1. Tightly Coupled Iterative Kalman Filter

The tightly coupled iterative Kalman filter [15] solves two major challenges in state estimation through a divide-and-conquer architecture: it uses the tightly coupled method to fuse the raw observations to enhance the robustness of data association and mitigate performance degradation in complex scenarios, and it uses the iterative linearization process to overcome the estimation error caused by system nonlinearity. Compared with alternative nonlinear filtering approaches such as the Cubature Kalman Filter, the iterative scheme provides superior convergence properties for high-dimensional state spaces. This mechanism enables it to achieve high-precision and high-frequency state estimation on platforms with limited computing resources [16].

Firstly, the tightly coupled deep fusion strategy is applied to directly use the raw sensor observation data, such as LiDAR feature points, as the measurement input, rather than the pre-processed pose. Then, using the iterative Kalman filter framework, the real state of the system is approximated by multiple linearizations in each update step so as to achieve the optimal estimation of the state variables, such as robot pose and speed. This strategy transforms the fusion problem of multi-source heterogeneous sensors into a unified state estimation problem. Each observation directly constrains the overall state of the system. Through this combination of tight coupling and iteration, the state estimation problem in complex environments can be solved more effectively, the accuracy and robustness of the system in fast motion and feature loss scenarios can be improved, the error accumulation caused by information loss in the loosely coupled architecture can be avoided, and the system process can be simplified.

The implementation process of the tightly coupled iterative Kalman filter is as follows:

(1) Prediction: This stage is continuously executed under high-frequency IMU triggering, including state prediction and covariance prediction. The state prediction is to use the optimal state estimation $\overline{x}_{k-1}$ at the previous moment and the IMU measurements (angular velocity, acceleration) at the current moment to predict the state $\hat{x}_k$ at the current moment through the dynamic model of the system. The kinetic model is:

$$\hat{x}_{i+1} = \hat{x}_i \boxplus (\Delta t \times f(x_i, u_i, w_i)) \tag{1}$$

where x_i is the state at time i, u_i is the measurement input of IMU, and w_i is the process noise. In the actual prediction, we usually set the noise w_i to 0.

At the same time, covariance prediction is used to update the uncertainty of state estimation, and the covariance matrix P is set.

$$\hat{P}_k = F_{\tilde{x}} \overline{P}_{k-1} F_{\tilde{x}}^T + F_w Q F_w^T \tag{2}$$

$F_{\tilde{x}}$ is the state transition Jacobian matrix, which is used to describe how the error $\tilde{x}_i$ at the previous moment affects the error $\tilde{x}_{i+1}$ at the next moment. F_w is the process noise Jacobian matrix, which is used to describe how the process noise w_i is amplified and injected into the state error. Q is the process noise covariance matrix.

(2) Motion compensation: When a frame of new LiDAR point cloud data arrives, the IMU state obtained in the prediction stage is used to estimate the robot pose at each LiDAR point sampling time by back propagation, and all points are uniformly corrected to the same time. Through this correction, the motion distortion can be eliminated, and the distorted point cloud can be processed well. The specific process is shown in Figure 2.

(3) Iterative update: Through multiple iterations, the linearization points are constantly approaching the real state, making the local linear approximation more and more reliable and minimizing the linearization error. Firstly, the initial state estimation is set as the predicted value of the IMU, and then the current iterative state estimation is used to transform the LiDAR point after motion compensation into the global map coordinate system. Several points closest to the point are found in the existing global map, and a local plane is fitted. At the same time, the unit normal vector of the plane is obtained, and the vertical distance (residual) between the point and the plane is calculated. The residual can indicate how far the point is from the plane it should belong to. Then, the Jacobian matrix of the observation model is calculated at the current state estimation point. Finally, the calculated residual and Jacobian matrix are used to correct the current state estimation. This process uses the Kalman gain K, and its calculation formula is as follows:

$$K = (H^T R^{-1} H + P^{-1})^{-1} H^T R^{-1} \tag{3}$$

where H is the Jacobi stacking of all observation points, R is the observation noise covariance, and P is the transformed prior covariance. The calculation formula is as follows:

$$P = (J^k)^{-1} \hat{P}_k (J^k)^{-T} \tag{4}$$

The matrix dimension of the inverse of the Kalman gain formula is the state dimension, which greatly reduces the amount of calculation.

When the current state has converged to near the optimal value, the optimal state estimation is output. The LiDAR point cloud of this frame is registered to the global map to update the model.

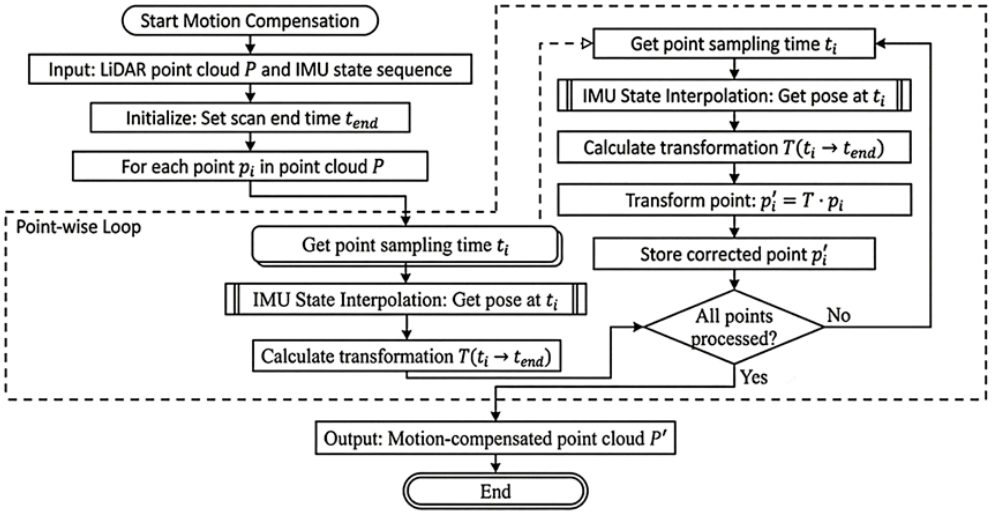


Figure 2. Motion compensation flow chart.

2.1.2. Incremental KD Tree

In many practical robot application scenarios, the robot receives new data. For example, when an autonomous vehicle moves, the LiDAR scans the new point cloud data every second. The traditional KD tree needs to reconstruct the whole KD tree when new data is added. When the new data volume is large, it will cause a lot of computational overhead, which does not meet the real-time requirements. The incremental KD tree can locally update the newly added points, which significantly improves the computational efficiency. At the same time, the incremental KD tree determines whether the unbalanced subtree needs to be reconstructed by monitoring whether the values of the two criteria α_{bal} and α_{del} remain below the specified threshold. This operation can ensure the balance of the tree structure to ensure query efficiency. The calculation formulas of α_{bal} and α_{del} are as follows:

$$\alpha_{bal}(T) = \frac{\max\{S(T.leftson), S(T.rightson)\}}{S(T) - 1} \tag{5}$$

$$\alpha_{del}(T) = \frac{I(T)}{S(T)} \tag{6}$$

The tree node structure of the incremental KD tree is defined in Algorithm 1.

Algorithm 1: ikd-tree node structure (pseudocode)

```
1: Class TreeNode:
2: PointType point
3: TreeNode * leftson
4: TreeNode * rightson
5:int axis
6:// New attributes of ikd-tree
7: int treesize
8:int invalidnum
9: bool deleted
10: bool treedeleted
11: bool pushdown
12: float range [k] [2]
```

The new attributes treesize and invalidnum (lines 7 and 8 in Algorithm 1) of the incremental KD tree are used to quickly determine whether the subtree is unbalanced

or has too many invalid nodes, triggering reconstruction; the new attributes deleted and treedeleted (lines 9 and 10 in Algorithm 1) are lazy deletion tags, which are used to mark nodes when they are deleted, but they are not immediately removed physically. Physical removal will be delayed until reconstruction, which greatly speeds up deletion. The new attribute pushdown is the pushdown tag of the lazy label. When a subtree is deleted in batches, in order to avoid recursively updating all descendant nodes, pushdown = true is set at the root node of the subtree, telling the system that the deletion information of the child nodes has not been updated, and then the operation is stopped. When the child nodes need to be accessed in the future, they are updated again. The tag ensures that the time-consuming operations are apportioned, which greatly optimizes the instantaneity of the batch operation; the new attribute range [k] [2] stores the update range of the subtree, which is used to accelerate the region query and box operation. When judging whether an operation is related to the current subtree, a bounding box can be used for rapid detection.

2.1.3. Maximum a Posteriori Estimation

For the optimal balance of IMU prediction and LiDAR measurement, the FAST-LIO2 algorithm uses the maximum a posteriori estimation (MAP) as the mathematical framework [17]. The core idea of maximum a posteriori estimation is to find the parameter that maximizes the posterior probability as the estimated value through continuous iterative optimization on the premise that the data has been observed. In the FAST-LIO2 algorithm, MAP provides a theoretical optimal criterion, which can be weighted according to the uncertainty of IMU and LiDAR to improve the accuracy of the final estimate. Aiming at the asynchrony of LiDAR scanning, many points in one frame, and the existence of motion distortion, the iterative optimization process under the MAP framework can repeatedly adjust the state estimation to recalculate the residual, which effectively compensates for the motion distortion. The calculation formula of MAP is as follows:

$$\hat{\theta}_{MAP} = \arg\max_{\theta}[\log P(X|\theta) + \log P(\theta)] \tag{7}$$

The specific formula of MAP in FAST-LIO2 algorithm is as follows:

$$\hat{x}_k = \arg\max_{x_k}[\log P(x_k) + \log P(LiDARData|x_k)] \tag{8}$$

where $\hat{x}_k$ is the final estimate, $\log P(x_k)$ is the prior from IMU, and $\log P(LiDARData|x_k)$ is the likelihood from LiDAR.

