

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology



Master's Thesis

Learning to Traverse: A Cross-Modal Comparison of Terrain Traversability Estimation Methods

Salvin George

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Sebastian Houben
Dr. Dirk Schulz (Fraunhofer FKIE)
Carlos Andrés Tampier Cotoras (Fraunhofer FKIE)
Allen Isaac Jose (Fraunhofer FKIE)

September 2026

© 2026 Salvin George. All Rights Reserved

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work. The report was, in part, written with the help of the AI assistants Claude, Codex, and Grammarly as described in the appendix. I am aware that content generated by AI systems is no substitute for careful scientific work, which is why all AI-generated content has been critically reviewed by me, and I take full responsibility for it.

Date	Salvin George

Abstract

Outdoor robots must judge whether the terrain ahead is safe to cross before committing to a path. This thesis compares camera-only, LiDAR-only, and fused traversability estimation under a unified evaluation protocol. The evaluation uses GA-Nav and WVN as camera methods, FASTC for LiDAR, and cost-level fusion on a shared bird's-eye-view grid across four domains: RELLIS-3D, GOOSE, and the forest and campus domains of a custom FKIE dataset. The camera-only condition projects semantic predictions onto the ground plane using monocular depth estimation. Despite producing useful image-plane semantic predictions, this camera-only bird's-eye-view (BEV) projection performs weakly across all four domains, with mean intersection-over-union (mIoU) between 0.0004 and 0.0396. In the fusion condition, the same semantic predictions are instead transferred using LiDAR geometry through point-painting, anchoring the camera evidence to measured returns and giving it high recall within the camera-LiDAR overlap region. The LiDAR-only baseline covers nearly the entire map, and its accuracy depends strongly on the domain. Obstacle recall reaches 0.945 on RELLIS-3D and falls to 0.107 on the out-of-distribution campus domain. Fusion adds little where the LiDAR baseline is already strong and raises campus obstacle recall from 0.107 to 0.169 where that baseline fails. A simple fixed-weight blend captures nearly all observed fusion gains. No rule that instead weights each modality by its own per-cell confidence signal consistently outperforms this fixed blend across domains and metrics, even though one fixed-weight variant (camera-heavy on WVN) trades small losses on some metrics for a gain on obstacle recall. These results are corroborated by structured cross-domain failure-mode analysis and by embedded benchmarking on a Jetson AGX Thor: the fused GA-Nav+FASTC pipeline runs at 9.23 Hz (sequential estimate from separately measured components), while the single-method TensorRT pipelines run faster in isolation (FASTC 62.5 Hz, WVN 44.3 Hz, GA-Nav 26.2 Hz), as expected given fusion composes multiple stages. Fusion also carries calibration, compute, and integration costs, most of which can be skipped at deployment time wherever the LiDAR baseline is already known to be reliable. Together, these results suggest that coverage and LiDAR baseline strength provide useful pre-deployment indicators of whether camera-LiDAR fusion is likely to justify its additional cost.

Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Challenges and difficulties . . . 2

1.2.1 Perceptual ambiguity of unstructured terrain . . . 2

1.2.2 Complementary but imperfect sensors . . . 2

1.2.3 Scarcity of ground truth and standardized benchmarks . . . 3

1.3 Problem statement . . . 3

1.3.1 Research questions . . . 3

1.3.2 Scope of the comparison . . . 4

1.3.3 Thesis contributions . . . 4

1.3.4 Thesis structure . . . 5

2 State of the art **6**

2.1 Problem framing and surveys . . . 6

2.2 Vision-based traversability estimation . . . 6

2.3 LiDAR-based traversability estimation . . . 7

2.4 Cross-modal fusion approaches . . . 8

2.5 Benchmark datasets for off-road perception . . . 9

2.6 Foundation models and language-guided navigation . . . 9

2.7 Limitations of previous work . . . 10

3 Methodology **11**

3.1 Setup . . . 11

3.2 Sensor calibration . . . 13

3.2.1 Sensor configuration . . . 13

3.2.2 Camera intrinsic calibration . . . 14

3.2.3 Camera-LiDAR calibration . . . 15

3.3 Calibration method comparison criteria . . . 18

3.4 Datasets . . . 18

3.4.1 Public benchmark datasets . . . 19

3.4.2 Custom ROVO dataset . . . 21

3.4.3 Custom ROVO dataset pre-processing and annotation . . . 22

4 Solution **27**
4.1 Proposed algorithm . . . 27
4.1.1 Shared bird's-eye-view grid . . . 27
4.1.2 Condition A: Camera-only semantics via monocular depth projection . . . 29
4.1.3 Point-painting as a fusion projection choice . . . 30
4.1.4 Condition B: LiDAR-only traversability . . . 32
4.1.5 Condition C: Cost-level fusion . . . 33
4.2 Implementation details . . . 34
4.2.1 Method class architecture . . . 34
4.2.2 Sequential evaluation protocol . . . 34
4.2.3 Checkpoint provenance and leak avoidance . . . 34
4.3 Deployment optimisation on the Jetson AGX Thor . . . 35
4.3.1 TensorRT deployment boundaries . . . 35
4.3.2 FASTC TensorRT deployment . . . 35
4.3.3 WVN TensorRT deployment . . . 37
4.3.4 GA-Nav TensorRT deployment . . . 37
4.3.5 Fusion and point-painting boundary . . . 37
4.3.6 Limitations of full-pipeline conversion . . . 37

5 Evaluation and results **39**
5.1 Sensor calibration . . . 39
5.1.1 Camera intrinsic calibration . . . 39
5.1.2 Camera-LiDAR extrinsic calibration . . . 40
5.1.3 Method comparison . . . 44
5.1.4 Selected calibration and system integration . . . 45
5.2 Predictive evaluation metrics . . . 45
5.3 Custom ROVO dataset evaluation . . . 46
5.4 Cross-dataset accuracy comparison . . . 47
5.4.1 Dataset composition . . . 47
5.4.2 Headline cross-dataset results . . . 48
5.4.3 Point-painted camera leg (fusion camera projection) . . . 50
5.5 Computational cost and deployment performance . . . 53
5.5.1 Benchmark protocols . . . 54
5.5.2 Desktop GPU stage-wise baseline . . . 54
5.5.3 TensorRT parity validation . . . 55
5.5.4 Matched PyTorch–TensorRT pipeline comparison . . . 56
5.5.5 Memory footprint and model size . . . 57
5.5.6 Corrected Condition C fusion runtime . . . 57
5.5.7 Deployment summary . . . 59

6 Coverage, competence, and the conditions for camera contribution in fusion 61
6.1 The coverage wall . . . 62
6.2 Contribution within coverage . . . 63
6.2.1 Coverage versus competence . . . 64
6.2.2 The effect of class-imbalance on GA-Nav's precision . . . 67
6.2.3 Range-dependent usability of the point-painted fusion leg . . . 68
6.2.4 Robustness of the fusion gain to evaluation-frame selection . . . 69
6.3 An oracle ceiling on camera contribution . . . 70
6.4 Fusion rule selection by backbone . . . 70
6.5 Reliability of confidence signals across domains . . . 72
6.6 The recall–precision trade across three safety regimes . . . 75
6.7 Domain shift and spatial coverage in campus fusion . . . 77
6.7.1 Effect of BEV densification alone . . . 77
6.7.2 Fusion-rule dependence of camera contribution . . . 78
6.7.3 Coupling between densification and fusion trust . . . 79
6.7.4 Domain-shift limits on camera contribution . . . 81
6.8 Summary . . . 81

7 Discussion 82
7.1 Generalisability of the findings . . . 82
7.1.1 Findings by evidential status . . . 82
7.1.2 Scoring the pre-registered predictions . . . 84
7.2 Limitations . . . 84
7.3 Failure-mode analysis . . . 86
7.3.1 Range . . . 86
7.3.2 Terrain . . . 87
7.3.3 Lighting . . . 87
7.3.4 Obstacle geometry . . . 88
7.3.5 Sensor density and configuration . . . 90
7.3.6 Summary of the failure-mode analysis . . . 91
7.4 Relation to the broader fusion literature . . . 92
7.5 From qualitative probe to quantitative finding . . . 92
7.6 Future work . . . 94
7.7 Deployment implications . . . 94

8 Conclusion 96
8.1 Summary of contributions . . . 96
8.2 Answering the research questions . . . 98
8.2.1 Comparing the modalities and where fusion helps . . . 98
8.2.2 What fusion costs . . . 99

8.3 Reflection on the original deliverables . . . 99
8.4 Outlook . . . 100

Appendix A Full fusion rule sweep results **101**
A.1 Fusion rule sweep (forest and RELLIS-3D) . . . 101
A.2 Obstacle-override threshold sweep . . . 101
A.3 Campus third-domain results . . . 101

Appendix B Deployment benchmark details **108**
B.1 FASTC ONNX export compatibility adaptations . . . 108
B.2 GA-Nav TensorRT pipeline benchmark . . . 109
B.3 WVN TensorRT pipeline benchmark . . . 109
B.4 FASTC TensorRT pipeline benchmark . . . 109
B.5 TensorRT parity validation details . . . 109
B.6 Network-only and image-plane TensorRT benchmarks . . . 109
B.7 Desktop GPU memory and model-size baseline . . . 109
B.8 Fusion-stage timing details . . . 109
B.9 Matched PyTorch–TensorRT memory diagnostics . . . 114

Appendix C Checkpoint provenance **116**

Appendix D Description of AI-generated content **117**

References **118**

1

Introduction

Unmanned systems are increasingly deployed across a wide range of domains, from autonomous underwater vehicles to robots used in extra-planetary exploration. However, the ability to identify traversable regions on the Earth's surface presents a wide variety of challenges. The types of terrains found on the Earth's surface are widely different and diverse. A plain surface could semantically be asphalt, concrete, mud, swamp, cobble, or even snow, and so on. Even if one could identify the surface texture, how could a system identify grass, which is traversable, and bushes or tree stubs, which could be obstacles depending on the platform?

Most prominent autonomous mobile robots today are tasked with solving navigation in structured environments. These systems need to identify other cars, pedestrians, and lanes for successful navigation, but because their work is mostly based on structured environments, the problem is easier. In unstructured environments, the problem is fundamentally different: there are no lane markings, no fixed catalogue of object classes, and no consistent visual appearance to anchor on. The same patch of ground can be traversable gravel in one lighting condition and, moments later, appear visually identical to a shadowed patch of loose sand that is not. Consequently, unstructured-terrain navigation cannot rely on the closed-set, appearance-driven assumptions that work well in cities, and instead must reason jointly about what a surface looks like and what it might actually be.

Terrain traversability estimation can be defined as the ability of a mobile robot to compute the feasibility and/or effort associated with the traversal of a certain portion of terrain [1].

1.1 Motivation

Autonomous ground platforms are increasingly deployed in domains where the terrain cannot be curated in advance: search-and-rescue, agriculture, forestry, planetary exploration, and off-road defence and inspection tasks all require a robot to decide, on the fly, whether a given patch of ground can be safely crossed [2, 3]. Unlike on-road driving, where the drivable area is largely determined by lane geometry, off-road traversability is a continuous, terrain-dependent property that changes with vegetation, moisture, slope, and clutter. Misjudging it can leave a robot stuck, tipped, or damaged far from assistance. This makes accurate, real-time traversability estimation a prerequisite for any level of autonomy in unstructured environments, rather than a convenience feature [4].

A large part of the recent traversability literature commits early to a single sensing modality. Approaches are typically purely vision-based semantic segmentation such as GANav [5], purely geometric LiDAR-based, or learned self-supervised methods trained from proprioceptive robot experience [6–8]. Each choice comes with a different trade-off. Cameras provide dense semantic cues (this looks like mud, that looks like grass), but they degrade under poor lighting and cannot directly measure geometry; LiDAR measures geometry precisely regardless of illumination, but on its own it cannot distinguish a load-bearing gravel path from visually identical loose sand at the same slope. Fusion approaches attempt to combine both [9–12], but the reported gains are usually demonstrated on a single platform and a single dataset, making it difficult to know whether a fusion strategy is actually necessary for a given application or whether a single, cheaper modality would have been sufficient. This thesis is motivated by that gap. Rather than proposing another isolated method, it directly and systematically compares a camera-only semantic approach, a LiDAR-only geometric approach, and a cost-level fusion (combining each modality's per-cell traversability cost through confidence-aware weighting, without fusing raw sensor data or intermediate features) of the two, holding platform, sensor rig, and evaluation protocol constant within each of four evaluation domains, so that the practical value of fusion can be assessed rather than assumed.

1.2 Challenges and difficulties

This section discusses three factors that make terrain traversability estimation substantially harder than perception in structured environments: the perceptual ambiguity of unstructured terrain, the complementary but imperfect nature of camera and LiDAR sensing, and the scarcity of ground truth (GT) and standardized benchmarks in this domain.

1.2.1 Perceptual ambiguity of unstructured terrain

Unlike the closed-set object catalogues used in structured-environment perception (cars, pedestrians, lane markings), unstructured terrain does not decompose cleanly into a fixed vocabulary of classes. The same semantic label, e.g. "grass", can be traversable when short and dry but effectively impassable when tall, wet, or hiding a ditch; conversely, visually similar textures (mud versus wet asphalt, gravel versus scree) can have very different load-bearing and slip properties. This ambiguity means that appearance alone, and geometry alone, are each individually insufficient, which motivates evaluating them side by side rather than assuming either is enough [13].

