applied sciences

MDPI

Article

Automatic Multi-Sensor Calibration for Autonomous Vehicles: A Rapid Approach to LiDAR and Camera Data Fusion

Stefano Arrigoni *, Francesca D'Amato and Hafeez Husain Cholakkal

Department of Mechanical Engineering, Politecnico di Milano, 20156 Milan, Italy; hafeezhusain.cholakkal@polimi.it (H.H.C.)
* Correspondence: stefano.arrigoni@polimi.it

Abstract

Precise sensor integration is crucial for autonomous vehicle (AV) navigation, yet traditional extrinsic calibration remains costly and labor-intensive. This study proposes an automated calibration approach that uses metaheuristic algorithms (Simulated Annealing (SA), Genetic Algorithms (GA), and Particle Swarm Optimization (PSO)) to independently optimize rotational and translational parameters, reducing cross-compensation errors. Bayesian optimization is used offline to define the search bounds (and tune hyperparameters), accelerating convergence, while computer vision techniques enhance automation by detecting geometric features using a checkerboard reference and a Huber estimator for noise handling. Experimental results demonstrate high accuracy with a single-pose acquisition, supporting multi-sensor configurations and reducing manual intervention, making the method practical for real-world AV applications.

Keywords: autonomous vehicles; sensor fusion; extrinsic calibration; metaheuristic algorithms; Bayesian optimization; computer vision; LiDAR

1. Introduction

The rapid advancement of autonomous vehicle (AV) technology has made precise sensor integration crucial for safe and efficient navigation. Sensor fusion relies on accurate extrinsic calibration to interpret complex environments and support real-time decisions. As AVs become more reliant on multiple sensors, such as cameras and LiDAR systems, ensuring that these devices are properly aligned is vital to achieving reliable, real-time data interpretation. In autonomous driving, precise extrinsic calibration is a prerequisite for reliable multi-sensor fusion; errors in the relative pose between LiDAR, cameras, and radar directly degrade detection, tracking, and localization performance as highlighted by several recent surveys on sensor fusion for autonomous vehicles [1,2].

Extrinsic calibration, which determines the relative position and orientation between different sensors, is key to effective sensor fusion and, by extension, to the safety and performance of autonomous systems. The operational importance of accurate calibration is also reflected in major autonomous driving benchmarks, which provide fully calibrated sensor suites (e.g., nuScenes, Waymo Open Dataset) and document detailed mounting parameters to ensure the validity of perception and fusion evaluations [3,4].

Manual and automatic target-based calibration methods differ primarily in the level of human intervention required during the calibration process. Manual approaches require engineers to explicitly establish correspondences between LiDAR point clouds and camera images using calibration targets, such as checkerboards [5–7], fiducial markers (e.g., ArUco

Academic Editors: Mahjoub Dridi, Yazan Mualla and Abdeljalil Abbas-Turki

Received: 11 December 2025
Revised: 31 December 2025
Accepted: 5 January 2026
Published: 2 February 2026

tags) [8], or custom planar targets [9]. Zhang's method [5], which remains a cornerstone in camera calibration, estimates the extrinsic transformation by observing a planar target from multiple viewpoints. Manual methods, though still a benchmark for calibration, have paved the way for increasingly automated approaches. The literature [10,11] highlights a clear trend toward developing calibration toolboxes that reduce reliance on calibration objects and human intervention. With the rise in autonomous vehicles, there is a growing need for calibration methods that are fully automatic, swift, and adaptable to various configurations, enabling faster, robust, and accurate sensor fusion. Automatic target-based calibration methods aim to reduce human intervention by automatically estimating correspondences between sensor data and calibration targets. In industrial automotive settings, offline calibration procedures are still the de facto standard, as they provide repeatable and certifiable results required by safety regulations. However, such procedures typically involve highly controlled environments, precise target placement, multiple acquisitions, and expert supervision, resulting in calibration times ranging from several tens of minutes to several hours per vehicle, depending on sensor configuration and required accuracy [12]. These constraints translate into significant operational costs, mainly driven by dedicated infrastructure and skilled personnel rather than by the calibration algorithm itself. Industry reports indicate that even minor sensor misalignments following low-speed collisions can lead to recalibration expenses of up to $5000 (USD), largely due to the need for specialized facilities and manual setup [13]. Consequently, calibration methods that relax environmental constraints while maintaining accuracy are highly desirable. The proposed approach addresses this need by increasing flexibility in target placement and acquisition geometry. Unlike conventional procedures that require close-range positioning and repeated fine adjustments, the method allows for reliable calibration with targets placed at distances of up to seven meters while robustly filtering noise and outliers. This flexibility reduces setup time and lowers dependency on rigid infrastructure, thereby enabling cost-efficient deployment in existing industrial workflows without sacrificing accuracy. Recent research has explored online and learning-based calibration techniques that estimate extrinsic parameters during vehicle operation by exploiting natural scene structure and motion cues [2,10,14]. While promising, these approaches typically require extended data collection, sufficiently rich environments, and vehicle motion. Moreover, their convergence properties and safety assurances remain challenging to certify in safety-critical workflows. For these reasons, online and learning-based calibration is commonly treated as a complementary refinement rather than a replacement for offline calibration, which remains a necessary initial step in industrial settings. More broadly, generative-AI and hybrid AI frameworks are increasingly investigated to support autonomy-related tasks such as simulation, behavior modeling, and system-level development, providing a complementary context for AV research [15,16]. Offline automatic target-based methods remain widely used in industrial and pre-deployment contexts due to their accuracy and repeatability. Representative approaches, such as the method proposed by Geiger et al. [6], leverage automatic detection of multiple checkerboards to reduce manual effort. Nevertheless, these pipelines still rely on controlled calibration environments, precise target placement, and multiple observations to achieve robust results. In practice, offline target-based calibration procedures often require dedicated infrastructure, including leveled floors, fixed target rigs, and controlled lighting conditions, and are associated with non-negligible operational costs and calibration times ranging from several minutes to over an hour when setup and validation are included [14]. Within this offline target-based category, Zhou et al. [17] introduces a single-pose calibration method that estimates the extrinsic transformation between a LiDAR sensor and a camera using geometric constraints derived from a checkerboard plane and its edges. This

approach relies on an iterative optimization process that minimizes three error terms, as outlined below:

1. Checkerboard normal-vector alignment: Ensures that the normal vector of the checkerboard in the LiDAR frame, denoted as $\mathbf{n}_L$, aligns with the normal in the camera frame, $\mathbf{n}_C$.
2. Alignment of the Checkerboard's Edges: The direction of the edges in the LiDAR frame, represented by the vector $\mathbf{d}_L$, must match the direction of the edges in the camera frame, $\mathbf{d}_C$, ensuring consistency in their representation.
3. Accurate Projection of Edge Centroids: The edge centroids must project consistently in the image plane, providing a positional constraint that ensures the checkerboard's edges are correctly positioned in the camera frame relative to the LiDAR frame.

Together, these error terms interact to refine the overall calibration process, with each term influencing the others. A discrepancy in one term, such as misalignment of the checkerboard's normal vector, can propagate and affect the accuracy of the edge alignment and centroid projection, potentially leading to cumulative errors. Therefore, minimizing all three terms simultaneously is essential for achieving high-precision calibration. The objective of this process is to determine the optimal transformation—rotation $\mathbf{R}_{CL}$ and translation $\mathbf{t}_{CL}$—that minimizes the sum of misalignments in the checkerboard's normal, edges, and centroids:

$$\min_{\mathbf{R}_{CL},\mathbf{t}_{CL}} \sum_{i=1}^{N} \left(\frac{1}{N_i} \sum_{m=1}^{N_i} \|\mathbf{n}_C \cdot (\mathbf{R}_{CL}\mathbf{p}_L + \mathbf{t}_{CL}) + \mathbf{d}_C\|^2 + \sum_{j=1}^{K_i} \sum_{k=1}^{K_{ij}} \left\|\mathbf{A}_{ij}(\mathbf{R}_{CL}\mathbf{q}_L + \mathbf{t}_{CL} - \mathbf{p}_C)\right\|^2 \right) \tag{1}$$

Specifically, $\mathbf{p}_L$ and $\mathbf{q}_L$ denote points in the LiDAR frame, while $\mathbf{p}_C$ represents corresponding points in the camera frame. The term $\mathbf{A}_{ij}$ is a projection matrix that removes components parallel to the edge direction $\mathbf{d}_C$. The variable N_i indicates the number of points on the checkerboard plane in the LiDAR frame, K_i represents the number of edges on the checkerboard in the camera frame, and K_{ij} denotes the number of points on the j-th edge in the camera frame. Although Zouh's method offers clear advantages in terms of efficiency, its reliance on a single observation of a planar calibration target introduces both strengths and weaknesses. On the one hand, this design significantly reduces data acquisition time and eliminates the need for repeated target placement or complex motion sequences. Consequently, the optimization problem involves only a small set of variables and constraints, ensuring fast convergence and low computational overhead—features that make the approach appealing in controlled calibration scenarios. On the other hand, the single-pose formulation inherently limits geometric redundancy, reducing the ability to compensate for errors. As a result, the accuracy of extrinsic parameter estimation becomes highly sensitive to plane fitting, edge extraction, and target placement. Measurement noise, partial occlusions, or uneven sampling of the checkerboard surface can propagate directly into the final solution, without the benefit of averaging across multiple observations. Verma et al. [18] explicitly analyze the impact of edge-based constraints on calibration accuracy. Their study shows that edge estimation accumulates uncertainty from both plane fitting and line fitting, whereas constraints based on the board normal depend solely on plane estimation. In addition, LiDAR beam divergence introduces systematic deviations in edge measurements, often on the order of several centimeters. To mitigate these effects, Verma et al. propose a cost function optimized via a Genetic Algorithm. While this approach improves robustness, it significantly increases computational cost and requires multiple target poses to ensure stable convergence, thereby in-

creasing acquisition time and reducing operational flexibility. More recent offline target-based approaches, such as Jeong et al. [19], focus on improving numerical stability through refined non-linear optimization. By introducing weighted residuals and iterative refinement, these methods reduce sensitivity to noise and outliers while preserving the same set of unknowns and geometric constraints as Zhou's formulation. However, the acquisition requirements remain unchanged, and in practice the reported accuracy gains are achieved through increased computational effort and the use of multiple observations. Gentilini et al. [20] address a different calibration scenario by jointly estimating extrinsic parameters across multiple sensors and multiple poses. Although this global optimization framework achieves high accuracy through redundancy, it substantially increases the dimensionality of the optimization problem and requires extended data acquisition, making it unsuitable for single-pose or rapid calibration scenarios. From an operational perspective, three quantitative factors are critical in practical calibration procedures: the number of required poses, which directly affects acquisition time; the dimensionality and complexity of the optimization problem, which influence convergence behavior and computational cost; and flexibility with respect to target placement and environmental conditions. Existing offline target-based methods primarily improve robustness by increasing redundancy, either through additional poses or higher computational effort. In contrast, the method proposed in this work operates under the same single-pose acquisition assumption as [17], while reducing sensitivity to noise and target placement constraints without increasing the number of poses or the dimensionality of the optimization problem. For this reason, Zhou's method represents the most appropriate baseline for quantitative comparison, as it enables a direct evaluation of robustness improvements under identical acquisition and operational conditions.