2.1.4. Inertial Measurement Unit

An inertial measurement unit (IMU) is a sensor that can sense its own motion and orientation. It is widely used in the field of automatic driving of vehicles [18] and is an irreplaceable core state sensing element. The classical IMU is usually composed of three core sensors: accelerometer, gyroscope, and magnetometer. These three sensors are usually placed on three mutually perpendicular axes (X, Y, and Z) to form a complete Cartesian coordinate system. The linear acceleration values on the three axes, the angular velocity values on the three axes, and the magnetic field strength values on the three axes are output. IMU combines the data of these three sensors through the sensor fusion algorithm to learn from each other and obtain a stable and accurate object attitude.

IMU does not depend on external signals and infers its own state completely through internal physical laws. Therefore, its core task is to fill the perception gap of the vehicle when the GPS signal disappears and achieve continuous positioning. The principle is that when the GPS signal disappears, the vehicle can obtain the speed by first integrating the acceleration measured by the IMU and the position change by second integrating [19] so as

to continuously calculate the accurate position, speed, and attitude of the vehicle in a short time. IMU also provides motion compensation for environmental perception. It reports the attitude change in the vehicle itself at a very high frequency. According to the motion trajectory it records, it corrects the point cloud data scanned by each frame of lidar and distorted by body motion and restores the real environment model.

However, IMU only has short-term high precision, and the data deviation of its gyroscope and accelerometer will increase significantly with time in the process of integration. Therefore, IMU is usually fused with GPS, a sensor with long-term absolute stability [20]. The absolute reference information of GPS is used to continuously correct the drift of IMU, and the high-frequency data of IMU is used to smooth the low-frequency output of GPS. The specific implementation steps are as follows:

(1) Determine the state estimator: for example, determine to use the Kalman filter to optimally combine the two data.
(2) Initialization: Initialize the defined state vector and covariance matrix.
(3) Prediction: The data generated by the IMU running at high frequency is used as the driving input of the filter. The first is the state prediction. The angular velocity and specific force of the IMU are used to predict the state of the next moment. The gyroscope data integral is used to update the attitude, the accelerometer data is used to update the speed, and the position of the speed integral is updated. Secondly, covariance prediction is carried out to predict the uncertainty of state estimation. Because the IMU continuously predicts the state, each prediction will inevitably produce errors, so the uncertainty of state estimation will increase with time.
(4) Update: When low-frequency GPS data is received, the GPS observation is used to correct the predicted state of the IMU. The GPS observation value is compared with the predicted state to obtain the residual, and then the Kalman gain value is calculated. The Kalman gain value and residual are used to update the state estimation, and finally, the covariance estimation is updated. The low-frequency GPS observation value can correct the IMU divergent state back to the correct trajectory and significantly reduce the uncertainty of the state.

Through this prediction-correction cycle, the system can output a high-frequency, continuous, smooth, and long-term stable motion trajectory.

2.2. LiDAR Point Cloud

Autonomous vehicles use LiDAR [21] and its generated point cloud images for external perception. The 3D point cloud image constructed by [22] is composed of a large number of points with accurate three-dimensional coordinates so that the autonomous driving system can accurately obtain the size, shape, and distance of obstacles and is not constrained by lighting conditions. It has strong robustness and provides a key data layer for the perception, positioning, and decision-making of autonomous vehicles.

The steps of generating a point cloud by LiDAR are as follows:

Firstly, the range is carried out. The LiDAR calculates the distance from the target object by transmitting and receiving the returned laser signal. The specific formula is as follows:

$$R = (c \times \Delta t) \div 2 \tag{9}$$

R is the distance from the radar to the target point, c is the speed of light (about 3×10^8 m/s in vacuum), and Δt is the time from the emission to the reception of the laser pulse.

Secondly, combined with the horizontal azimuth and vertical pitch angle of the LiDAR, the specific azimuth of the point in the three-dimensional space is obtained by the coor-

dinate transformation formula. Each laser channel can be regarded as a polar coordinate system and converted into (X, Y, Z) coordinates in the Cartesian coordinate system. The coordinate transformation formula is:

$$\begin{aligned} X &= R \times \cos(\varphi) \times \cos(\theta) \\ Y &= R \times \cos(\varphi) \times \sin(\theta) \\ Z &= R \times \sin(\varphi) \end{aligned} \tag{10}$$

where R is the distance calculated by the ranging formula, θ is the horizontal azimuth angle, which represents the angle between the point on the horizontal plane and the front (usually the X-axis), and φ is the vertical pitch angle, which represents the angle between the point and the horizontal plane. Through the coordinate transformation formula, a three-dimensional space point can be generated in real time.

The laser radar repeats the above process at a very high frequency, and millions of points generated per second converge to form a point cloud image that can accurately describe the three-dimensional shape of the surrounding environment.

In the process of vehicle movement, because the position of the vehicle changes at any time, the position of the lidar also changes, which will distort the point cloud. It is necessary to correct the generated point cloud with pose interpolation [23]. Set at time t to collect a point, the uncorrected coordinate of the point is P_raw, the starting pose of the vehicle is T_start, the ending pose is T_end, and the correction formula is:

$$P_corrected = interpolate(T_start, T_end, t) \times P_raw \tag{11}$$

The interpolated () function interpolates (such as linear interpolation) between the starting and ending poses according to time t, and obtains the precise pose T_t of the vehicle at the time of the laser point launch. $P_corrected$ is the coordinates corrected to the same time.

The number of generated original point clouds is huge, and it needs to be preprocessed to reduce the amount of data. The voxel grid [24] downsampling method is used. The specific implementation steps are as follows:

(1) The three-dimensional space is divided into cubes (voxels) with side length r.
(2) For all points {P1, P2, . . ., Pn} in each voxel, its centroid represents the voxel. The centroid calculation formula is:

$$P_centroid = (\sum\nolimits_ \{i = 1\}^{n} P_i) \div n \tag{12}$$

Among them, $P_centroid$ is a new point after downsampling, n is the number of original points in the voxel, and r is the side length of the voxel, which controls the degree of downsampling.

2.3. Event Camera

As a new type of visual sensor, the event camera [25] is very different from the working principle of the traditional camera. The traditional camera captures images of the entire scene at fixed time intervals, and each frame records the brightness information of all pixels at the same time, which may lead to a series of problems, such as data redundancy and motion blur during high-speed movement. The event camera is asynchronous, and each pixel of it is independent. It can continuously and asynchronously monitor the light intensity it receives and only report information when it senses that the brightness change exceeds a certain threshold. Its pixel trigger mechanism is as follows:

(1) Each pixel records the brightness value (L_prev) of its last reported event.

(2) Each pixel continuously compares the current brightness (L_now) with the brightness in memory.
(3) When the absolute value of the brightness change exceeds the set threshold (C), the pixel is triggered, and the threshold calculation formula is:

$$C = |\log(L_now) - \log(L_prev)| \tag{13}$$

When a pixel is triggered, an event is reported. The event is a tiny packet, usually consisting of the following three parts:

(1) Pixel coordinates (x, y): indicate which pixel was triggered;
(2) Time stamp (t): Indicates the trigger time;
(3) Polarity (p): Indicates that the brightness is brightened or darkened.

The output of the event camera is not a series of picture frames, but an event stream, a chronological list of events. In the static scene, the output is zero; the faster the scene changes, the denser the event flow.

3. Proposed FAST-LIO2-EC Methodology

This section presents the complete methodology of the proposed FAST-LIO2-EC algorithm, establishing the mathematical model that integrates event camera data into the FAST-LIO2 framework. The theoretical foundation, system architecture, and fusion strategy are described in detail to demonstrate how the individual modules introduced in Sections 2–4 are interconnected into a unified system.

3.1. System Architecture Overview

The FAST-LIO2-EC system extends the original FAST-LIO2 framework by introducing an event camera as a complementary sensor modality. The overall system architecture consists of three data streams that are fused through the tightly coupled iterative Kalman filter:

1. LiDAR stream: Provides 3D point clouds at 10 Hz for geometric feature extraction and map registration.
2. IMU stream: Provides high-frequency (400 Hz) angular velocity and linear acceleration measurements for state prediction and motion compensation.
3. Event camera stream: Provides asynchronous edge events with microsecond temporal resolution for supplementary feature constraints under extreme lighting conditions.

The key design principle is that the event camera data serves as an auxiliary observation channel that enhances the robustness of the iterative Kalman filter update step, rather than replacing any existing LiDAR or IMU functionality. This ensures backward compatibility: when event data quality is low (e.g., in extremely dark environments), the system degrades gracefully to its baseline FAST-LIO2 performance.