1.2.2 Complementary but imperfect sensors

Cameras and LiDAR fail in different, largely uncorrelated ways. Cameras lose reliability under direct sun glare, deep shadow, dust, or at night, and provide no direct depth or slope information; the Ouster OS-128 LiDAR used in this work, on the other hand, is largely lighting-invariant but produces a comparatively sparse point cloud at range (sensor configuration detailed in Section 3.2.1), struggles to resolve thin or low-lying vegetation, and carries no semantic information of its own. Combining the two requires an accurate camera–LiDAR extrinsic calibration (Chapter 3) so that semantic labels from the

camera and geometric measurements from the LiDAR can be associated with the same physical surface point; any residual calibration error directly degrades the quality of the fused traversability estimate.

1.2.3 Scarcity of ground truth and standardized benchmarks

Public unstructured-terrain datasets such as RELLIS-3D [14], RUGD [15], and GOOSE [16] have substantially improved the availability of labelled off-road data, but they are collected on platforms, sensor configurations, and geographic terrain types that do not necessarily match the target robot and deployment site used in this work. Because traversability additionally depends on platform-specific properties (ground clearance, wheel or track configuration, mass), labels generated for one robot do not transfer cleanly to another. This motivated the collection and annotation of a custom dataset for the ROVO platform used in this thesis. Its final annotation used direct dual-modality labelling. RGB images were labelled in the image plane, and LiDAR scans were labelled directly in bird's-eye-view (BEV), rather than projecting camera labels onto the LiDAR point cloud. Section 3.4.3 describes how this BEV grid is constructed from LiDAR scans. Calibration and projection errors nevertheless remain part of the experimental setup for the camera projection and fusion stages rather than a solved prerequisite [17].

1.3 Problem statement

Building on the challenges outlined above, this section defines the problem addressed by this thesis: the research questions it answers, the scope within which the comparison is conducted, and the resulting contributions.

1.3.1 Research questions

This thesis addresses the following questions:

1. How does a camera-only, semantic-segmentation-based traversability estimate compare to a LiDAR-only, geometry-based estimate on the same unstructured outdoor terrain and the same robot platform?
2. Does cost-level fusion of the two modalities produce a measurably more reliable traversability estimate than either modality alone, and under what terrain conditions does the advantage of fusion appear or disappear?
3. What is the practical cost, in calibration effort, computational load, and implementation complexity, of achieving that fusion, and how do these costs weigh against the accuracy gains documented in this thesis?

For the first question above, the custom FKIE forest and campus domains use one fixed platform and sensor rig, so that any difference between them is not attributable to rig geometry; RELLIS-3D and GOOSE, collected on different platforms with different sensor placements, instead test whether the same qualitative pattern holds across genuinely different rigs. Because camera-LiDAR overlap, and therefore

how much of the scene fusion can actually draw on, is set by mounting height, pitch, and field of view, a rig with substantially different geometry would plausibly shift the exact numbers reported here, even if the underlying mechanism (coverage bounds fusion contribution) is expected to generalise.

1.3.2 Scope of the comparison

The camera-only condition in this comparison uses UniDepth [18] monocular depth projection and does not use LiDAR support. LiDAR-supported point-painting [19] is used only for the camera leg of the cost-level fusion condition; the two should not be treated as the same camera evaluation.

The comparison in this thesis evaluates camera-semantic, LiDAR-based, and fused traversability estimation under a common scoring protocol. Within each evaluation domain, the methods operate on the same synchronized sensor data, shared BEV grid, class taxonomy, and scoring footprint. The study covers the RELLIS-3D and GOOSE public domains, together with forest and campus recordings collected on the Fraunhofer ROVO platform using an IDS uEye camera and Ouster OS-128 LiDAR; GOOSE serves as an independent cross-domain generalisation probe and receives lighter follow-up analysis than RELLIS-3D, forest, and campus. This scoping choice was made deliberately to manage the thesis's overall length. GOOSE was evaluated under the same protocol as the other three domains, so its results are included as a further generalisation check, but it was not analysed in the same depth to keep the primary fusion narrative focused on RELLIS-3D, forest, and campus. Three conditions are compared throughout this thesis: the camera-only condition (Condition A), which projects the camera's UniDepth predictions to BEV without any LiDAR support; the LiDAR-only condition (Condition B); and the fusion condition (Condition C), where point-painting is used to bring the camera evidence onto the LiDAR-supported BEV grid. The short labels A, B, and C are used throughout this thesis as compact references to these three conditions, particularly in tables, figures, and cross-references. Fusion is evaluated through fixed-weight, confidence-aware, and rule-based cost-level combinations. This design makes it possible to distinguish improvements caused by fusion from those caused by domain, baseline strength, or sensor configuration. Estimation methods that rely on proprioceptive, self-supervised, or simulation-only training [8, 20] are discussed in Chapter 2 for context but are not re-implemented or benchmarked here.

1.3.3 Thesis contributions

This thesis contributes: (i) a camera–LiDAR extrinsic calibration methodology for the ROVO platform, comparing three calibration approaches and selecting the best-performing one, applied to establish the sensor rig used throughout this work (Chapter 3); (ii) a custom-annotated terrain traversability dataset generated from that calibrated rig, with its ground-truth conventions described in Chapter 3 and known limitations discussed in Section 7.2; and (iii) a controlled, cross-modal comparison of camera-only, LiDAR-only, and fused traversability estimation on that dataset, together with a discussion of the practical trade-offs each approach entails.

1.3.4 Thesis structure

The remainder of this thesis is organised as follows. Chapter 2 reviews prior work on camera-based, LiDAR-based, and fusion approaches to traversability estimation. Chapter 3 describes the experimental methodology, covering the ROVO platform's hardware and sensor calibration, the evaluation criteria, and the datasets used. Chapter 4 presents the shared BEV representation and the camera-only, LiDAR-only, and fusion pipelines compared in this thesis, including the fusion rules evaluated. Chapter 5 reports the calibration results and the headline predictive metrics across all four evaluation domains. Chapter 6 extends these results with a focused analysis of camera coverage, competence, and the specific conditions under which fusion helps, going beyond the cross-dataset comparison in Chapter 5. Chapter 7 discusses generalisability, limitations, failure modes, and relation to prior work. Chapter 8 concludes with a summary of contributions and directions for future work.

2

State of the art

Terrain traversability estimation has been studied from three broadly distinct angles: purely visual (camera) methods that reason about terrain appearance, purely geometric (LiDAR) methods that reason about terrain shape, and fusion methods that attempt to combine the two. This chapter reviews representative work in each category, followed by the benchmark datasets commonly used to train and evaluate these methods, and a brief note on the emerging but distinct paradigm of foundation-model- and language-guided navigation. The chapter closes by identifying the specific gaps in the literature that this thesis addresses.

2.1 Problem framing and surveys

Several surveys have mapped out the traversability estimation problem and its terminology. Sevastopoulos and Konstantopoulos [3] and Borges et al. [2] both provide broad taxonomies of traversability estimation methods for mobile and ground vehicles, organising the field along sensor modality, output representation (binary mask, cost map, risk map), and learning paradigm (classical geometric, supervised, self-supervised). Shu et al. [4] and Arafin et al. [21] update this picture with more recent trends, in particular the shift from hand-engineered geometric features toward learned and hybrid multimodal approaches, and both explicitly flag sensor fusion and cross-domain generalisation as open problems rather than solved ones. Wijayathunga et al. [13] and Benrabah et al. [17] focus specifically on the challenges of scene understanding in unstructured outdoor environments and on how traversability *risk* should be assessed and reported, respectively; the latter is a useful reminder that a traversability estimate is only as useful as the confidence or risk information attached to it. E. Carvalho et al. [1] narrow the scope to forestry applications, illustrating how traversability estimation requirements shift with domain (dense canopy, deformable ground cover) even within the broader unstructured-terrain problem. Taken together, these surveys motivate the definition of terrain traversability estimation used in Chapter 1 and confirm that a controlled comparison of modalities, rather than a single new algorithm, remains a needed contribution to the field.

2.2 Vision-based traversability estimation

Camera-only approaches estimate traversability from the semantic and textural appearance of the terrain. GANav [5] introduced a group-wise attention mechanism for coarse-grained terrain segmentation,

classifying regions by navigability level rather than object identity, and reported mIoU improvements of 2.25–39.05% on RUGD and 5.17–19.06% on RELLIS-3D over prior segmentation networks, together with reduced trajectory roughness when deployed on Jackal and Husky platforms. Trinity [22] extends this direction by jointly performing class-specific semantic segmentation and class-agnostic terrain segmentation in a single transformer-based network, trained in part on a synthetic dataset (RUGDSynth) to address the scarcity of labelled unstructured-terrain imagery.

A parallel line of work avoids manual terrain labels altogether through self-supervision. The Wild Visual Navigation (WVN) line of work [8, 23] learns traversability online from proprioceptive robot experience, using the robot's own successful and unsuccessful traversal attempts as supervision rather than hand-labelled semantic classes. Seo et al. [6] and Cho and Chung [7] similarly use self-supervision, the latter introducing a risk-aware self-training scheme that adapts to a robot's own navigation experience over time. V-STRONG [24] and WayFASTER [25] both target self-supervised, vision-based traversability specifically for off-road navigation, the latter emphasising increased navigation awareness through predicted uncertainty. Most recently, ViTA [26] adapts large vision foundation models (built on SAM2) to the traversability task, introducing learnable traversability prompts and perspective-diversified training to handle the annotation ambiguity inherent to traversability labels, and distilling geometric cues so that slope and elevation can be reasoned about from RGB input alone; the authors report state-of-the-art intersection-over-union (IoU) and precision with markedly fewer false positives across several off-road datasets. A sufficiently strong vision backbone may partially compensate for missing geometric sensing, which motivates measuring how much a camera-only estimate loses relative to LiDAR or fusion.

Beyond semantic segmentation, monocular metric depth estimation offers an alternative route to lifting camera predictions into a common BEV grid without requiring LiDAR support. Recent foundation models such as Depth Anything V2 [27], Metric3D v2 [28], and UniDepth [18] predict metric-scale depth from a single image, differing primarily in their training data and architectural approach to resolving scale ambiguity. This thesis empirically screens all three on the target platform's own sensor data (Section 4.1.2) rather than adopting a single method on the basis of published benchmarks alone, since generalization to the specific camera, terrain, and range distribution used here cannot be assumed from cross-dataset leaderboards.

2.3 LiDAR-based traversability estimation

Purely geometric approaches infer traversability from the shape of the terrain in the point cloud, without relying on visual appearance. PUTN [29] fits local planes to the point cloud to identify uneven but traversable terrain for navigation planning. FASTC [30] instead applies an attentional deep network directly to point cloud data for fast semantic traversability classification. Santo et al. propose a hybrid neural-geometric ground segmentation method [31] and, in follow-up work, TE-NeXt [32], a 3D sparse convolutional network operating directly on LiDAR point clouds for traversability estimation; both aim to retain the lighting-invariance of LiDAR while recovering some of the fine-grained discrimination that purely geometric heuristics lack. EAT [33] takes an environment-agnostic approach explicitly designed to generalise traversability estimates across different unstructured settings without per-environment retuning.

Across this body of work, geometric methods are consistently reported as robust to illumination but limited in their ability to distinguish visually distinct but geometrically similar surfaces (e.g. solid ground versus tall grass at a similar height profile), which is precisely the failure mode that a camera-based semantic signal is best positioned to resolve.

2.4 Cross-modal fusion approaches

A smaller body of work fuses camera and LiDAR signals directly for traversability estimation. Sock et al. [9] propose a probabilistic traversability map generated jointly from 3D LiDAR and camera data, an early example of Bayesian fusion. Zhou et al. [10] and Zhang et al. [11] both report terrain traversability mapping and analysis pipelines that combine LiDAR-based structural cues with camera-based semantic cues, generally finding that LiDAR alone is well suited to strongly structured obstacles (walls, ledges) while vision is needed to resolve semantically distinct but geometrically flat regions (e.g. lawn versus pavement). Guo et al. [34] present a further LiDAR-camera fusion pipeline for uneven outdoor terrain. UFO [35] explicitly models the uncertainty of each modality before fusing camera and LiDAR information into a semantic terrain map, which is closer in spirit to the confidence-aware rule evaluated within the cost-level fusion condition in this thesis than the earlier deterministic fusion schemes. Most recently, Beycimen et al. [36] report a real-time framework that fuses appearance- and geometry-based traversability estimates with dynamic model selection for autonomous robots operating in dynamic environments, which is, to the best of the author's knowledge, the most directly comparable recent system to the fusion condition evaluated in this work. Outside the traversability literature specifically, HydraFusion [12] demonstrates context-aware selective sensor fusion for autonomous vehicle perception more broadly, showing that the benefit of a sensor pair depends on operating conditions, an observation directly relevant to the research questions in Chapter 1. Wang et al. [37] and Atas et al. [20] address related but narrower problems: rigid-terrain path planning for wheeled vehicles, and transferring simulation-trained traversability models to real-world deployment, respectively.

