Leveraging Road Scene Geometry for Sensor Calibration in Visual Traffic Monitoring Systems

A THESIS

submitted by

Yatin Rajendra Phalak

(MT23108)

in partial fulfillment of the requirements
for the award of the degree

MASTER OF TECHNOLOGY

COMPUTER SCIENCE ENGINEERING

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

MAY 2025

THESIS CERTIFICATE

This is to certify that the thesis titled **Leveraging Road Scene Geometry for Sensor Calibration in Visual Traffic Monitoring Systems**, submitted by **Yatin Rajendra Phalak**, to the Indraprastha Institute of Information Technology, Delhi, for the award of the degree of Master of Technology, is a bona fide record of the research work carried out by him under our supervision. The contents of this thesis, in whole or in part, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Saket Anand
Thesis Supervisor
Associate Professor
Dept. of CSE and ECE
IIIT Delhi, 110020

Dr. Arani Bhattacharya
Thesis Supervisor
Associate Professor
Dept. of CSE
IIIT Delhi, 110020

Place: New Delhi

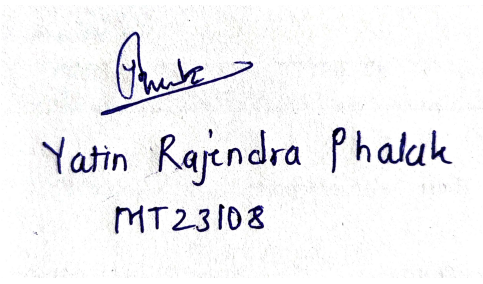
Acknowledgements

I would like to express my profound gratitude to my thesis advisors, **Dr. Saket Anand** and **Dr. Arani Bhattacharya**, for their exceptional mentorship, continuous guidance, and unwavering support throughout the course of this research. Their deep expertise, constructive feedback, and insightful discussions have been instrumental in shaping both the technical and conceptual aspects of this work. I remain sincerely indebted to them for the opportunity to work under their supervision and for their confidence in my abilities.

This research was supported by the **Vehant Technologies Fellowship Program**, for which I am sincerely thankful. I am especially grateful to **Dr. Krishan Sharma** and **Dr. Renu Rameshan** of Vehant Technologies for their invaluable advice and mentorship, particularly in deepening my understanding of computer vision and contributing significantly to the homography estimation component of this work. Their guidance has greatly enriched the scientific rigor of this thesis.

I also wish to acknowledge the academic environment at **IIIT-Delhi**, including the faculty and my peers, for fostering an intellectually stimulating atmosphere and providing helpful feedback throughout the research process.

Lastly, I am profoundly thankful to my family for their unwavering emotional support, patience, and encouragement. Their faith in me has been a source of constant strength, and I am deeply grateful for their enduring presence throughout this journey.



Abstract

This thesis investigates how the inherent geometric structure of road scenes can be effectively leveraged to solve two critical problems in visual traffic monitoring systems: sensor calibration and vehicle speed estimation. Instead of relying on manual calibration procedures or artificial targets, this work builds on the insight that road environments themselves provide rich geometric cues such as planar surfaces, consistent structures, and vanishing lines that can be used for robust computation across modalities.

The first part addresses the challenge of LiDAR–camera cross-calibration. Here, planar surfaces commonly found in road scenes such as the ground, walls, and signboards are extracted from LiDAR point clouds and camera images. These geometries are matched using graph-based techniques, and a robust transformation is computed using model fitting to estimate the extrinsic parameters between the sensors. This enables automatic calibration in dynamic and unstructured environments, making it suitable for real-world traffic applications.

The second part focuses on monocular vehicle speed estimation. A homography between the image plane and road surface is estimated by combining depth prediction, road segmentation, and intrinsic approximation using either vanishing point geometry or learning-based models. Once established, this homography maps image-space vehicle motion into real-world distances, allowing accurate speed computation from a single camera without requiring additional sensors or per-frame depth.

By unifying both problems under the principle of exploiting road scene geometry, this thesis presents a cohesive and practical framework for vision-based traffic monitoring. The methods are validated across multiple challenging datasets and demonstrate robust performance even in the absence of ideal calibration conditions or sensor accuracy, enabling scalable deployment in real-world traffic systems.

Contents

List of Tables

List of Figures

ABBREVIATIONS

LiDAR	Light Detection and Ranging
RANSAC	Random Sample Consensus
ADAS	Advanced Driver Assistance Systems
CT-ICP	Continuous-Time Iterative Closest Point
VPs	Vanishing Points
PnP	Perspective-n-Point
ICP	Iterative Closest Point
MI	Mutual Information
NID	Normalised Information Distance
P3L	Perspective-Three-Line
DLT	Direct Linear Transform
K	Intrinsic Camera Matrix
IAC	Image of the Absolute Conic
LSD	Line Segment Detector
IQR	Interquartile Range
GT	Ground Truth
FoV	Field of View
R	Rotation
t	Translation
DOF	Degrees of Freedom
KDE	Kernel Density Estimation
Dist	Distance
SVD	Singular Value Decomposition

Chapter 1

Introduction

Visual traffic monitoring systems are essential components of intelligent transportation networks. They provide critical support for managing traffic congestion, enforcing road safety regulations, collecting data for urban planning, and enabling automated event detection. These systems rely on extracting reliable spatial and temporal information from visual inputs to interpret vehicle behavior, road usage patterns, and environmental context. At the heart of this capability lies the need for accurate measurement of position, motion, and scale from images or video streams.

While traditional traffic monitoring relies on fixed infrastructure, the advent of autonomous vehicles introduces the potential for a dynamic, mobile form of traffic surveillance. Autonomous vehicles, already equipped with sophisticated sensor suites and computational power, can serve as moving traffic monitoring units. As they navigate through urban environments, these vehicles can collect and process rich visual and geometric data in real time. This opens the door to scalable, flexible, and real-time traffic monitoring that is not bound to fixed locations.

Autonomous vehicles rely on a suite of sensors, cameras, and LiDARs, each contributing with complementary information about the environment. To interpret this data coherently, it must be fused in a common reference frame, which requires accurate extrinsic calibration between sensors. This calibration aligns sensor measurements spatially, enabling tasks like object detection, localization, and obstacle avoidance. Without it, even high-quality data cannot be meaningfully combined, leading to perception errors. Reliable, synchronized, and well-calibrated sensor fusion is thus essential not only for safe and effective autonomous operation but also for deriving insights from traffic data.

The successful deployment of traffic monitoring systems critically depends on accurate calibration and periodic recalibration of the associated sensors. In this thesis, we propose a fully automatic framework for sensor calibration tailored to traffic monitoring applications. Our work addresses two specific scenarios:

1. **Autonomous systems with onboard sensors**, where we introduce a novel, target-free method for automatic cross-calibration between LiDAR and camera systems.

2. **Fixed monocular camera-based traffic monitoring systems**, where we develop a geometry-driven approach for accurate real-world distance and speed estimation using a monocular camera input.

1.1 LiDAR-camera cross calibration

In autonomous systems and advanced driver assistance systems (ADAS), precise sensor fusion is critical to achieving reliable perception of the environment. Among the most common sensor pairs used for 3D perception are LiDAR (Light Detection and Ranging) and cameras. LiDAR provides accurate 3D geometric information about the environment in the form of sparse point clouds, while cameras offer rich semantic and texture information. To enable effective multi-sensor fusion, it is essential to determine the extrinsic calibration between the LiDAR and camera i.e., the rigid transformation that maps 3D points from the LiDAR coordinate frame to the camera coordinate frame.

LiDAR-camera cross calibration aims to compute the rotation and translation between these two coordinate systems. Accurate calibration allows projecting LiDAR points onto the image plane, enabling tasks like 3D object detection, semantic segmentation of point clouds, depth supervision from LiDAR for monocular models, and visual localization.

Challenges in LiDAR-Camera Cross Calibration

Accurate LiDAR-camera cross calibration is essential for fusing depth and visual data in autonomous systems. However, achieving robust calibration in real-world scenarios presents multiple technical challenges:

- **Dynamic Environments:** Autonomous vehicles operate in continuously changing environments, making stable data acquisition difficult for calibration purposes [3].

- **Sensor Misalignment and Noise:** Slight misalignment or sensor noise can drastically degrade calibration accuracy, impacting subsequent perception tasks.

- **Sparse LiDAR Data:** LiDAR sensors provide sparse point clouds, particularly when the vehicle is moving rapidly, which makes matching data points between sensors challenging.

- **Complex and Manual Procedures:** Many existing calibration methods require manual intervention, making the calibration process labor-intensive and prone to human errors.
- **Lack of Universal Calibration Targets:** Finding or designing calibration targets suitable for various sensors and scenarios is challenging, limiting the versatility of calibration methods.
- **Low-Resolution LiDAR Sensors:** Low-resolution LiDAR sensors produce sparse point clouds with fewer data points, which reduces the effectiveness of feature-matching algorithms and decreases calibration accuracy.

The process of LiDAR-camera calibration has been extensively studied and can be broadly classified into two categories: target-based and targetless methods. Target-based methods rely on specially designed calibration patterns such as checkerboards, fiducial markers, or polygonal boards that are simultaneously visible to both sensors. Early works such as [4] and [5] used planar targets and intensity-based detection to estimate extrinsic parameters by solving geometric alignment problems like the Perspective-n-Point (PnP) formulation. However, such methods often require controlled setups, careful placement of calibration targets, and manual annotation, making them impractical for large-scale deployments or dynamic, unstructured environments.

To overcome these limitations, targetless methods have emerged, aiming to estimate extrinsic parameters without relying on artificial patterns. These approaches utilize natural features in the environment or exploit motion over time. A common class of targetless methods is based on scene geometry alignment, where edges, planes, or intensity gradients observed in LiDAR data are matched with corresponding structures in the camera image. These include both indirect feature-based methods, which extract and match edges or lines [6, 7], and direct methods, which rely on statistical similarity between sensor modalities, such as maximizing mutual information (MI) or normalized MI between image intensity and LiDAR reflectivity [8, 9]. More recently, targetless methods leveraging deep learning-based approaches have emerged to learn features from sensor data, and use those to solve for the extrinsic cross calibration problem, [10, 11, 12], which has problems like densification of LiDAR data, which fails on sparse LiDAR data. However, in this work, instead of extracting features, we make use of the observed scene structure, specifically the road environment, and leverage its dominant planar surfaces to perform extrinsic cross-calibration between the LiDAR and camera.

While target-based methods remain the most accurate, targetless and learning-based approaches offer greater flexibility and automation. This thesis builds upon these advancements to develop a geometry-driven, targetless calibration framework suited for

real-world urban environments and sparse scenes, addressing the limitations of prior work in scalability and automation.

We propose a geometry-based LiDAR-camera calibration method that leverages planar structures extracted from sparse LiDAR scans and 3D camera points reconstructed using depth and intrinsics. The planar correspondences are established using bipartite graph matching, followed by robust rotation and translation estimation via RANSAC. This pipeline will be discussed in detail in the chapter 4.

1.2 Homography-based Distance Estimation

Static camera systems remain central to current traffic surveillance deployments. Installed at intersections, highways, and critical junctions, these cameras are used for a variety of applications, including vehicle counting, speed estimation, incident detection, and law enforcement. Their fixed viewpoints make them ideal for long-term monitoring and data aggregation. However, extracting precise real-world measurements from monocular images, such as estimating distances and speeds, remains a significant challenge, particularly in the absence of depth sensors or elaborate calibration setups.

Traffic cameras are widely used for speed estimation, distance measurement, and vehicle tracking. However, converting image observations into real-world measurements requires accurate calibration. Since monocular cameras lack depth information, both intrinsic and extrinsic parameters are essential to relate image coordinates to physical distances. Without calibration, tasks like speed monitoring or vehicle localization become unreliable, making calibration a critical step in static visual traffic systems.

Homography-based distance estimation between the image and the road plane is a valuable technique for vehicle speed detection, but it encounters several practical limitations. Monocular vision lacks direct depth information, and learning-based depth methods often struggle in unseen environments due to domain shift, affecting distance accuracy.

There's also a trade-off between local and global road plane approximations. Local homographies work on small flat regions but don't generalize well, while global ones can suffer from distortions on uneven roads. Additional issues like camera placement, motion blur, occlusions, and detection errors further impact accuracy, making robust and adaptable methods essential for real-world deployment.

Most existing works depend on either visual patterns (lane markings), temporal cues (motion), or statistical priors (vehicle sizes), manual calibration, using known physical measurements from the scene [13, 14]. These dependencies limit generalization in uncontrolled settings, particularly when scenes are sparse, dynamic, or lack repeated vehicle

passes.

In this work, we propose a fully automatic, geometry driven framework for monocular vehicle speed estimation based on robust image-to-road homography computation. The central idea is to leverage road scene geometry extracted from a single RGB image via monocular depth estimation and intrinsic calibration to derive a homography transformation that maps image pixels to accurate, metric ground-plane coordinates. By fitting a dominant road plane using RANSAC on the depth data, we achieve a robust and scale-consistent mapping, enabling precise real-world distance estimation necessary for speed computation.

The intrinsic parameters of a camera, particularly the focal length, play a crucial role in establishing the correct scale in monocular vision systems. Without accurate intrinsics, the mapping from image pixels to 3D metric space becomes ambiguous, leading to incorrect depth and scale estimations.

Unlike previous approaches that rely on manual measurements, vehicle tracking over time, or assumptions about road markings, our method requires only a single RGB image. If Intrinsic parameters are not available, we first estimate the camera intrinsic parameters from vanishing points in the scene, assuming a pinhole camera model with zero skew and a centered principal point. Vanishing points corresponding to two orthogonal directions in the scene allow us to recover focal length and other intrinsics as outlined in [15], or using the deep learning methods to estimate the intrnsic[16]. This step ensures geometric consistency for downstream 3D estimation and projection.

Overall, our method provides a lightweight, manual calibration-free alternative to traditional speed estimation techniques, suitable for real-time deployment in urban and highway traffic monitoring systems, especially in scenes lacking lane markings, structured road geometry, or repeated vehicle patterns.

1.3 Contributions of the Thesis

This thesis presents a geometry-driven, automatic calibration framework addressing two major challenges in traffic monitoring: LiDAR-camera cross calibration for mobile systems and homography-based distance estimation from monocular images in fixed-camera setups. Our contributions span robust plane extraction, feature matching, and homography estimation grounded in both classical geometry and deep learning.

1.3.1 LiDAR-Camera Cross Calibration

We propose a target-free, geometry-based method for extrinsic calibration between LiDAR and camera sensors. By detecting planar structures in both modalities and matching them via graph optimization, we achieve robust and automatic cross-sensor alignment.

- **Plane Detection in LiDAR Point Clouds:** The Progressive-X algorithm is employed to detect planes in LiDAR point clouds efficiently. This method ensures robust multi-model fitting and enables accurate geometric feature extraction from LiDAR data [17].
- **Plane Detection in Camera Images:** A 3D point cloud is reconstructed using metric3D depth data to extract planar surfaces in camera images, allowing reliable feature matching between LiDAR and camera data.
- **Bipartite Graph Matching using Hungarian Algorithm:** We use the Hungarian graph matching algorithm to match the detected geometric entities in the point cloud.
- **Point Cloud Registration Using RANSAC:** A robust registration method is implemented using the RANSAC algorithm to align LiDAR and camera data. This step eliminates outliers and enhances the accuracy of the calibration process.

1.3.2 Homography based Distance Estimation

We introduce a homography estimation framework for fixed monocular cameras by leveraging estimated camera intrinsics and monocular depth. This enables accurate mapping from image coordinates to real-world metric space for distance and speed estimation.

- **Intrinsic Approximation:** Camera intrinsics can be approximated either by utilizing three mutually orthogonal vanishing points (VPs), by assuming square pixels with zero skew (commonly referred to as the "seasonal assumption"), or through deep learning-based intrinsic parameter estimation techniques.
- **Depth Based Homography Estimation:** Given a planar surface in the scene and the estimated camera intrinsics, homography can be computed using a depth estimation method combined with RANSAC to fit the planar transformation for a fixed viewpoint robustly.

By integrating these contributions, this research aims to provide an automated and robust LiDAR-camera calibration method tailored for real-world autonomous driving

applications and an automated homography estimation pipeline suitable for extracting speeds of vehicles.

1.4 Thesis Organization

This thesis is structured into six chapters, each covering a different aspect of the research.

- **Chapter 1: Introduction**: Provides an overview of the research problem, motivation, key challenges, and contributions of this work.
- **Chapter 2: Literature Review**: Reviews existing methods for LiDAR-camera calibration, highlighting their strengths and limitations, and identifies the research gap addressed in this thesis.
- **Chapter 3: LiDAR camera cross calibration**: Details the proposed approach, including plane detection in LiDAR and camera images, point cloud registration using RANSAC, and feature matching techniques, experimental setup, datasets used, evaluation metrics, and presents the performance analysis of the proposed method..
- **Chapter 4: Homography based distance estimation**: Details the proposed approach, including VP-based Intrinsic approximation and Unidpeth-based Intrinsic extraction, Homography estimation using RANSAC, experimental setup, datasets used, evaluation metrics, and presents the performance analysis of the proposed method..
- **Chapter 5: Discussion and Future Work**: Analyzes the results, compares them with state-of-the-art methods, and discusses strengths, limitations, and areas for improvement suggests potential directions for future improvements in LiDAR-camera calibration..
- **Chapter 6: Conclusion**: Summarizes the contributions of this research.

This structure ensures a logical flow of information, guiding the reader through the problem, existing solutions, proposed methodology, results, and conclusions.

Chapter 2

Literature Review

This chapter provides a comprehensive review of existing techniques for LiDAR-Camera cross calibration and Speed estimation. By examining the current state of the art for both LiDAR-Camera cross calibration and Speed estimation modules, we aim to identify the gaps and challenges that can be addressed in our methodology. The first part reviews existing methods for LiDAR-camera cross calibration, categorized into target-based and targetless approaches, including both classical geometric techniques and recent learning-based methods. It highlights the trade-offs between accuracy, automation, and real-world applicability. The second part focuses on image-to-plane homography estimation for monocular speed measurement, analyzing calibration methods based on lane markings, vehicle motion, manual measurements, and statistical vehicle dimensions. A comparative summary and overview of relevant datasets are also provided to contextualize the evaluation of proposed methods.

2.1 LiDAR-camera cross calibration

LiDAR-camera calibration has been extensively explored through various methods that can broadly be categorized into target-based and targetless approaches. Each category presents distinct trade-offs in terms of accuracy, automation, and applicability to real-world autonomous driving scenarios.

2.1.1 Target-Based Methods

Target-based methods remain among the most accurate and widely adopted techniques for extrinsic calibration of the LiDAR camera. These methods typically rely on specially designed calibration targets, such as checkerboards, polygonal boards, or fiducial marker arrays, which are simultaneously visible in camera images and LiDAR point clouds. Early

works like [4] utilized a planar checkerboard to extract the geometry of a plane in both modalities, solved the calibration problem by minimizing point-to-plane distances, aligning LiDAR points to the image-observed plane. Method required manual annotation of LiDAR data and assumed pre-calibrated camera intrinsics. [18] also used the planar calibration pattern to solve the extrinsic calibration problem, minimizing the difference in distance from the camera origin to each plane, and minimizing point-to-plane distances to get accurate calibration later.

More recent efforts have improved the automation and robustness of these methods. For instance, [5] leveraged the reflectivity profile of LiDAR data to detect checkerboard patterns automatically. This allowed for precise extraction of 3D corners directly from LiDAR, which were then matched to 2D corners in the image using a PnP algorithm. Similarly, [19] focused on calibrating solid-state LiDARs with non-repetitive scan patterns by temporally accumulating point clouds and refining spatial features, enabling accurate detection of calibration board corners using intensity cues. These approaches eliminated the need for manual intervention and were validated with real-world hardware, showing high accuracy and repeatability, similar to the preceding two methods [20] uses grid search over multiple checkerboard frames in LiDAR frame to get the best fit center of circle followded by minimizing pixel reprojection errors followed by Non-linear optimization.

Other methods have extended the class of usable targets and feature types.[21] moved beyond checkerboards by using polygonal boards or checkerboard edge lines, leveraging features like 3D edges, lines, and planes. These methods formulated calibration as a combination of point-to-line, point-to-plane, and normal-direction alignment constraints. Some approaches, such as [22] and [23], employed ArUco or circle markers to establish 3D–3D correspondences between LiDAR and camera observations, solving the extrinsic transformation using variants of the Iterative Closest Point (ICP) algorithm. More recently, [20] introduced a hybrid calibration board combining checkerboard and circular holes to optimize camera intrinsics and extrinsics in a unified framework jointly. Meanwhile, [24] tackled the challenge of calibrating event cameras with LiDARs using checkerboard detection and a globally optimal solver based on point-to-plane and point-to-edge constraints.

These target-based methods are known for their high precision, with many reporting sub-centimeter translation and rotation errors under one degree. As a result, target-based calibration remains a foundational approach in applications requiring accurate sensor fusion, such as autonomous vehicles, robotics, and 3D mapping, offering an effective balance between accuracy, automation, and practical feasibility.

While target-based methods offer high precision, they often require manual intervention and controlled environments, making them less practical for dynamic scenarios like

autonomous driving.

2.1.2 Targetless Methods

Targetless methods eliminate the need for physical calibration patterns, instead leveraging natural scene features or motion cues. These methods are more scalable and convenient for real-world deployment.

2.1.2.1 Scene based methods

Scene based methods depends upon consistancy between both camera scene and LiDAR scene where features of LiDAR scene are projected onto camera scenes and their consistancy is measured with some metrices.

Recent research has focused on developing targetless extrinsic calibration methods between LiDAR and camera sensors, eliminating the need for artificial patterns like checkerboards. These approaches generally fall into three main categories: mutual information (MI)-based methods, feature-based methods, and semantic or hybrid learning-based approaches.

There are two major approches towards measuring consistancy : Direct and Indirect

The indirect method starts by detecting feature points, such as edges, from both camera and LiDAR data. It then calculates the reprojection error between the corresponding 2D and 3D points[6] [7]. This method generally achieves good convergence thanks to the use of distinctive features and reliable point matching. However, it relies on the presence of rich geometric details in the environment to extract enough features. Since much of the texture information is ignored in this process, its accuracy tends to be lower than that of the direct method.

In contrast, the direct method compares intensity values from pixels and LiDAR points directly. Due to the significant differences in how LiDAR and cameras capture intensity, directly comparing these values doesn't work well. To address this, metrics based on mutual information (MI) have been developed, which assess the consistency of intensity patterns by looking at their co-occurrence rather than direct differences [8] [9]. Building on this idea, the normalized information distance (NID) metric has been introduced, which adheres to metric space properties and offers improved robustness compared to traditional MI [12].

Early works such as [8] and [9] employed MI and normalized MI to align grayscale camera images with LiDAR reflectivity or surface normal projections. These methods optimize for statistical dependence between the sensor outputs and can work in unstruc-

tured environments. However, they often rely on a strong correlation between LiDAR intensity and image brightness, which may not hold in all scenes. Additionally, they are sensitive to initialization and can be computationally intensive.