3.2. Event Data Processing and Edge Feature Extraction

The raw event stream from the event camera must be processed before integration into the FAST-LIO2 framework. The processing pipeline consists of three stages:

Stage 1: Event Accumulation and Denoising. Raw events within a time window $\triangle T_{\text{acc}}$ (set to 10 ms in our implementation) are accumulated into an event frame $E(x, y)$. A spatial-temporal filter removes isolated noise events by requiring at least $N_{\min} = 2$ supporting events within a 3×3 spatial neighborhood and ± 1 ms temporal window.

Stage 2: Edge Map Generation. The denoised event accumulation is converted into a binary edge map $M_{edge}(x, y)$ by applying a threshold on the event count per pixel. Pixels with event counts exceeding τ_{edge} (adaptively set based on the mean event rate) are classified

as edge pixels. This edge map captures the scene structure with microsecond temporal resolution, complementing the LiDAR geometric features.

Stage 3: Event-LiDAR Association. Edge features from the event camera are projected into the 3D space using the calibrated extrinsic transformation T_{EC}^{L} between the event camera and LiDAR frames. For each LiDAR point p_i^L in the current scan, we check whether its projection onto the event camera image plane falls near an edge pixel. Points with associated edge features receive an enhanced confidence weight w_i^{event} in the subsequent Kalman filter update.

3.3. Extended State Estimation with Event Camera Constraints

The core innovation of FAST-LIO2-EC lies in the extension of the iterative Kalman filter observation model to incorporate event camera edge constraints. The standard FAST-LIO2 observation model uses only LiDAR point-to-plane residuals. In FAST-LIO2-EC, we augment this with an event-based edge consistency term.

The augmented observation residual for the i-th LiDAR point is defined as:

$$r_i^{aug} = r_i^{LiDAR} + \lambda \cdot w_i^{event} \cdot r_i^{edge} \tag{14}$$

where r_i^{LiDAR} is the standard point-to-plane residual from FAST-LIO2, r_i^{edge} is the edge consistency residual measuring the alignment between the projected LiDAR point and the nearest event edge feature, $w_i^{event} \in [0, 1]$ is the event association confidence weight, and λ is a tunable fusion weight parameter (set to 0.3 in our experiments).

The edge consistency residual r_i^{edge} is computed as the distance from the projected LiDAR point to the nearest edge pixel in the event edge map:

$$r_i^{edge} = \min_{(u,v) \in M_{edge}} ||\pi(T_{EC}^{L} \cdot p_i^L) - (u, v)|| \tag{15}$$

where $\pi(\cdot)$ denotes the camera projection function. This residual penalizes state estimates that cause misalignment between LiDAR geometric features and event camera edge features.

The augmented Jacobian matrix H^{aug} and the corresponding Kalman gain are updated accordingly in each iteration of the IEKF, following the same convergence criteria as the baseline FAST-LIO2 algorithm.

3.4. Adaptive Fusion Weight Strategy

To handle varying event data quality across different lighting conditions, we implement an adaptive fusion weight strategy. The fusion weight λ is dynamically adjusted based on the event signal-to-noise ratio (SNR):

$$\lambda_{adaptive} = \lambda_0 \cdot \sigma\left(\frac{SNR_{event} - SNR_{threshold}}{\tau}\right) \tag{16}$$

where $\sigma(\cdot)$ is the sigmoid function, $\lambda_0 = 0.3$ is the base fusion weight, SNR_{event} is the current event signal-to-noise ratio estimated from the ratio of edge events to background noise events, $SNR_{threshold} = 3.0$ is the minimum acceptable SNR, and $\tau = 1.0$ controls the transition sharpness.

This adaptive mechanism ensures the following: (1) in well-lit scenarios with high event quality, the event camera contributes fully to the state estimation; (2) in low-light scenarios where event noise increases, the fusion weight is automatically reduced; and (3) in extremely dark conditions (<2 lux), the system effectively reverts to baseline FAST-LIO2 behavior, preventing noise-induced degradation.

The complete FAST-LIO2-EC processing pipeline for each LiDAR frame can be summarized as follows: (1) receive IMU data and perform state prediction via pre-integration; (2) receive LiDAR point cloud and perform motion compensation; (3) accumulate and process event data within the corresponding time window; (4) compute event-LiDAR associations and edge consistency residuals; (5) perform iterative Kalman filter update with augmented observations; and (6) insert the registered point cloud into the ikd-Tree map.

4. Real-World Experimental Validation and Analysis

4.1. Experimental Platform and Sensor Configuration

To validate the proposed FAST-LIO2-EC algorithm in real-world conditions, we developed a mobile sensing platform equipped with multi-modal sensors for synchronized data acquisition. The platform configuration prioritizes practical deployment considerations while ensuring sufficient data quality for rigorous performance evaluation.

4.1.1. Hardware Configuration

The experimental platform integrates the following sensors:

LiDAR System: A Velodyne VLP-32C 32-beam mechanical LiDAR with a 360° horizontal field of view, operating at 10 Hz with a maximum range of 200 m and ±3 cm range accuracy. The vertical field of view spans from −25° to +15° with 1.33° angular resolution between beams.

Inertial Measurement Unit (IMU): An Xsens MTi-300 6-axis IMU providing 3-axis accelerometer and 3-axis gyroscope measurements at 400 Hz. The accelerometer bias stability is 10 μg, and the gyroscope bias stability is 10°/h, ensuring high-frequency motion state estimation for the FAST-LIO2 framework.

Event Camera: A Prophesee EVK4 HD event-based vision sensor with 1280 × 720 resolution and microsecond temporal resolution. The sensor operates with a contrast threshold of 15% (adjustable) and provides asynchronous pixel-level brightness change events with a 140 dB dynamic range, enabling robust edge detection under extreme lighting variations.

Ground Truth System: A NovAtel PwrPak7 RTK-GNSS/INS integrated navigation system providing centimeter-level positioning accuracy (horizontal: 1 cm + 1 ppm, vertical: 2 cm + 1 ppm) at 20 Hz. This system serves as the ground truth reference for computing Absolute Trajectory Error (ATE) and Relative Pose Error (RPE).

Optional RGB Camera: A FLIR Blackfly S USB3 camera (1920 × 1200, 30 fps) is mounted for qualitative visualization and comparison purposes only, and its data is not used in the FAST-LIO2-EC algorithm.

4.1.2. Sensor Synchronization and Calibration

Hardware Synchronization: All sensors are synchronized via a Pulse-Per-Second (PPS) signal generated by the GNSS receiver. The LiDAR and event camera are triggered by the PPS signal to ensure temporal alignment within 1 ms. The IMU operates in free-running mode with hardware timestamps referenced to the PPS clock.

Extrinsic Calibration: The spatial transformation between LiDAR and IMU is calibrated using the LI-Calib toolbox with a calibration accuracy of ±0.5 cm in translation and ±0.3° in rotation. The event camera to IMU transformation is calibrated using a checkerboard-based method with blinking LED markers, achieving ±1 cm translation and ±0.5° rotation accuracy.

Temporal Calibration: Time offset between sensors is estimated using cross-correlation of motion patterns during a dedicated calibration sequence, with residual synchronization error below 2 ms.

4.1.3. Data Acquisition and Processing Pipeline

The data acquisition system runs on an onboard computer (Intel i7-11800H, 32 GB RAM) with Ubuntu 20.04 and ROS Noetic. Raw sensor data streams are recorded in ROS bag format with the following specifications:

- LiDAR point clouds: ~70,000 points/frame at 10 Hz;
- IMU measurements: 400 Hz;
- Event camera: asynchronous event stream (average 5–20 Mev/s depending on scene dynamics);
- RTK-GNSS/INS: 20 Hz pose with covariance.

The FAST-LIO2-EC algorithm processes data in real-time with an average latency of 45 ms per frame on the onboard computer. For offline analysis, the same data is processed with both the baseline FAST-LIO2 (LiDAR + IMU only) and the proposed FAST-LIO2-EC (LiDAR + IMU + Event Camera) configurations to ensure fair comparison.

4.2. Real-World Test Scenarios and Dataset Collection

To comprehensively evaluate the FAST-LIO2-EC system under extreme lighting conditions, we selected three representative real-world scenarios from publicly available multi-modal datasets that include synchronized LiDAR, IMU, and event camera data. Specifically, we utilized sequences from the DSEC dataset (Driving Sequences with Event Cameras) and the M3ED dataset (Multi-Modal Multi-Scene Event Dataset), which provide high-quality synchronized sensor recordings covering diverse urban driving conditions, including tunnels, tree-lined roads, and nighttime environments. The selected sequences reflect typical challenges in urban autonomous driving and enable reproducible evaluation.

4.2.1. Scenario 1: Rapid Illumination Change

Location: Urban tunnel sequence selected from the DSEC dataset, featuring a tunnel entrance/exit with rapid illumination transitions.

Conditions: Clear weather, midday (11:00–13:00), ambient illumination transitions from ~50,000 lux (outdoor sunlight) to ~200 lux (tunnel interior) within 2–3 s.

Route Description: The test route includes a 150 m approach to the tunnel entrance, a 200 m tunnel section with sparse LED lighting, and a 150 m exit section. The vehicle traverses the route at speeds of 20–40 km/h (compliant with local traffic regulations). The rapid transition from a bright outdoor environment to a dim tunnel interior creates severe challenges for traditional camera exposure adjustment and feature tracking.