RoadRunner [38] and its extension RoadRunner M&M [39] predict traversability and elevation directly from camera images (and, in the M&M variant, LiDAR voxel maps) at multiple spatial ranges and resolutions, using a hindsight-fused reference stack and satellite elevation data as training supervision; this is one of the few works in the reviewed literature that already combines camera and LiDAR inputs within a single learned model rather than treating fusion as a separate downstream step.

Despite this range of fusion work, very few of the reviewed papers report a controlled comparison against the same method's camera-only and LiDAR-only ablations on identical data; fusion is more often presented as the final system, with the unimodal alternatives either omitted or taken from a different paper's numbers on a different dataset and platform. Table 2.1 summarises this gap for the three works most directly comparable to the camera-only, LiDAR-only, and fusion conditions evaluated in this thesis: each was validated on a different dataset and platform, and only one reports real-world (rather than dataset-only or simulation) testing.

The fusion approaches surveyed above operate at the level of combining evidence, cue, or map estimates once both modalities are already available in a shared representation; none address the lower-

Table 2.1: Summary of the most directly comparable prior camera-only, LiDAR-only, and fusion works, illustrating the lack of a shared platform, dataset, or real-world validation across them.

Work	Sensors	Environment	Dataset	Urban/Rural	Real-world tests
Frey et al. (WVN) [23]	RGB camera	Forests, parks, grasslands	Self-collected (field)	Rural	Yes (ANYmal, 1.4 km)
Chen et al. (FASTC) [30]	3D LiDAR	On-/off-road terrain	SemanticKITTI + RELLIS-3D	Both	No (dataset only)
Guo et al. [34]	RGB camera + 3D LiDAR	Uneven outdoor terrain	RUGD (+ Gazebo)	Rural	No (simulation)

level question of how camera and LiDAR data are spatially associated before that combination happens. PointPainting [19], originally proposed for 3D object detection, addresses this by projecting LiDAR returns into the camera image and attaching each return's semantic label before aggregating in the target representation, an association mechanism this thesis adopts for the camera leg of the fusion condition (Section 4.1.3).

2.5 Benchmark datasets for off-road perception

Progress in vision-based terrain segmentation has been closely tied to the availability of labelled datasets. RUGD [15] and RELLIS-3D [14] are the two most widely used unstructured-terrain benchmarks in the papers reviewed above, the latter additionally providing synchronised camera and LiDAR data with 3D semantic annotations. GOOSE [16] extends this coverage with a larger and more diverse set of off-road scenes, and TartanGround [40] targets large-scale ground robot perception and navigation more broadly. Cityscapes [41] and PASCAL VOC [42] provide structured-environment and general-object benchmarks; they are not applicable to unstructured terrain directly, but several of the segmentation backbones adapted for off-road use (including GANav-style architectures) are pretrained on them, and they illustrate how much narrower and more standardised the structured-environment perception problem is by comparison. As noted in Chapter 1, none of these public datasets was collected on the ROVO platform's specific sensor rig, which motivated the custom dataset collection described in Chapter 3 and evaluated in Chapter 5.

2.6 Foundation models and language-guided navigation

A separate and rapidly growing line of work applies large vision-language and vision-language-action models to outdoor robot navigation, generally targeting goal-directed or instruction-following behaviour rather than terrain-level traversability estimation directly. Examples include ViNT [43] and NoMaD [44] as general-purpose visual navigation policies, LM-Nav [45] and VLM-GroNav [46] as language- and vision-language-grounded navigation systems for outdoor environments, and RT-2 [47] and PaLM-E [48] as broader vision-language-action foundation models applied across robotic manipulation and navigation tasks. GPT-4off [49] similarly distils traversability probability estimates from a large multimodal model into a lightweight on-board network. Firoozi et al. [50] and Marshall and Barron [51] both survey

the promises and open questions of foundation models in robotics more broadly, the latter offering a more skeptical view of how ”foundational” current transformer-based robot models actually are. These approaches are conceptually adjacent to this thesis in that they also fuse camera information with an additional modality (language, or a pretrained embedding space) to make navigation-relevant decisions, but they target higher-level task and goal reasoning rather than the pixel/point-level traversability estimate compared in this work, and are therefore discussed here only for context rather than benchmarked directly.

2.7 Limitations of previous work

Three gaps recur across the literature reviewed above. First, camera-only, LiDAR-only, and fusion methods are almost always evaluated as separate lines of work, each on its own preferred dataset and platform; even where fusion is proposed [9–11, 35, 36], a matched unimodal baseline collected on the same sensor rig and the same terrain is rarely reported, which makes it difficult to attribute reported gains specifically to fusion rather than to differences in data or tuning. Second, the calibration quality underlying camera-LiDAR fusion is typically treated as a solved precondition rather than a reported source of error, despite the fact that projection-based fusion and annotation pipelines are directly sensitive to extrinsic calibration accuracy (Chapter 3). Third, public benchmark datasets, while increasingly diverse [14–16, 40], do not cover every combination of platform, sensor rig, and terrain type that a given research group may need, which motivates custom data collection whenever the deployment platform differs meaningfully from those used to build the public datasets. This thesis addresses these three gaps directly by calibrating and evaluating a single camera-LiDAR rig on the ROVO platform (Chapter 3), collecting a matched custom dataset for that rig, and comparing camera-only, LiDAR-only, and cost-level fusion traversability estimates under identical conditions. The fusion condition evaluates a rule space comprising fixed blends, confidence weighting, sharpening, ordinal pooling, gating, and obstacle overrides (Chapters 3 and 5).

3

Methodology

This chapter describes the experimental methodology used throughout the thesis. It covers the hardware and dataset setup, the predictive evaluation metrics used to score each condition, the sensor calibration procedure, the evaluation criteria applied across domains, and the datasets used for evaluation.

3.1 Setup

The experiments use synchronized camera and LiDAR data collected with a ROVO unmanned ground vehicle, a base platform manufactured by Mattro (part of HAWE Hydraulik SE) and outfitted by Fraunhofer FKIE with the sensor payload described below. The platform is equipped with a forward-facing IDS uEye UI-3060CP RGB camera and two Ouster OS0-070-128U-AX LiDARs. The camera lens has a horizontal field of view of approximately 85° and a vertical field of view of approximately 60°, computed from the calibrated intrinsics (Section 5.1.1). The camera records images at a resolution of 968×608 pixels using BGR8 encoding. Each LiDAR is a 128-beam spinning sensor operating at approximately 10 Hz.

The camera and the two LiDARs are rigidly mounted on the ROVO platform, with the camera observing a forward-facing sector and the two LiDARs together providing near-360-degree geometric coverage around the vehicle. Their relative pose is therefore fixed during data collection. The camera–LiDAR extrinsic transformation was estimated and validated using the calibration procedures described in Section 3.2.1. This transformation is used to project LiDAR points into the camera image for point-painting and to transfer camera-derived traversability evidence to the shared BEV evaluation grid.

The physical sensor placement differs across the public benchmark platforms used in the cross-domain evaluation. The measured mounting positions and sensor heights for the RELLIS-3D, GOOSE, and FKIE/ROVO platforms are shown in Figure 3.1. These differences are relevant because mounting height and camera pitch affect the camera's near-field visibility and therefore its measurable BEV coverage.

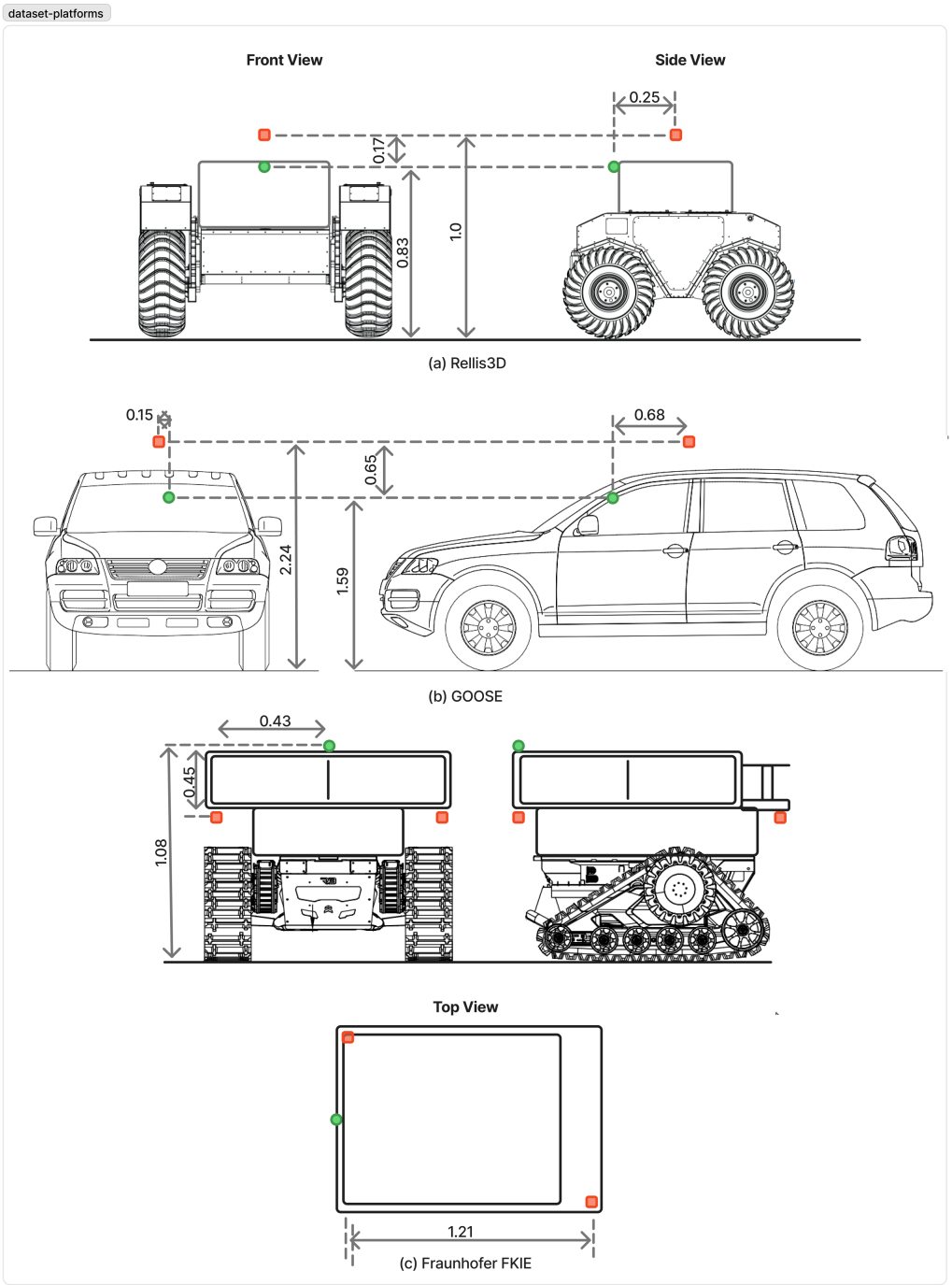


Figure 3.1: Physically measured sensor placement on the RELLIS-3D, GOOSE, and FKIE robot platforms: front view (left), side view (right), and, for FKIE, an additional top view. Camera positions are shown in green and LiDAR positions in red; all mounting heights are given in metres.

These three methods were selected as the best-performing openly testable implementations available for their respective modality at the time this work began, GA-Nav and WVN for camera-based estimation and FASTC for LiDAR-based estimation, so the comparison could be built on reproducible baselines whose performance could be independently verified rather than relying on claims from closed-source or otherwise untestable alternatives. PyTorch is retained as a broader evaluation and deployment dependency: the FASTC, GA-Nav, and WVN implementations were benchmarked in their original PyTorch paths and their deployable components were exported or validated against TensorRT in Chapter 5.

3.2 Sensor calibration

Both sensors used in this thesis, the IDS uEye camera and the Ouster OS-128 LiDAR, must be calibrated before their measurements can be combined. Two calibration steps are required: the camera's intrinsic parameters, which describe how the sensor itself maps the world onto its image plane, and the camera-LiDAR extrinsic transform, which describes how the two sensors are positioned relative to each other on the platform. The following subsections describe each of them, along with the extrinsic calibration methods evaluated and the one selected for the reported results.

3.2.1 Sensor configuration

The ROVO robot platform is equipped with the following sensors relevant to this calibration:

- **LiDAR:** Two Ouster OS-128, 128-beam spinning LiDARs, mounted diagonally at the front-right and rear-left corners of the platform. This calibration uses the front unit.
- **Camera:** IDS uEye UI-3060CP, 968×608 pixels, bgr8 encoding, fixed focal length. Image topic: `/rovo1/ueye_cam/ueye_froody/image_raw` (sensor_msgs/Image).

The camera is mounted forward-facing on the robot payload platform. The front LiDAR is mounted at the front-right corner of the platform with a slight downward pitch to capture the terrain ahead; the rear LiDAR is mounted at the rear-left corner. The two sensors share an overlapping field of view covering the forward terrain region relevant to traversability estimation. Figure 3.2 shows the platform.