In the literature, offline target-based LiDAR–camera calibration is typically addressed using conventional numerical optimization methods. Ref. [17] formulate the single-pose calibration as a non-linear least squares problem solved iteratively with the Levenberg–Marquardt algorithm. Ref. [19] extend this approach through weighted residuals and iterative refinement, while ref. [20] employ a Gauss–Newton scheme for multi-pose, multi-sensor calibration, leveraging redundancy across observations to improve stability. These gradient-based strategies perform well for smooth, differentiable cost functions but become less effective when the optimization landscape exhibits strong nonlinearity, multimodality, or non-differentiable components. The calibration problem considered in this work involves a custom cost function with precisely these characteristics, making conventional numerical methods inadequate. To overcome this challenge, metaheuristic optimization algorithms are investigated. Metaheuristics have proven successful in autonomous driving tasks such as nonlinear MPC-based motion planning and obstacle avoidance, where Firefly algorithms achieved efficient convergence under non-convex constraints [21]. Although the focus here is extrinsic calibration, the underlying optimization landscape shares similar properties—strong nonlinearity, multimodality, and lack of analytical gradients—justifying the adoption of metaheuristic techniques. Despite extensive research on LiDAR–camera calibration, robust and rapid procedures remain essential for in-vehicle deployment, where frequent recalibration and minimal human intervention are required; recent frameworks emphasize this need in real autonomous platforms [22]. To address these requirements, three widely used metaheuristic algorithms—Simulated Annealing (SA), Genetic Algorithms (GA), and Particle Swarm Optimization (PSO)—are evaluated and compared against traditional numerical methods. Unlike gradient-based approaches, which are prone to local minima in highly non-linear and multimodal spaces, metaheuristics employ global search strategies that balance exploration and exploitation, enabling efficient approximation of optimal solutions [23]. The comparison of different optimization algorithms such as SA, GA, and PSO was motivated by the "No Free Lunch" theorem [24], which

states that no single algorithm universally outperforms all others across every problem. Consequently, the most effective algorithm for this calibration task was identified through experimental evaluation. According to literature guidelines, GA is typically well suited for problems requiring extensive global exploration, PSO is effective for rapid convergence in smoother search spaces, and SA performs well for stochastic local search. However, these tendencies are general observations and do not guarantee superior performance for every specific problem. To exploit the complementary strengths of these algorithms, reduce cross-compensation effects, and simplify the optimization problem, the procedure was divided into two independent 3D phases—one for rotation and one for translation. Each phase was solved using the algorithm shown to perform best in practice, with the rotation matrix represented via Euler angles to both reduce the number of optimization variables and maintain the orthogonality of the matrix during the optimization process. To maximize the performance of the selected metaheuristic algorithms, Bayesian optimization was employed to define the search space and tune algorithm parameters offline. This approach automatically selects ideal ranges for relative yaw, pitch, roll, and translation between sensors, enhancing both optimization efficiency and adaptability to various sensor configurations.Furthermore, computer vision tools were employed to automate the selection of geometric features in both camera and LiDAR frames using checkerboard dimensions as the sole reference, while an innovative strategy incorporating a Huber estimator was devised to filter out errors and outliers. This approach ensured that the optimization process focused on reliable data, minimized noise impact, and improved overall precision, even when calibration relies on a single pose.

In Section 2, we present the methodology of our framework, detailing the implementation of the cost functions. Section 3 details the architecture of the proposed method, emphasizing Bayesian optimization for process automation. Section 4 outlines the experimental setup, followed by Section 5, which presents the results. Finally, Section 6 discusses conclusions and future directions.

2. Fundamentals of Metaheuristic Algorithms

The term metaheuristic stems from the Greek words “meta” (beyond) and “heuristic” (to find) [25]. According to Laporte and Osman [23], metaheuristics comprise iterative methods that guide subordinate heuristics by combining diverse strategies to explore and exploit the search space, using learning mechanisms to approach near-optimal solutions efficiently. While no universal definition exists, metaheuristics generally refer to adaptable algorithms that overcome the limitations of traditional optimization methods—particularly for nonlinear, multimodal, or high-dimensional problems where gradient-based or direct search approaches often fail. They rely on diversification (global search) and intensification (local exploitation) [26]. Widely applied across domains like NP-hard problem solving, robotics, and image processing [27], metaheuristics are typically classified by their source of inspiration: evolutionary algorithms, swarm intelligence, physical-law-based methods, and miscellaneous strategies [28,29]. Building on this foundation, the metaheuristic algorithms employed in this study—each representing a different category—are described in detail below:

- Genetic algorithm: Genetic Algorithms (GAs), developed by John Holland in the 1960s, emulate the process of natural adaptation through computational means [30]. Inspired by biological evolution, GAs operate on a population of candidate solutions called “chromosomes,” each composed of “genes” representing variables or traits that can take on different values, known as “alleles.” The fitness of a chromosome, evaluated by a fitness function to be maximized, determines its probability of being selected for reproduction.

The algorithm follows a cycle of steps: it begins with the initialization of a population of n_{pop} individuals encoded as bit strings, balancing randomness, compositionality, and generality to ensure a diverse search space. Next, each chromosome's fitness is evaluated [31]. Then, offspring are generated by applying genetic operators—selection, crossover, and mutation—to produce m new individuals. The parent and offspring populations are merged, evaluated, and sorted, and the best individuals are selected to form the next generation. This process iterates for a predefined number of generations, typically ranging from 50 to 500. The overall workflow of the Genetic Algorithm is illustrated in Figure 1.

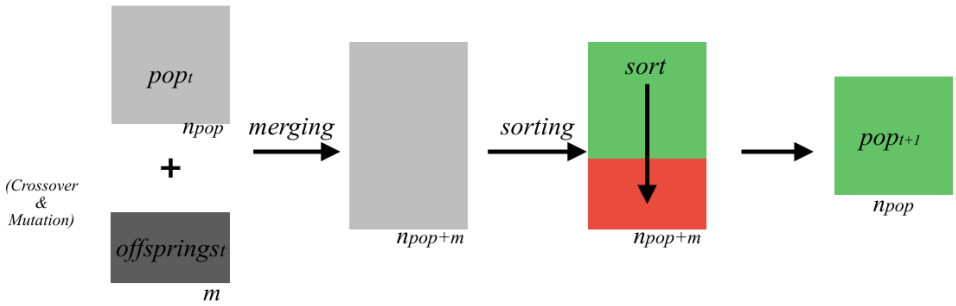


Figure 1. Genetic Algorithm workflow.

Because GAs are stochastic, repeated runs with the same parameters can yield different results. Thus, results are usually averaged over multiple runs to provide robust performance metrics. In the automotive domain, Genetic Algorithms have also been successfully applied to nonlinear Model Predictive Control (MPC) trajectory planning for autonomous driving, demonstrating high convergence reliability under complex vehicle-dynamics constraints [32]. Such applications confirm the suitability of GAs for non-convex optimization problems similar to those addressed in this work. The three key genetic operators balance exploration and exploitation within the search:

- Selection favors fitter individuals to propagate their traits. A common method is roulette wheel selection, where the probability of selecting chromosome i is proportional to its fitness:

$$P(i) = \frac{f_i}{\sum_{j=1}^{n_{pop}} f_j} \tag{2}$$

  Here, f_i is the fitness of chromosome i and n_{pop} is the population size. Although simple, this method may cause premature convergence by quickly favoring highly fit individuals. Alternatives such as tournament selection—choosing the best from random subsets—and ranking selection—assigning selection probabilities based on relative fitness ranks—help maintain diversity and prevent dominance [33].
- Crossover mimics biological recombination by exchanging gene segments between two parents. Techniques like one-point, two-point, or uniform crossover combine traits to create potentially superior offspring, enhancing the search for optimal solutions [34].
- Mutation introduces random changes in gene alleles, injecting novel genetic material to prevent premature convergence and maintain population diversity. Mutation promotes exploration, complementing the exploitative nature of crossover.

Convergence is typically defined when 95% of the population shares the same allele at each gene locus, indicating a high degree of genetic uniformity. This balance

of operators allows GAs to effectively navigate complex search spaces, evolving populations toward fitter solutions over successive generations.

- Particle Swarm Optimization: Particle Swarm Optimization (PSO), introduced by Kennedy and Eberhart in 1995, is inspired by the social behavior of bird flocks and is designed to optimize continuous nonlinear functions [35]. The algorithm employs a swarm of particles, where each particle represents a candidate solution characterized by its position and velocity within the search space. Over successive iterations, particles adjust their trajectories by leveraging both their individual experience and the collective knowledge of the swarm, ultimately converging toward optimal solutions. Starting from randomly initialized positions and velocities, the velocity and position of each particle k at iteration $i+1$ are updated as follows:

$$\begin{aligned} V_k(i+1) = w \cdot V_k(i) \\ + c_1 \cdot r_1 \cdot (pbest_k - X_k(i)) \\ + c_2 \cdot r_2 \cdot (gbest - X_k(i)) \end{aligned} \tag{3}$$

$$X_k(i+1) = X_k(i) + V_k(i+1) \tag{4}$$

The velocity update equation consists of three components: the momentum term $w \cdot V_k(i)$, which helps maintain the particle's current direction and balances exploration and exploitation; the cognitive term $c_1 \cdot r_1 \cdot (pbest_k - X_k(i))$, which draws the particle toward its personal best position to encourage individual learning; and the social term $c_2 \cdot r_2 \cdot (gbest - X_k(i))$, which attracts the particle toward the global best position discovered by the swarm, promoting collective learning.