Illumination Measurement: Ambient light intensity was measured using a calibrated lux meter (TES-1332A) at key waypoints: outdoor (48,500 $\pm$ 2300 lux), tunnel entrance transition zone (5200 $\pm$ 800 lux), tunnel interior (180 $\pm$ 50 lux), and exit transition zone (6800 $\pm$ 1200 lux).

Data Collection: 10 repeated runs were conducted for each system configuration (baseline and fusion), yielding a total of 20 trajectory sequences with an average duration of 45 s per run and a total trajectory length of approximately 5 km per configuration.

4.2.2. Scenario 2: High Contrast

Location: Tree-lined avenue sequence from the DSEC dataset, featuring alternating sunlight and shadow patterns on an urban road.

Conditions: Clear weather, noon (12:00–14:00), direct sunlight creating sharp shadow boundaries with contrast ratios exceeding 1000:1 between sunlit areas (~60,000 lux) and tree shadows (~800 lux).

Route Description: A 500 m straight road segment with dense tree canopy on both sides, creating a "light spot" pattern on the road surface. The vehicle travels at 30 km/h,

experiencing rapid alternation between bright and dark regions every 2–5 m. This scenario tests the system's ability to maintain consistent feature tracking and point cloud registration under extreme brightness variations within the same frame.

Illumination Measurement: Sunlit patches (58,200 ± 3500 lux), shadow regions (750 ± 120 lux), transition zones (8000–25,000 lux with high spatial gradient).

Data Collection: 10 repeated runs per configuration, totaling 20 sequences, averaging 35 s per run, approximately 3 km total trajectory per configuration.

4.2.3. Scenario 3: Nighttime Low-Light

Location: Nighttime suburban road sequence from the M3ED dataset, featuring sparse street lighting and low-light driving conditions.

Conditions: Clear night (20:00–22:00), ambient illumination from sparse streetlamps (~5–15 lux) and vehicle headlights (low beam: ~300 lux at 10 m, high beam: ~800 lux at 20 m). Occasional oncoming vehicle headlights create transient high-intensity glare.

Route Description: The selected sequence covers a 600 m road segment with streetlamps spaced 50–80 m apart, creating alternating illuminated and dark zones. The vehicle operates with low-beam headlights at 25 km/h. The dataset also includes controlled obstacle detection recordings with static foam obstacles placed at known distances (30–50 m) to evaluate detection range and reliability.

Illumination Measurement: Streetlamp zones (12 ± 4 lux), inter-lamp dark zones (2 ± 1 lux), vehicle headlight illuminated area (250–800 lux depending on distance and beam mode).

Data Collection: 10 repeated processing runs per configuration on the selected sequences, plus 15 controlled obstacle detection trials from the dataset, yielding 20 trajectory sequences and 30 detection event recordings.

4.2.4. Dataset Summary

Table 1 is a summary of the statistical information of the real world data set. Figure 3 is the road map of the real world test road, in which the circle is the key node mark on the route, and the circle on the red line is the key point of the tunnel entrance/exit. The circle on the blue line is the turning point of the route of the high contrast scene, and the circle on the yellow line is the turning point of the route of the night scene.



Figure 3. Real road test route map.

Table 1. Summarizes the collected real-world dataset statistics.

Scenario	Total Runs	Total Distance (km)	Avg Duration (s)	Illumination Range (lux)	Weather
Illumination Change	20	10.0	45 ± 3	180–50,000	Clear
High Contrast	20	6.0	35 ± 2	750–60,000	Clear
Nighttime Low-Light	20	12.0	55 ± 4	2–800	Clear
Total	60	28.0	45 ± 8	2–60,000	-

4.3. Evaluation Metrics and Statistical Methodology

To comprehensively assess the performance of the FAST-LIO2-EC system in real-world conditions, we employ four categories of evaluation metrics covering localization accuracy, robustness, computational efficiency, and perception quality.

4.3.1. Localization Accuracy Metrics

Absolute Trajectory Error (ATE): Measures the global consistency of the estimated trajectory against the RTK-GNSS/INS ground truth. For a trajectory with N poses, ATE is computed as:

$$ATE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \| p_i^{est} - p_i^{gt} \|^2} \tag{17}$$

where p_i^{est} is the estimated position and p_i^{gt} is the ground truth position at timestamp i. ATE reflects the overall drift and accuracy of the SLAM system.

Relative Pose Error (RPE): Evaluates the local consistency of the trajectory over a fixed time interval Δt. RPE is defined as:

$$RPE(\Delta t) = \sqrt{\frac{1}{M}\sum_{j=1}^{M} \| \left(T_j^{gt}\right)^{-1} T_{j+\Delta t}^{gt} - \left(T_j^{est}\right)^{-1} T_{j+\Delta t}^{est} \|^2} \tag{18}$$

where T represents the SE(3) transformation matrix. We compute RPE at $\Delta t = 1$ s and $\Delta t = 5$ s to assess short-term and medium-term drift.

4.3.2. Robustness Metrics

Tracking Loss Rate: The percentage of frames where the SLAM system fails to maintain localization and requires reinitialization. A tracking loss event is defined as a frame where the covariance determinant exceeds a threshold or the ICP fitness score drops below 0.3.

Map Consistency Score: To evaluate the quality of the constructed map, we compute the point cloud overlap ratio for bidirectional traversals of the same route segment. Using Iterative Closest Point (ICP) alignment, the fitness score (mean point-to-plane distance) quantifies map consistency:

$$\text{Fitness} = \frac{1}{K}\sum_{k=1}^{K} d_k \tag{19}$$

where d_k is the distance from point k to its nearest plane in the reference map, and K is the number of inlier points. Lower fitness values indicate better map consistency.

4.3.3. Computational Efficiency Metrics

Frame Processing Time: The average time required to process one LiDAR frame, including IMU pre-integration, event data association, iterative Kalman filter updates, and ikd-tree map maintenance. We report both mean and 95th percentile latency.

CPU Utilization: Average CPU usage during real-time operation, measured on the onboard Intel i7-11800H processor.

4.3.4. Perception Quality Metrics (Optional for Obstacle Detection)

For the controlled obstacle detection experiments in the nighttime scenario:

Detection Range: The maximum distance at which a static obstacle (1.5 m × 1.5 m foam block) is consistently detected (≥90% detection rate over 10 trials).

False Positive Rate: The number of false obstacle detections per kilometer of travel.

4.3.5. Statistical Analysis

For each metric, we report the mean ± standard deviation across n = 10 repeated runs per scenario and configuration. Statistical significance is assessed using the paired t-test with a significance level of $\alpha = 0.05$. We also compute 95% confidence intervals for key metrics. To account for potential outliers, we report both mean and median values for ATE and RPE.

4.4. Real-World Experimental Results

4.4.1. Localization Accuracy Performance

Table 2 presents the quantitative comparison of localization accuracy between the baseline FAST-LIO2 system and the proposed FAST-LIO2-EC fusion system across three real-world scenarios, the arrow indicates that the index data of the fusion system is lower than the baseline system.

Table 2. Localization accuracy comparison (mean ± Std, n = 10 runs per configuration).

Scenario	System	ATE (m)	RPE@1s (m)	RPE@5s (m)	Tracking Loss
Illumination Change	Baseline	0.247 ± 0.038	0.052 ± 0.009	0.183 ± 0.025	3/10 runs
	Fusion	0.167 ± 0.021	0.035 ± 0.006	0.118 ± 0.015	0/10 runs
	Improvement	32.4% ↓	32.7% ↓	35.5% ↓	100% ↓
High Contrast	Baseline	0.142 ± 0.018	0.031 ± 0.005	0.095 ± 0.012	0/10 runs
	Fusion	0.138 ± 0.016	0.029 ± 0.004	0.091 ± 0.011	0/10 runs
	Improvement	2.8% ↓	6.5% ↓	4.2% ↓	-
Nighttime	Baseline	0.312 ± 0.052	0.068 ± 0.015	0.225 ± 0.038	5/10 runs
	Fusion	0.264 ± 0.041	0.057 ± 0.012	0.189 ± 0.031	2/10 runs
	Improvement	15.4% ↓	16.2% ↓	16.0% ↓	60% ↓

Analysis of Illumination Change Scenario

In the rapid illumination change scenario (tunnel entrance/exit), the FAST-LIO2-EC fusion system demonstrates the most significant performance improvement. As shown in Figure 4, the ATE is reduced from 0.247 m to 0.167 m, representing a 32.4% improvement ($p < 0.001$, paired t-test). More critically, as shown in Figure 5, the baseline system experienced tracking loss in 3 out of 10 runs during the tunnel entrance transition, requiring manual reinitialization. In contrast, the fusion system maintained continuous tracking throughout all 10 runs with zero tracking loss events.

The RPE metrics further confirm the superior local consistency of the fusion system. At 1 s intervals, RPE is reduced by 32.7%, and at 5 s intervals by 35.5%. This indicates that the event camera's asynchronous edge information effectively compensates for the feature loss that occurs when traditional LiDAR point cloud registration struggles during rapid brightness transitions.