Feature-based methods extract geometric features, such as edges or lines, that are observable in both modalities. [25] aligned the Canny edges of the image with depth discontinuity edges of the LiDAR through correlation. Similarly, CamVox [26] used multi-feature edge alignment with coordinate descent for fast calibration, particularly suited for Livox sensors. [6] achieved pixel-level accuracy using a point-to-line optimization between natural edges, although it required high-resolution LiDAR input. [27] introduced CRLF, focusing on long line features in road scenes, solving an analytic perspective-3-line (P3L) problem followed by refinement, though its applicability is limited to structured environments.

Popular targetless calibration approaches, such as *Direct Visual LiDAR Calibration*[12], leverage keypoint-based feature matching. In this approach, a virtual camera image is generated by projecting LiDAR points into the image plane, and classical features such as SIFT are extracted from both the LiDAR-based image and the real camera image. These features are then matched to estimate the extrinsic parameters using robust estimation techniques like RANSAC. While effective in dense and static environments, this method suffers from significant limitations in practical scenarios. LiDAR data can be extremely sparse, making it difficult to extract meaningful keypoints for reliable SIFT feature detection. As a result, the feature correspondence process becomes unreliable, leading to inaccurate or unstable calibration estimates. Since LiDAR captures the scene over a sweep of time rather than a single instant, moving objects and changing geometry degrade the matching quality and violate the static-scene assumption. To address this static scene, we use frame-by-frame geometry extraction for synchronized LiDAR-camera frames, so that instead of looking at the static scene, we can now look into the geometry of local synchronized frames.

Semantic-based methods utilize high-level scene understanding.[7] extended MI concepts to semantic labels, using semantic mutual information to align LiDAR and image segmentations, though this requires pretrained segmentation models. More recently, hybrid methods like [12] combined keypoint matching (via SuperGlue) and MI refinement in a publicly available toolbox, while [28] introduced a zero-training calibration method leveraging the Segment Anything Model (SAM) and geometric consistency.

Overall, scene-based and semantic methods remain more interpretable and adaptable in diverse conditions, while deep learning approaches are evolving toward real-time and self-supervised calibration, albeit with domain-specific constraints.

2.2 Image-to-Plane Homography Estimation

Accurate camera calibration is fundamental for vehicle speed measurement using monocular cameras. Calibration involves estimating both intrinsic (camera's internal characteristics) and extrinsic (camera's position and orientation) parameters, and most importantly, deriving the scene scale which enables converting image measurements into real-world distances.

Several approaches have been explored in the literature for traffic camera calibration, categorized mainly into four types: methods based on lane markings, methods based on vehicle movements, methods using manual measurements, and automatic calibration based on vehicle statistics.

2.2.1 Methods Based on Acquired Lane Markings

Early works leveraged road line markings to estimate camera parameters.

He and Yung [29] proposed using lane markings as a calibration pattern, rectifying images to address perspective distortions, and utilizing shadows of vehicles for ground plane localization. Similarly, Cathey and Dailey [30] estimated vanishing points from detected lane markings and inferred the scale using known stripe lengths. Grammatikopoulos et al. [31] assumed tilt only along the Y-axis, finding vanishing points from lane markings with minimal manual measurements.

You and Zheng [32] combined detection of vanishing points from lane markings and poles or pedestrians to calibrate the system.

Limitations: These methods heavily depend on the presence, visibility, and consistency of lane markings, making them unsuitable in scenarios like construction zones or poorly marked rural roads.

2.2.2 Methods Based on Vehicles' Movement

An alternative direction uses the motion of vehicles themselves for calibration.

Dubská et al. [33] developed a method utilizing two vanishing points detected from feature points and vehicle edges, enabling fully automatic calibration. Schoepflin and Dailey [34] used an activity map of vehicle motion to detect lane boundaries and vanishing points. Filipiak et al. [35] leveraged sequences of detected license plates combined with evolutionary optimization to estimate camera parameters.

Advantages: Unlike lane-based methods, these approaches do not require visible road

markings. However, they typically need observations over a sufficiently large number of passing vehicles to achieve accurate calibration.

2.2.3 Methods Using Manual Measurements

Manual-intervention-based methods explicitly measure certain scene dimensions to calibrate the camera.

Maduro et al. [13] used known angles and line marking lengths for calibration. Nurhadiyatna et al. [36] assumed a zero-pan camera and measured known distances manually. Sina et al. [37] tailored their approach for night conditions using headlights detection and manual angle measurements. Luvizon et al. [14] proposed detecting and tracking license plates, using known real-world sizes for calibration.

Drawbacks: These methods require physically stopping or accessing the road to perform measurements, making them impractical for large-scale deployment and costly in operational settings.

2.2.4 Automatic Calibration Using Vehicle Dimension Statistics

To overcome limitations of manual or scene-dependent methods, Dubská et al. [33] introduced a fully automatic calibration method based on vehicle dimension statistics.

Their approach detects two vanishing points: the first vanishing point (VP1) estimated from tracked vehicle features, and the second vanishing point (VP2) extracted from edges on moving vehicles. Camera parameters are then recovered assuming standard intrinsic properties (e.g., centered principal point, no skew). To obtain the scene scale, known mean dimensions of vehicles (such as average length and width) are used.

This method requires no manual input and can adapt to arbitrary viewpoints, a major advantage over previous works.

2.2.5 Comparative Analysis of Methods

A summarized comparison of calibration methods (as presented by Sochor et al. [2]) shows that only a few methods are fully automatic and independent of viewpoint constraints notably Dubská et al. [33].

Other methods either depend on assumptions like zero pan angle [36, 38] or require manual scale measurements [13, 14].

Furthermore, scale inference varies among methods: some rely on known scene measurements [30, 29], while others leverage vehicle statistics [33].

In real-world deployment, automation and robustness to varying viewpoints are critical, thus favoring methods like Dubská et al.'s.

2.2.6 Existing Datasets for Evaluation

Most previous visual speed measurement works were evaluated on small, private datasets with limited vehicle counts, often relying on imprecise ground truth like induction loops [34], GPS [33], or speedometers [38].

Sochor et al. [2] addressed this critical gap by providing a large-scale, publicly available dataset with 20,865 vehicles annotated with high-accuracy speed ground truth obtained via LIDAR optical gates, significantly enhancing the reliability and scope of evaluation for future algorithms.

Chapter 3

Homography-based Distance Estimation

In monocular vehicle speed estimation, homography-based distance estimation provides a practical alternative to sensor-dependent approaches. By computing a homography matrix that maps the ground plane to the image plane, it becomes possible to estimate real-world distances using only camera images. This transformation enables tracking of vehicle displacement across consecutive frames, and when combined with temporal data, facilitates accurate speed computation. The approach eliminates the need for depth sensors and is well-suited for real-time traffic monitoring in resource-constrained settings.

3.1 Mathematical Preliminaries

Definition 1 (Homography Estimation Problem)**.** *[39] A* ***homography*** *is a projective transformation that maps points from one planar surface to their corresponding points in another planar surface, typically from two different frames of reference.*

Given a set of $n \geq 4$ *corresponding 2D points between two views or planes:*

$$(x_i, y_i) \leftrightarrow (x_i', y_i'), \quad i = 1, 2, \ldots, n,$$

the objective is to find a homography matrix

$$H \in \mathbb{R}^{3\times 3}, \quad \textit{with } \det(H) \neq 0,$$

such that the following projective relation holds:

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix},$$

where $\sim$ denotes equality up to a non-zero scale factor (i.e., in homogeneous coordinates).

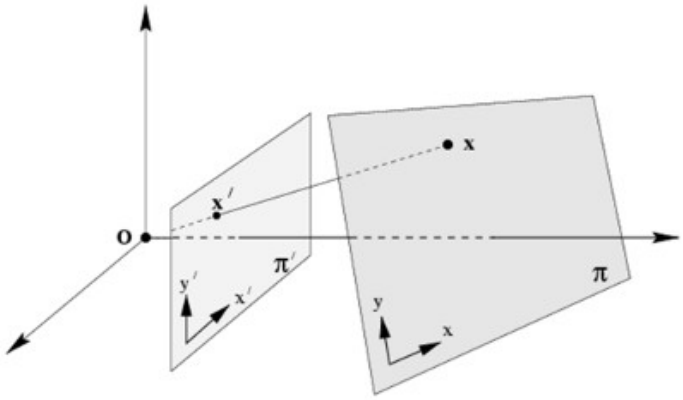


Figure 3.1: *Illustration of homography mapping between planar views.*

Definition 2 (EstimateHomography($\mathbf{p}^n_{\text{src}}, \mathbf{p}^n_{\text{dst}}, \tau, \eta$))**.** *[40] Let $\mathbf{p}^n_{src} = \{(x_i, y_i)\}$ and $\mathbf{p}^n_{dst} = \{(x_i', y_i')\}$ be n putative correspondences. The goal is to estimate the best homography H that aligns the points while minimizing geometric error under a robust estimation framework.*

Objective (RANSAC): *Iteratively select minimal subsets to compute H using DLT (Direct Linear Transform), and minimize the reprojection error for inliers with threshold τ:*

$$\left\| \begin{bmatrix} x_i' \\ y_i' \end{bmatrix} - proj\left(H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \right) \right\|^2 < \tau,$$

where proj$(\cdot)$ denotes conversion from homogeneous to Cartesian coordinates by normalizing with the third component.

The RANSAC algorithm repeats this process until a solution is found with confidence at least η, ensuring robustness to outliers. The final solution is obtained by running DLT on RANSAC inliers.

Solution via Direct Linear Transform (DLT)[39][41]

The goal of homography estimation is to find a projective transformation matrix $H \in \mathbb{R}^{3\times3}$ such that for a point $\mathbf{x} = [x, y, 1]^\top$ in the source image and its corresponding point

$\mathbf{x}' = [x', y', 1]^\top$ in the destination image, the relation

$$\mathbf{x}' \sim H\mathbf{x}$$

holds, where $\sim$ denotes equality up to scale due to the use of homogeneous coordinates.

To eliminate the unknown scale factor, we use the cross product constraint:

$$\mathbf{x}' \times (H\mathbf{x}) = 0,$$

which leads to a set of linear equations. For each point correspondence $(\mathbf{x}, \mathbf{x}')$, this yields two linearly independent equations of the form:

$$\begin{bmatrix} 0 & 0 & 0 & -wx & -wy & -w & y'x & y'y & y' \\ wx & wy & w & 0 & 0 & 0 & -x'x & -x'y & -x' \end{bmatrix} \mathbf{h} = 0,$$

where $\mathbf{h}$ is a 9×1 vector representing the elements of H:

$$\mathbf{h} = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33}]^\top.$$

Stacking these equations for $n \geq 4$ point correspondences results in a homogeneous system:

$$A\mathbf{h} = 0,$$

with $A \in \mathbb{R}^{2n \times 9}$. The trivial solution $\mathbf{h} = 0$ is not meaningful, so we solve for $\mathbf{h}$ subject to the constraint $\|\mathbf{h}\| = 1$. This is achieved by computing the singular value decomposition (SVD) of A:

$$A = U\Sigma V^\top,$$

and selecting $\mathbf{h}$ as the last column of V, corresponding to the smallest singular value.

Finally, $\mathbf{h}$ is reshaped into the 3×3 homography matrix H, and normalized such that $h_{33} = 1$. In practice, the DLT estimate is refined by applying RANSAC to identify inliers and recomputing H using only the consistent correspondences.

3.2 Problem Statement: Metric Distance Estimation using Homography

The core objective of this work is to estimate real-world distances between two points observed in a monocular camera frame, specifically in road scenes captured by a static, intrinsically and extrinsically calibrated camera. Mapping 2D pixel coordinates to 3D

metric coordinates on the road plane enables various downstream applications such as vehicle localization, motion analysis, and speed estimation. However, the primary emphasis here is on the accurate recovery of metric distances from image-space measurements.

As illustrated in Figure 3.2, a point on the image plane $\mathbf{P}_{\text{src}} = (u, v, 1)^T$ corresponds to a physical point $\mathbf{P}_{\text{dst}} = (X, Y, Z)^T$ in Euclidean space, located on the road surface. Assuming that the road can be approximated as a horizontal plane (i.e., $Z = 0$), the transformation between image coordinates and world coordinates on this plane is governed by a homography matrix $\mathbf{H} \in \mathbb{R}^{3\times3}$, which encapsulates the projective geometry induced by the camera-road configuration.

Given the homography $\mathbf{H}$, the mapping from image to world coordinates (up to scale) is expressed as:

$$\lambda \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad \lambda \neq 0$$

This transformation allows each point in the image to be projected onto the ground plane, yielding its corresponding metric coordinates in Euclidean space. Consequently, the distance between two such points can be directly computed as:

$$D = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

This metric distance can be combined with known frame intervals to estimate vehicle speed.

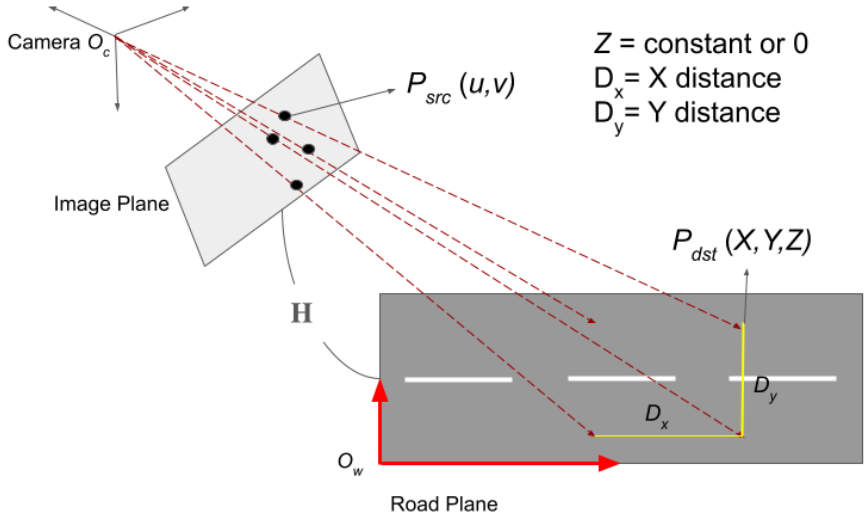


Figure 3.2: Monocular Homography estimation problem

3.3 Proposed Solution

The proposed approach aims to estimate a robust *image-to-road homography* matrix $\mathbf{H}_{\text{final}}$ that maps 2D road points from the image plane to a canonical road plane in real-world metric coordinates. This homography allows consistent and low-computation *metric distance estimation* across frames, which is then used to compute vehicle speed. The block diagram in Figure 3.3 illustrates the full pipeline, while Figure 3.2 demonstrates the geometric mapping enabled by homography.

3.3.1 Motivation

A natural question arises: *Why not use per-frame depth estimation to compute distances?*

While depth provides direct 3D information, using it for per-frame distance and speed estimation is computationally expensive. Each frame would require:

- Dense depth inference,
- 3D reconstruction,
- Metric distance calculation, and
- Error correction for depth inaccuracies.

In contrast, the proposed method uses depth and semantic segmentation *only once* to compute a homography between the road in the image and a canonical ground plane. Once estimated, this homography enables *lightweight, frame-wise 2D distance computation* in the metric domain by simply projecting pixel coordinates through a precomputed transformation. Thus, we significantly reduce both computational load and sensitivity to temporal depth noise.

3.3.2 Pipeline Overview

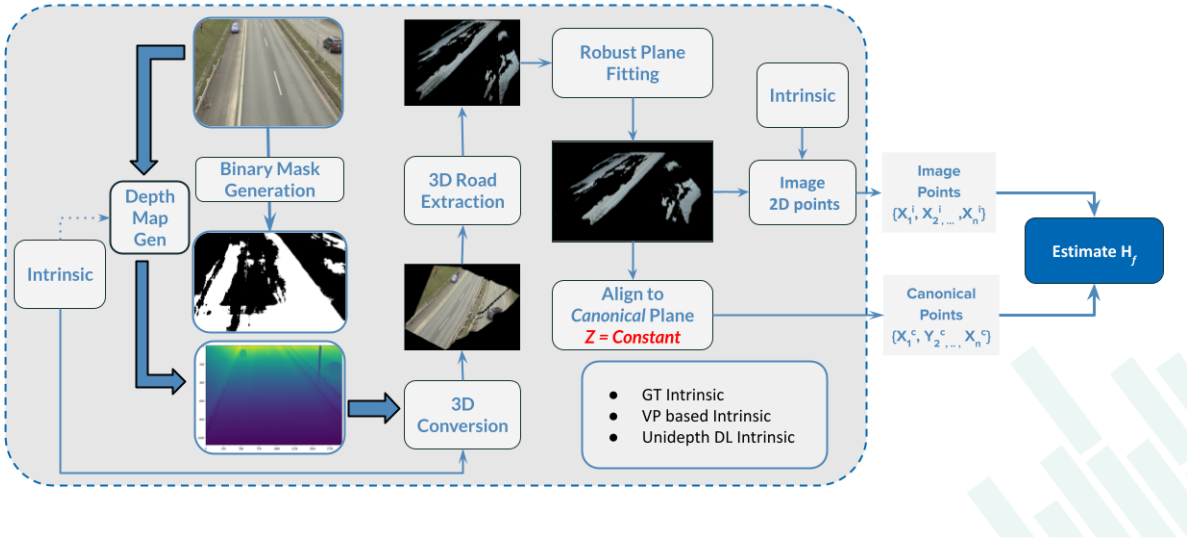


Figure 3.3: Block diagram of the proposed homography estimation pipeline.

The proposed method consists of the following main stages:

3.3.2.1 Depth Estimation and Road Segmentation

To extract meaningful 3D information from a single RGB image, we require both geometric depth and semantic understanding of the road area. This helps isolate only those pixels that belong to the drivable road surface, avoiding interference from background objects.

Given an input RGB image and intrinsic matrix $\mathbf{K}$, two parallel branches are used:

- A pre-trained *depth estimation model* predicts the depth map of the scene.
- A *semantic segmentation model* generates a binary road mask.

3.3.2.2 3D Road Point Cloud Construction

To analyze the geometry of the road in metric space, the depth map must be unprojected into 3D, and only the road surface needs to be retained to avoid noise.

Using the intrinsic matrix $\mathbf{K}$ and the predicted depth, a dense 3D point cloud is constructed. The binary mask is applied to retain only road points, resulting in a filtered set of 3D road points $\mathbf{X} = [X, Y, Z]^\top$ in the camera coordinate frame.

3.3.2.3 Robust Plane Fitting using RANSAC, Canonical Plane and Correspondence Construction

As depth predictions may be noisy and real roads are approximately planar, a robust method is needed to extract a clean dominant road plane. Furthermore, establishing a known reference plane is essential for metric reasoning.

RANSAC is applied to the 3D road points to estimate a dominant ground plane. Inlier points are retained based on a geometric distance threshold. The corresponding 2D image coordinates $\mathbf{P}_{\text{src}}$ for these inlier points are computed using camera intrinsics. A canonical metric plane (e.g., $Z = 0$) is defined and a set of ideal 2D points $\mathbf{P}_{\text{dst}}$ is constructed to represent real-world positions.

3.3.2.4 Normalization and Homography Estimation

To improve numerical stability and reduce the effect of scale variation in point coordinates, normalization is applied before estimating the transformation.

The source and destination points are normalized:

$$\mathbf{P}_{\text{src-norm}} = \mathbf{T}_{\text{src}} \cdot \mathbf{P}_{\text{src}}, \quad \mathbf{P}_{\text{dst-norm}} = \mathbf{T}_{\text{dst}} \cdot \mathbf{P}_{\text{dst}}$$

A robust homography $\mathbf{H}_n$ is computed using RANSAC. The final denormalized homography is:

$$\mathbf{H}_{\text{final}} = \mathbf{T}_{\text{dst}}^{-1} \cdot \mathbf{H}_n \cdot \mathbf{T}_{\text{src}}$$

3.3.2.5 Real-World Distance Estimation

To estimate vehicle speed or displacement in metric units, pixel coordinates need to be converted to real-world coordinates on the road surface.

Using $\mathbf{H}_{\text{final}}$, any road image point $\mathbf{P}_{\text{src}}$ can be mapped to its real-world coordinate $\mathbf{P}_{\text{dst}}$:

$$\mathbf{P}_{\text{dst}} = \mathbf{H}_{\text{final}} \cdot \mathbf{P}_{\text{src}}$$

This enables accurate and consistent computation of displacements

3.4 Experimental Setup

In this section, we detail the experimental pipeline used for depth-based road scene analysis and LiDAR-camera calibration. We evaluate multiple monocular depth estimation

models, Metric3D, UniDepth, and ZoeDepth, each offering distinct capabilities with respect to metric scale estimation and dependency on camera intrinsics. For semantic segmentation, we employ DeepLabv3+ to obtain binary road masks, enabling accurate geometric isolation of the drivable surface.

To support depth-based processing, we explore three methods for intrinsic parameter estimation: ground-truth usage, vanishing-point-based geometric inference, and intrinsic prediction via UniDepth. Additionally, we describe our robust homography estimation algorithm, which combines depth, segmentation, and plane fitting techniques to establish metric mappings from the image to the canonical ground plane.

This experimental setup forms the backbone for evaluating real-world scale consistency and sensor calibration accuracy in traffic surveillance applications.

3.4.1 Depth Estimation Models

We utilize a range of monocular depth estimation models to infer metric depth from single RGB images. These models vary in their reliance on camera intrinsics, architecture, and target outputs, and are selected based on their suitability for geometric reasoning in traffic scenes.

3.4.1.1 Metric3D Depth Estimation using Canonical Camera[42]

Metric3D estimates depth by first transforming the input image into a canonical camera space, standardizing the geometry across scenes. The overall pipeline is as follows:

$$\texttt{Input Image } I \rightarrow \texttt{Canonical Space Transformation} \rightarrow \texttt{Transformed Image } I_c \rightarrow \texttt{Depth Network} \rightarrow \texttt{Predicted Depth } D_c \rightarrow \texttt{De-Canonical Transformation} \rightarrow \texttt{Final Depth } D$$

During training, both the input image and ground-truth depth are mapped into this canonical space using known camera intrinsics. The model predicts depth in this canonical space, which is then transformed back to the original image coordinate frame during inference.

The focal length is a key component of the intrinsic matrix of the camera and plays a critical role in the estimation of the monocular depth. Two images with similar appearance can correspond to vastly different physical distances if captured using different focal lengths. Without accurate focal length, the problem of inferring metric depth from a single image becomes ill-posed, as the scale of projection cannot be recovered reliably.

Hence, precise knowledge of intrinsic parameters, particularly focal length, is essential to ensure accurate metric depth prediction.

3.4.1.2 Unidepth Depth Estimation Module[16]

UniDepth is a transformer-based monocular depth estimation framework that jointly predicts metric depth and camera pose from a single RGB image. It leverages cross-attention and self-attention mechanisms to extract shared feature embeddings, enabling consistent depth and camera estimation without requiring camera intrinsics or additional supervision.