Here, the parameters are defined as follows: w is the inertia weight that controls the influence of the particle's previous velocity; c_1 and c_2 are acceleration coefficients that weigh the relative contributions of personal and social experiences; r_1 and r_2 are random scalars uniformly sampled from the interval $[0, 1]$ at each iteration, introducing stochastic variability to avoid premature convergence; $V_k(i)$ and $X_k(i)$ represent the current velocity and position of particle k, respectively; $pbest_k$ is the best position found so far by particle k; and $gbest$ is the best position found by the entire swarm.

The effectiveness of PSO is governed by key parameters that balance exploration and exploitation. The inertia weight w influences how strongly a particle relies on its previous velocity, with larger values encouraging broader exploration and smaller values focusing the search around promising areas. The acceleration coefficients c_1 and c_2 control the relative importance of individual learning versus social learning within the swarm. Additionally, although not explicitly included in the standard velocity update equations, a maximum velocity parameter $V_{\max}$ is often introduced in some implementations to cap the particle velocity, preventing excessive jumps in the search space and facilitating smoother convergence.

By iteratively updating particles' velocities and positions in accordance with these principles, PSO efficiently balances the trade-off between global exploration and local exploitation, thereby converging toward near-optimal solutions. Moreover, PSO and its accelerated variants have been integrated into nonlinear MPC frameworks for real-time motion planning by the authors, achieving both computational efficiency and solution robustness [36]. This evidence further supports the adoption of PSO in the present study as a reliable global-search optimizer for highly nonlinear calibration problems.

- Simulated Annealing: Simulated Annealing (SA), introduced by Kirkpatrick et al. [37], is an optimization algorithm inspired by the metallurgical process of annealing, where a crystalline solid is heated to a high-energy state and then slowly cooled to reach a

stable, low-energy configuration with minimal defects. Analogously, SA starts from a random solution and gradually lowers a control parameter called temperature T, exploring neighboring solutions at each step. Unlike greedy methods, SA probabilistically accepts worse solutions early on, enabling escape from local minima and a more thorough search of the solution space. As T decreases, the algorithm becomes more selective, converging to an optimal or near-optimal solution.

The analogy between physical annealing and simulated annealing is summarized as follows:

Physical Annealing	Simulated Annealing
System States	Solutions
Energy	Cost
Perturbed State	Neighboring Solutions
Temperature	Control Parameter
Final State	Heuristic Solution

The SA algorithm proceeds through these steps:

1. Initialization: Start with a random initial solution S_0 and initial temperature T_0.
2. Main Loop: While the stopping condition is unmet:
   - Generate a neighboring solution $S^{'}$ by perturbing the current solution S and compute its cost $E(S^{'})$.
   - Calculate the energy difference $\Delta E = E(S^{'}) - E(S)$.
     - If $\Delta E \leq 0$, accept $S^{'}$ as the new solution.
     - If $\Delta E > 0$, accept $S^{'}$ with probability $e^{-\Delta E/T}$, allowing occasional uphill moves to avoid local minima.
   - Update the temperature according to the cooling schedule, typically exponentially:

$$T_i = \alpha T_{i-1} \tag{5}$$

where $\alpha \in (0, 1)$ controls the cooling rate [38].

The overall workflow of the SA Algorithm is illustrated in Figure 2.

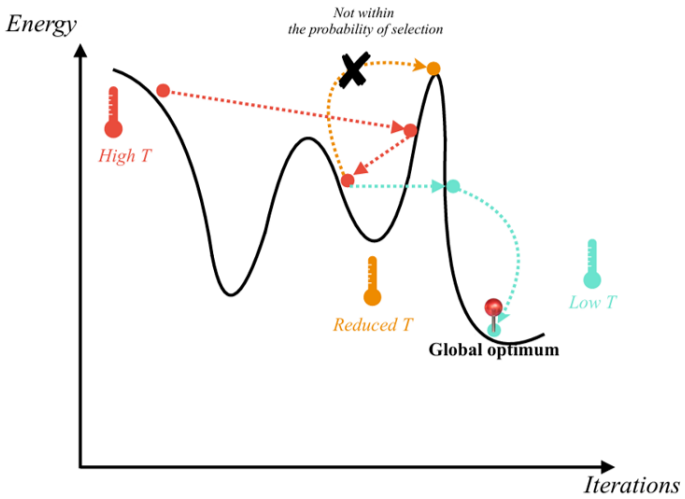


Figure 2. SA Algorithm workflow.

This gradual cooling and probabilistic acceptance strategy enables SA to balance exploration and exploitation effectively, making it a powerful metaheuristic for global optimization.

3. Methodology

In this section, the methodology and the main steps of the extrinsic calibration process are presented. The sensor architecture consists of three main devices: an OS1 LiDAR (Ouster, Inc., San Francisco, CA, USA), a Ladybug5+ omnidirectional camera (Teledyne FLIR Integrated Imaging Solutions, Inc., Richmond, BC, Canada), and a ZED X stereo camera (Stereolabs, Inc., San Francisco, CA, USA), as summarized in Table 1. These sensors include both active (LiDAR) and passive (cameras) devices, whose combination enables sensor fusion for a more complete and reliable perception of the environment [39].

Table 1. Sensor setup used for the extrinsic calibration.

Sensor Type	Sensor Name
LiDAR	 OS1 LiDAR
Omnidirectional camera	 Ladybug5+
Stereo camera	 ZED X

The procedure begins with an initial recording of approximately one minute to analyze and minimize measurement noise. The data from all sensors are recorded and synchronized. A checkerboard calibration target with a 9×7 grid of 0.108 m squares is employed to establish correspondences across the modalities.

Specifically, rather than relying on traditional board edges, which can introduce fitting errors, the focus is on the checkerboard's center and normal, thereby reducing inaccuracies. RANSAC and DBSCAN clustering are used to isolate the checkerboard plane from the LiDAR data to accurately detect the chosen features. An approach using the Huber estimator is implemented to refine the results by prioritizing points that closely match the expected dimensions of the checkerboard. This ensures more reliable coordinates even in the presence of noise.

The procedure calculates the width and height of the checkerboard for each acquisition i by measuring the distances between corner points. The total deviation is then computed as:

$$\text{total\_deviation}_i = \frac{|\text{width}_i - W| + |\text{height}_i - H|}{W + H} \tag{6}$$

with H and W being the known dimensions of the board. This deviation is used to assign weights to the acquisitions, which are then combined in the weighted estimator:

$$\hat{X} = \frac{\sum w_i X_i}{\sum w_i} \tag{7}$$

where X_i represents the data from each acquisition, and w_i is the weight assigned to that acquisition.

For the camera data, the coordinates of the four corners are derived from the checkerboard's dimensions and expressed in the standard reference frame at the upper-left corner in the pixel frame. These points, along with the center (calculated as the average of the detected corners), are then transformed into the world coordinate system. The normal vector is obtained by transforming a point along the checkerboard's z-axis into the camera frame. Despite some intrinsic noise, high resolution minimizes errors, and the Huber estimator refines the coordinates, ensuring reliable calibration.

At the end of this process, the features are represented as 3xN_{poses} matrices, where the rows represent the x, y, z coordinates (Figure 3) while the columns represent the number of poses:

$$\text{Lidar frame: } C_{L.j},\ j = 1 \dots 4,\ C_L^*,\ N_l$$
$$\text{Camera frame: } C_{C,j},\ j = 1 \dots 4,\ C_C^*,\ N_c$$

with C_j denoting the corners, C^* the center of the board, and **N** the normals. For reference, the corners and edges of the board expressed in the LiDAR reference frame are shown in Figure 3.

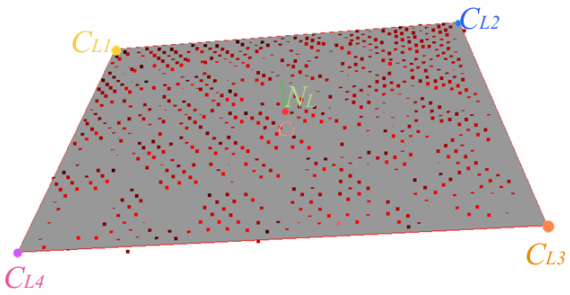


Figure 3. Geometrical features extracted from the LiDAR reference frame.

At this point, the extrinsic calibration problem is redefined as a streamlined optimization task that directly minimizes alignment errors between sensor frames, diverging from traditional methods that rely on point cloud-to-pixel correspondences and varied feature sets. By structuring the problem in this way, straightforward optimization algorithms can target solutions directly, bypassing the need for preliminary approximations or suboptimal results requiring later refinement with non-linear methods, thus achieving both efficiency and precision in calibration. Further optimization is achieved by representing rotations using Euler angles rather than a full 3 × 3 matrix. This reduces the number of optimization variables from nine to three and inherently preserves the orthogonality of the rotation matrix throughout the optimization process. Directly optimizing the individual matrix elements would require enforcing orthogonality at each iteration, as independent updates by the optimization algorithms could otherwise produce invalid transformations. By using Euler angles, the optimization is simplified while ensuring that all resulting transformations remain valid homogeneous transformations. The overall formulation is guided by geometric observability and numerical stability considerations rather than empirical tuning. In particular, the decomposition of the optimization into sequential stages and the selection of complementary geometric error terms are designed to reduce parameter coupling and improve convergence robustness. Detailed definitions of each cost component and their role in the optimization are provided in the following subsections. This approach leads to a multi-objective cost function that integrates both rotational and translational discrepancies,

with each component balanced by experimentally determined weights to ensure accurate alignment. To avoid cross-compensation between translation and rotation, calibration is divided into two phases and implemented in MATLAB as follows.

Phase 1: The first phase aims to align the checkerboards within the camera's reference frame by minimizing three angular errors, producing an initial guess for the rotation that maps the LiDAR points first to the ZED X frame and then to the Ladybug5+ frame.