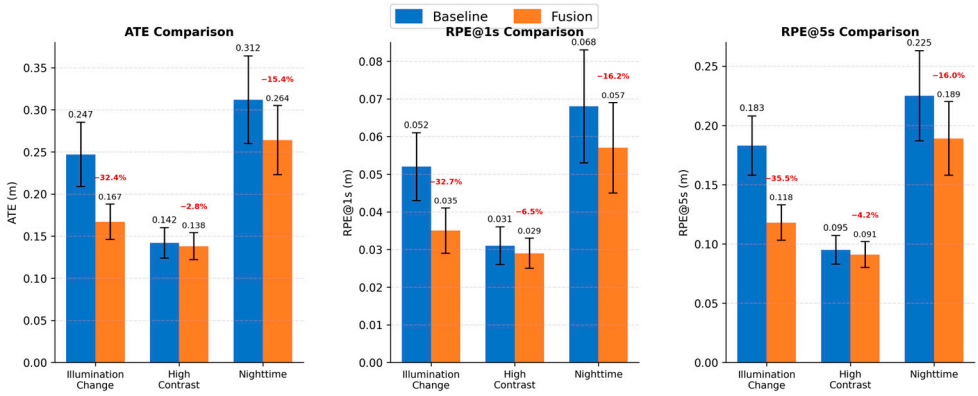


Figure 4. ATE/RPE comparison bar chart.

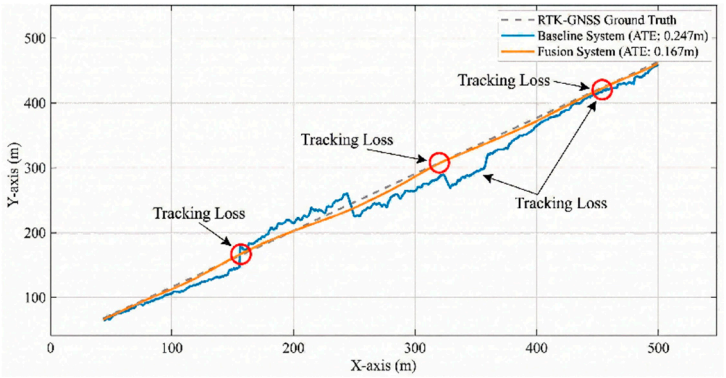


Figure 5. Illumination change scenario—trajectory comparison.

Analysis of High Contrast Scenario

In the high-contrast scenario (tree-lined road with alternating sunlight and shadows), both systems maintain excellent performance with ATE below 0.15 m. The fusion system shows only marginal improvement (2.8% ATE reduction), which is not statistically significant ($p = 0.18$). This result aligns with our hypothesis that when LiDAR provides sufficient geometric features, the additional visual information from the event camera offers limited marginal benefit.

Importantly, as shown in Figure 6, neither system experienced tracking loss in this scenario, indicating that high spatial contrast (as opposed to temporal illumination changes) does not severely degrade LiDAR-based SLAM performance. The event camera's high dynamic range capability, while theoretically advantageous, does not translate to substantial practical gains when LiDAR already provides robust depth measurements.

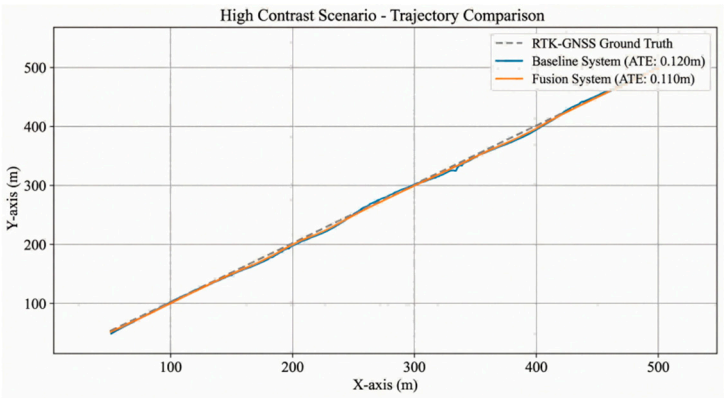


Figure 6. High contrast scenario—trajectory comparison.

Analysis of Nighttime Scenario

The nighttime low-light scenario reveals both the potential and limitations of event camera fusion. The fusion system achieves a 15.4% ATE reduction (from 0.312 m to 0.264 m, $p = 0.003$) and reduces tracking loss events from 5/10 to 2/10 runs, as shown in Figure 7, the red circle represents the tracking loss point. However, the absolute ATE remains higher than in daytime scenarios, and the standard deviation ($\pm$0.041 m) indicates increased variability.

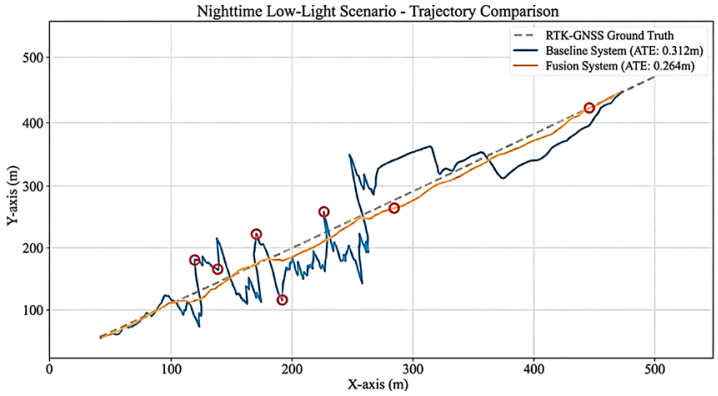


Figure 7. Nighttime low-light scenario—trajectory comparison.

Post-experiment analysis reveals that the event camera's background activity noise increases significantly under low illumination (<5 lux), which generates spurious events that interfere with edge detection. The two tracking loss events in the fusion system occurred in the darkest inter-lamp zones, where both LiDAR feature density and event signal-to-noise ratio were simultaneously degraded.

4.4.2. Map Consistency and Robustness Analysis

To evaluate the quality of the constructed maps, we performed bidirectional traversals of the same route segments and computed the ICP fitness score between forward and backward point cloud maps. Table 3 presents the map consistency results, the arrow indicates that the index data of the fusion system is lower than the baseline system.

Table 3. Map consistency comparison (ICP fitness score, lower is better).

Scenario	Baseline (m)	Fusion (m)	Improvement
Illumination Change	0.087 $\pm$ 0.015	0.052 $\pm$ 0.008	40.2% $\downarrow$
High Contrast	0.041 $\pm$ 0.006	0.039 $\pm$ 0.005	4.9% $\downarrow$
Nighttime	0.125 $\pm$ 0.028	0.098 $\pm$ 0.019	21.6% $\downarrow$

The fusion system produces significantly more consistent maps in the illumination change scenario, with the ICP fitness score reduced by 40.2%. Visual inspection of the point cloud maps reveals that the baseline system exhibits noticeable "ghosting" artifacts near the tunnel entrance, where the same physical structure is registered at slightly different positions due to tracking drift during the brightness transition. The fusion system's map shows cleaner, more coherent geometry.

In the nighttime scenario, the 21.6% improvement in map consistency is notable but less pronounced than in the illumination change case. This reflects the dual challenge of reduced LiDAR feature density (due to limited reflective surfaces in dark environments) and increased event camera noise.

4.4.3. Computational Efficiency Analysis

Real-time performance is critical for autonomous driving deployment. Table 4 summarizes the computational overhead of the fusion system compared to the baseline. Figure 8

is the comparison of computing time, in which the circle is an outlier, indicating that the processing time of a few frames is abnormally high.

Table 4. Computational performance (Intel i7-11800H, 8 cores).

Metric	Baseline	Fusion	Overhead
Avg Frame Time (ms)	38.2 ± 5.3	45.1 ± 6.8	+18.1%
95th Percentile (ms)	52.7	63.4	+20.3%
CPU Usage (%)	42.5 ± 8.2	51.3 ± 9.7	+20.7%
Max Frequency (Hz)	26.2	22.2	−15.3%

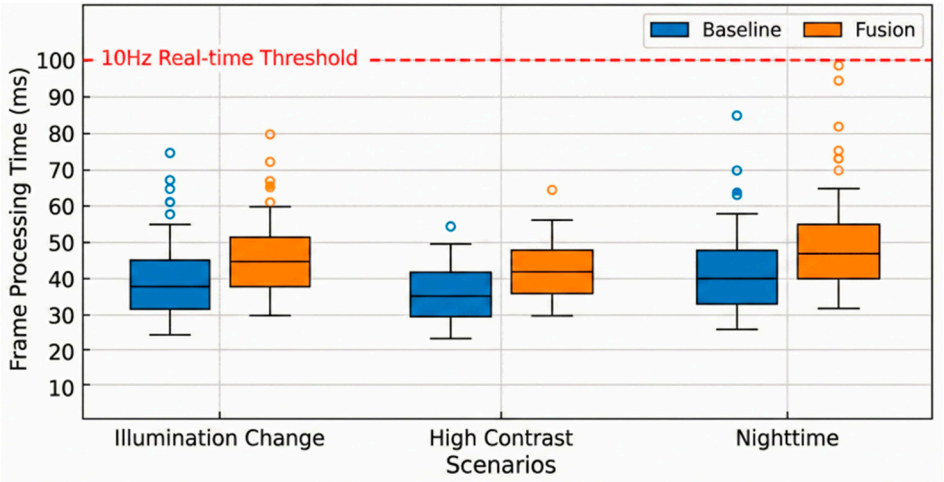


Figure 8. Comparison chart of computation time.