3.4.1.3 3D PointCloud using ZoeDepth[43]

ZoeDepth estimates a 3D point cloud from a single image by first predicting a depth map and then backprojecting the depth values into 3D space using a pinhole camera model. The intrinsic matrix is estimated based on the image resolution and a fixed field of view (FOV) of 55°.

Given an image of width W and height H, the focal length f is computed as:

$$f = \frac{0.5 \times W}{\tan(0.5 \times 55^\circ)}$$

Assuming the principal point lies at the image center:

$$c_x = 0.5 \times W, \quad c_y = 0.5 \times H$$

3.4.2 Semantic Segmentation Module: DeepLabv3+[1]

To obtain a binary mask for the road, we use *DeepLabv3+*, a state-of-the-art semantic segmentation model. It combines **Atrous Spatial Pyramid Pooling (ASPP)** to capture multi-scale context with a **decoder module** that refines segmentation boundaries using low-level features. This enables accurate identification of road regions in complex scenes. In our pipeline, DeepLabv3+ is used to extract a **binary road mask** from each input image, effectively isolating drivable areas for subsequent geometric processing.

3.4.3 Intrinsic Estimation

Accurate intrinsic parameters, particularly the focal length f, are essential for obtaining metric-consistent depth from monocular images. Many depth estimation frameworks, such as Metric3D [42], rely on these parameters to correctly scale the predicted depth. If incorrect intrinsics are provided, the resulting depth will also be erroneous. We explore three strategies to obtain intrinsics under different assumptions.

3.4.3.1 Ground-Truth Intrinsics (If Available)

If ground-truth camera intrinsics are available (e.g., from datasets like nuScenes or KITTI), they can be directly used for metric depth estimation without requiring Intrinsic approximation.

3.4.3.2 Joint Intrinsic and Depth Estimation via UniDepth

When ground-truth intrinsics are unavailable, we can use the UniDepth framework [16], which jointly estimates depth and camera intrinsics from monocular images. It leverages multi-view geometric consistency and does not require any calibration targets or ground-truth depth. This approach is highly effective in uncalibrated or large-scale settings.

3.4.3.3 Vanishing Point-Based Intrinsic Estimation

We also implement a geometric method for estimating focal length based on vanishing points (VPs). The method assumes a pinhole camera model with:

- Squared pixels ($f_x = f_y$)
- Centered principal point ($c_x = w/2$, $c_y = h/2$)

The pipeline is illustrated in Figure 3.4 and summarized as follows:

3.4.3.3.1 Focal Length Estimation Using Orthogonal Vanishing Points Chapter 5[39][44]
We use progressive-x[17] multi-modal fitting to get the Vanishing points. Given two vanishing points $\mathbf{v}_1$ and $\mathbf{v}_2$ corresponding to orthogonal directions in 3D space, the orthogonality constraint on the Image of the Absolute Conic (IAC), denoted by $\boldsymbol{\omega}$, is used:

$$\mathbf{v}_1^\top \boldsymbol{\omega} \mathbf{v}_2 = 0 \tag{3.1}$$

Assuming the camera has no skew, square pixels, and the principal point is at the image center (c_x, c_y), the intrinsic matrix $\mathbf{K}$ is defined as:

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

The Image of the Absolute Conic is given by:

$$\boldsymbol{\omega} = (\mathbf{K}\mathbf{K}^\top)^{-1} \tag{3.3}$$

Substituting this into the orthogonality condition yields the following constraint for focal length:

$$f^2 = -(v_{1x} - c_x)(v_{2x} - c_x) - (v_{1y} - c_y)(v_{2y} - c_y) \tag{3.4}$$

This relation is valid only if the vanishing points correspond to orthogonal directions in 3D. To ensure this, we compute the angle θ between the vectors $\mathbf{v}_1 - \mathbf{c}$ and $\mathbf{v}_2 - \mathbf{c}$, where $\mathbf{c} = [c_x, c_y, 1]^\top$ is the image center. We accept the pair as orthogonal if:

$$\theta \in [60^\circ, 150^\circ] \tag{3.5}$$

This constraint helps eliminate noisy or nearly parallel vanishing point pairs that do not provide reliable information about the focal length.

The algorithm1 estimates the camera's focal length by leveraging vanishing points detected across multiple frames in a video sequence. For each selected frame, line segments are extracted using a Line Segment Detector (LSD)[45], and vanishing points are computed using the Progressive-X estimator, which robustly fits multiple dominant directions.

Pairs of vanishing points that form approximately orthogonal directions (i.e., angles between 60° and 150° relative to the image center) are identified. For each such pair, the focal length is computed using the orthogonality constraint on the Image of the Absolute Conic (IAC). Repeating this process across frames yields a set of focal length estimates.

Using the described algorithm, we obtain a set of focal length estimates by applying the orthogonality constraint to multiple vanishing point pairs extracted across sampled video frames. Since vanishing point detection can be sensitive to noise and outliers in the line segments, some of the estimated focal lengths may deviate significantly from the true value.

To enhance the robustness of the final focal length estimation, we perform statistical refinement using the Interquartile Range (IQR) method. Specifically, we compute the first quartile Q_1 and the third quartile Q_3 of the focal length distribution and discard any values lying outside the range:

$$[Q_1 - 1.5 \times \text{IQR},\ Q_3 + 1.5 \times \text{IQR}], \tag{3.6}$$

where $\text{IQR} = Q_3 - Q_1$ is the interquartile range. This filtering step effectively removes statistical outliers while preserving the central tendency of the distribution.

The refined focal length f is then computed as the arithmetic mean of the remaining inlier values:

$$f = \frac{1}{N} \sum_{i=1}^{N} f_i, \tag{3.7}$$

where f_i are the inlier focal lengths after IQR filtering, and N is the number of inliers. This statistically robust estimate of f is subsequently used as the intrinsic focal length in further computations.

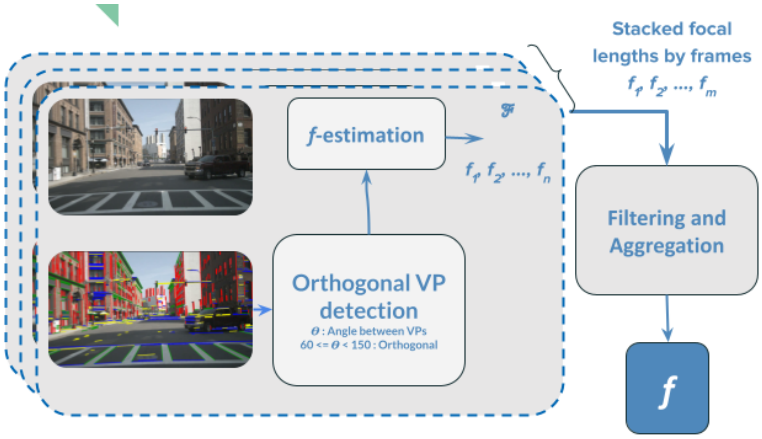


Figure 3.4: Pipeline for focal length estimation from vanishing points.

3.4.4 Algorithm

This Algorithm2 estimates a planar homography between road regions in an image and a canonical ground plane using monocular depth and semantic segmentation. It begins by predicting a dense depth map using a pretrained model and extracting a road mask using a segmentation model. These are combined to reconstruct a partial 3D point cloud corresponding to road surface geometry.

Algorithm 1 Intrinsic Estimation using Orthogonal Vanishing Points from n Frames[44]

Input: Video file path $\mathcal{V}$, frame skip interval n, line detector LSD, VP estimator ProgressiveX
Output: List of estimated focal lengths $\mathcal{F}$

$\mathcal{F} \leftarrow \emptyset$
for each frame I_k at index k where $k \mod n = 0$ **do**
  Convert to grayscale: $G \leftarrow \textsc{Grayscale}(I_k)$
  Detect line segments: $\mathcal{L} \leftarrow \text{LSD}(G)$
  Filter segments shorter than threshold
  Normalize line weights
  Estimate vanishing points: $\{\mathbf{v}_i\} \leftarrow \textsc{ProgressiveX}(\mathcal{L})$
  Normalize $\mathbf{v}_i$ to homogeneous coordinates
  Compute image center: $\mathbf{c} = [w/2, h/2, 1]$
  for all pairs $(\mathbf{v}_i, \mathbf{v}_j)$ **do**
    Compute angle $\theta = \angle(\mathbf{v}_i - \mathbf{c}, \mathbf{v}_j - \mathbf{c})$
    if $60^\circ \leq \theta \leq 150^\circ$ **then**
      Estimate focal length:

$$f = \sqrt{-(v_{ix} - c_x)(v_{jx} - c_x) - (v_{iy} - c_y)(v_{jy} - c_y)}$$

      if f is valid **then**
        Append f to $\mathcal{F}$
      end if
    end if
  end for
end for
return $\mathcal{F}$

Next, RANSAC is used to robustly fit a dominant ground plane to the segmented road points. The inlier 3D points are projected back to 2D image coordinates (source points), while the same 3D points are also aligned to a canonical plane (destination points). After normalization, a robust homography is estimated between the point sets and subsequently denormalized to yield a final 3×3 homography matrix that maps road image pixels to a fronto-parallel ground reference frame.

Algorithm 2 Homography Estimation from Image and Intrinsics

Input: RGB image I, camera intrinsics K, depth model $\mathcal{D}$, segmentation model $\mathcal{S}$, transform $\mathcal{T}$
Output: Homography matrix $\mathbf{H}$

Step 1: Depth and Segmentation
    Estimate depth map: $\mathbf{D} \leftarrow \textsc{EstimateDepth}(\mathcal{D}, I, K)$
    Generate road segmentation mask: $\mathbf{M} \leftarrow \textsc{GetRoadMask}(\mathcal{S}, I, \mathcal{T})$

Step 2: 3D Point Cloud and Road Extraction
    Reconstruct 3D point cloud: $\mathbf{P} \leftarrow \textsc{BackprojectPoints}(\mathbf{D}, K)$
    Extract road points: $\mathbf{P}_{\text{road}} \leftarrow \mathbf{P}[\mathbf{M} == 1]$

Step 3: Plane Fitting and Correspondences
    Fit dominant ground plane: $\pi \leftarrow \textsc{FitPlaneRANSAC}(\mathbf{P}_{\text{road}})$
    Project inliers to 2D: $\mathbf{p}_{\text{src}} \leftarrow \textsc{ProjectToImage}(\mathbf{P}_{\text{road}}, K)$
    Align 3D plane points to canonical plane: $(\mathbf{p}_{\text{dst}}, R, \mathbf{P}_{\text{dst}}) \leftarrow \textsc{AlignToPlaneNormal}(\mathbf{P}_{\text{road}}, \pi)$

Step 4: Normalization and Homography Computation
    Normalize source points: $(T_s, \mathbf{p}^n_{\text{src}}) \leftarrow \textsc{NormalizePoints}(\mathbf{p}_{\text{src}})$
    Normalize destination points: $(T_d, \mathbf{p}^n_{\text{dst}}) \leftarrow \textsc{NormalizePoints}(\mathbf{p}_{\text{dst}})$
    Estimate normalized homography:

$$\mathbf{H}_n \leftarrow \textsc{EstimateHomography}(\mathbf{p}^n_{\text{src}}, \mathbf{p}^n_{\text{dst}}, \tau, \eta)2$$

    Denormalize:

$$\mathbf{H} \leftarrow T_d^{-1} \cdot \mathbf{H}_n \cdot T_s$$

return H

3.5 Dataset Description and Challenges

3.5.1 BrnoCompSpeed Dataset[2]

Figure3.5 represents sample images for BrnoCompSpeed Dataset. The BrnoCompSpeed dataset is designed specifically for the task of vehicle speed estimation using monocular camera inputs. It consists of 18 high-resolution video sequences captured across six different sessions, where each session includes three synchronized views: *left*, *center*, and *right*.

Table 3.1: BrnoCompSpeed Dataset Characteristics

Property	**Description**
No. of Video Sequences	18 videos across 6 sessions (3 views per session)
Camera Calibration	✓ Vanishing Point-based (Intrinsic with assumptions)
Ground Truth Speeds	✓ LiDAR sensor
Ground Truth Depth	✗ Not available
Multi-view Setup	✓ Left, Center, Right
Planar Road Assumption	✓ Mostly flat roads
Traffic Complexity	Moderate (urban traffic)
Environmental Challenges	Minimal
Use Case	Academic benchmarking for speed estimation and Distance estimation

This dataset is well-suited for testing homography estimation pipelines that require high-fidelity pixel-to-world distance mapping using only monocular camera data.

(a) Center View (b) Left View (c) Right view

Figure 3.5: Samples frames BrnoCompSpeed

3.5.2 NuScenes Dataset[3] and IIITD Dataset

Figure3.6 represents sample images for Nuscene Dataset. The nuScenes dataset, developed by Motional, is a large-scale, multimodal dataset primarily designed for autonomous driving research. Where the camera and LiDAR are mounted on the car's top. For

homography-based road-to-image distance estimation, nuScenes provides several critical advantages:

Table 3.2: nuScenes Dataset Characteristics

Property	**Description**
No. of Scenes	1000 annotated scenes with synchronized sensors
Camera Calibration	✓ Full intrinsics and extrinsics
Ground Truth Speeds	✗ Not explicitly available
Ground Truth Depth	✓ Provided via LiDAR
Multi-view Setup	✓ 6 cameras (360° view)
Planar Road Assumption	✓ Mostly flat roads
Traffic Complexity	Structured (autonomous driving setting)
Environmental Challenges	Moderate (urban, varying light)
Use Case	Validation of monocular-to-metric 3D estimation

Although the full *nuScenes* dataset contains 1,000 scenes, for the purposes of homography-based ground-to-image transformation, we select a subset of 20 representative scenes. These were carefully chosen to cover diverse road geometries and lighting conditions while ensuring sufficient ground-plane visibility.

Each *nuScenes* scene contains approximately 400 images captured from the six cameras at 2 Hz. However, in our experiments:

We restrict analysis to the front-facing camera (`CAM_FRONT`) to focus on forward road-view scenes relevant for vehicle speed and road geometry estimation.

From each selected scene, we extract a consistent subset of 100 frames, evenly spaced across the scene's timeline, for image-LiDAR correspondence generation and homography fitting.

This controlled sampling ensures balanced spatial and temporal representation while keeping computational requirements tractable.

Summary of Usage

- Total scenes used: 20 out of 1,000
- Images per scene considered: 100 out of ∼400
- Camera used: Only `CAM_FRONT`
- Use case: Ground-to-image homography estimation using road-surface LiDAR projections and calibrated intrinsics/extrinsics

In summary, the nuScenes dataset enables homography estimation using precise 3D data, making it highly suitable for validating monocular vision-based distance estimation approaches against accurate ground-truth depth.

Figure 3.6: Samples frames Nuscene

Figure3.7 represents sample images for IIIT Dataset.

Table 3.3: IIITD Dataset Characteristics

Property	**Description**
No. of Scenes	734 frames (considered as scenes)
Camera Calibration	✓ Full intrinsics and extrinsics
Ground Truth Speeds	✗ Not explicitly available
Ground Truth Depth	✓ Provided via LiDAR
Multi-view Setup	✗ Single-view only
Traffic Complexity	Very low to moderate (inside IIITD campus)
Environmental Challenges	Moderate (urban, varying lighting)
Use Case	Validation of monocular-to-metric 3D estimation

Figure 3.7: Samples frames IIITD Dataset

3.5.3 Vehant Dataset

Figure3.8 represents sample images for Vehant Dataset. The Vehant dataset comprises multiple video sequences captured on Indian roads, offering realistic and challenging scenarios for vehicle speed estimation. It is particularly valuable for evaluating homography-based methods in non-ideal conditions.

Table 3.4: Vehant Dataset Characteristics

Property	**Description**
No. of Video Sequences	Multiple real-world traffic videos (India)
Camera Calibration	✗ Not provided
Ground Truth Speeds	✗ Not available
Ground Truth Depth	✗ Not available
Multi-view Setup	✗ Single-view recordings
Planar Road Assumption	✗ Mostly flat roads
Traffic Complexity	High (dense, unstructured, varied vehicles)
Environmental Challenges	High (occlusion, lighting, weather variation)
Use Case	Deployment-oriented evaluation in non-ideal settings

Figure 3.8: Samples frames Vehant Dataset

3.6 Empirical Evaluation and Results

3.6.1 Evalution Metrics

The evaluation metric used in this study is the *Relative Distance Percentage Error (RDPE)* Euclidean distance. This metric assesses the accuracy of estimated distances with respect to their corresponding ground truth distances.

Let the ground truth distance be denoted by d_{gt} and the predicted distance be d_{pred}. Then, the relative percentage error is computed as:

$$\text{RDPE} = \left(\frac{d_{\text{pred}}}{d_{\text{gt}}} \right) \times 100\%$$

- Down-road: Distance measured along the road.
- Cross-road: Distance measured accross the road.

3.6.2 Quantitative Results

Tables 3.5, 3.6, and 3.7 present a comprehensive evaluation of distance errors resulting from homography estimation across four datasets: BrnoCompSpeed, Vehant, NuScenes,

and IIITD. These evaluations compare different strategies for estimating camera intrinsic namely, ground truth intrinsics (GT Int), Unidepth-predicted intrinsics, and vanishing point (VP)-based approximated intrinsics.

From Table 3.5, we observe that the lowest errors are achieved when ground truth intrinsics are available, validating the importance of accurate calibration. However, in scenarios where GT intrinsics are not available, our proposed method using vanishing points for intrinsic estimation performs competitively, achieving lower mean errors (Down-road: 16.56, Cross-road: 18.96) compared to the Unidepth-based intrinsic estimation (Down-road: 30.57, Cross-road: 21.97). Additionally, while Unidepth-based intrinsics provide consistent performance, the VP-based approximation occasionally results in significantly lower errors for specific scenes, though with higher variance across others.

For the Vehant dataset (Table 3.6), where GT intrinsics are unavailable, both Unidepth and VP-based intrinsics show comparable average errors (20.23 and 22.70, respectively). However, variability is evident in certain scenes that exhibit very low errors (e.g., Video 2), while others (e.g., Video 8) have errors exceeding 40%. This highlights that the accuracy of VP-based intrinsics heavily depends on the quality of vanishing point detection and the structure of the scene.

Table 3.7 shows a slightly different trend: for NuScenes, the lowest error is obtained using Unidepth-based intrinsics (10.78 vs. 13.26 for GT), suggesting favorable alignment between Unidepth's predictions and the dataset's camera parameters. In contrast, for the IIITD dataset, the GT intrinsics yield better accuracy (11.23 vs. 12.73 for Unidepth).

In summary, while ground truth intrinsics consistently provide the best performance, our proposed VP-based intrinsic estimation method serves as a viable alternative in their absence, often outperforming generic learning-based intrinsic predictions. The results further suggest that scene structure and calibration quality significantly influence the final homography accuracy. A more detailed error analysis is presented in the subsequent sections to explain the observed trends.

Table 3.5: BrnoCompSpeed Distance Error for Homography

Session	**Alignment**	**Metric3D + GT Int ↓**		**Metric3D + Unidepth Int ↓**		**Metric3D + VP Int ↓**		**Unidepth Depth + Unidepth Int ↓**		**3D : ZoeDepth ↓**	
		Down-road	Cross-road	Down-road	Cross-road	Down-road	Cross-road	Down-road	Cross-road	Down-road	Cross-road
Session 1	Center	34.90	35.28	27.93	34.86	12.83	35.60	26.66	10.07	52.27	57.87
	Left	18.30	3.18	33.96	16.80	27.73	14.22	34.85	21.05	73.08	63.13
	Right	15.36	5.56	35.18	33.68	27.53	14.76	23.32	19.07	39.33	46.40
Session 2	Center	5.46	18.25	33.25	22.82	28.84	17.49	33.25	22.83	59.80	47.17
	Left	9.15	18.38	31.54	19.17	32.17	18.20	32.63	15.63	55.74	50.29
	Right	8.67	20.63	20.61	19.89	15.51	21.78	30.94	31.78	50.96	39.10
Session 3	Center	13.10	22.29	25.97	21.10	6.57	18.54	30.04	23.95	47.71	38.34
	Left	4.69	16.31	31.29	21.25	24.48	20.72	29.80	18.24	74.18	68.43
	Right	18.85	21.19	34.29	21.48	7.71	14.95	29.42	21.47	45.04	45.07
Session 4	Center	3.83	17.92	37.09	18.82	11.49	18.14	44.08	24.12	61.10	44.67
	Left	4.63	15.36	32.76	19.43	4.63	15.30	37.95	22.06	60.28	47.62
	Right	2.13	11.34	31.07	22.89	6.87	8.75	31.56	20.17	49.28	43.51
Session 5	Center	11.34	16.66	29.89	28.49	16.87	29.29	26.88	27.40	90.87	89.75
	Left	12.87	15.91	26.67	21.89	37.54	32.94	22.18	16.93	51.60	53.00
	Right	10.68	18.87	27.15	25.18	12.89	20.61	19.18	25.63	43.10	55.08
Session 6	Center	2.12	17.06	46.57	18.03	2.53	17.27	44.98	21.55	73.27	56.39
	Left	3.71	16.55	19.76	10.55	16.81	7.55	6.45	7.17	58.12	52.41
	Right	4.43	15.71	25.21	19.18	5.03	15.23	32.42	23.66	72.68	63.42
Mean		**10.23**	**17.03**	30.57	21.97	**16.56**	**18.96**	29.81	20.71	58.80	53.42

Table 3.6: Distance error comparison for Vehant Data using different intrinsic estimation methods

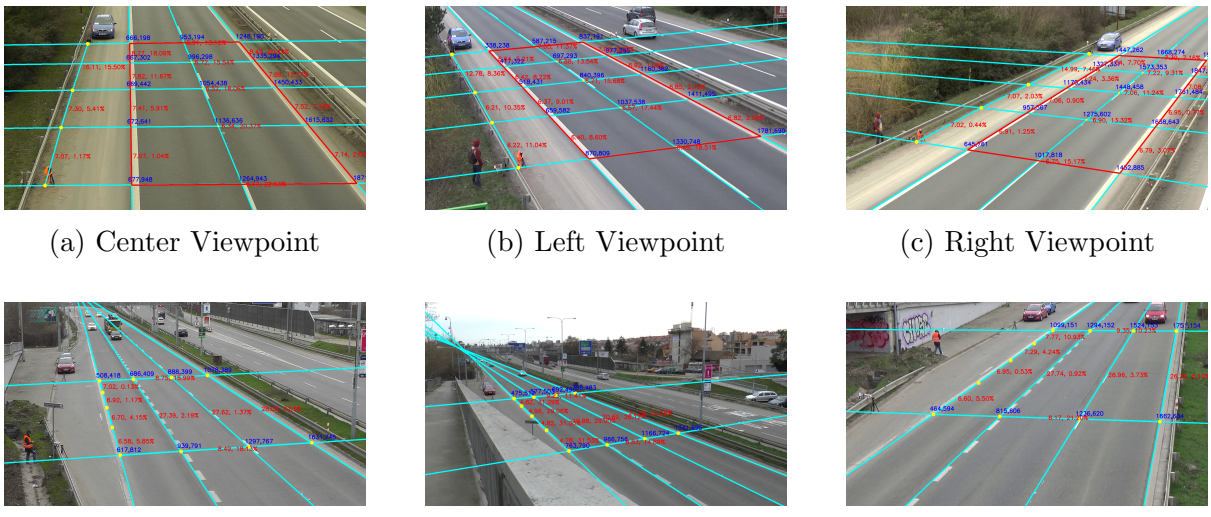
Vehant Video	**Metric3D + Unidepth Int ↓**	**Metric3D + VP Int ↓**
Video 1	29.86	36.52
Video 2	9.17	8.63
Video 3	13.13	11.89
Video 4	22.76	36.48
Video 5	11.26	22.59
Video 6	14.26	12.65
Video 7	16.36	10.88
Video 8	45.03	41.99
Average	**20.23**	**22.70**

Table 3.7: Distance error comparison across Nuscene and IIIT Dataset

Dataset	**Metric3D + GT Int ↓**	**Metric3D + Unidepth Int ↓**
NuScenes	13.26	10.78
IIITD Data	11.23	12.73

3.6.3 Qualitative Results

Figure 3.9 represents qualitative results for the BrnoCompSpeed Dataset.