Figure 3.2: Photograph of the ROVO platform, a base platform manufactured by Mattro (part of HAWE Hydraulik SE) and outfitted by Fraunhofer FKIE with the sensor payload described in this chapter, showing the tracked chassis, forward-mounted camera, and roof-mounted GNSS receiver (photo: Dirk Schulz and Allen Isaac Jose).

3.2.2 Camera intrinsic calibration

Accurate intrinsic calibration of the IDS uEye UI-3060CP camera is a prerequisite for the downstream tasks of this work. The intrinsic parameters (focal length, principal point, lens distortion) determine how 3D points are projected onto the image plane, and any error in these parameters propagates directly into

1. the camera-LiDAR extrinsic estimate, since extrinsic optimisation assumes correct intrinsics,
2. the projection of LiDAR points onto camera images used for point-painting (Section 4.1.3), the camera-to-BEV construction used in the fusion condition.

The calibration was performed once before the camera was rigidly mounted to the ROVO platform.

The classical pinhole projection model with two-coefficient radial and two-coefficient tangential distortion ("plumb bob" model) was adopted, since the lens used is a standard, non-fisheye optic with no significant equidistant behaviour. The model has nine free parameters (focal length f_x, f_y, principal point c_x, c_y, radial distortion k_1, k_2 and tangential distortion p_1, p_2), with k_3 fixed to zero given the chosen two-coefficient radial parameretisation. This model is consistent with the conventions of sensor_msgs/CameraInfo, allowing the calibration to be consumed directly by all ROS 2 nodes downstream.

A 6×6 AprilGrid target with 105 mm tags from the tag36h11 family and a 26.25 mm inter-tag spacing was used (corresponding to a Kalibr tagSpacing ratio of 0.25). AprilGrid was preferred over plain checkerboards because every tag carries a unique identifier, allowing partial views of the board to contribute to the calibration. This is particularly useful for ensuring observation coverage at the image

periphery, where lens distortion is strongest. The target was rigidly mounted to a flat backing to avoid the planarity errors that occur with unsupported paper printouts.

Image data was collected by recording a rosbag of the `/ueye_cam/froody/image_raw` topic (resolution 968 × 608, BGR8 encoding, manual exposure and gain, focus locked) while moving the calibration target through varied poses. Approximately 250 frames sampled uniformly across an 85-second recording were used, of which 245 were retained after filtering for anomalous reprojection error (Chapter 5). Coverage was deliberately distributed across multiple positions in the image, three distance ranges, and pitch/yaw/roll tilts of up to 45° to ensure that the calibration would be well-constrained in all parameters. The bag recording was decoded into image frames using the rosbags library.

The intrinsic calibration itself was carried out using the MATLAB R2024b Camera Calibrator app (Computer Vision Toolbox) [52]. MATLAB was chosen over Kalibr after preliminary attempts with the latter on an A4-sized AprilGrid printout and a larger 76 × 76 cm rigid board failed to converge, a known issue when pose diversity is borderline or the board is non-planar. Kalibr's documentation emphasises the need for suitable target visibility and sufficiently informative calibration motion [53]. MATLAB's solver, which combines AprilTag detection with bundle adjustment over the AprilGrid model, proved more tolerant of the recorded pose distribution and produced a clean solution from the same data the Kalibr pipeline rejected.

3.2.3 Camera-LiDAR calibration

With the camera intrinsics calibrated, the next step was to estimate the extrinsic transformation between the camera and the front-mounted Ouster OS-128 LiDAR, one of the platform's two identical units. This transformation consists of a rotation and translation that defines the LiDAR's pose relative to the camera, and is essential for projecting LiDAR points onto the camera image plane for point-painting and related methods. Only the LiDAR provides direct 3D range measurements, so a LiDAR-observed point p_{lidar} is first transformed into camera-centred coordinates using the rigid-body extrinsic transform $T_{\text{cam}\leftarrow\text{lidar}}$:

$$p_{\text{camera}} = T_{\text{cam}\leftarrow\text{lidar}} \cdot p_{\text{lidar}} \tag{3.1}$$

The resulting point p_{camera} is still a 3D coordinate expressed in the camera's frame; it is not itself a pixel. Projecting it onto the 2D image plane is a separate step, performed using the camera's intrinsic parameters (Section 5.1.1).

This transform is represented as a 4x4 homogeneous matrix comprising a 3x3 rotation matrix R and a 3x1 translation vector t. In practice it is parametrised as [x, y, z, qx, qy, qz, qw], where the first three elements are the translation and the last four are the quaternion components of the rotation.

The calibration is relevant to three components of the system, though not always as a direct input:

- **Condition C (cost-level fusion):** combining the camera and LiDAR traversability costmaps requires pixel-to-point correspondence, which depends directly on $T_{\text{cam}\leftarrow\text{lidar}}$.
- **Custom dataset annotation:** the final dataset used direct dual-modality annotation, with RGB

labels assigned in the image plane and LiDAR labels assigned directly in BEV, so the calibration transform was not used to generate these ground-truth labels. Calibration error still matters for this component indirectly, since the labels are evaluated against calibration-dependent predictions in Condition C (above).

- **Projection validation:** verifying that LiDAR points from known scene geometry (e.g. wall edges, door frames) project onto the correct pixels in the camera image serves as a qualitative calibration quality check. The projection implementation itself was separately validated using synthetic geometry with known ground truth, independent of the real calibration values, so this real-scene check primarily stresses the calibration parameters rather than being confounded with implementation correctness.

Method 1: `ros2_calib` (ika-rwth-aachen)

Principle `ros2_calib` [54] is a multi-sensor calibration tool developed at ika RWTH Aachen University. It supports LiDAR-to-Camera and LiDAR-to-LiDAR extrinsic calibration with a graphical interface for pair selection and direct URDF (Unified Robot Description Format) export of the resulting transforms. The tool is designed specifically for ROS2 workflows and operates natively on MCAP (the standard ROS2 bag-recording container format) recordings.

Like the MATLAB approach, `ros2_calib` is a **target-based method** requiring a plain checkerboard. It uses a similar algorithmic approach: checkerboard corner detection in camera images, plane fitting in LiDAR point clouds, and joint optimisation of the extrinsic transform from the resulting geometric correspondences. The primary differentiating feature is the direct URDF export, which eliminates the manual step of converting the calibration result into a static transform publisher or URDF joint definition.

Processing Pipeline

1. **Data collection:** Synchronised image–point cloud pairs are extracted from the ROS2 bag. The point cloud frame used is `rovo1/lidar/front/lidar`, the actual LiDAR data frame published by the Ouster driver.

2. **Target detection:** A checkerboard pattern is detected in both the camera image (using 2D corner detection with the known intrinsics) and in the LiDAR intensity/range data (using planar segmentation).

3. **Optimisation:** The 6-DOF extrinsic transform is estimated by minimising the geometric consistency between corresponding 3D checkerboard planes detected in the two sensors. The optimiser runs over all collected pairs simultaneously to improve robustness.

4. **Output:** The result is expressed as a 4×4 homogeneous transformation matrix in ROS TF (transform-tree) convention, specifically, the pose of the camera frame (`froody`) in the LiDAR frame (`rovo1/lidar/front/lidar`), given as translation (x, y, z) and rotation (roll, pitch, yaw).

Method 2: MATLAB Lidar Camera Calibrator (target-based)

Principle The MATLAB Lidar Camera Calibrator app implements a target-based extrinsic calibration method. A planar checkerboard target with known geometry is presented to both sensors simultaneously. The algorithm detects the checkerboard corners in the camera image using standard Harris corner detection and fits a plane to the checkerboard region in the LiDAR point cloud using RANSAC (random sample consensus)-based plane fitting. The 3D plane normal and 2D corner positions provide geometric constraints from which the 6-DoF extrinsic transform is estimated by minimising a joint reprojection and plane-to-plane error.

Processing Pipeline The MATLAB Camera-LiDAR Calibration Toolbox (R2026a) provides a structured, GUI-assisted calibration workflow. The processing pipeline differs from `ros2_calib` in the following aspects:

1. **Data ingestion:** The same synchronised image–point cloud pairs are loaded into MATLAB. The point clouds are read from PCD (Point Cloud Data) files extracted from the ROS2 bag.
2. **Target detection:** The checkerboard corners are detected in each 2D image using the known camera intrinsics. Simultaneously, the checkerboard plane is segmented from the LiDAR point cloud using RANSAC-based plane fitting. MATLAB estimates the 3D pose of the checkerboard from both sensors independently.
3. **Optimisation:** The extrinsic transform is estimated by solving a least-squares minimisation over the correspondences between the 3D checkerboard poses as seen by each sensor. This is a feature-based approach using the checkerboard as a structured landmark.
4. **Output:** The result is a `rigidtform3d` object with the transformation matrix A in the form $[R \mid t;\ 0\ 0\ 0\ 1]$, where the transform maps points from the LiDAR frame to the camera frame (opposite convention to ROS TF).

Synchronized Pair Extraction Because MATLAB's `lidarCameraCalibrator` requires pre-extracted image and point cloud files rather than a ROS 2 bag directly, a custom extraction pipeline was developed using the `rosbags` Python library (version 0.11.0).

To guarantee temporal alignment, the extraction script anchors synchronization on the sparse, stop-triggered camera frames. For each camera frame, the nearest LiDAR scan is identified via binary search on the LiDAR timestamp array. Pairs with an absolute time difference exceeding 200 ms were rejected; however, given the stationary robot setup, this window provided effectively perfect synchronization. The `camera_info` topic, published as a latched topic by the IDS uEye driver, was captured once at startup and applied uniformly to all frames.

The pipeline decodes the ROS 2 messages (converting PointCloud2 byte buffers directly into Open3D (latest pip release)-compatible `.pcd` files via vectorized operations) and outputs the synchronized pairs into the flat directory structure required by MATLAB's solver.

Method 3: Koide `direct_visual_lidar_calibration` (targetless)

Principle `direct_visual_lidar_calibration` is a targetless extrinsic calibration method proposed by Koide et al. [55] that does not require a dedicated calibration target. Instead, it estimates the camera–LiDAR extrinsic transform by maximising the consistency between a camera image and a LiDAR intensity image rendered from a dense 3D point cloud of the scene.

The method consists of three main stages:

1. **Preprocessing:** Multiple LiDAR scans are accumulated into a dense point cloud map using CT-ICP. The accumulated map is then projected into the camera frame using the known camera intrinsics, and voxel downsampling is applied to reduce memory consumption.

2. **Initial guess estimation:** The method establishes 2D–3D correspondences between the camera image and the point cloud using SuperPoint and SuperGlue. These correspondences are used to compute an initial estimate of the sensor rotation and translation.

3. **Fine calibration:** The initial estimate is refined through gradient-based optimisation that minimises the photometric inconsistency between the rendered LiDAR image and the real camera image.

The method was built from source in a ROS 2 Jazzy workspace, since the available pre-built container images were incompatible with the recorded MCAP bag files. It was run with its default dependency stack (GTSAM, Ceres, Iridescence) and the standard SuperPoint–SuperGlue feature matching [56] for initial pose estimation, following the settings in the tool's own documentation.

This targetless approach was ultimately unable to produce a reliable camera-LiDAR extrinsic estimate, as discussed in Chapter 5; `ros2_calib` (Method 1) was therefore adopted as the primary extrinsic calibration for all reported results.

3.3 Calibration method comparison criteria

The calibration methods were compared using both quantitative and qualitative criteria. Quantitative criteria included reprojection error, translation and rotation consistency across image–point cloud pairs, and the ability of each method to produce a valid 6-DoF transform. Qualitative evaluation was performed by projecting LiDAR points into the camera image and visually assessing their alignment with structural scene features such as wall edges, door frames, and foreground objects. In addition, practical considerations including ROS 2 compatibility, preprocessing effort, and sensitivity to scene content were considered when assessing the suitability of each method for the overall system.

3.4 Datasets

Throughout this thesis, *dataset* denotes RELLIS-3D, GOOSE, or the custom FKIE dataset; *domain* denotes one of the four evaluation domains, RELLIS-3D, GOOSE, FKIE forest, or FKIE campus; and *sequence* denotes a named recording within a domain.

This work draws on three datasets that differ in platform, sensor rig, and annotation scheme: two public off-road benchmarks (RELLIS-3D, GOOSE) and the custom ROVO dataset collected for this thesis

(Section 3.2.1). Table 3.1 summarises the platform, sensor, and annotation specifications of all three side by side; the subsections that follow give the full description of each.

Table 3.1: Comparison of platform, sensor, and annotation specifications across the three datasets used in this thesis.