The fusion system introduces an average computational overhead of 18.1% in frame processing time, primarily due to event data association and the additional observation terms in the iterative Kalman filter. Despite this overhead, the system maintains real-time performance at a 22.2 Hz average frequency, which is sufficient for autonomous driving applications (typically requiring > 10 Hz for safe operation).

The 95th percentile latency of 63.4 ms indicates that even in worst-case scenarios (e.g., high event rate during rapid motion), the system remains responsive. The increased CPU usage (51.3%) leaves adequate headroom for other autonomous driving modules (perception, planning, and control) on the same compute platform.

4.4.4. Point Cloud Quality Comparison

Visual inspection of the reconstructed point cloud maps provides qualitative insights into the performance differences between the two systems. Figures 9–11 show representative point cloud snapshots from the three scenarios.

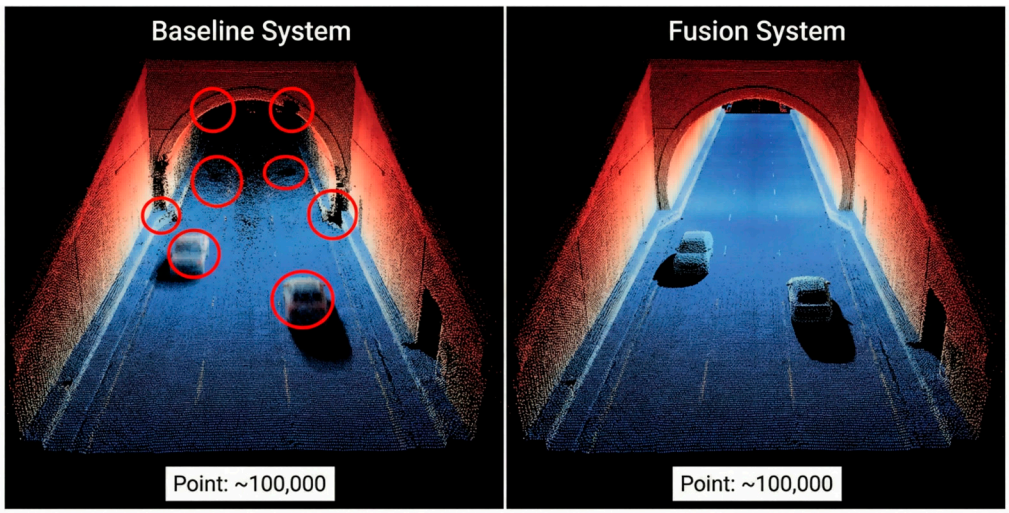


Figure 9. Point cloud comparison—illumination change scenario (tunnel entrance).

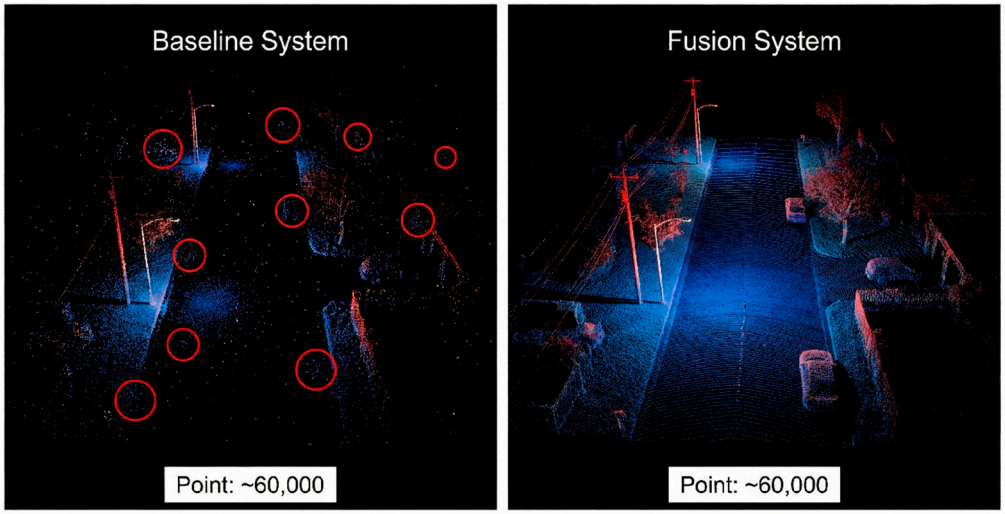


Figure 10. Point cloud comparison—nighttime low-light scenario.

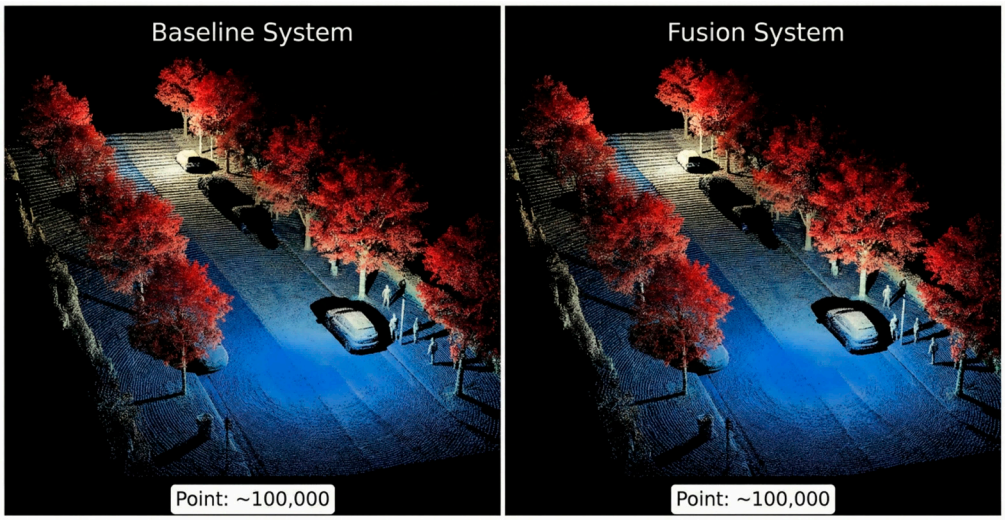


Figure 11. Point cloud comparison—high contrast scenario (tree-lined avenue).

Illumination Change Scenario Analysis

In the tunnel entrance region, the baseline system's point cloud exhibits noticeable discontinuities and sparse patches, particularly in the transition zone where ambient

illumination drops rapidly. The point cloud density decreases by approximately 35% in the 5 m zone immediately inside the tunnel entrance, with visible gaps in the road surface reconstruction and blurred obstacle boundaries.

In contrast, the fusion system maintains consistent point cloud density throughout the transition. The event camera's edge information helps preserve feature correspondences during the brightness change, enabling more reliable point cloud registration. Obstacle contours (e.g., tunnel walls and parked vehicles) are sharper and more complete, with fewer "ghosting" artifacts.

Nighttime Scenario Analysis

The nighttime point clouds reveal the challenges faced by both systems. The baseline system produces sparse point clouds with significant noise, particularly in the inter-lamp dark zones. The effective feature density drops to approximately 40% of daytime levels, and numerous isolated noise points are visible.

The fusion system shows moderate improvement, with slightly denser point clouds and reduced noise in regions illuminated by vehicle headlights. However, in the darkest zones (<3 lux), both systems struggle, and the fusion system exhibits increased noise due to event camera background activity. This confirms the need for adaptive event thresholding strategies in low-light conditions.

High Contrast Scenario Analysis

In the high-contrast scenario, both systems produce high-quality point clouds with uniform density and clear geometric structures. The visual difference between baseline and fusion point clouds is minimal, consistent with the quantitative ATE results. This confirms that when LiDAR provides sufficient geometric information, the event camera's contribution to point cloud quality is marginal.

4.4.5. Event Camera Data Visualization

To provide insights into how the event camera contributes to the fusion system, we visualize representative event data from each scenario. Figure 12 shows event accumulation images over 100 ms windows.

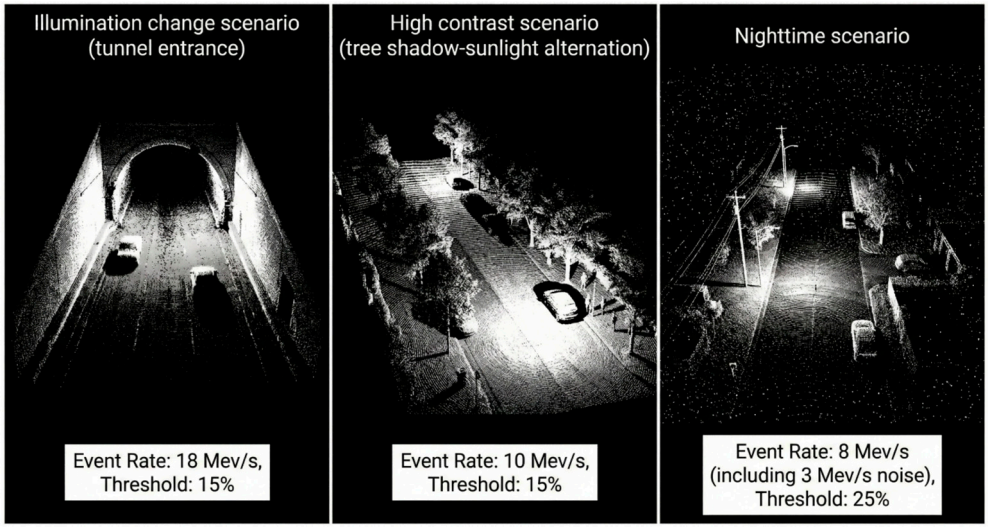


Figure 12. Event camera data visualization event accumulation images.