- Normal Error Alignment: The first error is the alignment between normal vectors. Given the normal matrices $\vec{N}_l$ and $\vec{N}_c$ ($3 \times N_{\text{extracted\_point}}$), and the Euler angles, a rotation matrix R is derived. For each point i, the normal vectors in Lidar frame $\vec{n}_{l_i}$ and camera frame $\vec{n}_{c_i}$ are computed as follows:

$$\vec{n}_{l_i} = \frac{\vec{N}_l[:, i]}{\|\vec{N}_l[:, i]\|}, \quad \vec{n}_{c_i} = \frac{\vec{N}_c[:, i]}{\|\vec{N}_c[:, i]\|} \tag{8}$$

Applying the rotation to the normal vector in the Lidar frame:

$$\vec{n}'_{l_i} = R \cdot \vec{n}_{l_i} \tag{9}$$

The alignment error is computed using the dot product, clamped between -1 and 1 to prevent numerical errors:

$$\text{dot\_product} = \vec{n}_{c_i} \cdot \vec{n}'_{l_i} = \cos(\theta) \tag{10}$$

The angular error θ (Figure 4) is computed as:

$$\theta = \arccos(\vec{n}_{c_i} \cdot (R \cdot \vec{n}_{l_i})) \tag{11}$$

Afterwards, Huber loss is applied to handle variations in reflections, distances, and other uncertainties affecting checkerboard poses.

Figure 4. Illustration of the angular error between the checkerboard normal vectors. The black and white checkerboard represents the plane observed in the camera frame, while the orange checkerboard corresponds to the same plane represented in the LiDAR frame. The red arrows denote the respective normal vectors in each coordinate frame, and the angular discrepancy between them defines the alignment (normal-to-normal) error.

This choice prioritizes reliable data while mitigating measurement errors, ensuring a robust cost function that improves algorithm convergence:

$$\text{huber\_err} = \begin{cases} \frac{1}{2} \cdot \text{angle\_err}^2 & \text{if } |\text{angle\_err}| \leq \delta \\ \delta \cdot (|\text{angle\_err}| - 0.5 \cdot \delta) & \text{otherwise} \end{cases} \tag{12}$$

where $\delta = 0.05$ rad based on the observed range of angular deviations in the calibration data. Finally, the Huber error for each normal vector pair is accumulated across all points. The final output, e_n, is the average Huber error across all N points:

$$e_n = \frac{\sum_{i=1}^{N_{points}} \text{normal\_huber\_error}_i}{N_{points}} \tag{13}$$

- Directional Alignment Error: The second error reflects the degree to which the diagonals in each frame are misaligned (Figure 5), following the same logic as the previous one, with the only difference being that the calculations are now performed on the vectors pointing from the center to each corner:

$$\begin{aligned} \vec{D}_{i,j,L} &= C_{i,j,L} - C^*_{i,L}, \\ \vec{D}_{i,j,C} &= C_{i,j,C} - C^*_{i,C}, \end{aligned} \qquad j = 1, \ldots, 4,\ i = 1, \ldots, N_{points} \tag{14}$$

Applying the rotation to the normalized vectors:

$$\vec{D}'_{i,j,l} = R \cdot \vec{D}_{i,j,l}, \qquad j = 1, \ldots, 4,\ i = 1, \ldots, N_{points}, \tag{15}$$

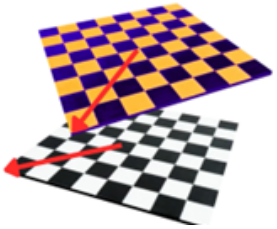


Figure 5. Illustration of the diagonal-direction angular error. The black and white checkerboard represents the plane in the camera frame, and the orange checkerboard represents the same plane in the LiDAR frame. The red arrows indicate the center-to-corner directional vectors defined for each checkerboard. The angular deviation between corresponding vectors quantifies the diagonal-alignment error.

At this point, the function performs the same calculations shown in Equations (10) and (12), applying a threshold of 0.05 rad for consistency, based on the same rationale outlined for the normal vectors. This yields the final directional alignment error:

$$e_{D,j} = \frac{\sum_{i=1}^{N_{points}} \text{direction\_huber\_error}_i}{N_{points}} (j = 1, \ldots, 4) \tag{16}$$

Summing over all corners:

$$e_{D,\text{tot}} = \sum_{j=1}^{4} e_{D,j} \tag{17}$$

- Perpendicularity Error: The third error ensures perpendicularity between the transformed normal vector and the corner directional vectors (Figure 6), computed via:

$$e_{P,\text{tot}} = \sum_{j=1}^{4} e_{P,j} \tag{18}$$

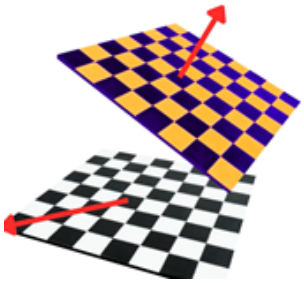


Figure 6. Illustration of the perpendicularity error. The orange checkerboard represents the plane in the LiDAR frame, with its associated normal vector shown in red. The black and white checkerboard corresponds to the plane in the camera frame, where the red arrow denotes a center-to-corner directional vector. The perpendicularity error is defined as the angular deviation from the expected orthogonality between the transformed normal vector and the corresponding directional vector.

To ensure this perpendicularity, the function calculates the arcsine of the cross product rather than using the arccosine, allowing it to capture small deviations effectively. Finally, the first objective function is formulated with weights:

$$w_n = 0.5, \quad w_d = 0.25, \quad w_p = 0.25 \tag{19}$$

Minimizing:

$$\arg \min_{\text{Euler angles}} J = w_n \cdot e_n + w_d \cdot e_{D,tot} + w_p \cdot e_{P,tot} \tag{20}$$

The weights were selected based on geometric robustness considerations, prioritizing normal vector alignment due to its higher stability compared to corner-based features. These parameters were fixed once and applied consistently across all experiments, and the optimization was observed to be stable under moderate variations of their values.

Phase 2: The second phase aims to determine the rigid transformation that precisely aligns the LiDAR point clouds with those in the camera frame, ensuring accurate calibration. To enhance the initial rotation estimate, both directional e_d and normal errors e_n are reintroduced to add redundancy, considering the variability of real-world data.

- Centers Alignment Error: A new function is implemented to refine the rigid transformation by ensuring precise alignment of centers and corners (Figure 7a,b).
  This function retrieves the center points in both frames across multiple poses and transforms the LiDAR centers into the Camera frame using the estimated rotation and translation:
  $$C_{i,L}^{*'} = R \cdot C_{i,L}^{*} + t \qquad i = 1, \ldots, N_{points} \tag{21}$$
  The alignment error is then quantified through the Euclidean distance:
  $$\text{error\_value}_i = \| C_{i,C}^{*} - C_{i,L}^{*'} \|, \qquad i = 1, \ldots, N_{points} \tag{22}$$
  To mitigate the influence of outliers, the Huber loss function is applied with a selected threshold $\delta = 0.0074$ m following Equation (12).
  The final error metric is computed as the average Huber loss across all poses:
  $$e_f = \frac{\sum_{i=1}^{N_{points}} \text{centers\_huber\_error}_i}{N_{points}} \tag{23}$$

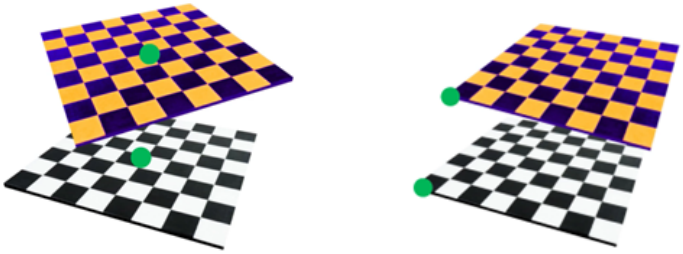
This process is systematically repeated for all corner points, producing e_{corners} as the alignment error for the corners, and resulting in an objective function that combines multiple weighted error components:

$$\arg\min_{\text{eul},t} J = w_n \cdot e_n + w_d \cdot e_d + w_{fc} \cdot e_f + w_f \cdot e_{\text{corners}} \tag{24}$$

where the weights are empirically determined:

$$w_n = 0.4, \quad w_{fc} = 0.3, \quad w_d = 0.05, \quad w_f = 0.25$$

The weights are experimentally selected, guided not only by the sources of error discussed previously but also by the amplification of the angular errors over distance. For example, when a rotational misalignment occurs, the perceived positional error of an object increases as the distance grows. This understanding also leads to the decision to divide the calibration problem into two stages, with a stronger emphasis on rotation in the initial phase to ensure precise angular alignment across multiple poses.



(**a**) Center alignment error (**b**) Corner alignment error

Figure 7. Visualization of the alignment errors used to refine the rigid transformation between LiDAR and Camera frames. The black and white checkerboard is represented in the Camera frame, the orange checkerboard in the LiDAR frame, and green points indicate the detected features. (**a**) shows the Euclidean distances between corresponding centers, defining the center alignment error. (**b**) shows the Euclidean distances between corresponding corners, defining the corner alignment error.

4. Bayesian Optimization Approach

To obtain an initial estimate of the rotation matrix, the normal vector was used due to its robustness, being influenced primarily by the plane-fitting error from RANSAC in the LiDAR frame:

$$R = \left((N_L N_L^\top)^{-1} \right) \cdot \left(N_L \cdot N^\top \right)^\top \tag{25}$$

where N_L and N represent the normal vectors in the LiDAR and camera frames. This matrix was then converted into Euler angles, providing a structured starting point for the optimization process. The next step consisted of defining the search space within which the optimal rotation matrix could be located. This required establishing a sufficiently good lower and upper bound around the initial estimate, taking into account both the relative orientation of the sensors and potential mounting errors, which are inherently unknown.

Manually exploring this search space through trial and error would have been time-consuming, prone to inconsistencies and contrary to the goal of an automated calibration process. To address this challenge, Bayesian optimization was employed to systematically identify optimal bounds for each algorithm (GA, SA, PSO), effectively defining the search space required to minimize the cost function across all approaches.

Bayesian optimization has proven to be a powerful technique for complex, expensive optimization problems, particularly when the objective function is nonlinear, non-convex, and difficult to model analytically. By using a Gaussian Process (GP) to probabilistically

model the objective function and iteratively refining estimates through an acquisition function such as Expected Improvement (EI), the method balances exploration and exploitation, enabling an efficient search without requiring explicit derivatives. This approach has been widely adopted in machine learning, especially for hyperparameter tuning in high-dimensional spaces, where traditional methods like grid search or random search are inefficient.