Property	**RELLIS-3D**	**GOOSE**	**Custom ROVO Dataset**
Platform	Clearpath Warthog UGV	MuCAR-3 (Volkswagen); the only GOOSE platform used in this thesis	ROVO UGV
Camera	Basler acA1920-50gc, 1920×1200, 16 mm lens	Prism RGB+NIR camera (model and resolution not stated in the GOOSE documentation)	IDS uEye UI-3060CP, 968×608, bgr8
LiDAR	Ouster OS1-64 (64-channel), plus a secondary Velodyne Ultra Puck (32-channel)	Roof-mounted 128-channel scanner, plus two 32-channel lateral scanners (±45°)	Two Ouster OS-128 LiDARs (128-channel each)
Sensor rate	10 Hz (sensors); GPS/IMU at 300/100 Hz	LiDAR approximately 10 Hz; all sensors PTP-synchronised	10 Hz
Annotated images	6,235 (pixel-wise)	10,000 (pixel-wise)	30 forest + 150 campus (pixel-wise, model-assisted)
Annotated LiDAR	13,556 scans (point-wise)	10,000 scans (point-wise)	30 scans, dual-labelled: point-wise + dense BEV raster
Semantic classes	20	64	4 traversability tiers + 1 ignore label (Table 3.2)
Environment	Off-road testing ground (runways, forested trails, pasture, hill), Texas A&M	Unstructured outdoor terrain across multiple German sites, three platforms	Fraunhofer FKIE campus grounds: grass, asphalt, test inclines, dense vegetation
Role in this thesis	Primary benchmark; seq. 00000–03 train, seq. 00004 held-out eval	Third, independent domain; cross-domain generalisation probe	Real-platform validation domain, dual camera+LiDAR ground truth

3.4.1 Public benchmark datasets

Two public off-road benchmarks are used alongside the custom ROVO dataset, RELLIS-3D and GOOSE, described in turn below. Each dataset's native class taxonomy is mapped onto this thesis's

shared four-tier scheme; the specific class-to-tier crosswalk is not published in this thesis (Section 7.3).

RELLIS-3D

RELLIS-3D [14] is a multimodal off-road perception dataset collected on the Rellis Campus of Texas A&M University, an off-road testing environment that includes runways, aprons, forested trails, pastures, and a lake-adjacent hill. It provides 6,235 pixel-wise annotated RGB images and 13,556 point-wise annotated LiDAR scans, synchronised across five recording sequences of approximately five minutes each, together with simultaneous localisation and mapping (SLAM)-derived ground-truth poses for every LiDAR frame. The dataset defines 20 semantic classes, including several traversability-relevant distinctions not present in most urban-driving benchmarks (separate *puddle* and *deep water* classes, *mud*, and *rubble* piles), reflecting its purpose as an off-road rather than on-road benchmark. The data was recorded on a Clearpath Warthog unmanned ground vehicle (UGV) equipped with a Basler acA1920-50gc RGB camera and an Ouster OS1-64 (64-channel) LiDAR as the primary annotated sensors, alongside a secondary 32-channel Velodyne LiDAR, a stereo camera, and GPS/IMU (Table 3.1); this 64-beam LiDAR is the sensor that motivates the domain-shift discussion against the ROVO platform's 128-beam Ouster OS-128 (Section 3.2.1).

In this thesis, RELLIS-3D is the primary public benchmark. Rather than use the published frame-level split files (`pt_train.lst`, `pt_val.lst`, and `pt_test.lst`), which distribute scans from the same sequences across training and test sets and thereby permit scene-level leakage, this work adopts a stricter sequence-isolated protocol. Sequences 00000–00003 form the training source and sequence 00004 is held out entirely for evaluation (Chapter 5); an internal `train_core`/`train_sel` partition within the training source is used for model selection. The partitions were verified disjoint ($\texttt{train_core} \cap \texttt{train_sel} = \varnothing$), and both share zero LiDAR-file inodes with sequence 00004. The exact `exp9` launch manifest that demonstrates the alternate selection reader is not committed, so this selection configuration is declared and its disjointness documented, rather than independently auditable from the repository alone. RELLIS-3D's native 20-class taxonomy is mapped onto the coarse four-tier traversability scheme $\{\text{free}, \text{low}, \text{medium}, \text{obstacle}\}$ used throughout this work, following the same class-mapping convention applied to the custom ROVO dataset (see the traversability class definition below).

GOOSE

GOOSE [16] is the German Outdoor and Offroad dataSEt, developed by Fraunhofer IOSB together with Universität der Bundeswehr München and the University of Koblenz. It comprises 10,000 pixel-wise and point-wise annotated RGB image/LiDAR point-cloud pairs collected across three robotic platforms, annotated according to a 64-class ontology designed specifically for unstructured outdoor perception, substantially broader and more fine-grained than RELLIS-3D's 20 classes (for example, distinguishing *low_grass*, *high_grass*, *moss*, and *scenery_vegetation* within the vegetation category alone). This thesis uses only recordings from the MuCAR-3 (Volkswagen) platform, not the Spot or excavator platforms of the GOOSE-Ex extension. MuCAR-3 carries a roof-mounted 128-channel LiDAR, a Prism RGB+NIR camera behind the windshield, four roof-mounted RGB cameras providing 360° coverage, and two 32-channel LiDARs tilted ±45° for lateral obstacle detection. Its LiDAR runs at approximately 10 Hz and all sensors

are synchronised through PTP (IEEE 1588). The Prism camera's more specific model number and resolution are not stated in the GOOSE documentation.

In this thesis, GOOSE serves as an independent evaluation domain. The full published validation split (544 frames) is used, without subsetting, primarily to probe cross-domain generalisation of models trained on RELLIS-3D and the custom ROVO dataset (Chapters 5–6). Its class ontology is mapped onto the same shared four-tier traversability scheme used for RELLIS-3D and the custom ROVO dataset.

3.4.2 Custom ROVO dataset

Neither RELLIS-3D nor GOOSE was collected on the ROVO platform or its sensor rig (Section 3.2.1), and neither covers the specific test terrain used to validate the system in this work. Platform-specific properties (ground clearance, sensor mounting height and field of view, wheel configuration) mean that traversability labels generated for another robot do not transfer cleanly (Chapter 1); this motivated collecting a custom dataset with the calibrated ROVO sensor rig itself (Figure 3.2). The site and the routes recorded or reconstructed for each collection run are shown in Figure 3.3.

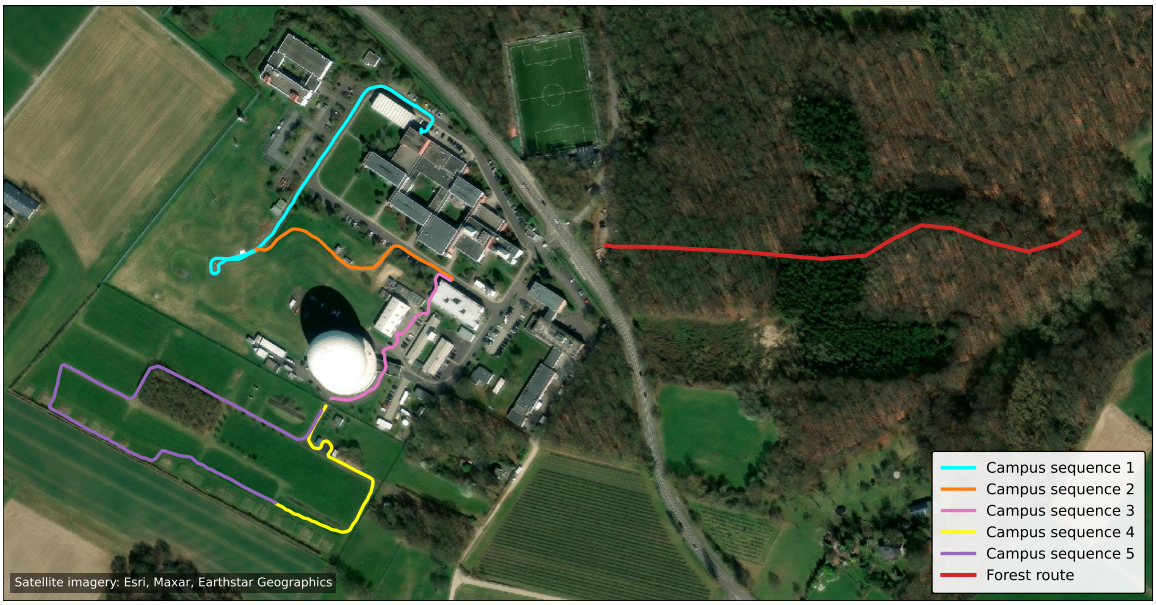


Figure 3.3: Recorded and reconstructed vehicle routes at the Fraunhofer FKIE site (Fraunhoferstraße 20, Wachtberg-Berkum), overlaid on satellite imagery. Campus sequences 1–5 show the GPS trace recorded during each campus data collection run. The forest route is reconstructed from the corresponding OpenStreetMap trail geometry, as the original recorded GPS log for the forest sequence could not be located, and therefore represents an approximate path rather than recorded telemetry.

Data was recorded as synchronised ROS 2 bags (front-facing RGB image stream and LiDAR point-cloud stream) at outdoor sites around the Fraunhofer FKIE campus. For every recording, all valid RGB–LiDAR pairs were first synchronised within the matching threshold, then equidistant pairs were sampled from the

resulting synchronised sequence for annotation, giving each annotated subset full temporal and spatial coverage of its route.

Two such collections were made on the same ROVO platform and sensor rig. The first, recorded on 1 July 2026 and referred to elsewhere as the "forest" collection, covers grassy areas, asphalt and paved surfaces, mild slopes, intentionally constructed test inclines, dense vegetation, trees, parked structures, small height changes, and open off-road paths; 30 RGB–LiDAR pairs were sampled for annotation (Section 3.4.3).

The second, recorded on 8 July 2026 at a research-campus site, spans a mixed built and unstructured environment, paved and cobbled surfaces, grass paths and open field, parked vehicles, buildings, fences, bins, and street furniture, deliberately chosen to place obstacle classes and ground materials outside the vegetated off-road distribution the camera methods (GA-Nav and WVN) were trained on. Five sequences of 30 annotated frames each were sampled (150 frames total). This campus collection uses the same camera calibration (validity re-confirmed on campus data) and the same four-tier annotation convention; because it serves specifically as a fourth, out-of-distribution evaluation domain rather than a second training-adjacent dataset, its full description and evaluation role are given alongside the results that use it (Section 5.3).

3.4.3 Custom ROVO dataset pre-processing and annotation

A custom RGB–LiDAR dataset was collected using the ROVO robotic platform in outdoor environments around the Fraunhofer FKIE campus. The purpose of this dataset is to evaluate traversability estimation under real platform conditions, where the robot encounters terrain and sensor effects that are not fully represented by public datasets. The recorded scenes include grassy areas, asphalt and paved surfaces, mild slopes, intentionally constructed test inclines, dense vegetation, trees, parked structures, small height changes, and open off-road paths.

The dataset is used as an additional evaluation domain for the comparison between camera-based, LiDAR-based, and fusion-based traversability estimation. It is particularly useful because the environment contains situations where visual appearance and LiDAR geometry provide different cues. For example, short grass may appear visually similar to taller grass, while LiDAR geometry can reveal raised terrain, slope changes, or object footprints. Conversely, camera images can provide semantic context that is not directly available from geometry alone.

Data preparation and annotation. The raw synchronized ROS2 streams were matched by nearest timestamp, keeping only RGB–LiDAR pairs within 0.05 s of each other, from which the 30 annotated frames were sampled as described above (Section 3.4.2). Each point cloud was stored in a KITTI-style (x, y, z, dummy) format for pipeline compatibility, since the recordings do not carry a real intensity channel; the original per-point fields were retained separately in case they become relevant later.

RGB images were labelled using SAM2-assisted segmentation, with masks precomputed for all selected frames and manually corrected where the automatic segmentation was inaccurate. LiDAR scans were labelled directly in a bird's-eye-view representation rather than by projecting camera labels, since the

LiDAR observes the full surroundings of the robot while the camera covers only a forward-facing frustum; this yields both a dense BEV raster label and a SemanticKITTI-style point-wise label per frame.

LiDAR BEV annotation. The LiDAR annotation process produced two complementary label representations. First, a dense BEV raster label was stored for each selected frame. Second, a point-wise label was exported for each LiDAR scan, assigning one traversability label to every LiDAR point. The point-wise labels follow the same general convention as SemanticKITTI-style LiDAR labels, while the BEV labels provide a dense spatial representation for visualization and BEV evaluation. The relationship between these two representations, and how it differs from the point-derived ground truth used for RELLIS-3D and GOOSE, is discussed in full in Section 3.4.3.

The LiDAR annotation was based primarily on physical geometry and traversability. This means that some regions can be labelled differently in the LiDAR annotation than in the image annotation when the LiDAR provides clearer geometric evidence. For example, a grassy test incline may appear visually similar to surrounding grass in the image, but in the LiDAR BEV it can appear as a clear raised terrain feature. In such cases, the LiDAR label follows the geometric traversability indicated by the point cloud.

Traversability class definition. The custom dataset uses four traversability classes and one unknown label:

$$\mathcal{Y} = \{0 : \text{free},\ 1 : \text{low},\ 2 : \text{medium},\ 3 : \text{obstacle},\ 255 : \text{unknown}\}.$$

The labels are not purely semantic categories. Instead, they represent the expected difficulty and risk for the ROVO robot to traverse a region. The guiding question during annotation was whether the robot could physically drive through the region, and with what level of cost or risk.