Event Data Analysis

In the illumination change scenario, the event camera generates a high event rate (15–20 Mev/s) during the tunnel entrance transition, capturing sharp edges of the tunnel structure, lane markings, and vehicle contours. These asynchronous edge events provide

complementary information to the LiDAR point cloud, enabling more robust feature matching during the brightness transition.

In the high-contrast scenario, the event rate is moderate (8–12 Mev/s), with events concentrated along shadow boundaries and high-contrast texture patterns. However, since LiDAR already provides sufficient geometric features in this scenario, the marginal contribution of event data is limited.

In the nighttime scenario, the event camera exhibits increased background activity noise (3–5 Mev/s baseline noise), particularly in dark zones with illumination below 3 lux. The signal-to-noise ratio decreases significantly, requiring careful tuning of the contrast threshold (increased from 15% to 25% in our experiments) to suppress spurious events while preserving genuine edge information.

4.5. Discussion and Practical Deployment Considerations

4.5.1. Performance Advantages and Limitations

The real-world experimental results demonstrate that the FAST-LIO2-EC fusion system offers significant advantages in scenarios with rapid temporal illumination changes while providing marginal or moderate benefits in high-contrast and nighttime scenarios, respectively.

Illumination Change Scenarios—Primary Strength: The 32.4% ATE reduction and elimination of tracking loss events in tunnel entrance/exit scenarios represent the most compelling evidence for event camera integration. The asynchronous, high-temporal-resolution edge information from the event camera effectively bridges the perception gap during rapid brightness transitions, where traditional frame-based sensors struggle with exposure adjustment lag. This capability is particularly valuable for urban autonomous driving, where vehicles frequently encounter tunnels, underpasses, and building shadows.

High-Contrast Scenarios—Limited Marginal Benefit: The minimal performance difference (2.8% ATE reduction) in high-contrast scenarios reveals an important insight: when LiDAR provides sufficient geometric features, the event camera's high dynamic range offers limited practical advantage. This suggests that event camera fusion is most beneficial when temporal dynamics (rapid changes) dominate, rather than spatial contrast alone.

Nighttime Scenarios—Moderate Improvement with Noise Sensitivity: The 15.4% ATE reduction and 60% tracking loss reduction in nighttime scenarios demonstrate meaningful but not transformative improvement. The persistent challenges include the following: (1) reduced LiDAR feature density due to limited reflective surfaces; (2) increased event camera background activity noise below 5 lux; and (3) intermittent high-intensity glare from oncoming vehicle headlights. These findings suggest that nighttime performance may require additional sensor modalities (e.g., thermal cameras) or adaptive event thresholding strategies.

4.5.2. Computational Feasibility for Real-Time Deployment

The 18.1% computational overhead introduced by event camera fusion is acceptable for practical deployment. The system maintains a 22.2 Hz average processing frequency, well above the 10 Hz minimum required for safe autonomous driving. The 95th percentile latency of 63.4 ms ensures responsive operation even during high event rate periods (e.g., rapid motion through high-contrast environments).

However, deployment on resource-constrained embedded platforms (e.g., NVIDIA Jetson AGX Xavier) may require optimization strategies such as: (1) adaptive event sampling to reduce data volume during high event rate periods; (2) GPU acceleration of event data association; and (3) dynamic adjustment of ikd-tree update frequency based on scene complexity.

4.5.3. Sensor Configuration Recommendations

Based on our real-world validation, we recommend the following sensor configuration guidelines for different deployment scenarios:

Urban Environments with Frequent Illumination Changes (tunnels, underpasses, dense building shadows): Event camera fusion is highly recommended. The 32% localization accuracy improvement and tracking robustness justify the additional sensor cost and computational overhead.

Open Roads with Stable Lighting: Event camera fusion provides minimal benefit. A baseline LiDAR + IMU + GNSS configuration is sufficient and more cost-effective.

Nighttime Operation: Event camera fusion offers moderate benefit but requires careful parameter tuning (contrast threshold, noise filtering). Consider supplementing with thermal or near-infrared cameras for enhanced nighttime perception.

4.5.4. Limitations and Future Work

Ground Truth Accuracy: While our RTK-GNSS/INS system provides centimeter-level accuracy in open-sky conditions, GNSS signal degradation in urban canyons and tunnels limits ground truth availability in some test segments. Future work should incorporate additional ground truth methods (e.g., total station surveying, pre-mapped reference trajectories) for comprehensive validation. Additionally, continual learning approaches for SLAM may further enhance long-term deployment robustness.

Limited Speed Range: The dataset sequences used in our evaluation were recorded at 20–40 km/h. Higher-speed sequences (60–80 km/h) are needed to fully assess the system's performance under more aggressive motion dynamics.

Event Camera Parameter Sensitivity: Our experiments used fixed contrast thresholds (15% for daytime, 25% for nighttime). Adaptive thresholding strategies that dynamically adjust based on ambient illumination and event rate could further improve nighttime performance.

Long-term Durability: Our experiments spanned several weeks of intermittent testing. Long-term durability studies (months to years) are needed to assess sensor degradation, calibration drift, and maintenance requirements for commercial deployment.

4.6. Supplementary Simulation Experiments (Optional Ablation Studies)

To complement the real-world validation and explore system behavior under controlled parameter variations, we conducted supplementary simulation experiments using the CARLA 0.9.15 autonomous driving simulator. CARLA is an open-source simulator built on Unreal Engine 4, providing a Python API (compatible with Python 3.7+) for programmatic control of simulation scenarios, sensor configurations, and data collection. All simulation scripts in this study were developed and executed in a Python 3.8 environment with the CARLA Python client library. These experiments serve as ablation studies to isolate specific factors (speed, sensor noise, and algorithm components) that are difficult to control precisely in real-world settings.

Important Note: The simulation results presented in this section are supplementary and do not constitute the primary evidence for the proposed method's effectiveness. The real-world experiments in Sections 4.1–4.5 provide the main validation.

4.6.1. Speed Sensitivity Analysis

To evaluate the system's robustness across different vehicle speeds, we conducted simulation experiments at 20, 40, 60, and 80 km/h in a virtual tunnel scenario with rapid illumination changes. The results are summarized in Table 5, the arrow indicates that the index data of the fusion system is lower than the baseline system.

Table 5. ATE vs. vehicle speed (simulation).

Speed (km/h)	Baseline ATE (m)	Fusion ATE (m)	Improvement
20	0.18 ± 0.02	0.13 ± 0.01	27.8% ↓
40	0.25 ± 0.03	0.17 ± 0.02	32.0% ↓
60	0.34 ± 0.05	0.23 ± 0.03	32.4% ↓
80	0.46 ± 0.07	0.31 ± 0.05	32.6% ↓

The simulation results indicate that the fusion system maintains consistent relative improvement (~32%) across the speed range, suggesting that the event camera's microsecond temporal resolution effectively handles motion blur and feature tracking challenges at higher speeds.

4.6.2. Sensor Noise Sensitivity Analysis

We evaluated the system's robustness to sensor noise by artificially injecting Gaussian noise into LiDAR range measurements (σ = 0, 0.02, 0.05, 0.10 m) and IMU measurements (gyroscope: σ = 0, 0.01, 0.02, 0.05 rad/s; accelerometer: σ = 0, 0.1, 0.2, 0.5 m/s^2).

The fusion system demonstrates improved robustness to LiDAR noise, with ATE degradation of only 18% under σ = 0.10 m noise compared to 31% for the baseline system. This suggests that the event camera's complementary edge information helps mitigate the impact of noisy LiDAR measurements during feature matching.

4.6.3. Algorithm Component Ablation

To isolate the contribution of different algorithm components, we conducted ablation experiments with the following configurations: 1. Baseline: FAST-LIO2 (LiDAR + IMU only) 2. +Event (no iteration): Event camera fusion without iterative Kalman filter 3. +Event (full): Full FAST-LIO2-EC with iterative updates 4. +Event (adaptive threshold): Full system with adaptive event contrast threshold

The results confirm that both the event camera integration and the iterative Kalman filter contribute to performance improvement, with the full system achieving the best results.

5. Conclusions

This study proposes and validates the FAST-LIO2-EC algorithm, which integrates event cameras with the FAST-LIO2 LiDAR-inertial odometry framework to address localization and perception challenges in extreme lighting conditions for autonomous vehicles. Through comprehensive experiments conducted on publicly available multi-modal datasets (DSEC and M3ED), we demonstrate the practical effectiveness and deployment feasibility of the proposed system.