(a) Center Viewpoint (b) Left Viewpoint (c) Right Viewpoint

(d) Center Viewpoint (e) Left Viewpoint (f) Right Viewpoint

Figure 3.9: Qualitative results for Homography-based distance estimation. (a,b) a is the estimated distance, and b is the percent error.

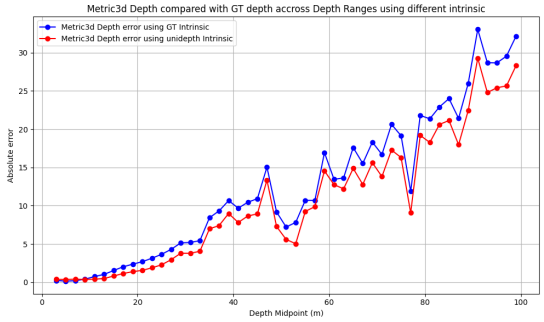
3.6.4 Error Analysis

As observed from Tables 3.5, 3.6, and 3.7, the distance errors are considerably higher when using depth estimation models such as UniDepth and ZoeDepth. This is primarily due to large inaccuracies in the predicted depth maps and the inherent assumptions made by ZoeDepth regarding camera intrinsic parameters such as fixed FoV. Because of these significant depth errors, we will focus the remainder of our analysis solely on results obtained using the Metric3D framework.

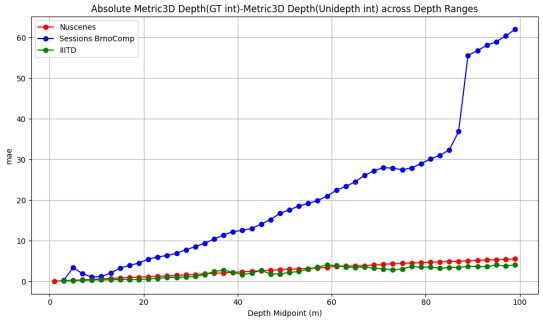
Across all datasets, it is clear that errors are lowest when ground truth (GT) intrinsic parameters are available. In scenarios where GT intrinsics are not provided, vanishing point (VP)-based intrinsic approximation yields lower errors than UniDepth-based intrinsics in the BrnoCompSpeed dataset, and performs comparably in the Vehant dataset. Interestingly, UniDepth-based intrinsics result in relatively lower errors in the NuScenes and IIITD datasets compared to their performance in BrnoCompSpeed and Vehant. This variation is primarily due to the underlying depth estimation errors, which in turn are influenced by the quality of the estimated intrinsics.

Figure 3.10a shows the depth error in the NuScenes and IIITD datasets, using different intrinsic estimation methods. Ground truth depth is derived from LiDAR, and we observe that the error increases with the distance from the camera. Figure 3.10b illustrates the absolute error when depth estimation using Metric3D + GT intrinsics is compared against depth estimation using Metric3D + UniDepth-based intrinsics. The BrnoCompSpeed dataset shows significantly higher errors when UniDepth intrinsics are used, whereas the NuScenes and IIITD datasets maintain lower errors.

This discrepancy is attributed to the larger deviation between the predicted UniDepth intrinsic and the actual GT intrinsic of the BrnoCompSpeed dataset, compared to the relatively smaller deviations found in the NuScenes and IIITD datasets, as shown in Table 3.8. A key reason for this deviation is the domain shift in data distribution. As illustrated in Figure3.11, the BrnoCompSpeed dataset exhibits substantial differences in the layout of the scene and the orientation of the viewpoint compared to the Nuscene and IIITD datasets, where the horizon is clearly visible. Specifically, BrnoCompSpeed includes a larger proportion of oblique road views, more extreme vanishing point orientations, and focal length characteristics that deviate from the typical patterns seen in standard driving datasets. These out-of-distribution characteristics cause UniDepth to generalize poorly, leading to erroneous intrinsic predictions. As depth errors compound through the homography pipeline, they ultimately manifest as large distance estimation inaccuracies on the ground plane.



(a) Metric3d Depth compared with GT depth across Depth Ranges using different intrinsic on Nuscene and IIITD Datasets



(b) Absolute error: Metric3D Depth(GT int) comapared with Metric3D Depth(Unidepth int) across Depth Ranges

Table 3.8: Unidepth predicted Intrinsic error (pixels)

Unidpeth Intrinsic Evaluation	**Nuscene ↓**	**BrnoCompSpeed ↓**	**IIITD Data ↓**
Mean Frobenius Absolute Error	100.353	947.2165	41.0633
Mean Frobenius Relative Error	0.0499	0.2263	0.0350
Mean fx Relative Error	0.0581	0.2464	0.0409
Mean fy Relative Error	0.0545	0.2349	0.0357
Mean cx Relative Error	0.0032	0.0027	0.0040
Mean cy Relative Error	0.0176	0.0163	0.0402

(a) Orientation 1 (Horizon visible)

(b) Orientation 2 (Horizon not visible)

Figure 3.11: Examples of different orientation types in datasets.

As shown in Table 3.9, the intrinsic estimation errors for the BrnoCompSpeed dataset are relatively high when using the vanishing point-based method. However, the corresponding distance estimation errors reported in Table 3.5 remain reasonably low. This apparent contradiction can be explained by the variance in the accuracy of intrinsic predictions across different scenes. While some intrinsics are significantly inaccurate and deviate from the ground truth, others are estimated with high precision. Consequently, although the overall intrinsic error appears high in aggregate, the correctly estimated intrinsics in certain scenes compensate for the poor estimates in others, resulting in a lower average error in homography-based distance estimation. We also observe inconsistency

in the errors obtained using the vanishing point (VP)-based intrinsic estimation method, where some scenes yield very low errors while others exhibit significantly high errors.

Table 3.9: Vanishing Points based approximated Intrinsic error (pixels)

VP based Intrinsic Evaluation	**BrnoCompSpeed ↓**
Mean Frobenius Absolute Error	652.5416
Mean Frobenius Relative Error	0.1968
Mean fx Relative Error	0.2100
Mean fy Relative Error	0.2100
Mean cx Relative Error	0.0000
Mean cy Relative Error	0.0000

We also observe a significant difference between down-road and cross-road distance estimation errors, even in cases where ground truth intrinsics are used. One possible reason for this discrepancy is the assumption of square pixels in the intrinsic matrix, where the focal lengths in the x and y directions are considered equal ($f_x = f_y$). In real-world cameras, especially in wide-angle or automotive setups, this assumption may not hold perfectly, potentially leading to anisotropic scaling effects in the depth-to-distance projection and causing uneven accuracy along different directions.

3.7 Comparative Analysis

Our method demonstrates superior performance across multiple datasets and scenes with varying complexity 3.103.113.123.13. On the BrnoCompSpeed dataset, we achieve an error of 10.23% using ground truth (GT) intrinsics and 16.56% using approximated intrinsic, on which the FARSEC speed estimation module [46] reports a significantly higher error of 26.34%. On the Vehant dataset, the error is 20.23% with UniDepth-predicted intrinsics and 22.7% with vanishing point (VP)-based intrinsic, which is almost equivalent. On the nuScenes and IIIT datasets, we report errors of 13.26% and 11.23% using GT intrinsics, and 10.78% and 12.73% with UniDepth intrinsics, respectively. Here, when the proposed methodology uses Unidepth Intrinsic, the error is quite low as compared to BrnoCompSpeed and Vehant Dataset. The reason for this is a change in the Dataset Distribution.

Table 3.10: Comparison of distance estimation error (%) across different methods and BrnoCompSpeed Dataset

Method (Dataset)	**Intrinsic**	**Error (%)**
FARSEC (BrnoCompSpeed) [46]	-	26.34
Ours (BrnoCompSpeed)	GT Intrinsic	10.23
	VP-Based Intrinsic	16.56

Table 3.11: Comparison of distance estimation error (%) across different methods and Vehant Dataset

Method (Dataset)	**Intrinsic**	**Error (%)**
Ours (Vehant)	UniDepth Intrinsic	20.23
	VP-Based Intrinsic	22.70

Table 3.12: Comparison of distance estimation error (%) across different methods and NuScene Dataset

Method (Dataset)	**Intrinsic**	**Error (%)**
Ours (nuScenes)	GT Intrinsic	13.26
	UniDepth Intrinsic	10.78

Table 3.13: Comparison of distance estimation error (%) across different methods and IIITD dataset

Method (Dataset)	**Intrinsic**	**Error (%)**
Ours (IIIT)	GT Intrinsic	11.23
	UniDepth Intrinsic	12.73

Our method is tested across multiple scenes and datasets, including challenging conditions such as varying road layouts and traffic environments. The overall performance also surpasses that of the FARSEC module, showcasing the robustness and adaptability of our proposed pipeline.

Chapter 4

LiDAR-Camera Cross Calibration

LiDAR–Camera cross calibration is a critical step in multi-sensor fusion pipelines used in autonomous vehicles, robotics, and intelligent perception systems. It involves estimating the rigid-body transformation between the coordinate frames of a 3D LiDAR sensor and a 2D camera. This transformation, defined by a rotation and translation, enables the projection of LiDAR point clouds onto the image plane, allowing for the accurate fusion of depth and visual information.

While the theoretical formulation of this calibration problem is well-established, real-world deployment introduces a range of practical challenges that significantly impact accuracy and robustness. One major issue is **sensor motion during calibration**. Slight vibrations or misalignments due to vehicle dynamics can cause relative drift between the LiDAR and camera, introducing inconsistencies in the observed correspondences.

Field-of-view misalignment is also a common challenge, especially in configurations where the LiDAR provides a broader horizontal coverage than the camera. This reduces the overlapping region available for calibration and limits the number of reliable correspondences between modalities.

Environmental factors such as **varying lighting conditions**, **adverse weather** (e.g., rain, fog, glare), and **scene sparsity** further complicate the calibration process. Sparse and non-uniform LiDAR point distributions make it difficult to establish consistent geometric matches with 2D image features.

This chapter addresses these challenges by leveraging geometric structures in the environment, specifically plane features, to perform targetless calibration. The method is designed to operate reliably in outdoor settings without requiring controlled conditions or manual intervention, making it practical for scalable real-world applications.

4.1 Mathematical Preliminaries

We aim to estimate the rigid-body transformation between the LiDAR frame L and the camera frame C, represented by a transformation matrix:

$$^{C}T_{L} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^{\top} & 1 \end{bmatrix},$$

where $\mathbf{R} \in SO(3)$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is a translation vector.

Given a planar surface in the LiDAR coordinate frame represented by the plane equation:

$$\mathbf{n}_L^{\top}\mathbf{x}_L + d_L = 0,$$

Where $\mathbf{n}_L \in \mathbb{R}^3$ is the normal vector and $d_L \in \mathbb{R}$ is the offset, the plane can be transformed into the camera coordinate frame as follows.

Definition 3 (Plane Transformation)**.** *The function* PlaneTransformation $(\mathbf{n}_L, d_L, \mathbf{R}, \mathbf{t}) \rightarrow (\mathbf{n}_C, d_C)$ *is defined as:*

$$\mathbf{n}_C = \mathbf{R}\,\mathbf{n}_L,$$

$$d_C = d_L - \mathbf{n}_C^{\top}\mathbf{t}.$$

Definition 4 (Plane Correspondence Verification)**.** *The function* VerifyPlaneCorrespondence $(\mathbf{n}_C, d_C, \mathbf{n}'_C, d'_C, \theta_{max}, \delta_{max}) \rightarrow$ *bool verifies consistency based on two criteria:*

- ***Normal Alignment:*** *The angular difference between the transformed normal* $\mathbf{n}_C$ *and the camera-detected normal* $\mathbf{n}'_C$ *must satisfy*

$$\arccos\left(\mathbf{n}_C^{\top}\mathbf{n}'_C\right) \leq \theta_{max}.$$

- ***Offset Difference:*** *The difference between the plane offsets must satisfy*

$$|d_C - d'_C| \leq \delta_{max}.$$

Definition 5 (Rotation and Translation Estimation from Plane Pairs)**.** *[18] Given matched plane pairs* $\{(\mathbf{n}_{L_i}, d_{L_i}), (\mathbf{n}_{C_i}, d_{C_i})\}$*, the rotation* $\mathbf{R}$ *and translation* $\mathbf{t}$ *are estimated as follows:*

Rotation: *Minimize*

$$\min_{\mathbf{R}\in SO(3)} \sum_i \|\mathbf{R}\mathbf{n}_{L_i} - \mathbf{n}_{C_i}\|^2.$$

Compute $\mathbf{M} = \sum_i \mathbf{n}_{L_i}\mathbf{n}_{C_i}^{\top}$*, apply SVD:* $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$*, and set* $\mathbf{R} = \mathbf{V}\mathbf{U}^{\top}$*. If* $\det(\mathbf{R}) < 0$*, flip the sign of the last column of* $\mathbf{V}$*.*

Translation: *Given* $\mathbf{R}$*, solve*

$$\mathbf{t} = \left(\sum_i \mathbf{n}_{C_i} \mathbf{n}_{C_i}^\top \right)^{-1} \left(\sum_i \mathbf{n}_{C_i} (d_{C_i} - d_{L_i}) \right),$$

if the matrix is invertible; otherwise, set $\mathbf{t} = \mathbf{0}$*.*

Definition 6 (Rotation and Translation Estimation from a Single Plane Pair)**.** *Given two planes defined by* $\mathbf{n}_1^\top \mathbf{x} + d_1 = 0$ *and* $\mathbf{n}_2^\top \mathbf{x} + d_2 = 0$*, where* $\mathbf{n}_1, \mathbf{n}_2 \in \mathbb{R}^3$ *are unit normals and* $d_1, d_2 \in \mathbb{R}$ *are offsets, the goal is to estimate a rotation* $\mathbf{R}$ *and translation* $\mathbf{t}$ *such that* $\mathbf{n}_2 \approx \mathbf{R}\mathbf{n}_1$*.*

For rotation, define the cross and dot products:

$$\mathbf{v} = \mathbf{n}_1 \times \mathbf{n}_2, \quad c = \mathbf{n}_1^\top \mathbf{n}_2, \quad s = \|\mathbf{v}\|.$$

Let the skew-symmetric matrix be

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix},$$

then the rotation matrix is

$$\mathbf{R} = \begin{cases} \mathbf{I}_3, & \text{if } s = 0, \\ \mathbf{I}_3 + [\mathbf{v}]_\times + \frac{1-c}{s^2} [\mathbf{v}]_\times^2, & \text{otherwise.} \end{cases}$$

For translation, normalize the offsets as $d_1' = -d_1/\|\mathbf{n}_1\|$*,* $d_2' = -d_2/\|\mathbf{n}_2\|$*, and compute*

$$\mathbf{t} = (d_2' - d_1')\mathbf{n}_2.$$

4.2 Problem Statement: LiDAR-Camera Cross Calibration

In autonomous perception systems, accurate multi-sensor fusion requires estimating the **extrinsic transformation** between a 3D LiDAR sensor and a 2D camera. The goal is to determine the **rigid-body transformation** from the LiDAR frame to the camera frame:

$$\mathbf{T}_{\text{LiDAR}\to\text{Camera}} \in SE(3), \quad \mathbf{T}_{\text{LiDAR}\to\text{Camera}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{4.1}$$

where:

- $\mathbf{R} \in SO(3)$ is the rotation matrix
- $\mathbf{t} \in \mathbb{R}^3$ is the translation vector

This transformation should satisfy the **projection constraint**:

$$\mathbf{P}_{\text{Camera}} = f_{\mathbf{R},\mathbf{t}}(\mathbf{P}_{\text{LiDAR}}) \tag{4.2}$$

where:

- $\mathbf{P}_{\text{LiDAR}} \in \mathbb{R}^3$ is a 3D Geometric Entity in the LiDAR frame
- $\mathbf{p}_{\text{Camera}} \in \mathbb{P}^2$ is the corresponding 3D Geometric Entity in the Camera frame

4.3 Proposed LiDAR–Camera Cross-Calibration Pipeline

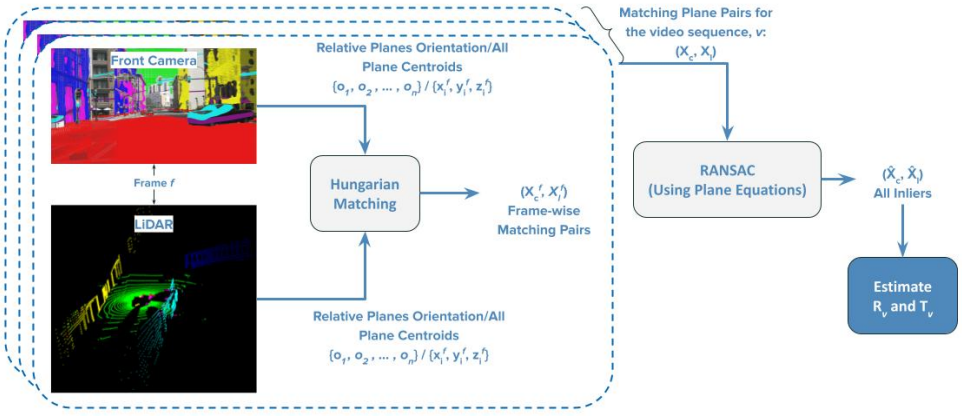


Figure 4.1: Overview of the proposed plane-based LiDAR-camera extrinsic calibration pipeline.

The proposed pipeline performs LiDAR–Camera cross-calibration by leveraging the geometric structure of the scene. The objective is to estimate the rotation and translation between the two sensor modalities using information extracted from static parts of the environment.

Geometric primitives specifically planes are extracted independently from the camera images and LiDAR scans. These planes represent reliable features in structured environments and serve as the basis for cross-sensor matching. The overall pipeline is composed of four main stages, described below.

4.3.1 Plane Extraction from Camera and LiDAR

For each synchronized frame, we extract planar structures from both the camera image and the corresponding LiDAR scan. These geometric entities serve as the fundamental units for establishing correspondences between the two sensors. The extraction process is performed independently for the camera and LiDAR data streams.

4.3.2 Graph-Based Plane Matching

After extracting the planes, we compute geometric similarity between planes from the camera and LiDAR using a graph-based matching approach. Here, each plane is considered a node in a graph, and the relationships between planes (e.g., orientation, relative distances) are encoded in the matching cost. The Hungarian algorithm[47] is then used to find optimal one-to-one matches between the two sets of planes for each frame. This results in a set of matched plane pairs.

4.3.3 Algorithm: Initial Transformation Estimation

The matched plane pairs obtained from each frame are collected over the entire video sequence. To remove incorrect matches and improve robustness, we apply RANSAC on these pairs. RANSAC helps eliminate outliers by evaluating which matches are consistent with a single rigid-body transformation model based on plane relationships. The output is a refined set of reliable matching plane pairs.

4.3.4 Estimation of Rotation and Translation

Using the set of inlier plane matches from RANSAC, we estimate the final transformation between the LiDAR and camera frames. This includes both the rotation and translation components. The result is a robust estimation of the extrinsic parameters that align the two sensors spatially.

4.4 Experimental Setup

4.4.1 Plane Extraction from LiDAR and Camera

Fig. 4.2 overview of the plane detection system.

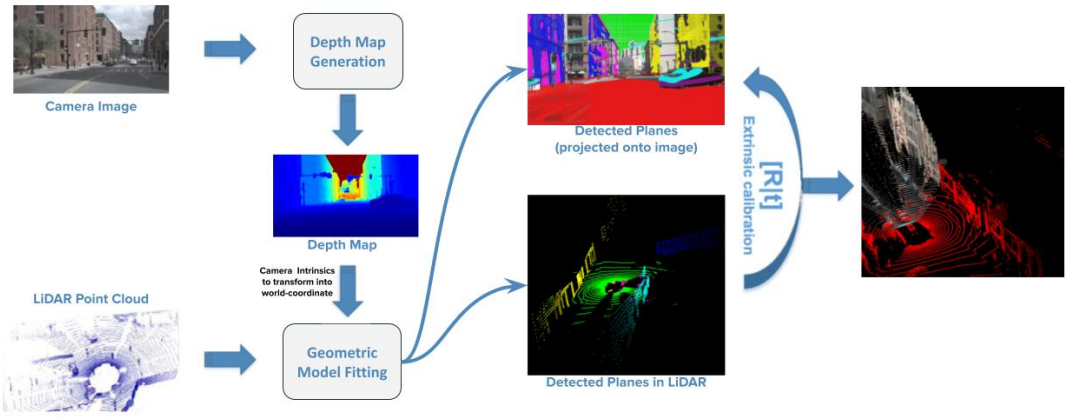


Figure 4.2: Overview of the plane detection and calibration pipeline.

Figure 4.3 represents planes detected in LiDAR and camera point cloud

4.4.1.1 Plane Detection in LiDAR

To detect planes from the LiDAR point cloud, we employ the **Progressive-X** multi-modal fitting algorithm [17], which is a robust and efficient variant of RANSAC specifically designed for multi-model fitting tasks.

For each LiDAR frame, Progressive-X produces a set of dominant planar surfaces represented by their normal vectors and offsets. These detected planes are geometric primitives for establishing correspondences with planes detected independently from the camera images.

The main advantages of using Progressive-X for LiDAR plane detection are:

- Robustness to noise and outliers in the LiDAR point cloud,
- Efficient extraction of multiple planes without prior knowledge of the number of planes,

- Superior stability and accuracy compared to other multi-model fitting algorithms.

4.4.1.2 Plane detection in camera

To detect planes from camera data, we first generate a dense depth map using the **Metric3D** monocular depth estimation network [42], which predicts metric-scale depth from a single RGB image. Its real-world depth outputs make it well-suited for accurate geometric reasoning in plane extraction.