These rules apply to the dense BEV raster label, not the point-wise export described above, since robot geometry is only meaningful at the cell level where physical traversal is being assessed. Two edge cases required rules beyond the table above. A flat ground surface underneath a parked vehicle, trailer, or other object was not labelled as free, since the space is physically occupied and cannot be traversed by the robot. High tree canopy was not automatically treated as an obstacle if the robot could pass underneath it; only vegetation at robot-body height or dense blocking vegetation was labelled as obstacle. This robot-conditioned treatment follows directly from the traversability label definition above. Because the classes represent risk and cost specific to the ROVO platform rather than generic semantic categories, robot geometry is a necessary input at the cell level, not an inconsistency in the labelling scheme. RELLIS-3D and GOOSE apply no equivalent geometric or robot-size reasoning, since their labels are a fixed semantic-class lookup from pre-existing public annotations rather than a purpose-built BEV traversability annotation.

Figure 3.4 shows one open and one cluttered scene from each domain. The parked vehicle in the campus example and the dense canopy in the forest examples illustrate the edge-case rules above. The large unknown fraction in every row reflects the sparsity of the point-label convention itself, most of the BEV grid receives no LiDAR return in a single scan, rather than any deficiency in the annotation.

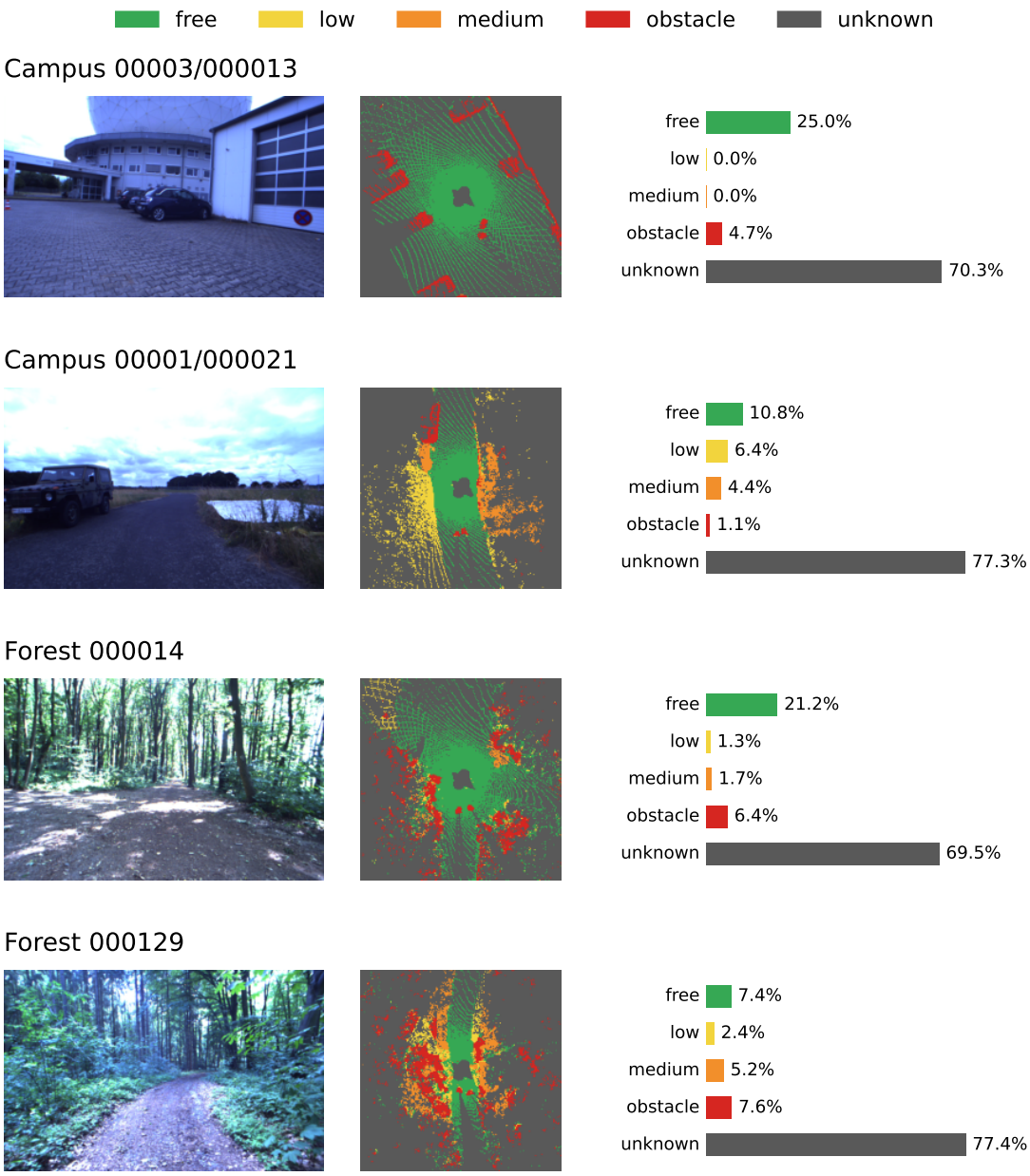


Figure 3.4: Representative campus and forest examples from the custom ROVO dataset: RGB frame, sparse point-label ground truth, and per-class pixel breakdown.

Table 3.2: Traversability class interpretation used for the custom ROVO dataset.

Label	Class	Interpretation
0	Free	Easily traversable surfaces such as asphalt, pavement, compact dirt paths, flat open ground, or clearly drivable worn tracks.
1	Low cost	Traversable terrain with minor additional difficulty, such as short grass, mowed grass, mild slopes, shallow unevenness, leaves, small sticks, or low vegetation that the robot can cross with little risk.
2	Medium cost	Traversable but uncertain or risky terrain, such as taller grass, dense low vegetation, raised bumps, test inclines, larger curb edges, side slopes, or regions where hidden unevenness may affect the robot.
3	Obstacle	Non-traversable or collision-risk regions, including tree trunks, dense bushes, walls, vehicles, trailers, metal structures, high curbs, large logs, large rocks, or vegetation that blocks the robot body.
255	Unknown	Regions where traversability cannot be judged reliably, including sky, far background, heavy occlusion, ambiguous boundaries, or areas outside the relevant annotation region.

Deviation from the initial projection-based annotation plan. The initial plan was to label semantic regions in the camera image and project those labels into the LiDAR point cloud using the calibrated camera–LiDAR transformation. This was abandoned because the camera's limited forward-facing field of view would leave LiDAR points outside it unlabelled, unsuitable for evaluating LiDAR-only and BEV fusion methods that operate on the full field around the robot. The final strategy instead used direct dual-modality annotation, RGB images labelled in the image plane and LiDAR scans labelled directly in BEV, giving independent but temporally synchronized labels for the two modalities and full 360-degree LiDAR ground truth rather than only labels within the camera frustum.

BEV grid and coordinate convention

FASTC's native LiDAR BEV annotation grid has a resolution of 0.10 m per cell and a size of 450×450 cells. The grid covers

$$X \in [-20, 25]\,\mathrm{m}, \qquad Y \in [-22.5, 22.5]\,\mathrm{m}.$$

The sensor is located at row 200 and column 225 in this grid. This position is not the geometric centre of the grid because the forward extent is asymmetric. The sensor offset must therefore be handled explicitly during visualization, cropping, and evaluation. Each LiDAR point is assigned to a cell using its horizontal position only, following $\mathrm{row} = \lfloor (x - X_{\min})/0.10 \rfloor$ and $\mathrm{col} = \lfloor (y - Y_{\min})/0.10 \rfloor$; height is not used in this projection, so points at different heights above the same ground location fall into the same cell. The traversability class for each cell was then assigned by a human annotator working from this top-down projection, using the geometric evidence described above.

This grid differs from the smaller RELLIS-style BEV grid used elsewhere in the evaluation, which is

centred around the sensor with a smaller spatial extent. For the custom ROVO dataset, two evaluation views are therefore relevant. The first is a sensor-centred crop that allows comparison with the public-dataset protocol. The second is the full native custom grid, which retains the larger spatial coverage and the near-field labels around the robot.

The custom dataset also contains labelled terrain close to the robot. Therefore, a range band selected for another dataset should not be applied blindly to the ROVO data. In particular, removing near-field cells would discard some of the most relevant terrain for local navigation.

Role of the custom dataset in the evaluation. Each frame's point-wise label count was checked against its LiDAR scan's point count, and label values were confirmed to fall within $\{0, 1, 2, 3, 255\}$, using a centralized workflow that reduces the risk of mismatched files or stale configurations.

The custom ROVO dataset is used to test whether the findings from public datasets also hold under real robotic sensing conditions. It provides scenes where camera appearance and LiDAR geometry can support or contradict each other. This makes it useful for analysing modality-specific failure modes. For example, tall grass may appear visually traversable but produce dense LiDAR returns; a raised test incline may be geometrically clear in LiDAR but visually similar to surrounding grass; and a flat surface under a vehicle may be geometrically flat but not physically traversable.

The dataset serves as an additional evaluation set and a diagnostic tool for understanding the behaviour of camera-only, LiDAR-only, and fusion-based traversability estimation methods in realistic outdoor conditions.

4

Solution

This chapter describes how the three experimental conditions compared in this thesis (camera-only semantics via monocular depth projection, Condition A; LiDAR-only, Condition B; and cost-level fusion, Condition C) are formulated and implemented on a shared evaluation footing. Section 4.1 defines the shared representation and the per-condition prediction and fusion pipeline; Section 4.2 describes the software architecture used to run all three conditions identically.

4.1 Proposed algorithm

This section describes how each of the three conditions, camera-only, LiDAR-only, and fusion, is formulated on a shared scoring representation. Section 4.1.1 defines that shared grid. The subsections that follow describe each condition's construction in turn, ending with the fusion condition that combines the camera and LiDAR branches at the cost level. The resulting data flow is shown in Figure 4.1.

4.1.1 Shared bird's-eye-view grid

All three conditions are scored on a common bird's-eye-view (BEV) grid rather than in each method's native representation: 200×200 cells at 0.1 m per cell (± 10 m), LiDAR-centred in frame `rovo/lidar/front/lidar`. This thesis refers to evaluation on this shared grid as Path-A. Its index mapping $\text{row} = \text{round}(99.5 - x/\text{res})$ and $\text{col} = \text{round}(99.5 - y/\text{res})$ expresses the ± 10 m sensor-centred convention. This grid uses the raw LiDAR frame directly, with no ground-levelling or tilt correction applied, so it inherits the sensor's own mounting pitch (Section 3.2.1) rather than being aligned to true horizontal. Using one shared grid as the scoring domain, rather than each method's native representation (for example FASTC's own 512×512 at 0.2 m internal grid), is what makes the camera-only, LiDAR-only, and fusion outputs directly comparable cell-for-cell; every method's native output, FASTC's included, is resampled onto this shared grid before scoring (Chapter 5).

Figure 4.2 shows what each method predicts on a frame of the RELLIS-3D dataset. Condition A is described in detail in the following subsection.

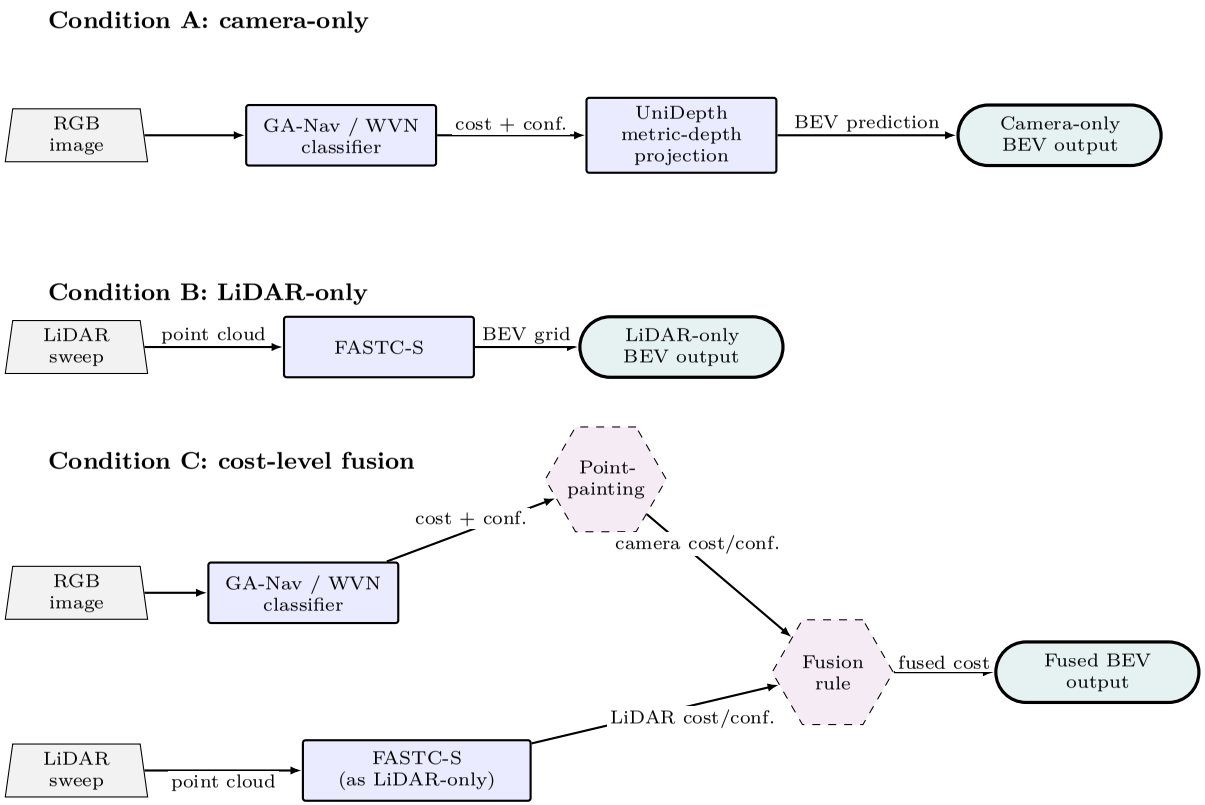


Figure 4.1: Data-flow diagram of the three condition pipelines, showing how each sensor input is transformed into a BEV output on the shared canonical grid. Trapezium nodes are raw sensor input, rectangles are learned models, hexagons are non-learned deterministic operations, and rounded terminator nodes are the final per-condition output. Condition C has separate camera and LiDAR branches that converge at the cost-level fusion rule. GA-Nav and WVN are evaluated as alternative camera classifiers in both Condition A and Condition C, each reported as a separate result, not combined with each other.