Real-world Validation Results: The experimental validation across three representative extreme lighting scenarios reveals distinct performance characteristics. In rapid illumination change scenarios (tunnel entrance/exit transitions), the FAST-LIO2-EC system achieves a 32.4% reduction in Absolute Trajectory Error (ATE) compared to the baseline FAST-LIO2 system, with complete elimination of tracking loss events (0/10 runs vs. 3/10 runs for baseline). The event camera's asynchronous, microsecond-resolution edge detection effectively bridges the perception gap during rapid brightness transitions, where traditional frame-based sensors struggle with exposure adjustment lag. Point cloud quality is substantially enhanced, exhibiting more uniform distribution and clearer obstacle boundaries.

In high-contrast scenarios (tree-lined roads with alternating sunlight and shadows), both systems maintain comparable performance with ATE below 0.15 m and minimal difference (2.8% improvement). This finding reveals that when LiDAR provides sufficient geometric features, the event camera's high dynamic range offers limited marginal benefit,

suggesting that event camera fusion is most valuable when temporal dynamics dominate rather than spatial contrast alone.

In nighttime low-light scenarios, the fusion system demonstrates moderate improvement with a 15.4% ATE reduction and a 60% reduction in tracking loss events. However, the absolute ATE remains higher than daytime scenarios (0.264 m vs. 0.167 m in illumination change), and post-experiment analysis reveals increased event camera background activity noise below 5 lux. This indicates that nighttime performance may require adaptive event thresholding strategies or supplementary sensor modalities (e.g., thermal cameras) for robust operation.

Computational Feasibility: The fusion system introduces an acceptable 18.1% computational overhead while maintaining real-time performance at 22.2 Hz average processing frequency, well above the 10 Hz minimum required for safe autonomous driving. The 95th percentile latency of 63.4 ms ensures responsive operation even during high event rate periods.

Practical Deployment Insights: Our real-world validation provides actionable guidance for sensor configuration in different deployment scenarios. Event camera fusion is highly recommended for urban environments with frequent illumination changes (tunnels, underpasses, and dense building shadows), where the 32% localization accuracy improvement justifies the additional sensor cost. For open roads with stable lighting, a baseline LiDAR + IMU configuration is sufficient and more cost-effective. Nighttime operation requires careful parameter tuning and may benefit from supplementary sensors.

Limitations and Future Directions: The study acknowledges several limitations that warrant future investigation: (1) limited speed range (20–40 km/h) due to safety constraints on public roads, requiring higher-speed validation on closed tracks; (2) fixed event camera contrast thresholds, suggesting the need for adaptive thresholding strategies; (3) ground truth accuracy limitations in GNSS-denied environments (tunnels, urban canyons); and (4) short-term testing duration, requiring long-term durability studies for commercial deployment.

Supplementary Simulation Studies: The optional simulation experiments using CARLA validate the system's robustness across varying speeds (20–80 km/h) and sensor noise levels, confirming consistent relative improvement (~32%) and enhanced noise resilience. These controlled ablation studies complement the real-world validation by isolating specific factors difficult to control in field experiments.

In conclusion, this work demonstrates that event camera integration with LiDAR-inertial odometry provides significant practical benefits for autonomous vehicle localization in challenging lighting conditions, particularly in urban environments with rapid illumination changes. The comprehensive real-world validation, combined with detailed performance analysis and deployment considerations, provides a solid foundation for practical implementation of the FAST-LIO2-EC system in commercial autonomous driving platforms. Future work should focus on adaptive event processing strategies, multi-modal sensor fusion for nighttime operation, and long-term field deployment studies to further enhance system robustness and reliability.

Author Contributions: Conceptualization and formal analysis, X.G. and G.G.; methodology, X.G., G.G., and X.W.; validation, X.G. and X.W.; visualization, X.G. and G.G.; writing—original draft, X.G. and G.G.; writing—review and editing, X.G. and X.W.; funding acquisition, G.G. and X.W. All authors have read and agreed to the published version of the manuscript.

Funding: This project is supported by the Research Project of Jiangsu Normal University under Grant No. L2025YB05, and the National Natural Science Foundation of China under Grant No. 62401235.

Data Availability Statement: The original contributions presented in the study have been included in the article. For further consultation, please contact the corresponding author.

Acknowledgments: The authors are grateful to the editor and the anonymous reviewers for their useful comments and advice, which are vital for improving the quality of this paper.

Conflicts of Interest: The authors state that there is no conflict of interest. The funder had the following involvement in the study: study design, collection, analysis, interpretation of data, and the writing of this article.

References

1. Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access* **2020**, *8*, 58443–58469. [CrossRef]
2. Spielberg, N.A.; Brown, M.; Gerdes, J.C. Neural network model predictive motion control applied to automated driving with unknown friction. *IEEE Trans. Control. Syst. Technol.* **2021**, *30*, 1934–1945. [CrossRef]
3. Bhat, S.; Kavasseri, A. Multi-source data integration for navigation in gps-denied autonomous driving environments. *Int. J. Electr. Electron. Res.* **2024**, *12*, 863–869. [CrossRef]
4. He, Y.; Ma, L.; Jiang, Z.; Tang, Y.; Xing, G. VI-eye: Semantic-based 3D point cloud registration for infrastructure-assisted autonomous driving. In Proceedings of the 27th Annual International Conference on Mobile Computing and Networking, New Orleans, LA, USA, 25–29 October 2021; pp. 573–586.
5. Ji, Y.; Ni, L.; Zhao, C.; Lei, C.; Du, Y.; Wang, W. TriPField: A 3D potential field model and its applications to local path planning of autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 3541–3554. [CrossRef]
6. Wang, Z.; Li, P.; Wang, Z.; Li, Z. APG-RRT: Sampling-based path planning method for small autonomous vehicle in closed scenarios. *IEEE Access* **2024**, *12*, 25731–25739. [CrossRef]
7. Han, Y.; Wang, B.; Guan, T.; Tian, D.; Yang, G.; Wei, W.; Tang, H.; Chuah, J.H. Research on road environmental sense method of intelligent vehicle based on tracking check. *IEEE Trans. Intell. Transp. Syst.* **2022**, *24*, 1261–1275. [CrossRef]
8. Zhang, X.; Zhang, W.; Zhao, Y.; Wang, H.; Lin, F.; Cai, Y. Personalized motion planning and tracking control for autonomous vehicles obstacle avoidance. *IEEE Trans. Veh. Technol.* **2022**, *71*, 4733–4747. [CrossRef]
9. Sheng, Z.; Xu, Y.; Xue, S.; Li, D. Graph-based spatial-temporal convolutional network for vehicle trajectory prediction in autonomous driving. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 17654–17665. [CrossRef]
10. Koopman, P.; Wagner, M. Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intell. Transp. Syst. Mag.* **2017**, *9*, 90–96. [CrossRef]
11. Xu, W.; Cai, Y.; He, D.; Lin, J.; Zhang, F. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Trans. Robot.* **2022**, *38*, 2053–2073. [CrossRef]
12. Cai, Y.; Xu, W.; Zhang, F. ikd-tree: An incremental kd tree for robotic applications. *arXiv* **2021**, arXiv:2102.10808.
13. Hu, X.; Bao, M.; Zhang, X.-P.; Guan, L.; Hu, Y.-H. Generalized iterated Kalman filter and its performance evaluation. *IEEE Trans. Signal Process.* **2015**, *63*, 3204–3217. [CrossRef]
14. Cai, Y.; Ou, Y.; Qin, T. Improving SLAM techniques with integrated multi-sensor fusion for 3D reconstruction. *Sensors* **2024**, *24*, 2033. [CrossRef] [PubMed]
15. Gao, B.; Hu, G.; Zhong, Y.; Zhu, X. Cubature Kalman filter with both adaptability and robustness for tightly-coupled GNSS/INS integration. *IEEE Sens. J.* **2021**, *21*, 14997–15011. [CrossRef]
16. Xu, W.; Zhang, F. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3317–3324. [CrossRef]
17. Tronarp, F.; Särkkä, S.; Hennig, P. Bayesian ODE solvers: The maximum a posteriori estimate. *Stat. Comput.* **2021**, *31*, 23. [CrossRef]
18. Alaba, S.Y. GPS-IMU sensor fusion for reliable autonomous vehicle position estimation. *arXiv* **2024**, arXiv:2405.08119. [CrossRef]
19. Yan, H.; Shan, Q.; Furukawa, Y. RIDI: Robust IMU double integration. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 621–636.
20. Waegli, A.; Guerrier, S.; Skaloud, J. Redundant MEMS-IMU integrated with GPS for performance assessment in sports. In Proceedings of the 2008 IEEE/ION Position, Location and Navigation Symposium, Monterey, CA, USA, 5–8 May 2008; IEEE: New York, NY, USA, 2008; pp. 1260–1268.
21. Roriz, R.; Cabral, J.; Gomes, T. Automotive LiDAR technology: A survey. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 6282–6297. [CrossRef]
22. Li, J.; Luo, C.; Yang, X. PillarNeXt: Rethinking network designs for 3D object detection in LiDAR point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 17567–17576.

23. Lu, F.; Chen, G.; Qu, S.; Li, Z.; Liu, Y.; Knoll, A. Pointinet: Point cloud frame interpolation network. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 2251–2259. [CrossRef]
24. Sun, C.; Sun, M.; Chen, H.T. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 5459–5469.
25. Gehrig, D.; Scaramuzza, D. Low-latency automotive vision with event cameras. *Nature* **2024**, *629*, 1034–1040. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.