The potential of Bayesian optimization extends beyond machine learning to metaheuristic algorithms, which also rely on careful parameter tuning for optimal performance. Similar to neural networks, the relationship between parameters and outcomes in metaheuristics is often complex and problem-dependent. Bayesian optimization overcomes the need for an explicit model by using sample evaluations of the objective function, making it particularly valuable for sensor calibration, where the search space is initially undefined and optimal solutions are not readily apparent.

Although only a few studies, such as [40], have explored the combination of Bayesian optimization with metaheuristics, the results demonstrate its effectiveness in improving algorithm performance through optimal parameter selection. This approach was successfully applied to the Hybrid Kernel EDA algorithm, and future work could extend its application to other metaheuristics.

By integrating Bayesian optimization, the search boundaries are tuned offline in a one-time configuration stage, prior to the actual calibration. Since the extrinsic calibration is static, this procedure is not executed at each calibration run but only when the sensor setup changes, and the resulting bounds are reused thereafter, significantly reducing the runtime of the subsequent optimization process. For this study, a broad initial range of $[[-200^\circ, 0^\circ],\ [0^\circ, 200^\circ]]$ was chosen, enabling the optimizer to iteratively refine the bounds and efficiently guide the algorithms toward the optimal solution.

For each candidate configuration, GA, SA, and PSO were executed 20 times, and the resulting cost values and execution times were averaged and normalized, yielding AvgCost and AvgTime. The optimization objective was defined as:

$$\text{obj} = 0.5 \cdot \frac{\text{AvgCost} - \text{Cost}_{\text{min}}}{\text{Cost}_{\text{max}} - \text{Cost}_{\text{min}}} + 0.5 \cdot \frac{\text{AvgTime} - \text{Time}_{\text{min}}}{\text{Time}_{\text{max}} - \text{Time}_{\text{min}}} \tag{26}$$

ensuring a trade-off between accuracy and efficiency.

The comparative analysis showed that all three algorithms achieved similar accuracy within the defined bounds, but their computational profiles differed significantly. SA offered stable performance but required substantially longer execution times. PSO performed competitively in terms of accuracy, but its sensitivity to parameter choices made it less consistent. GA, instead, consistently combined low average cost with the shortest execution times, confirming its suitability for the first calibration phase. For this reason, only GA results are shown in the following figures, as it was ultimately selected for the subsequent parameter-tuning stage (see Figures 8–10). Table 2 summarizes the performance of all three algorithms after Bayesian optimization. From the table, it is evident that GA (roulette selection) achieved the lowest average cost and shortest average execution time, confirming the choice of GA for the subsequent parameter tuning stage. The table also highlights that although SA was the most robust to parameter changes, its execution time was a limiting factor, and PSO required careful bound selection to achieve similar performance.

Table 2. Performance of GA, PSO, and SA after Bayesian optimization for phase one (one pose). GA was selected as it offered the best trade-off between Avg. Cost and Avg. Time. Only GA parameter trends are reported in the following figures.

Algorithm	Avg. Time (s)	Avg. Cost	Best Cost
GA (Roulette Selection)	0.064	0.0033	0.0024
GA (Tournament Selection)	0.080	0.0035	0.0024
PSO	0.083	0.0035	0.0024
SA	1.1	0.0044	0.0024

Detailed analysis of GA parameters showed that higher crossover percentage (Pc) reduced cost but increased execution time, while mutation percentage (Pm) introduced cost fluctuations and slightly increased execution time. The crossover range factor (gamma) maintained stable times until 0.5, then spiked, and the mutation rate (mu) minimized cost at 0.2 while increasing execution time at 0.15.

After completing the robustness analysis in phase one, the optimized rotation matrix R^* was exported as the starting point for phase two. This matrix provided a reliable initial estimate, which was then used both to refine R^* and to define the translation vector:

$$t_{lc} = \text{mean}(C_C^* - R^* \cdot C_L^*) \tag{27}$$

where C_C^* and C_L^* represent the corresponding calibration points in the camera and LiDAR frames.

The search space for phase two was established around these initial estimates, with lower and upper bounds adjusted to account for sensor orientation and potential mounting errors. Bayesian optimization was again employed to refine the bounds and guide parameter selection. All three algorithms—GA, SA, and PSO—were evaluated to assess cost and execution time. However, only PSO results are presented in the figures, as it provided the most favorable compromise between accuracy and computational efficiency.

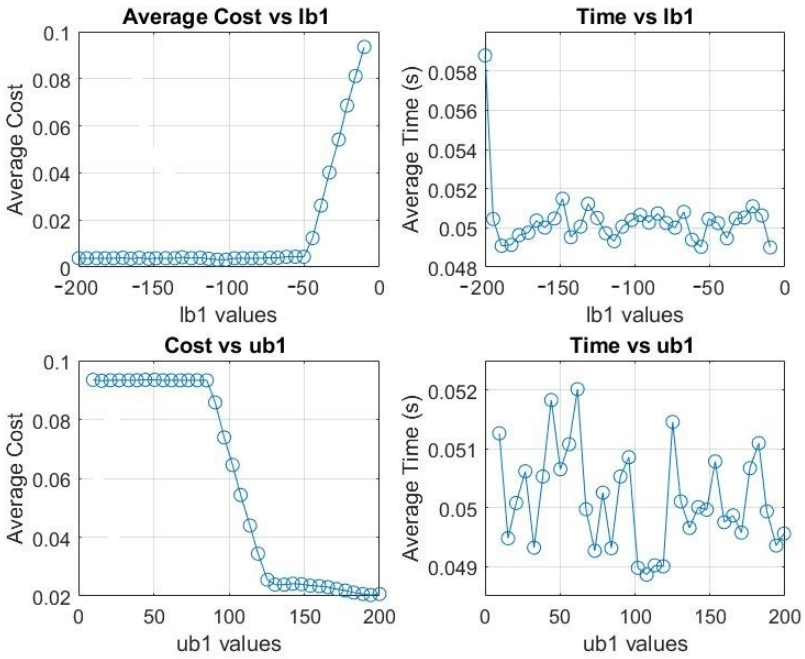


Figure 8. GA fitness and execution time vs. lower and upper bounds expressed in degrees.

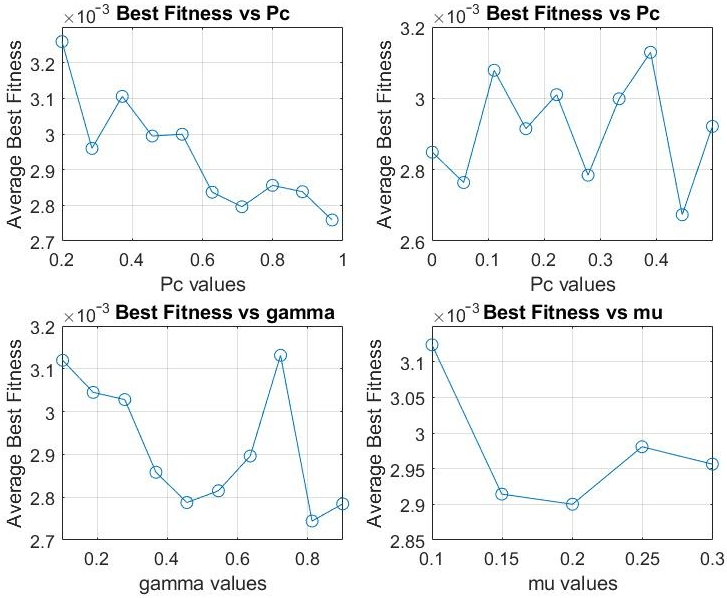


Figure 9. GAfitness trend with respect to parameters during phase one.

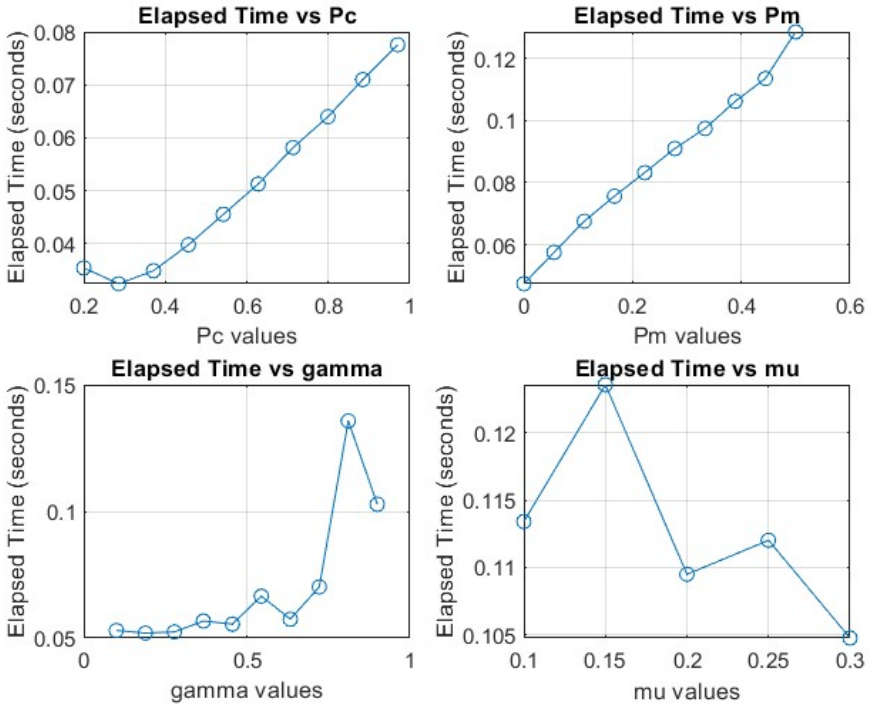


Figure 10. GA execution time with respect to parameters during phase one.

Cost analysis revealed that SA generally maintained lower values (around 4.4×10^{-4}) across the search space, while PSO achieved comparable results, with slightly higher peaks at the bounds. GA, in contrast, reached costs up to 10^{-3}. Execution time considerations highlighted that SA required up to 30 s, whereas PSO and GA completed the computations up to 100 times faster (See Table 3).