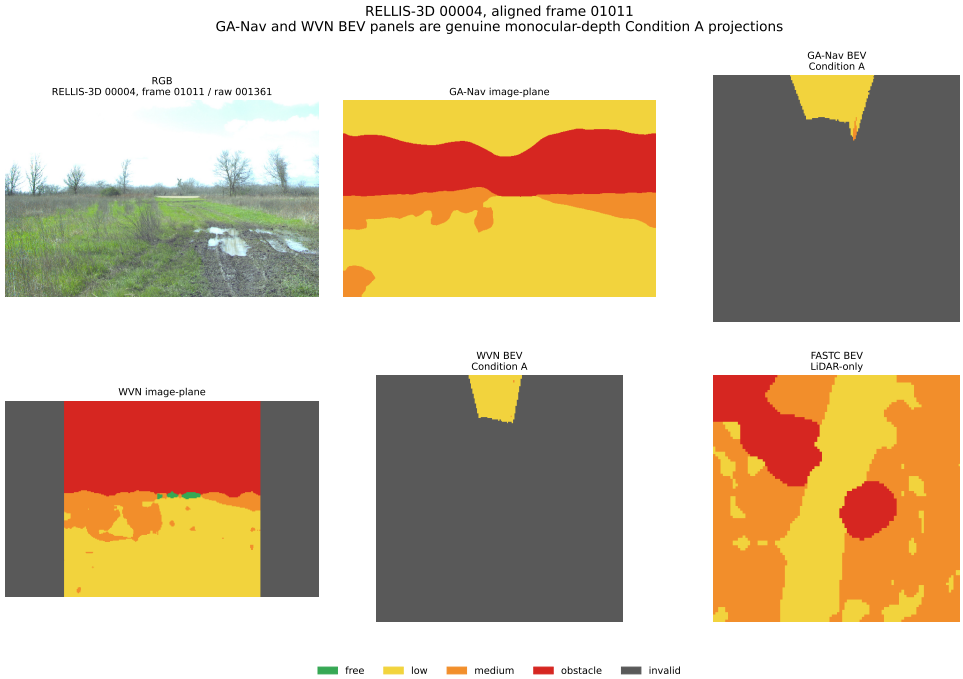


Figure 4.2: Qualitative predictions from all three methods on RELLIS-3D sequence 00004, frame 01011 (near-training-distribution domain). GA-Nav and WVN image-plane predictions are shown alongside their Condition A monocular-depth BEV projections; both BEV projections are narrow because camera-only projection reaches only a small fraction of the scene, a structural limitation examined in Chapter 5. FASTC's LiDAR-based BEV, by contrast, covers nearly the entire grid. In the panels, free is green, low is yellow, medium is orange, obstacle is red, invalid is grey. The full four-tier class definition is given in Section 3.4.3.

4.1.2 Condition A: Camera-only semantics via monocular depth projection

Two independent camera-based estimators are evaluated for the camera-only condition:

- **WVN.** A frozen STEGO/DINO-ViT dense feature extractor feeds a trained four-class head, producing per-pixel traversability logits, an expected cost on the tier scale $\{0, 0.33, 0.66, 1.0\}$, and a reconstruction-error-based confidence signal [8].
- **GA-Nav.** A retrained SegFormer [57] MiT-b0 backbone with a class-attention head produces the same four-tier cost via a direct class-to-cost lookup, with entropy of the softmax output used as the confidence signal.

Under the camera-only condition, these image-plane predictions are projected to BEV using UniDepth-v2 [18] monocular depth estimates, without using LiDAR returns to determine whether a cell is populated. A cell is camera-valid when the camera-only depth projection reaches it; coverage is therefore determined by the camera projection and its estimated depth, rather than by LiDAR support. This deliberately

isolates the camera modality. The resulting camera-only BEV estimates are weak across all four evaluation domains, reflecting the difficulty of recovering reliable BEV position and obstacle geometry from a monocular image. The LiDAR-supported point-painted camera leg used for fusion is defined separately in Section 4.1.3.

Selecting the monocular depth model for Condition A. Three monocular depth models were evaluated as candidates for the camera-only BEV projection: DepthAnything [27], Metric3D [28], and UniDepth. The comparison was a depth-quality diagnostic on 15 FKIE forest frames against LiDAR ground truth, using correlation to LiDAR range, metric-scale ratio, BEV coverage, and near-field fill. It was not a scored end-to-end traversability evaluation. UniDepth gave the strongest global correlation (0.729 ± 0.063) and a near-unity metric-scale ratio (0.929 ± 0.165), indicating minimal systematic depth bias, without a material saturation-ceiling artefact. DepthAnything showed severely inflated near-field scale (1.436 ± 0.277). Metric3D produced slightly higher coverage and near-field fill (0.156 and 0.217), but under-ranged at longer distances (metric-scale ratio 0.745, indicating systematic under-estimation) and showed the only material ceiling artefact. UniDepth was therefore selected for all camera-only evaluation. Figure 4.3 provides qualitative examples from three of the 15 frames; the quantitative comparison is the basis for the selection. The diagnostic is recorded and version-controlled in the project repository. This model-selection decision is separate from Section 4.1.3, which asks why point-painting was selected over camera-only projection as a fusion construction.

4.1.3 Point-painting as a fusion projection choice

Camera predictions must reach the shared BEV grid before they can be combined with LiDAR costs. Fixed-ground-plane projection and monocular depth lifting with UniDepth-v2 were tested as camera-only alternatives, but proved inadequate for reliable obstacle projection. Because LiDAR is already available for fusion, point-painting was evaluated as a fusion design choice: LiDAR returns are projected into the camera image and the semantic label at each return is attached before aggregation in BEV.

The remainder of this section reports that comparison in detail.

A dedicated comparison on RELLIS-3D sequence 00004 (1,359 frames) tested three BEV constructions per backbone against the same full-support scoring used elsewhere in this thesis: a fixed-ground-plane projection, a monocular-depth-lifted projection (UniDepth-v2), and point-painting.

When scored on camera-valid support only (the cells the camera actually reaches), the comparison is sharper than the full-support numbers alone suggest. Point-painting beats monocular depth primarily on accuracy within the cells both methods reach, not merely on how many cells they cover. Camera-valid obstacle IoU is 0.4455 vs. 0.0640 for GA-Nav and 0.3341 vs. 0.0141 for WVN, an order-of-magnitude accuracy gap that coverage (7.61 % vs. 6.02 % for GA-Nav, 4.68 % vs. 3.68 % for WVN) does not come close to explaining.

The fixed-plane projection should be described as degenerate for obstacle detection specifically, not wholesale: on camera-valid support its low-tier (free/low-cost terrain) IoU is 0.911 (GA-Nav) and 0.951 (WVN), comparable to point-painting's 0.933 and 0.963. The ground-plane assumption is correct for

Table 4.1: Camera-to-BEV projection comparison on RELLIS-3D sequence 00004 (1,359 frames). All values use full support over GT-valid Path-A cells; unseen cells therefore count as false negatives.

Projection method	Backbone	Obstacle IoU	Recall	Precision	Coverage
Fixed-plane	GA-Nav	0.0000	0.0000	0.0000	5.20%
Fixed-plane	WVN	0.0000	0.0000	0.0000	3.17%
UniDepth	GA-Nav	0.00098	0.00098	0.5990	6.02%
UniDepth	WVN	0.000089	0.000089	0.5123	3.68%
Point-painted	GA-Nav	0.00825	0.00830	0.5951	7.61%
Point-painted	WVN	0.00225	0.00225	0.5737	4.68%

ground and fails for vertical structure, because projecting every BEV cell at a fixed ground height smears vertical objects to wherever the ray happens to meet the plane, destroying the spatial correspondence obstacles depend on. A separate analysis on the forest domain shows the same fixed-plane failure mode, though less extreme: full-support GA-Nav obstacle recall there is 0.1437, not zero. This confirms the failure is not specific to RELLIS-3D, though literal zero recall is only established for the RELLIS-3D comparison presented here. The failure is genuine rather than an artifact of an empty denominator: 18,887 obstacle-ground-truth cells fall inside GA-Nav's fixed-plane valid mask, of which the construction predicts obstacle on 20, with zero true positives among them.

Returning to the RELLIS-3D comparison, monocular depth lifting (UniDepth-v2) avoids an exact zero but is far from competitive with point-painting: camera-valid obstacle IoU is roughly 7× below point-painting on GA-Nav (0.0640 vs. 0.4455) and roughly 24× below on WVN (0.0141 vs. 0.3341). Genuine camera-only coverage under UniDepth-v2 (6.02 % GA-Nav, 3.68 % WVN) is also lower than point-painting's LiDAR-supported coverage on the same sequence (7.61 %, 4.68 %), so removing the LiDAR dependency does not even buy back coverage; it loses both coverage and accuracy at once. This is not a coverage policy choice: UniDepth predicts a depth value for every pixel, but each pixel only registers in the shared grid if its predicted depth places it within the fixed ±10 m window, so depth-estimation error and perspective (near-field pixels covering a small ground area, far-field pixels covering a large one) both work against coverage rather than for it. Point-painting is therefore the best available camera-to-BEV construction of those tested, not an unexamined default: among the tested constructions it is the least-degenerate option for obstacle projection. It was therefore adopted for the camera leg of Condition C because LiDAR is available in there. This conclusion is a fusion methodology choice, not a claim about camera-only performance; the camera-only projection results are reported separately in the Condition A row of the cross-dataset comparison. Because the same trained classifier underlies both projection methods, the camera-valid obstacle-IoU gap above (0.0640 for GA-Nav and 0.0141 for WVN under UniDepth, versus 0.4455 and 0.3341 under point-painting) isolates projection accuracy from classifier accuracy. Holding the classifier fixed while only the projection changes reproduces most of the accuracy loss, so the camera-only condition's much lower full-support scores reflect projection and coverage limitations rather than a weaker underlying classifier.

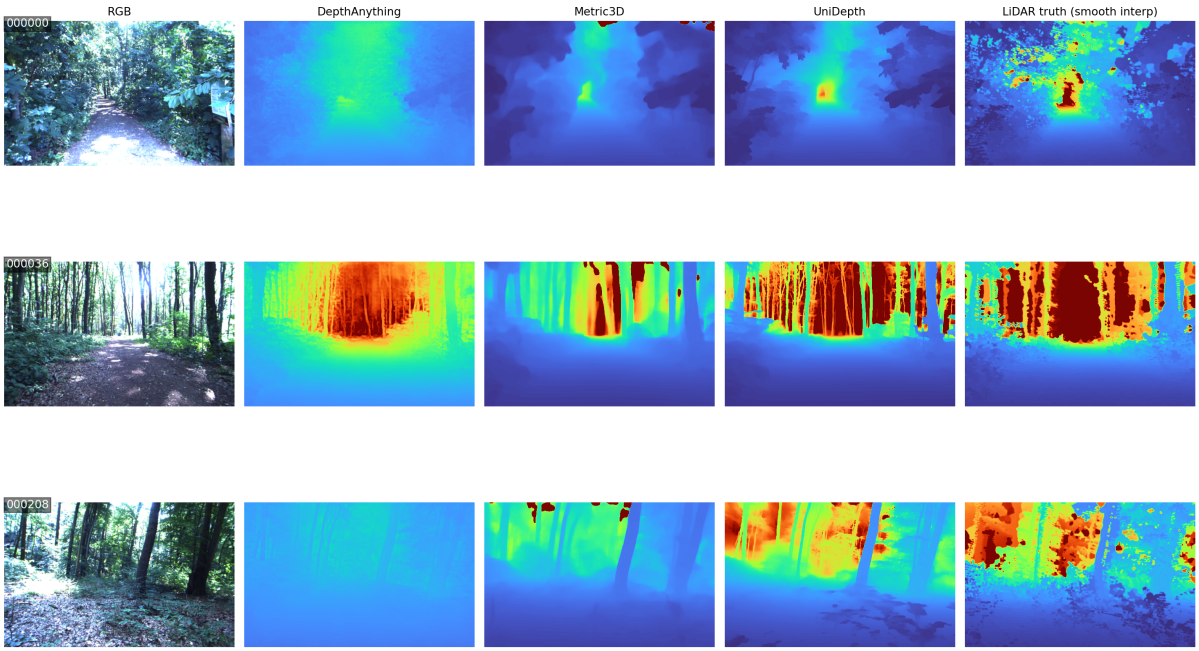


Figure 4.3: Qualitative depth-model comparison on three of the 15 FKIE forest frames used in the depth-quality diagnostic. Rows show frames 000000, 000036, and 000208; columns show RGB, DepthAnything, Metric3D, UniDepth, and smoothed/interpolated LiDAR truth. The figure is supporting qualitative evidence only; the model selection is based on the full 15-frame quantitative comparison reported in the text.