Given the depth map, we back-project each pixel into the camera coordinate frame using the standard pinhole model. For a pixel at (u, v) with depth z, the corresponding 3D point (X, Y, Z) is computed as:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = z\,\mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix},$$

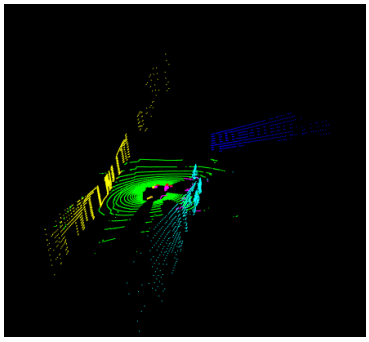
where $\mathbf{K}$ is the intrinsic camera matrix.

This yields a 3D point cloud representing the scene structure. We then apply RANSAC to fit planes by sampling minimal sets, generating plane hypotheses, and selecting inliers. Dominant planes with maximal support are retained, each defined by its normal vector and offset.

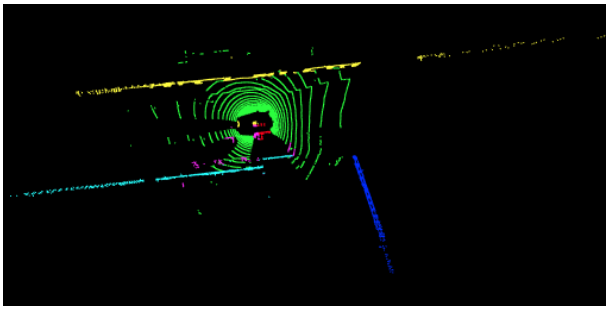
These extracted planar surfaces, derived from camera observations, are used for cross-modal matching and calibration.

Summary of steps:

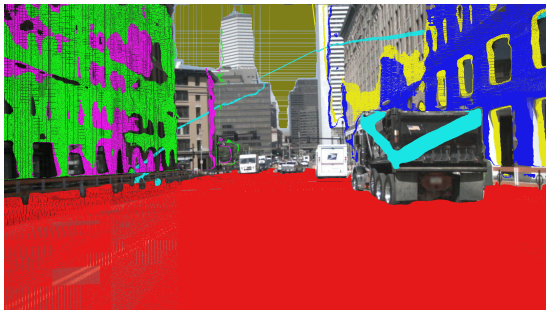
- Depth estimation via Metric3D,
- 3D point cloud reconstruction using known intrinsics,
- Plane fitting using RANSAC.



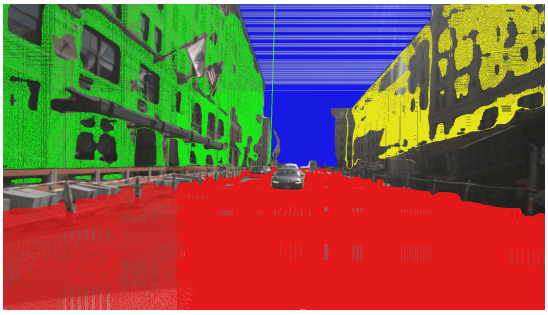
(a) Side view of planes in LiDAR



(b) Top view of planes in LiDAR



(c) Planes in front camera



(d) Planes in back camera

Figure 4.3: visual results of planes in LiDAR and image point clouds

4.4.2 Graph-Based Plane Matching Using Hungarian Algorithm

Planes detected independently from the LiDAR and the camera need to be matched before they can be used for calibration. We use the **Hungarian algorithm** to perform this matching efficiently and optimally.

The Hungarian algorithm is a classical method for solving the *assignment problem*, where the goal is to optimally assign elements from one set to elements of another based on a given assignment cost. In our case, the two sets are:

- Planes detected from the LiDAR point cloud,
- Planes detected from the camera-derived 3D point cloud.

Each possible pairing between a LiDAR plane and a camera plane has an associated **matching cost**, which we define based on three geometric criteria:

- The Euclidean distance between the centroids of the two planes,
- The angular difference between their normal vectors,

The Hungarian algorithm finds the set of plane matches that minimizes the total cost across all assigned pairs, ensuring that each plane is matched to at most one plane in the other set optimally.

Algorithm 3 Hungarian Graph Matching Algorithm (Kuhn–Munkres)[47]

Input: Cost matrix $C \in \mathbb{R}^{n \times n}$ representing pairwise assignment costs
Output: Optimal assignment $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$

Step 1: Row and Column Reduction
    Subtract row-wise minimum from each row of C
    Subtract column-wise minimum from each column of C

Step 2: Initial Zero Covering
    Cover all zeros in C using minimum number of lines
    if number of lines $= n$ **then return** assignment using zeros

Step 3: Adjust Matrix
    Find smallest uncovered value δ
    Subtract δ from all uncovered elements
    Add δ to elements covered twice
    Repeat zero covering until n assignments possible

Step 4: Assignment
    Select independent zeros to construct optimal matching π
return assignment π

This algorithm guarantees finding the *globally optimal* matching with the minimum total cost. This optimal matching ensures that the subsequent calibration steps operate on the most geometrically consistent plane correspondences.

4.4.2.1 Cost Matrix Construction for Plane Matching

To match planes detected from LiDAR and camera frames, we construct cost matrices that quantify how similar or dissimilar any two planes are. Four different cost matrices are computed based on both absolute and relative geometric properties of planes:

4.4.2.1.1 Euclidean Distance between Plane Centroids For each plane, we compute its centroid in 3D space. The matching cost between a LiDAR plane and a camera plane is defined as the Euclidean distance between their centroids:

$$\text{Cost}(\text{LiDAR plane}_i, \text{Camera plane}_j) = \|\mathbf{c}_i^L - \mathbf{c}_j^C\|_2,$$

where $\mathbf{c}_i^L$ and $\mathbf{c}_j^C$ are the centroids of the LiDAR and camera planes, respectively.

This metric captures spatial proximity and is useful for preserving plane positions across modalities.

Limitation: Centroid-based matching ignores plane orientation. It can fail in cases where two planes have similar centroids but opposite normals (e.g., flipped or back-facing planes). In such scenarios, the planes may appear close in space but are geometrically inconsistent, leading to incorrect matches. To overcome the sensitivity of Euclidean distance to differing coordinate frames in LiDAR-camera calibration, we propose a cost metric based on the cost between relative plane orientations. Instead of comparing absolute orientations, which are affected by global rotation, we compute pairwise angular relationships between planes within each modality. These intra-modal relative orientations remain consistent across coordinate systems, making them suitable for cross-modal matching.

4.4.2.1.2 Modified Symmetric Difference Based on Relative Orientations

The symmetric difference between two relative orientation sets must account for minor numerical variations, particularly when working with real-valued angular data derived from noisy sensor inputs. To address this, we define a *modified symmetric difference* that introduces a tolerance threshold ϵ when comparing elements across the two sets.

Given two sets of relative angles, A and B, the modified intersection is first computed by identifying all elements in A that are within ϵ of at least one element in B:

$$A \cap_\epsilon B = \{x \in A \mid \exists y \in B \text{ such that } |x - y| \leq \epsilon\}.$$

Using this approximate intersection, the modified symmetric difference is defined as the union of elements from A and B that do not belong to this ϵ-neighborhood intersection:

$$A \,\Delta_\epsilon\, B = \{x \in A \mid \forall y \in A \cap_\epsilon B,\ |x - y| > \epsilon\} \cup \{y \in B \mid \forall x \in A \cap_\epsilon B,\ |y - x| > \epsilon\}.$$

This formulation ensures that angle values close to each other, within a tolerance ϵ, are treated as equivalent, thus reducing the influence of small numerical discrepancies or sensor noise. The result is a more robust and stable matching cost, particularly in scenarios where exact floating-point comparisons would exaggerate dissimilarities between geometrically similar plane configurations.

4.4.2.1.3 Jaccard Distance Based on Relative Orientations

We compute the **Jaccard similarity** between relative orientation vectors to quantify the alignment between local directional patterns. To accommodate slight variations in values, we use an ϵ-tolerant intersection.

The **Jaccard similarity** is defined as:

$$\text{Jaccard Similarity}_\epsilon = \frac{|A \cap_\epsilon B|}{|A \cup B| + \delta},$$

where A and B are two sets of relative angles, and δ is a small constant (e.g., 10^{-6}) added for numerical stability.

The ϵ-approximate intersection is defined as:

$$A \cap_\epsilon B = \{x \in A \mid \exists y \in B \text{ such that } |x - y| \leq \epsilon\}.$$

The union is computed using the standard set union:

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}.$$

The **matching cost** is then defined as the complement of the Jaccard similarity:

$$\text{Cost} = 1 - \text{Jaccard Similarity}_\epsilon,$$

which reflects the dissimilarity between the orientation patterns. Lower cost values indicate stronger similarity between the local orientation vectors.

4.4.3 Algorithm: Initial Transformation Estimation

Figure 4.1 denotes an overview of the proposed plane-based LiDAR-camera extrinsic calibration pipeline.

The proposed calibration pipeline (Algorithm 4) estimates the extrinsic transformation $(\mathbf{R}, \mathbf{t})$ between a LiDAR and a camera using planar surfaces detected independently in each modality.

LiDAR planes are extracted using the Progressive-X algorithm, which robustly fits planes in sparse, noisy data. Camera-side planes are reconstructed by predicting depth via Metric3D, back-projecting it using intrinsic parameters, and applying RANSAC for plane fitting.

Matched planes are identified by comparing geometric features such as center distances, normal differences, and local context. These comparisons form cost matrices used in the Hungarian algorithm to find optimal correspondences.

Outliers are removed using an RANSAC loop. For each iteration, three plane pairs are sampled to estimate $(\mathbf{R}, \mathbf{t})$ via `CombinedRotationTranslationSVD`. The solution

Algorithm 4 Complete Pipeline: Plane-Based LiDAR-Camera Extrinsic Calibration

Input: LiDAR point cloud $\mathcal{P}_L$, camera RGB image $\mathcal{I}_C$, camera intrinsic matrix $\mathbf{K}$, thresholds $\theta_{\max}$, $\delta_{\max}$, maximum iterations N_{iter}

function LiDARCameraCalibrationPipeline($\mathcal{P}_L$, $\mathcal{I}_C$, $\mathbf{K}$, $\theta_{\max}$, $\delta_{\max}$, N_{iter})
    $\mathcal{S}_L \leftarrow$ DetectPlanesLiDAR($\mathcal{P}_L$)
    $\mathcal{S}_C \leftarrow$ DetectPlanesCamera($\mathcal{I}_C$, $\mathbf{K}$)
    $\mathcal{P} \leftarrow$ MatchPlanes($\mathcal{S}_L$, $\mathcal{S}_C$)
    $(\mathbf{R}^*, \mathbf{t}^*, \mathcal{I}^*) \leftarrow$ EstimateRotationTranslationRANSAC($\mathcal{P}$, $\theta_{\max}$, $\delta_{\max}$, N_{iter})
    return $\mathbf{R}^*$, $\mathbf{t}^*$

function DetectPlanesLiDAR($\mathcal{P}_L$)4.4.1.1
    Apply Progressive-X multi-model fitting
    return set of planes $\mathcal{S}_L$

function DetectPlanesCamera($\mathcal{I}_C$, $\mathbf{K}$)4.4.1.2
    $\mathcal{D} \leftarrow$ EstimateDepthMapMetric3D($\mathcal{I}_C$)
    $\mathcal{P}_C \leftarrow$ ReconstructPointCloud($\mathcal{D}$, $\mathbf{K}$)
    $\mathcal{S}_C \leftarrow$ RANSACPlaneFitting($\mathcal{P}_C$)
    return set of planes $\mathcal{S}_C$

function MatchPlanes($\mathcal{S}_L$, $\mathcal{S}_C$)
    Construct cost matrices (centroid distance, orientation difference, etc.)4.4.2.1
    Solve assignment using Hungarian Algorithm4.4.2
    return matched plane pairs $\mathcal{P}$

function EstimateRotationTranslationRANSAC($\mathcal{P}$, $\theta_{\max}$, $\delta_{\max}$, N_{iter})
    Initialize: $\mathbf{R}^* \leftarrow \mathbf{I}_3$, $\mathbf{t}^* \leftarrow \mathbf{0}_3$, $N^* \leftarrow 0$
    for $k = 1$ **to** N_{iter} **do**
        Sample 3/1 random plane correspondences
        $(\mathbf{R}, \mathbf{t}) \leftarrow$ CombinedRotationTranslationSVD(sampled pairs) 5/ PlaneToRotationTranslation($\mathbf{n}_L$, d_L, $\mathbf{n}_C$, d_C)6
        Initialize inlier set $\mathcal{I} \leftarrow \emptyset$
        for each plane pair $(\mathbf{n}_L, d_L), (\mathbf{n}_C, d_C)$
            $(\tilde{\mathbf{n}}_C, \tilde{d}_C) \leftarrow$ PlaneTransformation($\mathbf{n}_L$, d_L, $\mathbf{R}$, $\mathbf{t}$) 3
            if VerifyPlaneCorrespondence ($\tilde{\mathbf{n}}_C$, $\tilde{d}_C$, $\mathbf{n}_C$, d_C, $\theta_{\max}$, $\delta_{\max}$) 4 **then**
                Add to $\mathcal{I}$
            end if
        end for
        if $|\mathcal{I}| > N^*$ **then**
            Update $\mathbf{R}^*, \mathbf{t}^*, \mathcal{I}^*$
        end if
    end for
    return $\mathbf{R}^*$, $\mathbf{t}^*$, $\mathcal{I}^*$

is evaluated on all matches using angular ($\theta_{\max}$) and offset ($\delta_{\max}$) thresholds, and the transformation with the highest inlier count is selected. For single-plane estimation, `PlaneToRotationTranslation` is used directly.

4.4.4 Estimation of Rotation and Translation

After identifying inlier plane correspondences via RANSAC, the final step is to estimate the rigid-body transformation rotation $\mathbf{R}$ and translation $\mathbf{t}$ between the LiDAR and camera frames. Two methods are considered for this estimation.

The first, **CombinedRotationTranslationSVD** 5, jointly estimates $\mathbf{R}$ and $\mathbf{t}$ from all inliers using a single optimization step. This involves SVD-based alignment of normals and closed-form offset correction, yielding a globally consistent solution.

The second, **pairwise plane averaging** Algorithm 5, computes $\mathbf{R}$ and $\mathbf{t}$ independently for each plane pair using PlaneToRotationTranslation 6. Normals are aligned via cross product and skew-symmetric matrix construction, translation is computed from averaging normalized offsets and the camera's normal direction.

After computing all per-pair estimates, rotations are aggregated into a matrix and projected back to $SO(3)$ using SVD, while translations are averaged vectorially. This method improves robustness by reducing the influence of individual erroneous matches.

Both methods produce the final extrinsic parameters $(\mathbf{R}, \mathbf{t})$, with **CombinedRotationTranslationSVD** preferred for rotation and **pairwise averaging** offering better robustness in translation.

Algorithm 5 Final Rotation and Translation Estimation from RANSAC Inliers

Input: Set of RANSAC inlier plane pairs $\mathcal{I}$

Method 1: Combined Estimation via SVD5

Method 2: Per-Pair Averaging Estimation

  function EstimateViaPairwiseAveraging($\mathcal{I}$)

    Initialize empty sets of rotations $\mathcal{R}$ and translations $\mathcal{T}$

    for each plane pair $(\mathbf{n}_L, d_L), (\mathbf{n}_C, d_C)$ in $\mathcal{I}$ **do**

      $(\mathbf{R}_{ij}, \mathbf{t}_{ij}) \leftarrow$ PlaneToRotationTranslation($\mathbf{n}_L$, d_L, $\mathbf{n}_C$, d_C)6

      Add $\mathbf{R}_{ij}$ to $\mathcal{R}$, $\mathbf{t}_{ij}$ to $\mathcal{T}$

    end for

    Compute sum of rotations: $\mathbf{M}_R = \sum_{\mathbf{R} \in \mathcal{R}} \mathbf{R}$

    Perform SVD: $\mathbf{M}_R = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$

    Compute averaged rotation: $\mathbf{R} = \mathbf{U}\mathbf{V}^\top$

    Correct $\mathbf{R}$ if $\det(\mathbf{R}) < 0$

    Compute averaged translation:

$$\mathbf{t} = \frac{1}{|\mathcal{T}|} \sum_{\mathbf{t}_i \in \mathcal{T}} \mathbf{t}_i$$

    return $\mathbf{R}$, $\mathbf{t}$

4.5 Dataset Description

4.5.1 Datasets Used for Cross-Calibration

NuScenes Dataset [3]

Figure 4.44.5 shows sample frames from the *nuScenes* dataset used in this work for LiDAR–camera cross-calibration. The dataset, developed by Motional, provides a multi-modal sensor suite mounted on a vehicle, including six cameras and a 32-beam Velodyne HDL-32E LiDAR. These sensors are rigidly attached to the ego-vehicle but mounted at different positions and heights, which creates spatial misalignments critical to resolve through cross-calibration.

Table 4.1: nuScenes Dataset Characteristics for LiDAR–Camera Cross-Calibration

Property	**Description**
Sensor Modalities	6 cameras, 1 Velodyne HDL-32E LiDAR, GPS/IMU
Camera–LiDAR Sync	Time-synchronized at 12 Hz (camera) and 20 Hz (LiDAR)
Ground Truth Extrinsics	✓ Provided (can be refined further)
Calibration Drift	✓ Observed due to motion and vibrations
LiDAR Sparsity	✓ Sparse scans
Viewpoint Discrepancy	✓ Different mounting heights and orientations
Planar Structures	✓ Available, but not uniformly across all frames
Dynamic Elements	✓ Moving vehicles and pedestrians cause occlusions
Use Case	Plane-based extrinsic refinement via 3D–2D alignment

Although *nuScenes* provides ground-truth extrinsic calibration for all sensors, practical usage reveals small drifts and misalignments due to time-varying vibrations, temperature-induced mechanical changes, and motion distortion in LiDAR scans. For tasks requiring sub-degree alignment like projection consistency or metric 3D reconstruction such errors must be corrected.

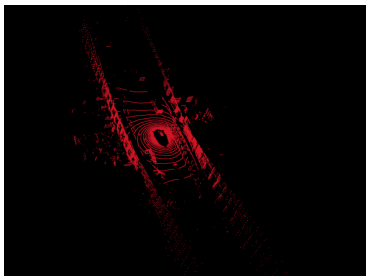
To that end, we utilize a curated subset of *nuScenes* data for calibration purposes:

Summary of Usage

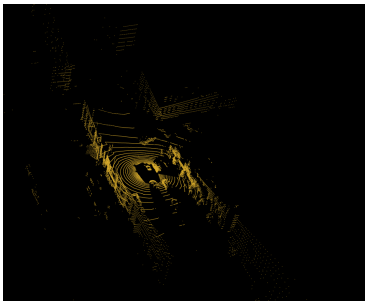
- Scenes selected: 85 to 340 (from 1,000 total)
- Frame pairs used: ∼230 time-synchronized LiDAR–camera samples
- Cameras used: Predominantly `CAM_FRONT`, aligned with LiDAR FOV
- Task: Refinement of LiDAR-to-camera extrinsics using planar road projections

Each selected sample consists of a calibrated camera image and the corresponding LiDAR point cloud. Planar surfaces such as roads, walls, and building facades are extracted and matched across the two sensor modalities to solve for a refined extrinsic transformation. These refined calibrations help mitigate small spatial inconsistencies and are essential for downstream tasks like 3D object localization and distance estimation.

In summary, the *nuScenes* dataset offers a rich testbed for cross-calibration due to its diverse scenes, full calibration metadata, and synchronized multimodal data. However, its real-world noise, sparse depth returns, and environmental variations also make calibration a non-trivial task, underscoring the need for robust refinement techniques.



(a) Nuscene Sample LiDAR Scan1



(b) Nuscene Sample LiDAR Scan2

Figure 4.4: Sample LiDAR scans for Nuscenes

(a) CAM FRONT

(b) CAM FRONT LEFT

(c) CAM FRONT RIGHT

(d) CAM BACK

(e) CAM BACK LEFT

(f) CAM BACK RIGHT

Figure 4.5: Nuscenes 6 Cams sample Images

4.6 Empirical Evaluation and Results

Evaluation Metrics

We use the following metrics to evaluate the accuracy of LiDAR-camera extrinsic calibration:

- **Rotation Angle Error (in degrees)**[48]:
  Measures the angular difference between the estimated rotation $\mathbf{R}_{\text{est}}$ and the ground truth rotation $\mathbf{R}_{\text{gt}}$. It is computed from the relative rotation matrix:

  $$\mathbf{R}_{\text{rel}} = \mathbf{R}_{\text{gt}}^{\top}\mathbf{R}_{\text{est}}$$

  The rotation angle θ is then obtained using the Rodrigues formula:

  $$\theta = \cos^{-1}\left(\frac{\text{trace}(\mathbf{R}_{\text{rel}}) - 1}{2}\right)$$

  The resulting angle is expressed in degrees.

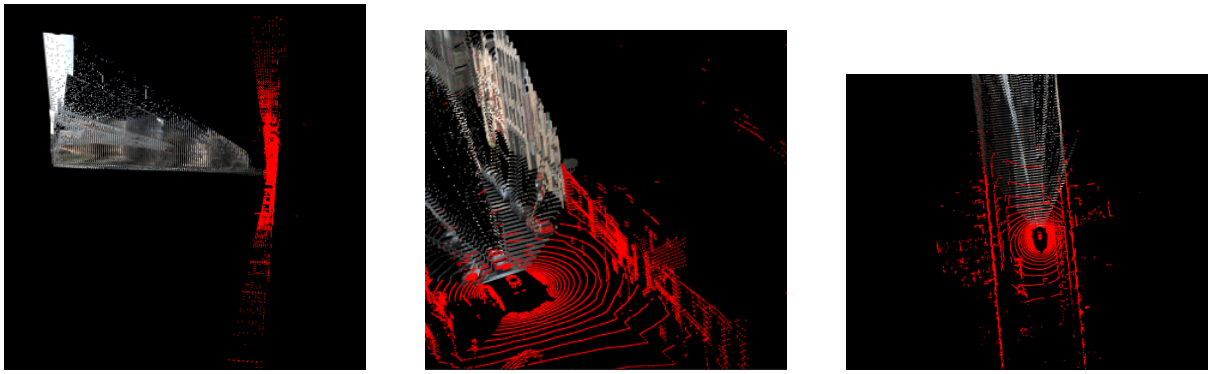
- **Translation Errors t_x, t_y, t_z (in meters)**:
  Represent the absolute differences between the estimated and ground truth translations along the x, y, and z axes:

  $$t_i = |t_{\text{est},i} - t_{\text{gt},i}| \quad \text{for } i \in \{x, y, z\}$$

  Each component of the translation error is reported in meters (m).