Following parameter tuning, PSO consistently produced solutions close to the minimum within reasonable time frames. Table 3 reports the numerical results, while Figures 11–13 illustrate fitness and execution time trends for PSO within the refined search space.

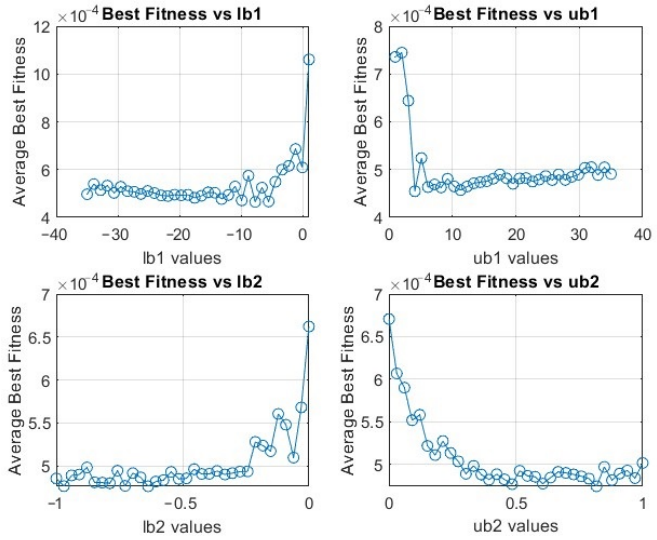


Figure 11. Trend of the average cost with respect to the bounds during phase two: ub_1, lb_1 in degrees, and ub_2, lb_2 in meters.

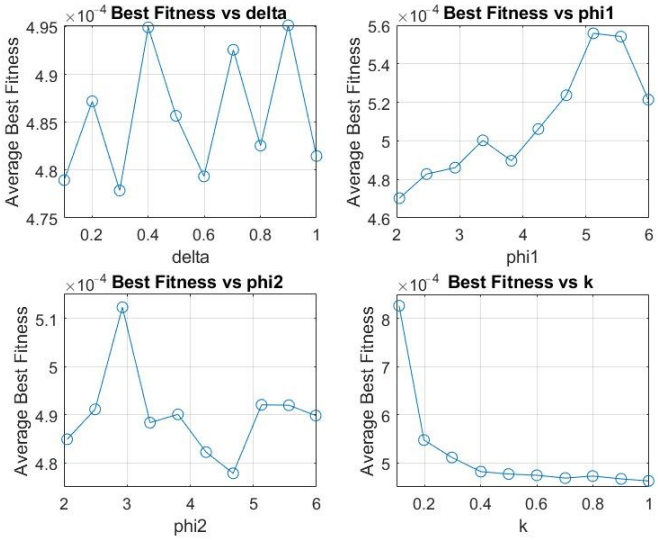


Figure 12. Cost trend of PSO with respect to parameters for phase two.

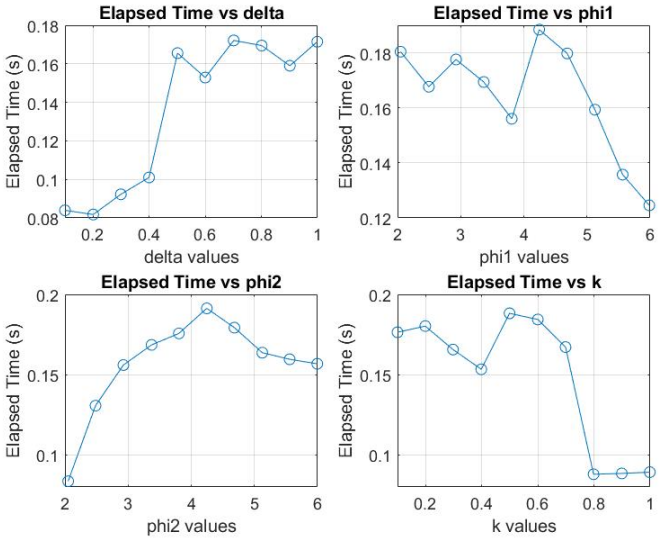


Figure 13. Average execution time trend of PSO with respect to parameters for phase two.

Table 3. Values obtained after Bayesian optimization for phase two, with only one pose.

Algorithm	Avg. Time (s)	Avg. Cost	Best Cost
GA (Roulette Wheel Selection)	0.0835	0.00065	0.00044
GA (Tournament Selection)	0.1	0.00065	0.00044
PSO	0.22	0.00051	0.00044
SA	9.03	0.00047	0.00044

As a result of the experimental evaluation guided by the No Free Lunch theorem, the optimization process adopts a hybrid approach, using a Genetic Algorithm (GA) for Phase 1 and Particle Swarm Optimization (PSO) for Phase 2. During Phase 1 (GA) of the hybrid optimization process, an adaptive mutation function is employed, as summarized in Algorithm 1. Instead of using a standard fixed mutation, the mutation intensity σ is dynamically adjusted based on the parent's fitness:

$$\delta = 1 - \frac{\text{parent.Fitness}}{\min(\text{pop.Fitness})}, \quad \sigma = \delta \cdot (\text{VarMax} - \text{VarMin}) \tag{28}$$

Selected genes of the parent are mutated using σ, enabling adaptive exploration of the search space while keeping values within the predefined bounds, VarMin and VarMax. This strategy improves convergence, reduces the number of tunable parameters, and prevents premature convergence during the GA phase. Additionally, 50% of the first-phase GA population is carried over to initialize Phase 2 (PSO), providing a heuristic that leverages promising solutions. Overall, this approach balances exploration and exploitation, enhances search efficiency, and ensures the discovery of high-quality solutions, as illustrated in the pseudocode below.

Algorithm 1 Hybrid GA-PSO Optimization

Def. $CostFunction_{GA}, CostFunction_{PSO}$
Init. GA $(n_{pop}, max\_gen, P_c, P_m, bounds)$
GA Phase1: Init. Pop $\in$ bounds
for $gen = 1 \rightarrow max\_gen$ **do**
    Eval fitness
    Selection $\rightarrow$ Crossover $\rightarrow$ AdaptiveMutation
    Update Pop
end for
Output: $eul_{first\_guess}, PopGA$
PSO Phase2: Def. bounds for eul, t
Init. Swarm: 50% PopGA, 50% rand
for $iter = 1 \rightarrow max\_iter$ **do**
    for each particle **do**
        Eval fitness
        Update velocity, position $\in$ bounds
        Update local, global best
    end for
end for
Return: eul_{best}, t_{best}

5. Experimental Setup

The design of the new platform benefits from the experience gained in our previous prototypical autonomous-vehicle testbed [41], where the overall system architecture, interfacing strategies and validation methodology were established on a full-scale electric vehicle. In the present work, a different hardware and computing architecture is adopted,

tailored to the specific multi-sensor configuration and calibration objectives described in Section 3.

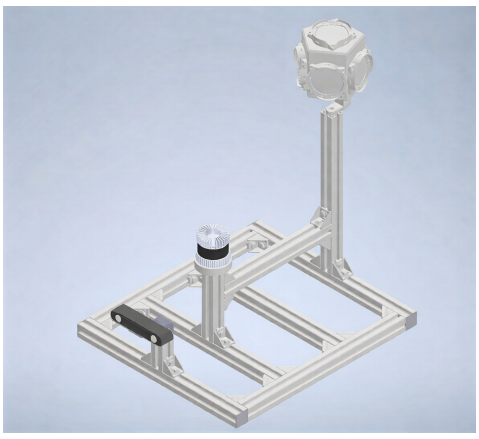
The sensor system used in this study integrates multiple sensing modalities within a Nissan Townstar van to achieve precise and reliable environmental perception (see Figure 14). The external sensor setup is composed of an OS1 LiDAR (mounted 270 mm above the base with a custom cylindrical flange), a Ladybug 5+ camera (elevated to 578 mm via a tripod adapter), and a ZED X stereo camera (positioned at 175 mm with an adjustable "L" bracket for pitch alignment). Due to the elevated mounting height of the external sensors, the resulting geometric configuration differs from that of standard passenger vehicles, affecting viewing angles and the spatial distribution of LiDAR points on calibration targets. To account for this aspect and to avoid overfitting the calibration procedure to a single fixed geometry, experiments were conducted by varying the distance between the sensor suite and the calibration target across multiple acquisition sessions. Inside the vehicle, additional modules are installed, including the IMU, the GPS board, the computing and storage units, and the power electronics required to supply the entire system. In particular, power is provided by two 24 V, 70 Ah battery groups, regulated through inverters that convert 24 V DC into 230 V AC, ensuring stable distribution while continuously monitoring voltage and current.

Data processing is distributed across two computing units: a NUC, which manages the inputs from the Ladybug camera, the IMU, and the DURO GPS, and a Jetson, which processes the data from the OS1 LiDAR and the ZED X stereo camera.

All components are interconnected via a network switch, which implements a modular software architecture that logically separates sensing, processing, and storage functions. This architecture enables efficient data flow between subsystems, ensures scalability for future expansion, and simplifies the integration and maintenance of heterogeneous modules.

The acquisition system is based on the open-source framework ROS. Sensor data streams are synchronized and collected in real time, then stored into dedicated log files. To optimize storage, data are automatically compressed and transferred to a NAS for long-term retention.

A modem ensures network synchronization across all devices, while a manual interface in the passenger compartment allows for real-time control of power distribution and system settings. To maintain signal integrity and minimize interference, all cables are routed through protective housings beneath the structure, resulting in a clean and efficient integration of power and data management for high-fidelity perception.



(**a**) CAD representation of the system setup, axonometric view

(**b**) Setup system, lateral view

Figure 14. Setup system mounted on the experimental vehicle.