4.6.1 Qualitative and Quantitative results for Centroid-Based Hungarian Matching

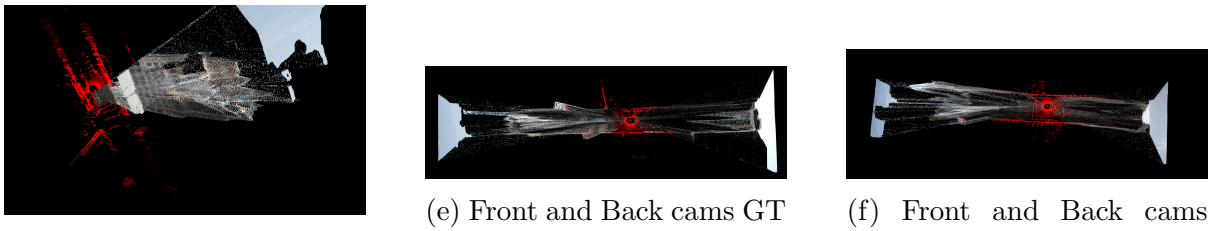
Figure 4.6 presents a comparative visualization of camera-LiDAR alignment results across different methods and viewpoints. The first row highlights the front camera views: (a) shows non-aligned without calibration data, (b) represents ground truth (GT) alignment, and (c) demonstrates our method's alignment, clearly improving point cloud consistency. The second row evaluates back camera and multi-view alignment: (d) is the unaligned back view, (e) shows GT alignment for front and back cameras, and (f) illustrates our method's performance in jointly aligning both views. Visually, our method achieves close conformity with GT, confirming its working.



(a) Front cam non aligned

(b) Front cam GT aligned

(c) Front cam ours aligned



(d) Back cam non aligned

(e) Front and Back cams GT aligned

(f) Front and Back cams ours aligned

Figure 4.6: visual results of LiDAR and image point clouds

Table 4.2 and Figure4.7 present results for the single-plane RANSAC method, where $\mathbf{R}$ and $\mathbf{t}$ are estimated from a single plane pair, constraining the solution to two degrees of freedom (DOF). This yields favorable results for the front camera due to only z-axis rotation, but leads to higher errors for other cameras with multi-directional motion highlighting the method's limitations in full 6-DOF estimation.

In contrast, the triple-plane RANSAC results (Table 4.3 and Figure4.8) assume three linearly independent planes, enabling full 6-DOF estimation. This leads to significantly lower errors across most views, where limited motion reduces the benefit of added constraints. While more accurate overall, the triple-plane method still exhibits some rotational anomalies, which are analyzed further in the next section.

Table 4.2: Errors Single Plane RANSAC Statistics

CAM_FRONT

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.02	0.24	3.56	8.50
Median	0.01	0.12	0.73	2.55
Min	0.00	0.00	0.73	0.81
Max	0.49	1.91	144.39	113.93

CAM_FRONT_LEFT

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.40	0.34	1.09	52.95
Median	0.20	0.19	0.74	42.23
Min	0.13	0.00	0.71	2.72
Max	5.90	1.87	10.30	177.93

CAM_FRONT_RIGHT

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.39	0.33	3.73	43.01
Median	0.23	0.24	0.74	32.77
Min	0.09	0.05	0.71	2.55
Max	7.96	1.93	146.47	178.67

CAM_BACK

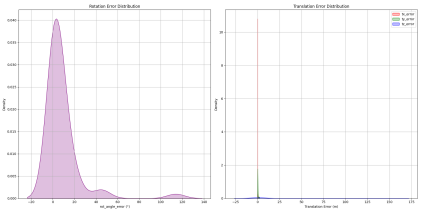
Error	t_x	t_y	t_z	θ_{rot}
Mean	0.16	0.37	19.74	114.48
Median	0.00	0.10	0.93	118.81
Min	0.00	0.00	0.91	1.26
Max	7.04	1.73	106.68	179.97

CAM_BACK_LEFT

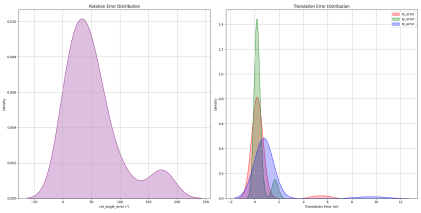
Error	t_x	t_y	t_z	θ_{rot}
Mean	0.30	0.37	6.99	80.18
Median	0.22	0.21	0.44	83.22
Min	0.21	0.00	0.38	4.46
Max	3.82	2.14	167.06	174.61

CAM_BACK_RIGHT

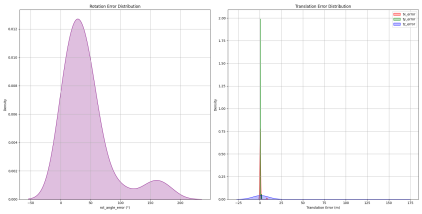
Error	t_x	t_y	t_z	θ_{rot}
Mean	0.30	0.37	6.99	80.18
Median	0.22	0.21	0.44	83.22
Min	0.21	0.00	0.38	4.46
Max	3.82	2.14	167.06	174.61



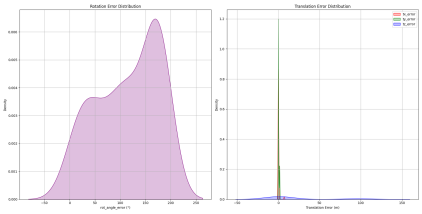
(a) CAM FRONT



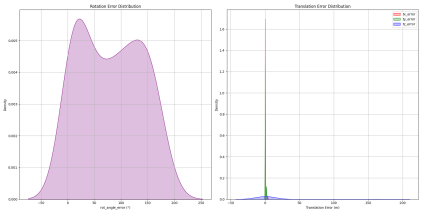
(b) CAM FRONT LEFT



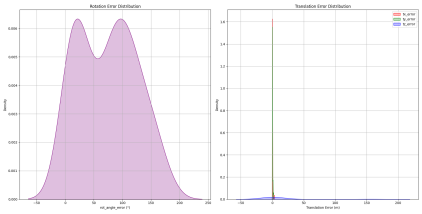
(c) CAM FRONT RIGHT



(d) CAM BACK



(e) CAM BACK LEFT



(f) CAM BACK RIGHT

Figure 4.7: Single plane RANSAC KDE plots for Rotation and translation error.

Table 4.3: Errors Triple plane RANSAC statistics

CAM_FRONT

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.05	0.20	0.74	25.21
Median	0.01	0.17	0.74	4.86
Min	0.00	0.01	0.70	0.97
Max	0.49	0.81	0.77	178.11

CAM_FRONT_LEFT

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.29	0.35	0.90	54.48
Median	0.20	0.26	0.74	27.50
Min	0.02	0.01	0.65	1.59
Max	4.94	2.17	8.73	179.28

CAM_FRONT_RIGHT

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.25	0.35	0.77	49.51
Median	0.23	0.29	0.74	26.53
Min	0.11	0.00	0.52	1.38
Max	0.66	1.30	2.21	179.42

CAM_BACK

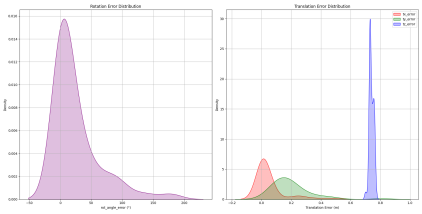
Error	t_x	t_y	t_z	θ_{rot}
Mean	0.19	0.36	0.91	81.14
Median	0.00	0.20	0.93	71.01
Min	0.00	0.00	0.36	3.76
Max	6.89	1.88	0.95	179.80

CAM_BACK_LEFT

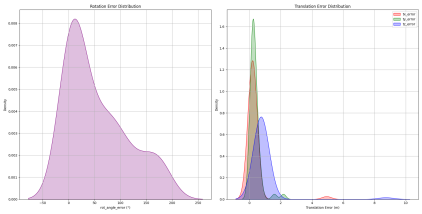
Error	t_x	t_y	t_z	θ_{rot}
Mean	0.23	0.36	0.44	71.34
Median	0.22	0.23	0.44	71.34
Min	0.19	0.00	0.36	0.33
Max	0.55	2.00	0.56	179.96

CAM_BACK_RIGHT

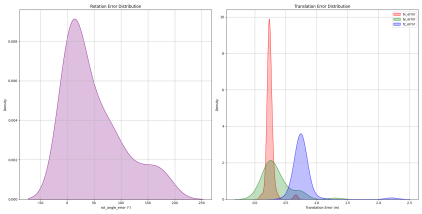
Error	t_x	t_y	t_z	θ_{rot}
Mean	0.24	0.51	0.46	69.28
Median	0.24	0.39	0.42	52.35
Min	0.13	0.09	0.12	0.73
Max	0.41	1.97	2.05	178.22



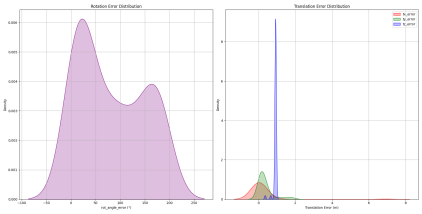
(a) CAM FRONT



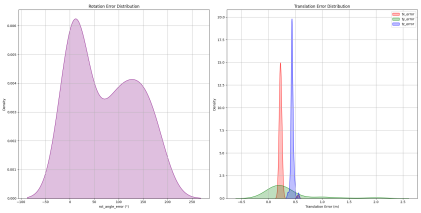
(b) CAM FRONT LEFT



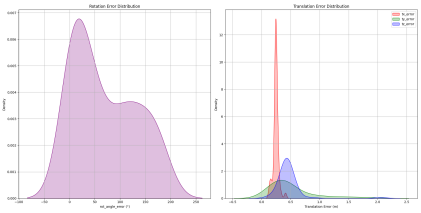
(c) CAM FRONT RIGHT



(d) CAM BACK



(e) CAM BACK LEFT



(f) CAM BACK RIGHT

Figure 4.8: Triple plane RANSAC KDE plots for Rotation and translation error.

Camera	LiDAR Frame	inliers
CAM_FRONT	LiDAR planes	0.1716
CAM_FRONT_RIGHT	LiDAR planes	0.1304
CAM_FRONT_LEFT	LiDAR planes	0.1360
CAM_BACK	LiDAR planes	0.1175
CAM_BACK_RIGHT	LiDAR planes	0.1070
CAM_BACK_LEFT	LiDAR planes	0.1308

Table 4.4: Inliers Hungarian Centroid Distance

4.6.2 Quantitative results for Relative orientation based Hungarian Matching (Modified Symmetric Difference)

As shown in Table 4.5, outperform the Euclidean distance between plane centroids. The KDE plot indicates that most scenes have errors concentrated around zero, although the issue of 180° rotational flips persists, as evidenced by secondary peaks near 180°.

Table 4.6 shows that the number of inliers obtained using the modified symmetric distance is relatively low, comparable to the centroid matching based on Euclidean distance.

Table 4.5: Errors RANSAC modified Symmetric Distance

CAM_FRONT

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.00	0.18	0.74	61.93
Median	0.00	0.19	0.73	32.60
Min	-0.15	-0.89	0.73	0.87
Max	0.17	0.57	0.83	179.60

CAM_FRONT_LEFT

Error	t_x	t_y	t_z	θ_{rot}
Mean	-0.03	0.25	1.07	79.70
Median	0.20	0.26	0.74	83.91
Min	-8.44	-0.10	0.60	1.02
Max	0.30	0.61	12.89	179.87

CAM_FRONT_RIGHT

Error	t_x	t_y	t_z	θ_{rot}
Mean	-0.03	0.33	1.04	52.67
Median	-0.23	0.31	0.74	23.91
Min	-0.53	-0.50	-0.09	0.92
Max	5.18	1.32	9.66	179.83

CAM_BACK

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.00	0.10	0.93	49.26
Median	0.00	0.13	0.93	12.55
Min	-0.10	-1.18	0.88	0.72
Max	0.22	0.44	1.38	179.03

CAM_BACK_LEFT

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.53	0.15	1.39	71.42
Median	0.22	0.23	0.44	64.25
Min	0.13	-8.13	0.36	1.44
Max	6.14	1.22	18.78	179.20

CAM_BACK_RIGHT

Error	t_x	t_y	t_z	θ_{rot}
Mean	-0.34	0.29	0.62	71.44
Median	-0.24	0.27	0.40	44.48
Min	-4.40	-0.28	0.18	1.69
Max	-0.16	1.95	9.94	179.70

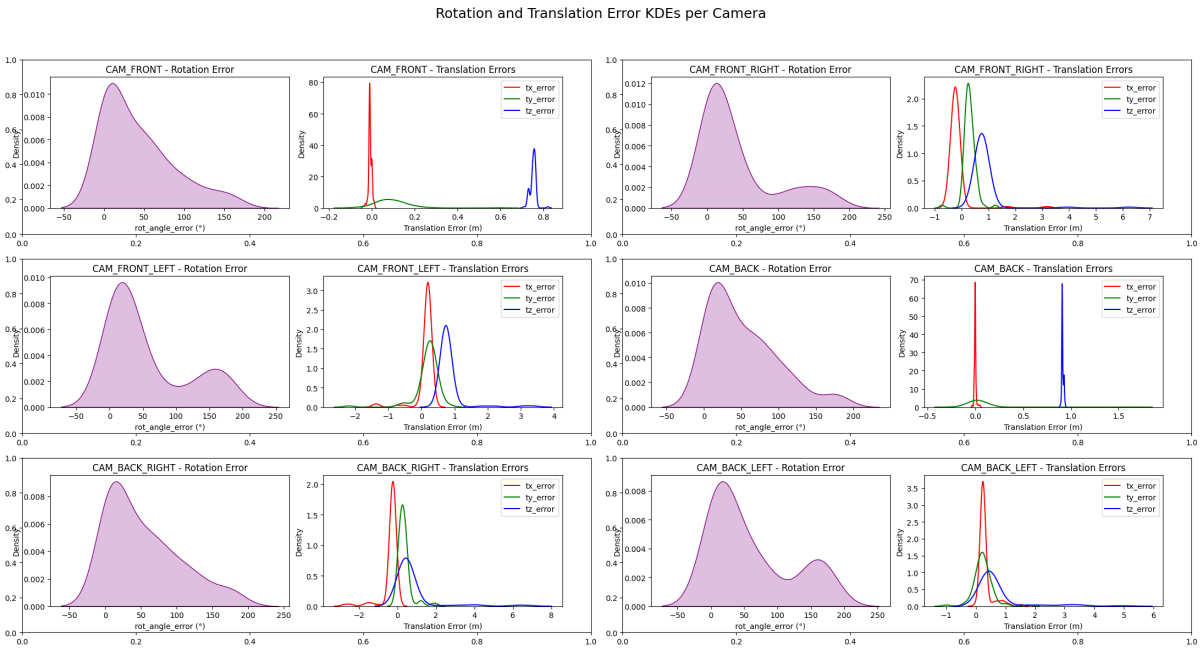


Figure 4.9: Error distribution across the scenes, Modified Symmetric Distance

Camera	LiDAR Frame	Mod Symm Dist
CAM_FRONT	LiDAR planes	0.1440
CAM_FRONT_RIGHT	LiDAR planes	0.1167
CAM_FRONT_LEFT	LiDAR planes	0.1235
CAM_BACK	LiDAR planes	0.1630
CAM_BACK_RIGHT	LiDAR planes	0.1098
CAM_BACK_LEFT	LiDAR planes	0.1072

Table 4.6: Inliers Hungarian Modified Symmetric Distance

4.6.3 Quantitative results for Relative orientation based Hungarian Matching (Jaccard Distance)

Tables 4.74.10 and 4.8 demonstrate that using Jaccard distance for plane matching yields consistent improvements in both rotation and translation errors across all camera views, as well as an increased inlier ratio, compared to the baseline method that relies on Euclidean distance between plane centroids.

Table 4.7: Errors RANSAC Jaccard Distance

CAM_FRONT

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.01	0.14	0.75	37.62
Median	0.00	0.13	0.75	20.37
Min	-0.12	-0.07	0.70	1.01
Max	0.55	0.60	0.81	175.48

CAM_FRONT_LEFT

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.16	0.01	0.81	43.19
Median	0.20	0.25	0.74	14.94
Min	-1.71	-22.04	0.16	1.63
Max	0.67	2.29	3.47	179.69

CAM_FRONT_RIGHT

Error	t_x	t_y	t_z	θ_{rot}
Mean	-0.22	0.10	0.80	37.04
Median	-0.23	0.25	0.75	8.57
Min	-1.10	-14.37	-0.24	0.95
Max	3.03	1.08	6.09	179.89

CAM_BACK

Error	t_x	t_y	t_z	θ_{rot}
Mean	-0.00	0.08	0.89	44.85
Median	0.00	0.07	0.91	26.85
Min	-0.14	-0.14	0.48	0.66
Max	0.18	0.32	0.95	178.95

CAM_BACK_LEFT

Error	t_x	t_y	t_z	θ_{rot}
Mean	0.27	0.06	0.51	54.25
Median	0.24	0.22	0.43	22.34
Min	0.16	-14.58	0.18	1.81
Max	1.39	2.33	3.93	179.83

CAM_BACK_RIGHT

Error	t_x	t_y	t_z	θ_{rot}
Mean	-0.26	0.28	0.41	61.33
Median	-0.24	0.24	0.42	17.67
Min	-0.82	-0.37	-0.43	1.23
Max	-0.15	3.17	1.87	179.85

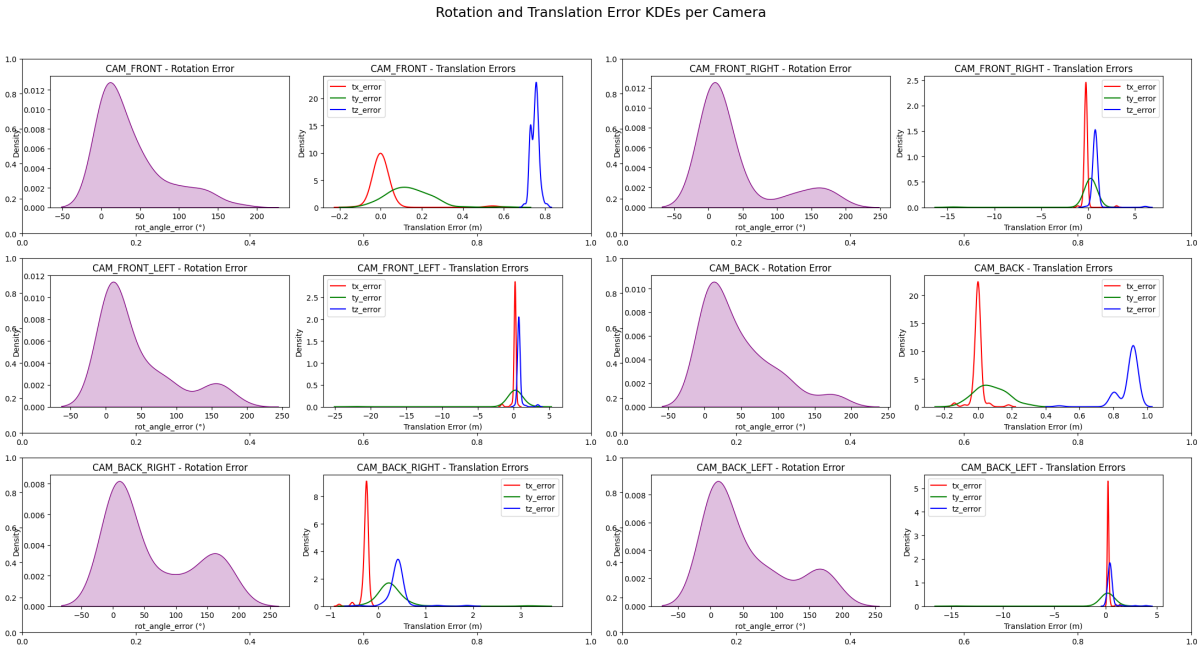


Figure 4.10: Error distribution across the scenes, Jaccard Distance

Camera	LiDAR Frame	Jaccard
CAM_FRONT	LiDAR planes	0.2241
CAM_FRONT_RIGHT	LiDAR planes	0.1629
CAM_FRONT_LEFT	LiDAR planes	0.1634
CAM_BACK	LiDAR planes	0.2151
CAM_BACK_RIGHT	LiDAR planes	0.1498
CAM_BACK_LEFT	LiDAR planes	0.1422

Table 4.8: Inliers Hungarian Jaccard Distance

4.6.4 Error Evaluation

Centroid-Based Hungarian Matching

For the single-plane RANSAC results, only the front camera provides usable outcomes. The front camera primarily involves motion in a single degree of freedom (DOF), which the single-plane method can handle correctly. However, for the other cameras, where the motion occurs in multiple directions, the single-plane approach fails to produce reliable results since it can only estimate two DOF due to the limitation of using a single plane for matching.

In the triple-plane RANSAC approach, cases of high estimation error can generally be attributed to a few key issues. One primary source of error is the incorrect identification of inlier sets during the RANSAC matching process. Since RANSAC seeks to maximize the number of inliers, it may select a larger but incorrect set of correspondences, which results in inaccurate transformation estimation. This issue becomes more pronounced when geometric outliers closely resemble genuine matches, misleading the selection process.

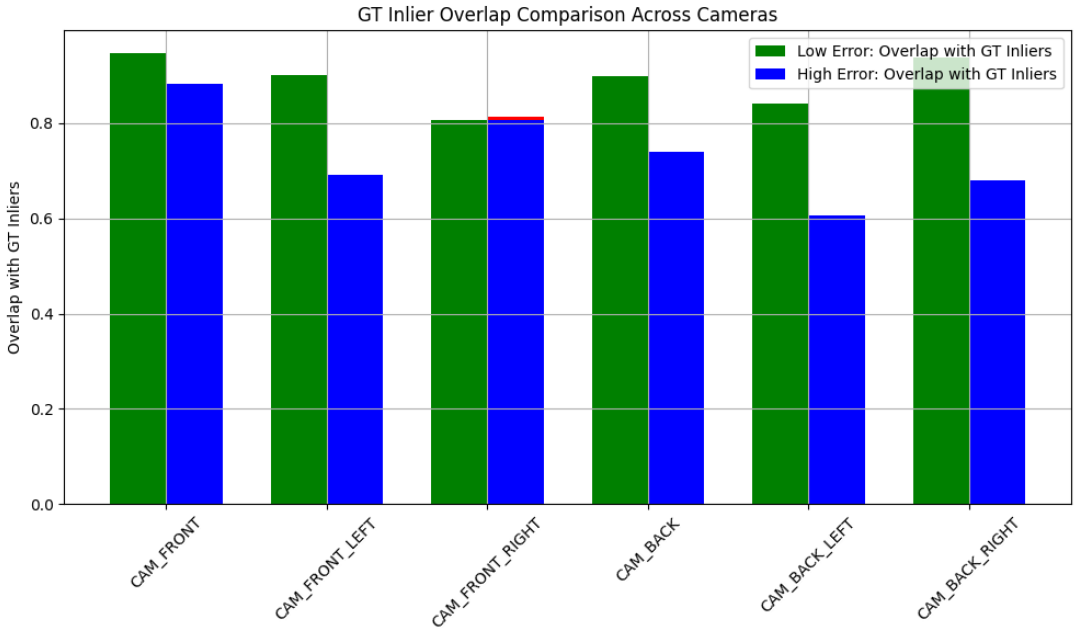


Figure 4.11: Centroid-based RANSAC inlier overlap with actual inliers

The bar plot4.11 shows overlap between RANSAC-selected inliers, obtained through centroid-based plane matching, and the actual ground truth inliers for estimating rotation and translation (R and T) across different camera views. A clear trend emerges: the intersection with ground truth inliers is significantly lower in scenarios where the proposed method results in high calibration error. In contrast, scenes with low calibration error exhibit much higher inlier overlap. This observation suggests a strong correlation between inlier overlap quality and calibration accuracy. Specifically, when the intersection is low, RANSAC is likely failing to identify the correct plane correspondences necessary for accurate pose estimation, leading to suboptimal transformation recovery.