6. Results

To demonstrate the effectiveness of the proposed calibration method, we evaluated its performance when reducing the number of target poses from nine to a single pose. The validation relied on a task-driven evaluation criterion directly relevant to autonomous driving: the lateral error perceived by the sensor system for an obstacle placed at a known distance from the sensor setup. This metric is critical because it directly impacts vehicle maneuvering and obstacle localization accuracy. In the experimental setup, a red spherical target (Figure 15) was positioned at a known distance from the sensor setup. Its center was detected in the image plane and reprojected into the LiDAR reference frame. From this reprojection, the azimuth deviation $\Delta\theta$ and elevation deviation $\Delta\psi$ were computed. The azimuth deviation was then converted into a lateral error L_{error} at the obstacle distance d_{obstacle} (measured from the sensor setup), according to:

$$L_{\text{error}} = d_{\text{obstacle}} \cdot \tan(\Delta\theta) \tag{29}$$

When reducing the calibration from nine poses to a single pose, the increase in lateral error was negligible:

$$L_{\text{error}} = 0.15 \text{ mm} \quad \text{for} \quad d_{\text{obstacle}} = 10 \text{ m from the sensor setup.}$$

Detailed results for both LiDAR-to-Ladybug and LiDAR-to-ZED X transformations are reported in Table 4. This minimal increase confirms that the single-pose configuration satisfies the accuracy requirements of autonomous driving applications while significantly reducing calibration complexity.

Table 4. Azimuth ($\Delta\theta$) and elevation ($\Delta\psi$) differences for LiDAR-LADYBUG and LiDAR-ZED X transformations. The target was placed 4.5–5.5 m from the board center.

Transformation	Poses	$\Delta\theta$ (°)	$\Delta\psi$ (°)
LiDAR-to-Ladybug	1	0.0152	0.11
LiDAR-to-ZED X	1	0.0027	0.013

Figure 15. Validation setup: the sensor system is shown on the left, while the red spherical target is shown on the right, with its center selected for computing the azimuth and elevation error deltas.

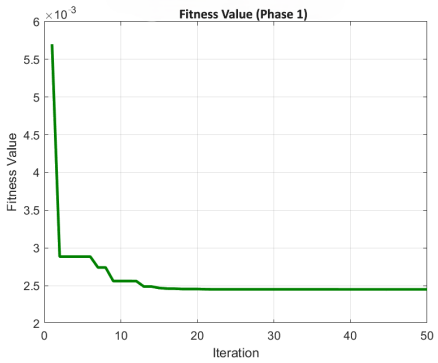
Once the number of poses was reduced to one, we compared the proposed method with the approach in [7]. Zhou's method (Table 5), while achieving a significantly lower elapsed time—processing a single frame in just 3.1 s compared to the 10.986 s required by the proposed method—relies exclusively on automatic detection of checkerboard dimensions.

This reliance, heavily influenced by external environmental conditions, leads to azimuth and elevation errors that are, on average, 217 times higher than those obtained using the proposed method, highlighting the necessity of managing uncertainties and leveraging stable features for precise and reliable extrinsic calibration.

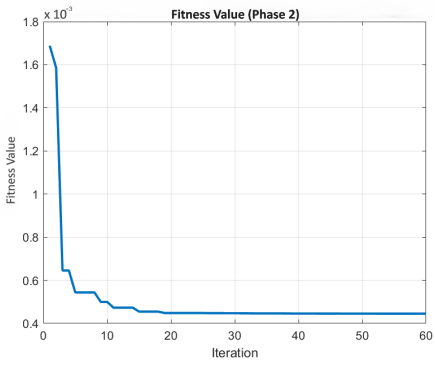
Table 5. Performance evaluation of Zhou's method.

Method	$\Delta\theta$ (°)	$\Delta\psi$ (°)	Time (s)
Zhou	0.9985	0.9138	3.1

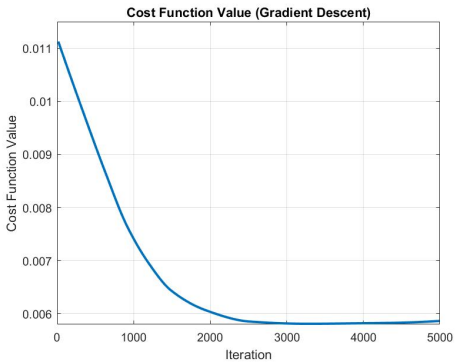
For completeness, the hybrid metaheuristic optimization approach was compared with classic gradient descent, included as a representative local optimization baseline. As shown in Figures 16 and 17 and Table 6, gradient descent frequently becomes trapped in local minima, resulting in suboptimal calibration. This comparison highlights the ability of metaheuristic strategies to explore the non-linear and multimodal cost landscape more effectively, achieving higher accuracy and robustness even when the initial estimate is poor. In contrast, the proposed approach successfully completes the optimization in just 0.284 s.



(**a**) Fitness value progression in GA objective function for Phase 1 (single pose).



(**b**) Fitness value progression in PSO objective function for Phase 2 (single pose).



(**c**) Optimization through Gradient Descent Method.

Figure 16. Comparison of optimization performance for single-pose objective functions: (**a**) Phase 1 GA-based optimization; (**b**) Phase 2 PSO-based optimization; (**c**) Optimization through Gradient Descent Method.

(**a**) Point cloud (Red dots in the image) transformed into the camera frame using the pose estimated through the Gradient Descent optimization.

(**b**) Point cloud (Red dots in the image) transformed into the camera frame using the pose estimated through the proposed Hybrid GA–PSO optimization.

Figure 17. Visualization of the point cloud after transformation into the camera coordinate frame using the pose estimates obtained through (**a**) Gradient Descent optimization; (**b**) the proposed Hybrid GA–PSO optimization method.

Table 6. Results with the Gradient Descent Method.

Delta Azimuth (°)	Delta Elevation (°)
0.6910	13.38

Building on the same task-driven validation strategy, the influence of target distance on calibration accuracy was systematically analyzed. This analysis serves a dual purpose: first, to identify a practical distance range that ensures calibration reliability while relaxing setup constraints and increasing flexibility in target placement; and second, to evaluate the robustness of the proposed method under varying effective geometric configurations induced by sensor mounting height and viewing angle. As illustrated in Figure 18 and summarized in Table 7, calibration error exhibits only a marginal increase when the target distance from the sensor setup varies between 4.53 m and 6.71 m. At approximately 7 m, the resulting angular deviations translate into lateral errors L_{error} that remain within the centimeter range. Such deviations are well below the thresholds that affect obstacle localization, lane-level perception, and trajectory planning in urban and highway autonomous driving scenarios. Beyond this distance, error growth becomes non-linear. Angular deviations $\Delta\theta$ of 1–2 degrees produce lateral displacements ranging from a few centimeters for near-field objects—such as those encountered during assisted parking—to approximately 30 cm at $d_{\text{obstacle}} = 10$ m. This larger distance is considered representative of scenarios where vehicles ahead are detected at extended ranges, for example, during highway driving. The comparison between these two cases illustrates the impact of distance on calibration accuracy: small lateral errors at short ranges may be acceptable for low-speed maneuvers, whereas the substantially larger deviations observed at longer ranges compromise the requirements of safety-critical autonomous driving tasks that demand precise obstacle positioning. Since sensor mounting height primarily affects calibration through changes in viewing geometry and angular sensitivity, expressing calibration validity in terms of target distance provides a vehicle-agnostic criterion that captures these effects in a practical and measurable form. Based on this analysis, 7 m is identified as the maximum reliable target distance for static offline calibration, providing a balanced compromise between operational flexibility and the accuracy constraints of autonomous driving applications.

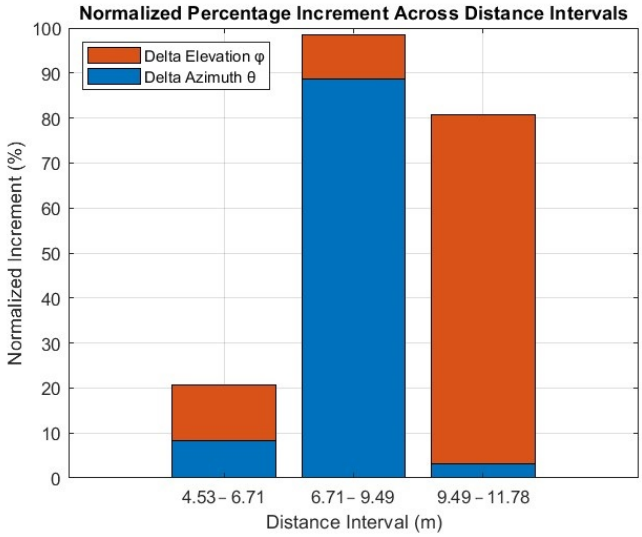


Figure 18. Percentage increase in calibration errors as the target distance grows.

Table 7. Normalized percentage increment for azimuth and elevation changes across distances.

Range (m)	ΔAzim (%)	ΔElev (%)
4.53–6.71	8.20	12.50
6.71–9.49	88.73	9.77
9.49–11.78	3.06	77.73

7. Conclusions and Future Work

This paper presented a metaheuristic-based framework for extrinsic sensor calibration tailored to autonomous-vehicle platforms. The proposed method satisfies the demanding requirements of AV perception pipelines by combining speed, adaptability, and precision. These properties align with the needs observed in large-scale benchmarks, where accurate and up-to-date calibration underpins reliable multi-sensor fusion [1,3].

For pairwise calibration tasks (e.g., Ladybug+LiDAR or ZED+LiDAR), the process completes in approximately 11 s at runtime, with the search space and metaheuristic parameters already defined during the offline configuration stage. Optimization parameters selected via Bayesian optimization yield robust and more accurate results than Zhou's state-of-the-art method. Although the computational time is slightly higher, this increase is justified by improved noise handling, enabling superior precision even up to a 7-m range without requiring a strictly controlled environment. The experiments also confirmed consistent performance across different sensor pairs and scene configurations, with repeatability errors below 0.5°, indicating readiness for integration into autonomous-vehicle calibration pipelines.

Future extensions of this work will address four key aspects:

- porting the implementation to C++ for real-time and embedded deployment;
- performing a systematic sensitivity analysis of weighting coefficients to better quantify their influence on calibration accuracy;
- extending the calibration range through distance-weighted costs and improved target reflectivity;
- validating robustness under diverse environmental conditions;
- unifying intrinsic and extrinsic calibration into a single optimization stage.

Collectively, these enhancements will strengthen the robustness and flexibility of the proposed calibration framework, extending its applicability to real-world autonomous

vehicles. The achieved accuracy, repeatability, and execution time demonstrate the maturity of the approach as a reliable calibration tool for current AV research and development while paving the way toward fully automated, high-precision calibration pipelines for next-generation intelligent transportation systems.

Author Contributions: Conceptualization, F.D. and H.H.C.; methodology, F.D. and S.A.; software, F.D.; validation, S.A., F.D. and H.H.C.; formal analysis, H.H.C.; investigation, F.D.; data curation, F.D.; writing—original draft preparation, F.D.; writing—review and editing, S.A.; supervision, S.A.; project administration, S.A.; funding acquisition, S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to (specify the reason for the restriction).

Acknowledgments: The authors would like to thank Milano Serravalle—Milano Tangenziali S.p.A. for their support in this work, in particular for providing access to the test vehicle and the measurement equipment used in the experimental campaign. During the preparation of this manuscript/study, the author(s) used ChatGPT 5.0 for the purposes of textual revision. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yeong, S.M.; Velasco-Hernandez, J.; Barry, C.; Walsh, J.J. Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review. *Sensors* **2021**, *21*, 2140. [CrossRef] [PubMed]
2. Wang, H.; Liu, J.; Dong, H.; Shao, Z. A Survey of the Multi-Sensor Fusion Object Detection Task in Autonomous Driving. *Sensors* **2025**, *25*, 2794. [CrossRef] [PubMed]
3. Caesar, H.; Bankiti, V.; Lang, A.H.; Vora, S.; Liong, V.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; Beijbom, O. nuScenes: A Multimodal Dataset for Autonomous Driving. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 11618–11628. [CrossRef]
4. Sun, P.; Kretzschmar, H.; Dotiwalla, X.; Chouard, A.; Patnaik, V.; Tsui, P.; Guo, J.; Zhou, Y.; Chai, Y.; Caine, B.; et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 2446–2454. [CrossRef]
5. Zhang, R.P.Q. Extrinsic Calibration of a Camera and Laser Range Finder (Improves Camera Calibration). In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan, 28 September–2 October 2004; Volume 3, pp. 2301–2306.
6. Geiger, A.; Moosmann, F.; Car, Ö.; Schuster, B. Automatic Camera and Range Sensor Calibration Using a Single Shot. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA), St. Paul, MN, USA, 14–18 May 2012; pp. 3936–3943.
7. Zhou, L.; Deng, Z. A New Algorithm for Computing the Projection Matrix Between a LiDAR and a Camera Based on Line Correspondences. In Proceedings of the 2012 IV International Congress on Ultra Modern Telecommunications and Control Systems, St. Petersburg, Russia, 3–5 October 2012; pp. 436–441.
8. Dhall, A.; Chelani, K.; Radhakrishnan, V.; Krishna, K.M. LiDAR-Camera Calibration Using 3D-3D Point Correspondences. *arXiv* **2017**, arXiv:1705.09785.
9. Guindel, C.; Beltrán, J.; Martín, D.; García, F. Automatic Extrinsic Calibration for LiDAR-Stereo Vehicle Sensor Setups. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 1–6.
10. An, P.; Ding, J.; Quan, S.; Yang, J.; Yang, Y.; Liu, Q.; Ma, J. Survey of Extrinsic Calibration on LiDAR-Camera System for Intelligent Vehicle: Challenges, Approaches, and Trends. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 15342–15366. [CrossRef]
11. Beltrán, J.; Guindel, C.; de la Escalera, A.; García, F. Automatic Extrinsic Calibration Method for LiDAR and Camera Sensor Setups. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 17677–17689. [CrossRef]
12. TEXA S.p.A. ADAS Radar & Camera Calibration Kit. Available online: https://www.texa.com/products/adas-radar-camera-calibration-kit/#ADAS_RCCS3 (accessed on 4 January 2026).

13. American Automobile Association. Cost of Advanced Driver Assistance Systems (ADAS) Repairs. Technical Report, American Automobile Association, Inc. 2023. Available online: https://newsroom.aaa.com/wp-content/uploads/2023/11/Report_Cost-of-ADAS-Repairs-FINAL-23.pdf (accessed on 4 January 2026).
14. Levinson, J.; Thrun, S. Automatic Online Calibration of Cameras and Lasers. In *Robotics: Science and Systems (RSS)*; KIT-MRT: Karlsruhe, Germany, 2013. Available online: https://www.roboticsproceedings.org/rss09/p29.pdf (accessed on 4 January 2026).
15. Wang, Y.; Xing, S.; Can, C.; Li, R.; Hua, H.; Tian, K.; Mo, Z.; Gao, X.; Wu, K.; Zhou, S.; et al. Generative AI for Autonomous Driving: Frontiers and Opportunities. *arXiv* **2025**, arXiv:2505.08854. [CrossRef]
16. Bilal, H.; Rehman, A.; Aslam, M.S.; Ullah, I.; Chang, W.J.; Kumar, N.; Almuhaideb, A.M. Hybrid TrafficAI: A Generative AI Framework for Real-Time Traffic Simulation and Adaptive Behavior Modeling. *IEEE Trans. Intell. Transp. Syst.* **2025**, 1–17. [CrossRef]
17. Zhou, L.; Li, Z.; Kaess, M. Automatic Extrinsic Calibration of a Camera and a 3D LiDAR Using Line and Plane Correspondences. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 5562–5569. [CrossRef]
18. Verma, S.; Berrio, J.S.; Worrall, S.; Nebot, E. Automatic Extrinsic Calibration Between a Camera and a 3D LiDAR Using 3D Point and Plane Correspondences. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 3906–3912.
19. Jeong, S.; Lee, J.; Kim, J.; Cho, S. LiDAR-camera calibration based on the characteristics of LiDAR sensor. *Sensors* **2024**, *24*. [CrossRef]
20. Gentilini, L.; Serio, P.; Donzella, V.; Pollini, L. A target-based multi-LiDAR multi-camera extrinsic calibration system *arXiv* **2025**. [CrossRef]
21. Inghilterra, G.; Arrigoni, S.; Braghin, F.; Cheli, F. Firefly Algorithm-Based Nonlinear MPC Trajectory Planner for Autonomous Driving. In Proceedings of the 2018 International Conference of Electrical and Electronic Technologies for Automotive (EETA), Milan, Italy, 9–11 July 2018; pp. 1–6. [CrossRef]
22. Walters, R.; Tan, L.; Ferrari, F. A Robust Framework for Online Extrinsic Calibration of Multi-Sensor Vehicle Platforms. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 8432–8439. [CrossRef]
23. Laporte, G.; Osman, I.H. Routing Problems: A Bibliography. *Ann. Oper. Res.* **1995**, *61*, 227–262. [CrossRef]
24. Igel, C. No Free Lunch Theorems: Limitations and Perspectives of Metaheuristics. In *Theory and Principled Methods for the Design of Metaheuristics*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 1–23.
25. Lazar, A.; Reynolds, R.G. Heuristic Knowledge Discovery for Archaeological Data Using Cultural Algorithms and Rough Sets. In *Heuristics and Optimization for Knowledge Discovery*; Abbass, H.A., Newton, C.S., Sarker, R.A., Eds.; Idea Group Publishing: Hershey, PA, USA, 2002; pp. 263–278.
26. Yang, X.S. *Nature-Inspired Optimization Algorithms*; Academic Press: Boca Raton, FL, USA, 2020.
27. Panwar, K.; Deep, K. Discrete Grey Wolf Optimizer for Symmetric Travelling Salesman Problem. *Appl. Soft Comput.* **2021**, *105*, 298. [CrossRef]
28. Molina, D.; Poyatos, J.; Ser, J.D.; García, S.; Hussain, A.; Herrera, F. Comprehensive Taxonomies of Nature- and Bio-Inspired Optimization: Inspiration Versus Algorithmic Behavior, Critical Analysis Recommendations. *Cogn. Comput.* **2020**, *12*, 897–939. [CrossRef]
29. Rajwar, K.; Deep, K.; Das, S. An Exhaustive Review of the Metaheuristic Algorithms for Search and Optimization: Taxonomy, Applications, and Open Challenges. *Artif. Intell. Rev.* **2023**, *56*, 13187–13257. [CrossRef] [PubMed]
30. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence*; University of Michigan Press: Ann Arbor, MI, USA, 1975; pp. 439–444.
31. Goldberg, D.E.; Deb, K. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In *Foundations of Genetic Algorithms*; Rawlins, G.J.E., Ed.; Morgan Kaufmann: Los Altos, CA, USA, 1991; pp. 69–93.
32. Arrigoni, S.; Braghin, F.; Cheli, F. MPC Trajectory Planner for Autonomous Driving Solved by Genetic Algorithm Technique. *Veh. Syst. Dyn.* **2022**, *60*, 4118–4143. [CrossRef]
33. Vanneschi, L.; Silva, S. Genetic Algorithms. In *Lectures on Intelligent Systems*; Natural Computing Series; Springer: Cham, Germany, 2023. [CrossRef]
34. De Jong, K. The Analysis and Behaviour of a Class of Genetic Adaptive Systems. Ph.D. Thesis, University of Michigan, Ann Arbor, MI, USA, 1975.
35. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948.
36. Arrigoni, S.; Trabalzini, E.; Bersani, M.; Braghin, F.; Cheli, F. Nonlinear MPC Motion Planner for Autonomous Vehicles Based on Accelerated Particle Swarm Optimization Algorithm. In Proceedings of the 2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), Turin, Italy, 2–4 July 2019; pp. 1–6.

37. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
38. Donnett, J.G. *Simulated Annealing and Code Partitioning for Distributed Multimicroprocessors*; Technical report; Department of Computer and Information Science, Queen's University: Kingston, ON, Canada, 1987.
39. Woo, A.; Fidan, B.; Melek, W.W. Localization for Autonomous Driving. In *Handbook of Position Location: Theory, Practice, and Advances*, 2nd ed.; Zekavat, S., Buehrer, R.M., Eds.; Wiley-IEEE Press: Hoboken, NJ, USA, 2019; pp. 1051–1087.
40. Roman, I.; Ceberio, J.; Mendiburu, A.; Lozano, J.A. Bayesian Optimization for Parameter Tuning in Evolutionary Algorithms. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 4839–4845.
41. Arrigoni, S.; Mentasti, S.; Cheli, F.; Matteucci, M.; Braghin, F. Design of a Prototypical Platform for Autonomous and Connected Vehicles. In Proceedings of the 2021 AEIT International Conference on Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), Torino, Italy, 17–19 November 2021; pp. 1–6. [CrossRef]