Moreover, in several cases, RANSAC was observed to return some inliers greater than the actual ground truth set, indicating classical failure modes where incorrect maximal matching subsets dominate the sampling process. To address the poor initial performance, the number of RANSAC iterations was increased from the theoretically calculated 4,603 (derived using a success probability of 0.99, an outlier ratio of 0.9, and a minimal sample size of 3 planes) to 20,000. However, this increase did not yield significant improvements in the results. These findings underscore that the combination of low inlier ratio4.4 and weak inlier overlap4.11 severely impairs RANSAC's ability to converge on the correct solution. Therefore, improving the robustness of inlier selection or incorporating additional geometric constraints may be necessary to guide RANSAC reliably in such high-outlier regimes.

Figure 4.13 represents instances where maximum matchings are found that do not intersect with GT inliers.

Plot 4.12 supports this analysis. Observing the left and right subplots, highlighting high-error cases, we observe that single-plane and triple-plane RANSAC of scenes with high errors yield more inliers than scenes with low errors. This overfitting behavior indicates that RANSAC has selected an erroneous but large inlier set, leading to large pose estimation errors despite a seemingly good inlier count. As we already saw, the overlap of RANSAC inliers and actual inliers for scenes with high errors is already less.

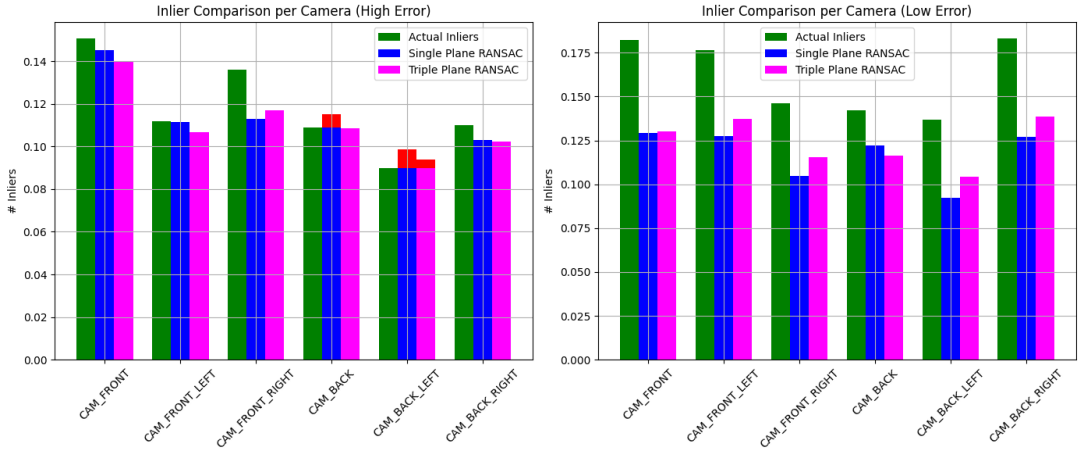
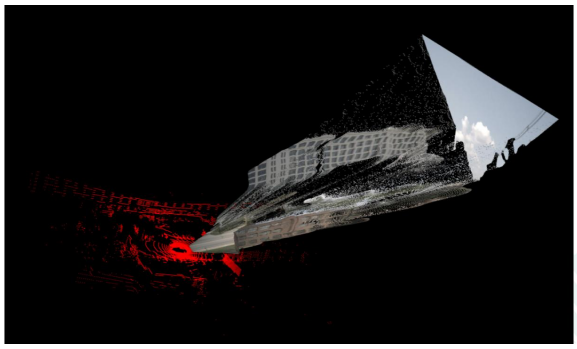
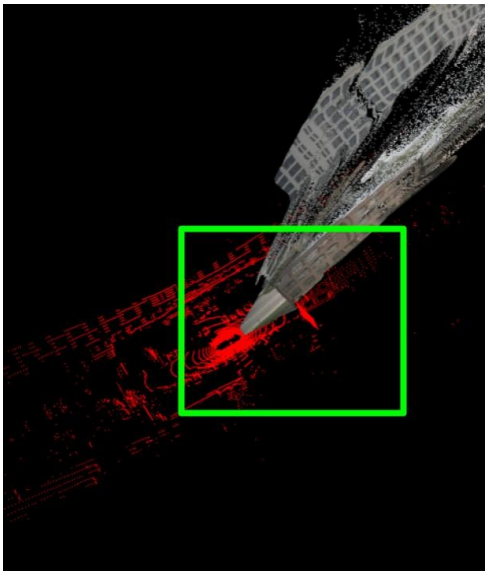


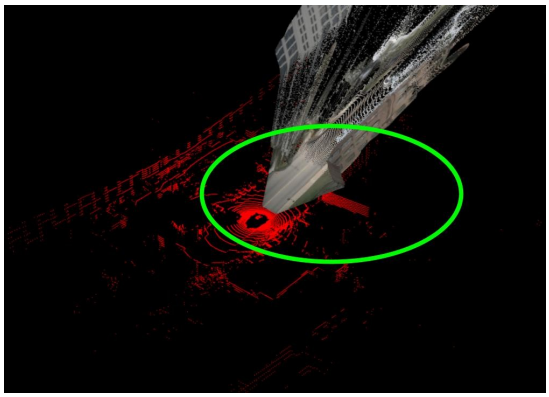
Figure 4.12: Centroid-based RANSAC inlier analysis



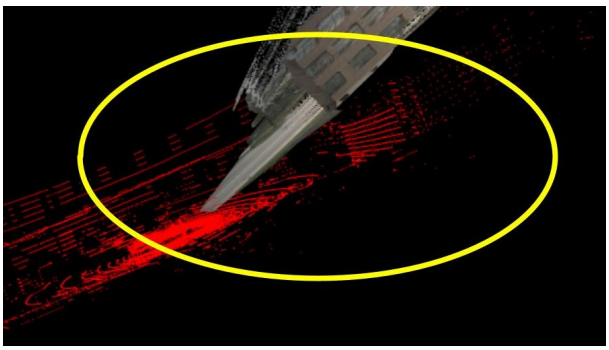
(a) Incorrect maximal Matching 1



(b) Incorrect maximal Matching 2



(c) Incorrect maximal Matching 3



(d) Incorrect maximal Matching 4

Figure 4.13: Incorrect maximal Matchings

The primary reason for low inlier ratios in the calibration pipeline is the presence of noisy or inconsistent plane detections. The Hungarian matching algorithm struggles to produce meaningful correspondences in cluttered scenes, often leading to poor matches. However, Hungarian matching performs well when planar surfaces are clearly and reliably detected in both modalities, leveraging centroid proximity to establish robust one-to-one assignments.

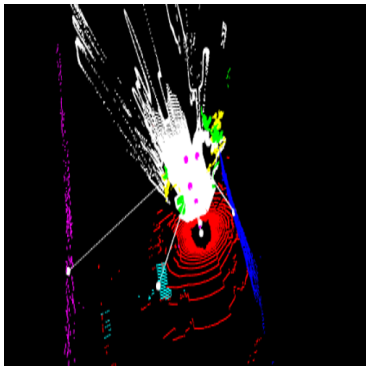
Figure 4.16 illustrates a failure case where the LiDAR detects planes that do not have corresponding counterparts in the camera, or vice versa. Since Hungarian matching enforces one-to-one assignments regardless of geometric overlap or semantic consistency, it produces erroneous matches in such scenarios, ultimately degrading the calibration quality.

In Figure 4.15, the LiDAR plane detections are of high quality, but the camera detections consist primarily of noise. The result is that multiple incorrect camera plane

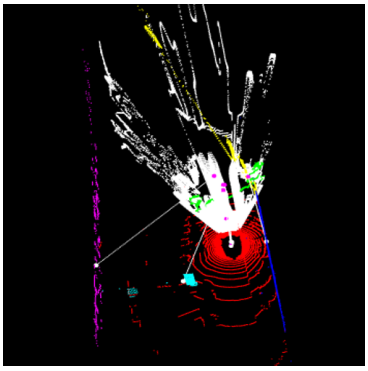
centroids (shown in pink) are matched to valid LiDAR centroids (shown in white), further degrading the correspondence quality. Such mismatches indicate that the visual plane extraction is highly sensitive to lighting conditions and textureless regions.

Figure 4.14 shows a mixed case, containing both well-matched and poorly matched plane pairs. The poorly matched regions exhibit clutter and dense, overlapping detections, whereas accurate matches are observed in areas with minimal noise and clean plane boundaries. This reinforces the importance of reliable plane segmentation as a prerequisite for effective matching.

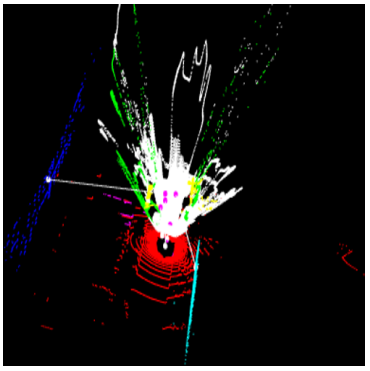
These observations suggest that improving the reliability of plane detections, particularly in the camera domain, and incorporating pre-matching filtering to reject noisy or inconsistent planes can significantly enhance the overall inlier ratio. This refinement is essential to ensure that RANSAC operates on geometrically meaningful correspondences, leading to more robust and accurate extrinsic calibration. Additionally, incorporating geometric constraints or semantic cues into the matching process could mitigate false matches in ambiguous scenarios.



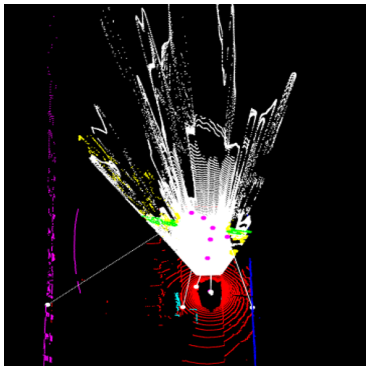
(a) Poor Hungarian matching 1



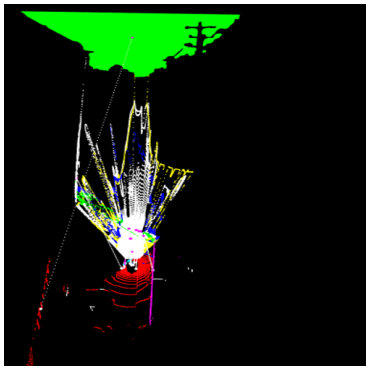
(b) Poor Hungarian matching 2



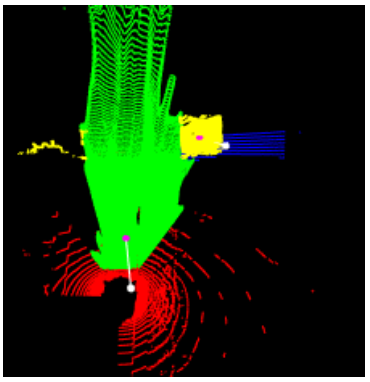
(c) Poor Hungarian matching 3



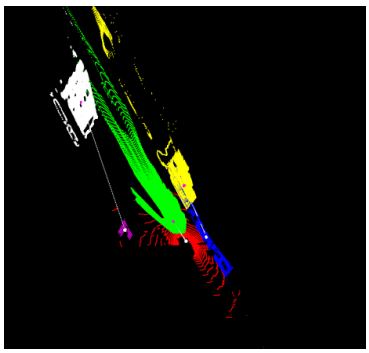
(d) Poor Hungarian matching 4



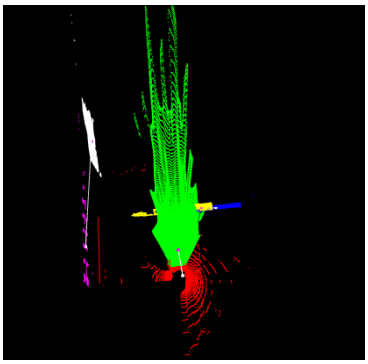
(e) Poor Hungarian matching 5



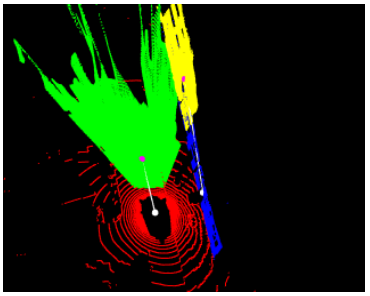
(f) Accurate Hungarian matching 1



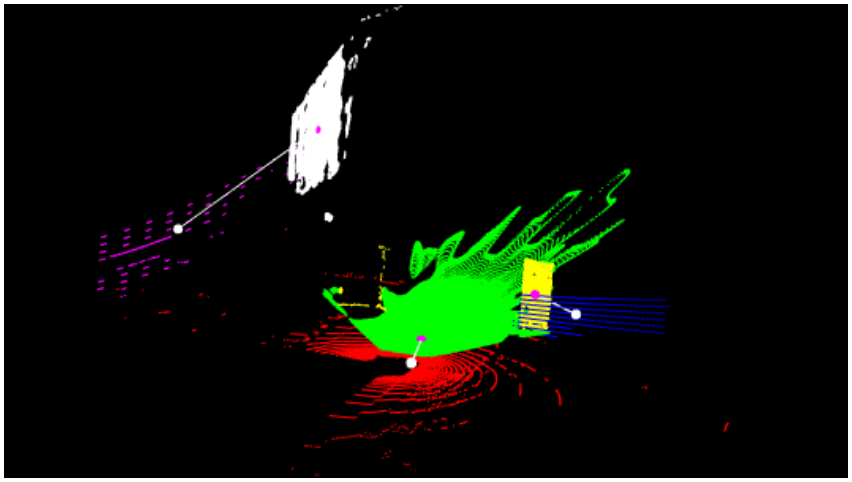
(g) Accurate Hungarian matching 2



(h) Accurate Hungarian matching 3

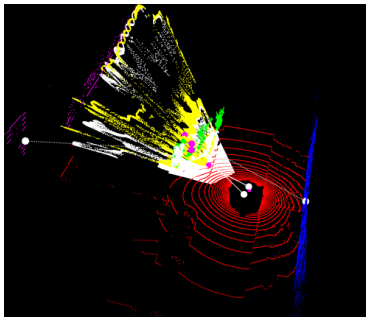


(i) Accurate Hungarian matching 4

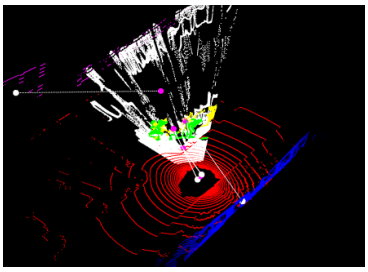


(j) Accurate Hungarian matching 5

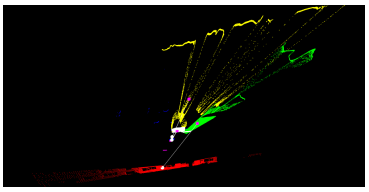
Figure 4.14: visual results of poor and accurate Hungarian matchings CAM FRONT



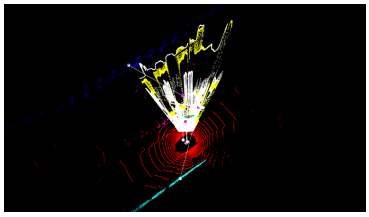
(a) Poor Hungarian matching 1



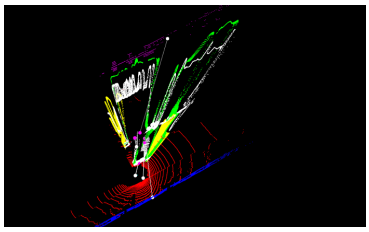
(b) Poor Hungarian matching 2



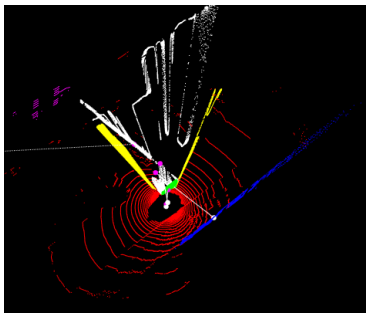
(c) Poor Hungarian matching 3



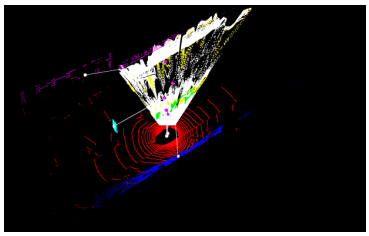
(d) Poor Hungarian matching 4



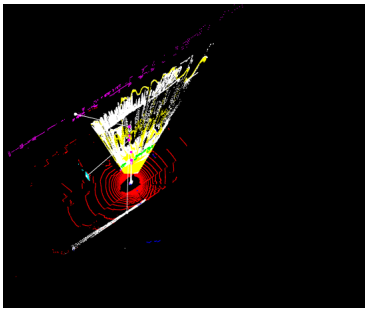
(e) Poor Hungarian matching 5



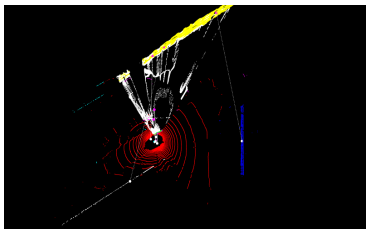
(f) Poor Hungarian matching 6



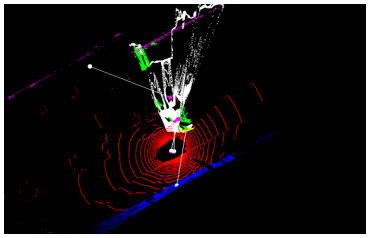
(g) Poor Hungarian matching 7



(h) Poor Hungarian matching 8

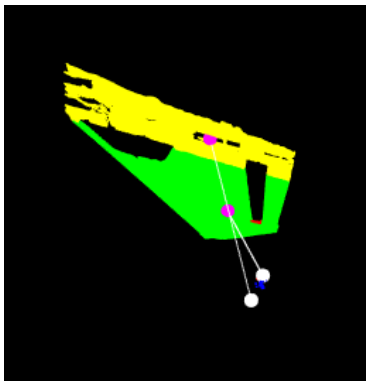


(i) Poor Hungarian matching 9

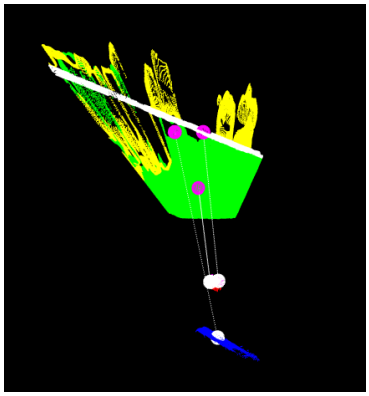


(j) Poor Hungarian matching 10

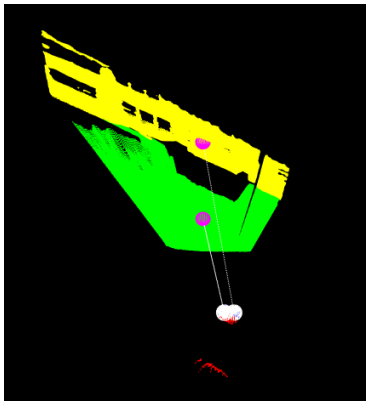
Figure 4.15: visual results of poor Hungarian matching CAM FRONT LEFT



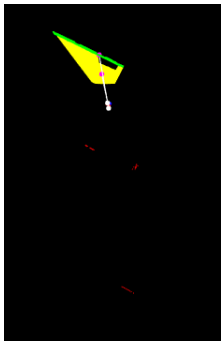
(a) Poor Hungarian matching 1



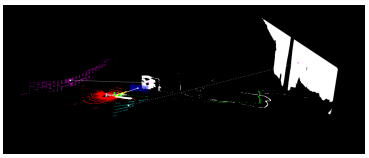
(b) Poor Hungarian matching 2



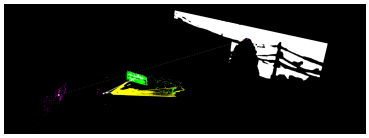
(c) Poor Hungarian matching 3



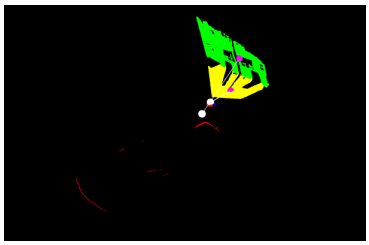
(d) Poor Hungarian matching 4



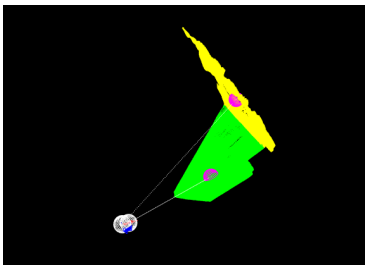
(e) Poor Hungarian matching 5



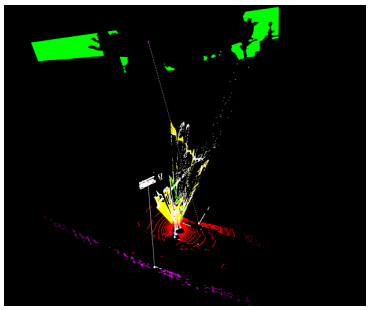
(f) Poor Hungarian matching 6



(g) Poor Hungarian matching 7



(h) Poor Hungarian matching 8



(i) Poor Hungarian matching 9

Figure 4.16: visual results of poor Hungarian matching CAM FRONT RIGHT

Another primary source of high rotational errors in the triple-plane RANSAC method is the occurrence of 180° flips in the estimated rotation matrices. Plots4.8a, 4.8b, 4.8c, 4.8d, 4.8e, 4.8f represent peaks around 180° such errors.These arise due to inherent ambiguities in plane geometry and the under-constrained nature of inter-modality matching. Figure4.17 represents such 180° flips in our method.

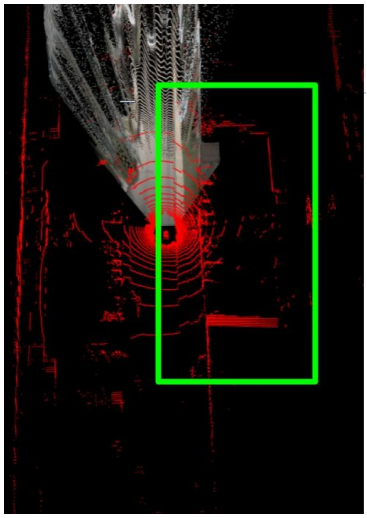
The following reasons might be behind 180° flips

Plane Normal Ambiguity A plane defined by $(\mathbf{n}, d)$ is equivalent to $(-\mathbf{n}, -d)$. If the normal direction is mismatched between LiDAR and camera planes, it can lead to a 180° flip in the estimated rotation. Although even after using proper normal convention, 180° flips are happening.

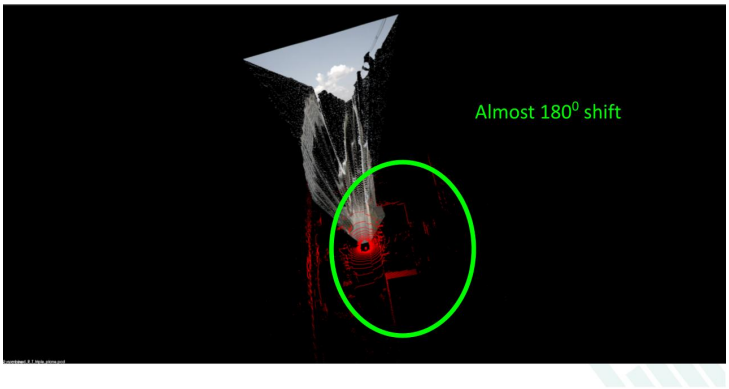
Symmetric Plane Configurations Poorly distributed or symmetric plane sets can admit multiple valid transformations. Optimization may converge to a mirrored (flipped) solution with minimal residual error.

Under-constrained Matching Without point-level correspondences, plane alignment is under-constrained. RANSAC combined with SVD-based estimation may select flipped solutions if no orientation priors or global constraints are applied.

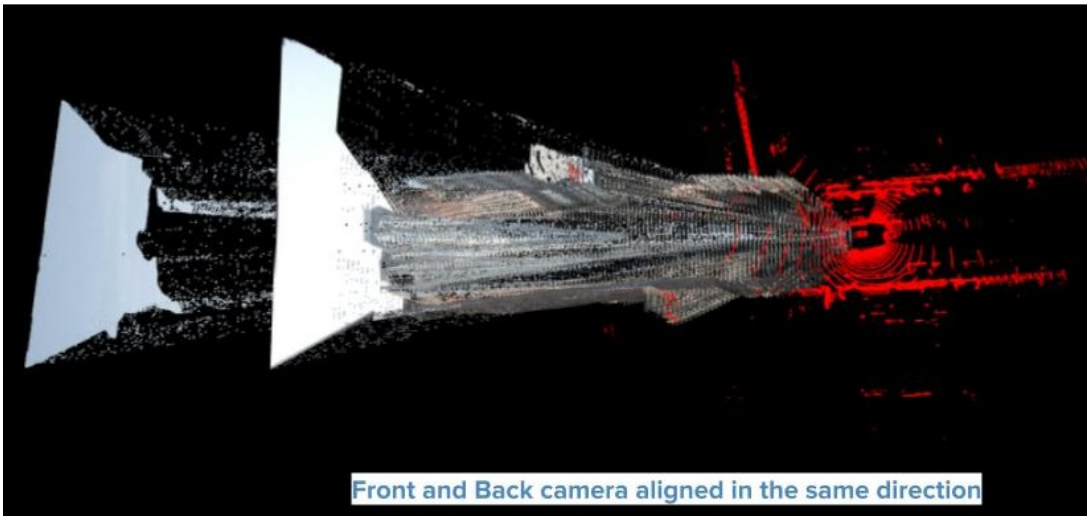
Difficulty in Detection and Open Question These errors are difficult to identify since flipped solutions may still yield high inlier counts when using direction-invariant metrics. Without ground truth or orientation priors, such erroneous results may appear plausible.



(a) 180° flip



(b) 180° flip



(c) Front cam and 180° flipped Back cam

Figure 4.17: visual results of 180° flips

The exact cause behind some of these high-error cases remains unclear and is left as a direction for future investigation.

Moreover, translation errors in some cases can be traced back to inaccuracies in the depth maps used to reconstruct 3D points from camera images. Errors in predicted depths lead to imprecise plane fitting, affecting the accuracy of the computed translation vector. These combined factors result in occasional outlier cases with high rotational and translational errors in the triple-plane RANSAC pipeline. Figure4.18 represents such errors in depth which lead to minor errors in translation and can lead to major errors in rotation

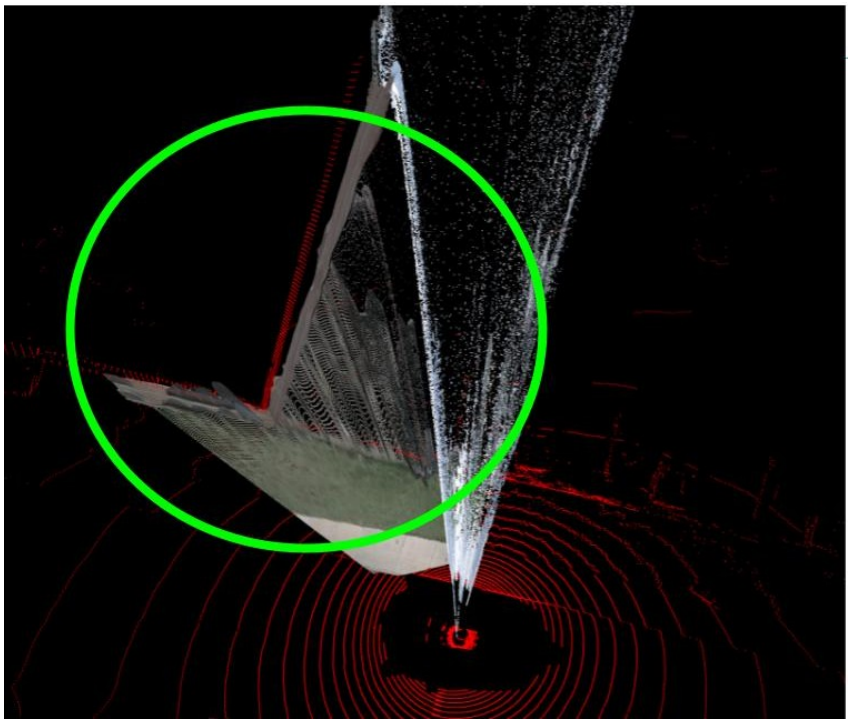


Figure 4.18: Error in depth

Relative Distance Hungarian Matching

In this setting, we employ a relative distance cost, specifically Modified symmetric distance and Jaccard distance, as the cost function for the Hungarian algorithm that matches planes across LiDAR and camera modalities. Compared to the baseline Euclidean distance between plane centroids, these relative cost result in improved performance in both error reduction and inlier ratios across all camera views. As evidenced in Tables 4.7 and 4.8, using Jaccard distance consistently leads to better correspondences and more robust pose estimation. The KDE plots (e.g., Figure **??**) further confirm that error distributions are concentrated around zero, indicating effective alignment in most scenes. Nonetheless, persistent peaks at 180° reveal a common failure mode associated with flipped rotational estimations.

These 180° rotational errors stem from several underlying factors: ambiguities in plane normal directions, symmetric plane configurations that permit mirrored solutions, and the inherently under-constrained nature of inter-modality plane matching without point-level supervision. Even when consistent normal conventions are enforced, RANSAC may still converge to flipped solutions that minimize geometric error due to a lack of orientation priors. Additionally, while generally well-behaved, translation errors can deteriorate due to noisy or inaccurate depth maps on the camera side. Such depth errors lead to suboptimal 3D point reconstruction and plane fitting, propagating into the final transformation estimates.

A deeper analysis reveals that the Hungarian algorithm's performance is highly dependent on the quality of plane detection. Figures 4.15 to 4.16 show that in cluttered scenes or under poor lighting conditions, visual plane detections are noisy or missing, leading to erroneous one-to-one assignments even when using refined distance metrics. These incorrect maximal matchings contribute to a lower inlier ratio and reduce the effectiveness of the RANSAC pipeline. Incorporating pre-matching filters to remove spurious planes, applying semantic or geometric priors, and enhancing camera-based plane segmentation are essential next steps to mitigate such issues and achieve consistently accurate calibration.

4.6.5 Comparative analysis

Table 4.9 presents a comparative analysis between the method proposed by Koide et al. (2021) [12] and our proposed approach, conducted on a single scene from the NuScenes dataset using the front-facing camera. The comparison includes rotation errors (Roll, Pitch, Yaw in degrees) and translation errors (Tx, Ty, Tz in meters) with respect to the ground truth.

Our method utilizes centroid-distance-based Hungarian matching of planar surfaces extracted from LiDAR and camera data, followed by a robust transformation estimation using a triple-plane RANSAC scheme. In contrast to Koide et al.'s method, which reports large rotation deviations (e.g., $-157.54°$ roll, $-82.15°$ yaw), our approach achieves substantially lower rotational errors and competitive translation performance. Most notably, roll and yaw errors are reduced by over two orders of magnitude, indicating improved alignment accuracy.

It is important to emphasize that this analysis is limited to a single scene and serves primarily as a proof-of-concept comparison. For broader generalization, further evaluation across diverse scenes is necessary.

Table 4.9: Error on NuScenes (Front Camera) (Koide et al. (2021) vs Ours)

Method	**Roll**	**Pitch**	**Yaw**	**Tx (m)**	**Ty (m)**	**Tz (m)**
Koide et al. (2021)[12]	-157.54	-59.72	-82.15	-0.264	-0.082	-0.148
Ours (Centroid cost)	2.20	-0.15	0.58	-0.246	0.645	-1.756

Chapter 5

Discussion and Future Work

5.1 Key Findings

For homography-based distance estimation, we demonstrated the ability to accurately estimate both on-road distances and distances at elevated positions above the road plane, provided the object height is known. This approach represents an effective first step toward monocular distance estimation. Additionally, we showed the practical utility of integrating monocular depth estimation modules and leveraging scene geometry such as vanishing points for approximating intrinsic parameters.

In the context of LiDAR-camera cross-calibration, we successfully employed scene geometric cues to achieve accurate extrinsic calibration, without relying on traditional calibration targets or manual intervention. These findings highlight the robustness and practicality of using structural features inherent in the environment for both calibration and metric distance estimation tasks.

5.2 Limitations of the Current Approach

For homography-based distance estimation, our approach relies heavily on the availability of accurate depth maps and precise intrinsic camera parameters. The approximation of intrinsics using vanishing points (VP) introduces additional error, as VP-based intrinsics are often less reliable, especially in cluttered or noisy estimated of intrinsic. Furthermore, our current pipeline does not incorporate any vehicle-specific information, such as dimensions, which could potentially enhance estimation accuracy.

In the context of LiDAR-camera cross-calibration, a key limitation is the reliance on planar surfaces alone for geometric alignment. Other structural features in the scene, such as vanishing points, line intersections, or curved surfaces, are not yet fully exploited.

Additionally, the accuracy of the calibration process is affected by the limitations of the depth estimation modules, especially in cases where depth predictions are unreliable. The current method for detecting planes in the camera frame also depends on the quality of depth estimation; thus, it may fail when depth predictions are inaccurate or inconsistent.

5.3 Future Research Directions and Potential Improvements

To improve the performance and robustness of both the homography-based distance estimation and the LiDAR-camera cross-calibration modules, the following directions are proposed:

Homography Estimation Module:

- **Refine intrinsic estimation:** The current approach to estimating camera intrinsics can be improved. Instead of relying solely on vanishing points, more robust methods should be explored to increase accuracy, especially in cluttered scenes.
- **Reduce or correct depth dependency:** The dependency on depth estimation can be minimized. Alternatively, if depth is used, it is important to correct the scale of the resulting 3D point cloud to ensure metric accuracy.
- **Use vehicle information for scale:** Known dimensions of vehicles on the road can be used to calibrate the scale of the scene. Several existing methods utilize the bird's-eye view of vehicles for this purpose.

LiDAR-Camera Cross-Calibration Module:

- **Improve plane detection:** The current plane detection method can be enhanced to handle noise and partial surfaces more effectively.
- **Incorporate heterogeneous features:** Beyond planes, other geometric cues such as lines, vanishing points, and intersections should be integrated to strengthen the calibration process.
- **Enhance data association through virtual projections and keypoint matching:** Project the LiDAR point cloud onto the camera image plane to generate a virtual depth image, enabling a denser and more structured representation. Then, perform feature matching using keypoints extracted from both the camera image and the projected LiDAR data to achieve more precise and detailed alignment.

- **Leverage deep graph matching:** Represent the scene's geometric structures as heterogeneous graphs and apply deep learning-based graph matching techniques to find robust and accurate correspondences between camera and LiDAR data.

Chapter 6

Conclusion

In conclusion, this thesis explored a approach to LiDAR-camera cross-calibration and monocular homography-based vehicle speed estimation. The proposed framework includes automated plane detection in both LiDAR and camera modalities, graph-based matching using the Hungarian algorithm, and RANSAC-based registration to estimate the extrinsic calibration parameters. Additionally, the homography estimation pipeline uses monocular depth, semantic segmentation, and vanishing point-based intrinsic estimation to recover ground-plane transformations from single images. This enables real-world distance and speed computation using only monocular vision, without reliance on multi-sensor setups.

While our methods show promising results across multiple datasets, several limitations remain. The LiDAR-camera calibration needs further refinement to improve robustness in highly dynamic environments and under low planar visibility. The intrinsic approximation methods, particularly vanishing point and UniDepth-based approaches, require more consistency across scenes, as performance varies depending on scene structure and orientation. In the homography estimation pipeline, estimating homography at specific image heights (e.g., vehicle centers) remains a challenge because it requires accurate knowledge of the corresponding 3D height on the road plane, which is currently unresolved. These issues highlight opportunities for future work in building more adaptive and geometry-aware calibration and estimation systems.

Overall, this thesis lays foundational work for deploying vision-based speed estimation in real-world traffic environments, leveraging road scene geometry, emphasizing automation, cost-efficiency, and cross-sensor compatibility.

References

[1] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," Aug. 2018. arXiv:1802.02611 [cs].

[2] J. Sochor, R. Juránek, J. Španěl, L. Maršík, A. Široký, A. Herout, and P. Zemčík, "Comprehensive dataset for automatic single camera visual speed measurement," *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11621–11631, 2020.

[4] Qilong Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, (Sendai, Japan), pp. 2301–2306, IEEE, 2004.

[5] W. Wang, K. Sakurada, and N. Kawaguchi, "Reflectance Intensity Assisted Automatic and Accurate Extrinsic Calibration of 3D LiDAR and Panoramic Camera Using a Printed Chessboard," *Remote Sensing*, vol. 9, p. 851, Aug. 2017. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute.

[6] C. Yuan, X. Liu, X. Hong, and F. Zhang, "Pixel-Level Extrinsic Self Calibration of High Resolution LiDAR and Camera in Targetless Environments," *IEEE Robotics and Automation Letters*, vol. 6, pp. 7517–7524, Oct. 2021.

[7] X. Liu, C. Yuan, and F. Zhang, "Targetless Extrinsic Calibration of Multiple Small FoV LiDARs and Cameras using Adaptive Voxelization," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–12, 2022. arXiv:2109.06550 [cs].

[8] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Automatic Targetless Extrinsic Calibration of a 3D Lidar and Camera by Maximizing Mutual Information," *Pro-*

ceedings of the AAAI Conference on Artificial Intelligence, vol. 26, pp. 2053–2059, Sept. 2021.

[9] Z. Taylor and J. Nieto, "Automatic Calibration of Lidar and Camera Images using Normalized Mutual Information,"

[10] G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna, "CalibNet: Geometrically Supervised Extrinsic Calibration using 3D Spatial Transformer Networks," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1110–1117, Oct. 2018. arXiv:1803.08181 [cs].

[11] X. Lv, B. Wang, Z. Dou, D. Ye, and S. Wang, "LCCNet: LiDAR and Camera Self-Calibration using Cost Volume Network," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, (Nashville, TN, USA), pp. 2888–2895, IEEE, June 2021.

[12] K. Koide, S. Oishi, M. Yokozuka, and A. Banno, "General, Single-shot, Target-less, and Automatic LiDAR-Camera Extrinsic Calibration Toolbox," Feb. 2023. arXiv:2302.05094 [cs].

[13] C. Maduro, K. Batista, P. Peixoto, and J. Batista, "Estimation of vehicle velocity and traffic intensity using rectified images," in *IEEE International Conference on Image Processing (ICIP)*, 2008.

[14] D. Luvizon, B. Nassu, and R. Minetto, "Vehicle speed estimation by license plate detection and tracking," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[15] J. M. Coughlan and A. L. Yuille, "Manhattan world: Compass direction from a single image by bayesian inference," *Proceedings Ninth IEEE International Conference on Computer Vision*, vol. 2, pp. 941–947, 2003.

[16] L. Piccinelli, Y.-H. Yang, C. Sakaridis, M. Segu, S. Li, L. V. Gool, and F. Yu, "UniDepth: Universal Monocular Metric Depth Estimation," Mar. 2024. arXiv:2403.18913 [cs].

[17] D. Barath and J. Matas, "Progressive-X: Efficient, Anytime, Multi-Model Fitting Algorithm," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, (Seoul, Korea (South)), pp. 3779–3787, IEEE, Oct. 2019.

[18] R. Unnikrishnan and M. Hebert, "Fast Extrinsic Calibration of a Laser Rangefinder to a Camera,"

[19] J. Cui, J. Niu, Z. Ouyang, Y. He, and D. Liu, "ACSC: Automatic Calibration for Non-repetitive Scanning Solid-State LiDAR and Camera Systems," Nov. 2020. arXiv:2011.08516 [cs].

[20] G. Yan, F. He, C. Shi, X. Cai, and Y. Li, "Joint Camera Intrinsic and LiDAR-Camera Extrinsic Calibration," Feb. 2023. arXiv:2202.13708 [cs].

[21] L. Zhou, Z. Li, and M. Kaess, "Automatic Extrinsic Calibration of a Camera and a 3D LiDAR Using Line and Plane Correspondences," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Madrid), pp. 5562–5569, IEEE, Oct. 2018.

[22] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna, "LiDAR-Camera Calibration using 3D-3D Point correspondences," May 2017. arXiv:1705.09785 [cs].

[23] J. Beltrán, C. Guindel, A. d. l. Escalera, and F. García, "Automatic Extrinsic Calibration Method for LiDAR and Camera Sensor Setups," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 17677–17689, Oct. 2022. arXiv:2101.04431 [cs].

[24] J. Jiao, F. Chen, H. Wei, J. Wu, and M. Liu, "LCE-Calib: Automatic LiDAR-Frame/Event Camera Extrinsic Calibration With A Globally Optimal Solution," Mar. 2023. arXiv:2303.09825 [cs].

[25] J. Levinson and S. Thrun, "Automatic Online Calibration of Cameras and Lasers," in *Robotics: Science and Systems IX*, Robotics: Science and Systems Foundation, June 2013.

[26] Y. Zhu, C. Zheng, C. Yuan, X. Huang, and X. Hong, "CamVox: A Low-cost and Accurate Lidar-assisted Visual SLAM System," Nov. 2020. arXiv:2011.11357 [cs].

[27] T. Ma, Z. Liu, G. Yan, and Y. Li, "CRLF: Automatic Calibration and Refinement based on Line Feature for LiDAR and Camera in Road Scenes," Mar. 2021. arXiv:2103.04558 [cs] version: 1.

[28] Z. Luo, G. Yan, and Y. Li, "Calib-Anything: Zero-training LiDAR-Camera Extrinsic Calibration Method Using Segment Anything," June 2023. arXiv:2306.02656 [cs].

[29] X. C. He and N. H. C. Yung, "A novel algorithm for estimating vehicle speed from two consecutive images," *IEEE Workshop on Applications of Computer Vision (WACV)*, 2007.

[30] F. Cathey and D. Dailey, "A novel technique to dynamically measure vehicle speed using uncalibrated roadway cameras," in *Intelligent Vehicles Symposium*, 2005.

[31] L. Grammatikopoulos, G. Karras, and E. Petsa, "Automatic estimation of vehicle speed from uncalibrated video sequences," in *International Symposium on Modern Technologies, Education and Professional Practice in Geodesy and Related Fields*, 2005.

[32] X. You and Y. Zheng, "An accurate and practical calibration method for roadside camera using two vanishing points," *Neurocomputing*, 2016.

[33] M. Dubská, J. Sochor, and A. Herout, "Automatic camera calibration for traffic understanding," in *British Machine Vision Conference (BMVC)*, 2014.

[34] T. Schoepflin and D. Dailey, "Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 2, pp. 90–98, 2003.

[35] P. Filipiak, B. Golenko, and C. Dolega, "Nsga-ii based auto-calibration of automatic number plate recognition camera for vehicle speed measurement," in *Applications of Evolutionary Computation (EvoApplications)*, 2016.

[36] A. Nurhadiyatna *et al.*, "Improved vehicle speed estimation using gaussian mixture model and hole filling algorithm," in *International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2013.

[37] I. Sina *et al.*, "Vehicle counting and speed measurement using headlight detection," in *International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2013.

[38] V.-H. Do *et al.*, "A simple camera calibration method for vehicle velocity estimation," in *International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2015.

[39] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, UK: Cambridge University Press, 2 ed., 2004.

[40] "OpenCV: Feature Matching + Homography to find Objects." Available: `https://docs.opencv.org/4.x/d1/de0/tutorial_py_feature_homography.html`.

[41] Y. Ma, S. Soatto, J. Kosecká, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*, vol. 26 of *Interdisciplinary Applied Mathematics*. New York, NY, USA: Springer, 2003.

[42] W. Yin, C. Zhang, H. Chen, Z. Cai, G. Yu, K. Wang, X. Chen, and C. Shen, "Metric3D: Towards Zero-shot Metric 3D Prediction from A Single Image," July 2023. arXiv:2307.10984 [cs].

[43] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, "ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth," Feb. 2023. arXiv:2302.12288 [cs].

[44] R. Orghidan, J. Salvi, M. Gordan, and B. Orza, "Camera calibration with two or three vanishing points,"

[45] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[46] L. Liebe, F. Sauerwald, S. Sawicki, M. Schneider, L. Schuhmann, T. Buz, P. Boes, A. Ahmadov, and G. d. Melo, "FARSEC: A Reproducible Framework for Automatic Real-Time Vehicle Speed Estimation Using Traffic Cameras," Sept. 2023. arXiv:2309.14468 [cs].

[47] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[48] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2012.