4.1.4 Condition B: LiDAR-only traversability

The LiDAR-only condition uses FASTC-S, a single-scan PointPillars-based network with a traversability completion module (Traver_Completion, a RegSeg-based BEV head), operating natively in BEV. The network produces a 512×512 grid at 0.2 m resolution over ± 51.2 m; its output is subsequently resampled to the shared ± 10 m, 0.1 m evaluation grid. It predicts traversability tiers together with a class-4 "unknown" label for cells it cannot classify. These unknown cells are masked during scoring rather than counted as an additional traversability class. This masking is distinct from the full-support convention used elsewhere in this thesis, under which cells a method does not predict on at all are counted as false negatives; here, the cells are within FASTC's coverage but explicitly flagged unknown, and are therefore excluded rather than penalised.

Because FASTC-S is single-scan while its training ground truth is a 141-scan aggregated label ($N = 141$, stride $S = 5$), the completion module infers occluded or otherwise unobserved cells from the single sweep.

This is the completion behaviour supervised by the aggregated ground truth. The evaluation uses a single FASTC checkpoint (`exp9_fw1`), carried unchanged across RELLIS-3D, FKIE forest, FKIE campus, and GOOSE; its full identifier is given in Appendix Table C.1. The full descriptions of these evaluation datasets are given in Section 3.4.

4.1.5 Condition C: Cost-level fusion

The fusion condition combines one of two alternative camera branches (GA-Nav or WVN, point-painted [19] onto the LiDAR-supported BEV grid) with the LiDAR branch (FASTC) at the cost level, on the shared grid defined in Section 4.1.1:

A traversability cost is the quantity a downstream planner uses to weigh a cell for path selection: higher cost means less preferable terrain, with the four discrete tiers mapped onto that same scale so camera and LiDAR outputs are directly combinable rather than requiring a separate reconciliation step. Fusing the two branches at this cost level, rather than at the raw prediction level, is what makes a single blend rule meaningful across modalities that otherwise output different things: a segmentation class versus a point-cloud classification.

1. **Fixed-weight fusion.** $\text{cost}_{\text{fused}} = w_{\text{cam}} \cdot \text{cost}_{\text{cam}} + w_{\text{lidar}} \cdot \text{cost}_{\text{lidar}}$, with weights normalised by the sum of *valid* weights. This means the weight only acts as a tie-break rule in cells where both legs are valid, not as a global mixing ratio over the whole grid: where only the LiDAR leg is valid the fused cost equals the LiDAR cost exactly, and camera-only cells pass through unchanged.

2. **Confidence-aware fusion.** The fixed weights are replaced by per-cell confidence drawn from each modality's own signal (WVN reconstruction-error, GA-Nav entropy, FASTC pillar density), each normalised to $[0, 1]$, and combined via the fusion rule identified in the rule study (Section 6.4).

The reported fusion result uses the per-domain best-performing rule: fixed $\alpha = 0.7$ (camera 0.7, LiDAR 0.3) on campus and confidence-weighted fusion on RELLIS-3D, forest, and GOOSE. Section 6.4 gives the full rule sweep.

Cost sampling and unknown-cell handling

The following conventions are applied uniformly across all three conditions when populating the shared grid: tiered cost values are sampled with nearest-neighbour interpolation (no synthetic mid-tier values), while confidence is sampled with bilinear interpolation. A cell is *unknown* if neither modality provides a valid value; it falls through to the LiDAR-only value if only one modality is valid at that cell; FASTC's own unknown/no-signal class (class 4) is always masked rather than treated as a traversability tier. A confidence of exactly zero causes that modality's cost to be excluded from the weighted combination at that cell, regardless of which modality reports it, leaving the fallback and validity rules above to determine the cell's value.

4.2 Implementation details

This section describes how the three conditions defined above are actually run: the shared software structure each method implements, the protocol used to evaluate them identically, and how checkpoints are tracked to avoid data leakage between training and evaluation.

4.2.1 Method class architecture

All three conditions (camera-only, LiDAR-only, and fusion) are implemented against a shared `BaseMethod` interface: a pure-Python class that consumes and returns `numpy` arrays with no ROS 2 dependency in the core prediction logic. This interface is wrapped twice, depending on how the method is run, without changing the method implementation itself:

- **Evaluation runner** – replays a recorded rosbag, calls the method class per frame, and scores the output against ground truth. This is the wrapper used for all results reported in Chapter 5.
- **ROS 2 deployment wrapper** – a thin (˜50-line) shell that subscribes to live sensor topics and publishes to `/traversability/costmap` for on-robot use (Section 4.3).

4.2.2 Sequential evaluation protocol

Only one method (camera-only, LiDAR-only, or fusion) is active at a time; every active method publishes to the same `/traversability/costmap` topic, and the metrics node is mode-agnostic: it scores whatever is published without needing to know which condition produced it. The same rosbag and ground truth are used for all three conditions on a given domain, so that method identity is the only variable between runs (Chapter 1).

4.2.3 Checkpoint provenance and leak avoidance

One checkpoint per condition was carried unchanged through every evaluation domain. The reported checkpoints are FASTC-S `exp9_fw1/model.pth.600`, GA-Nav `ganav_4tier/iter_38000.pth`, and WVN `wvn_head.pt`; their full SHA-256 values and the excluded alternatives are recorded in Appendix Table C.1. Checkpoints disqualified by leakage were excluded from all reported results. This follows the sequence-isolated RELLIS-3D protocol described in Chapter 3, which reserves sequence 00004 entirely for evaluation.

GA-Nav in particular required a deliberate isolation procedure: the originally available `ganav_rellis_6.pth` checkpoint was found to have been trained on a split that leaked 911 frames of the held-out evaluation sequence (00004), and was therefore discarded rather than used. A replacement four-tier GA-Nav model was retrained from an ImageNet-initialised MiT-b0 backbone (`load_from None`), on sequences 00000–03 only, with no exposure to sequence 00004 at any stage, the same isolation rule applied to every checkpoint used in this thesis. Checkpoint identity (via file hash) was verified to match across all evaluation domains before a result was reported.

4.3 Deployment optimisation on the Jetson AGX Thor

This work also investigates whether the selected approaches can be executed on an embedded robotic platform. The deployment target is the NVIDIA Jetson AGX Thor. The methods considered for deployment are the LiDAR-based FASTC branch and the camera-based GA-Nav and WVN branches.

The deployment workflow follows a PyTorch–ONNX–TensorRT path, consistent with established ONNX compilation and TensorRT deployment workflows [58, 59]. First, the deployable neural-network component of each method is exported from PyTorch to the Open Neural Network Exchange (ONNX) format. The ONNX graph is then compiled into an NVIDIA TensorRT engine using FP16 precision. TensorRT performs deployment-oriented graph optimisation, including operator fusion, kernel selection, memory planning, and reduced-precision execution.

However, TensorRT does not convert an entire Python inference program. Sensor loading, image resizing, LiDAR voxelisation, geometric projection, post-processing, and output formatting remain outside the TensorRT engine. Therefore, this work distinguishes between the exported neural-network boundary and the complete method pipeline. The implementation described in this chapter defines the TensorRT boundary for each method, while the corresponding runtime measurements are reported in Chapter 5.5.

4.3.1 TensorRT deployment boundaries

The three traversability methods have different input representations and preprocessing requirements. Consequently, a different TensorRT export boundary is used for each method.

- **FASTC:** TensorRT receives pre-voxelised LiDAR tensors and executes the neural-network component after voxelisation. Raw point-cloud loading and voxelisation remain outside TensorRT.
- **WVN:** TensorRT accelerates the frozen STEGO dense-code extractor. The WVN head, reconstruction-based confidence computation, expected-cost computation, and image-to-BEV projection remain outside TensorRT.
- **GA-Nav:** TensorRT accelerates the exported `encode_decode` image-logits network. Image preprocessing, softmax, expected-cost computation, entropy-confidence computation, and image-to-BEV projection remain outside TensorRT.

This design preserves the original method semantics while accelerating the most computationally relevant neural tensor graphs.

4.3.2 FASTC TensorRT deployment

FASTC, WVN, and GA-Nav each define a different boundary for what is captured in the exported TensorRT graph, summarised in Table 4.2. In every case, components outside this boundary were kept in Python for one of three reasons: they are data-dependent operations that cannot be captured in a static graph (FASTC's voxelisation), calibration-dependent geometric operations rather than learned

Table 4.2: TensorRT export boundaries for the three deployed methods. Class 4 in the FASTC output denotes the established unknown/no-signal class, handled during post-processing rather than as a traversability tier.

Method	TensorRT boundary	Input tensor	Output tensor
FASTC	Voxel encoder + middle BEV encoder + RegSeg classifier	`voxels` $\in \mathbb{R}^{N\times 55\times 4}$ `num_points` $\in \mathbb{R}^{N}$ `coords` $\in \mathbb{R}^{N\times 4}$	`logits` $\in \mathbb{R}^{1\times 5\times 512\times 512}$
WVN	STEGO dense-code extractor only	`image` $\in \mathbb{R}^{1\times 3\times 224\times 224}$	`code` $\in \mathbb{R}^{1\times 90\times 224\times 224}$
GA-Nav	Complete `encode_decode` network	`image` $\in \mathbb{R}^{1\times 3\times 375\times 600}$	`logits` $\in \mathbb{R}^{1\times 4\times 375\times 600}$

computation (the image-to-BEV projection shared by WVN and GA-Nav), or components small enough that exporting them offered negligible benefit while adding coupling risk (WVN's head, tied to its reconstruction-based confidence computation).

FASTC processes a LiDAR point cloud using a voxel-based representation, voxelising the raw scan with the `PointToVoxel` operator, an operation that remains outside the exported graph since it is data-dependent.

ONNX export compatibility adaptations

The original FASTC implementation could not initially be compiled directly by TensorRT because several PyTorch operations exported to ONNX operators that were ambiguous under dynamic input shapes or unsupported by the TensorRT ONNX parser. Two source-level adaptations were introduced to resolve this: an ambiguous squeeze operation in the PointPillars feature encoder was made explicit to the singleton point dimension, and a sequence-based channel-splitting operation in the RegSeg backbone, which exported to unsupported ONNX sequence operators, was replaced with explicit tensor slicing. Both changes preserve the trained parameters and mathematical behaviour of the network while making the exported graph unambiguous to the TensorRT parser; full implementation detail is given in Appendix B.

TensorRT engine generation

Because the number of occupied LiDAR voxels varies across scans, the FASTC TensorRT engine uses a dynamic optimisation profile with $N_{\min} = 1$, $N_{\text{opt}} = 4000$, and $N_{\max} = 12000$. The successful engine construction demonstrates that the FASTC neural-network component can be executed by TensorRT; the complete deployment latency, however, must still include voxelisation and post-processing, reported in Section 5.5.

4.3.3 WVN TensorRT deployment

WVN uses a frozen visual feature extractor followed by a lightweight trained head [8]. In the implementation used in this thesis, dense image features are produced by a STEGO/DINO-based feature extractor and then processed by the WVN head to obtain traversability logits, expected cost, and reconstruction-based confidence (Table 4.2). The head is retained outside TensorRT because it is lightweight compared with the visual feature extractor and is tightly coupled to the reconstruction-based confidence computation used by the method.

4.3.4 GA-Nav TensorRT deployment

GA-Nav is a camera-based semantic traversability method implemented through the MMSegmentation framework. Its exported TensorRT boundary covers the complete `encode_decode` image-logits network (Table 4.2). The downstream softmax, expected-cost computation, entropy-confidence computation, and image-to-BEV projection remain outside the TensorRT graph, keeping the engine focused on the neural segmentation model while retaining the same post-processing semantics as the original thesis implementation.

4.3.5 Fusion and point-painting boundary

The fusion condition combines the LiDAR and camera branches after their individual predictions have been produced. The camera branch is not used through its camera-only flat-ground BEV projection in this setting. This is the same fixed-floor projector implemented in the deployed GA-Nav method class (shared with WVN); it differs from the camera-only condition's evaluated projection, which used offline UniDepth-based depth predictions that were not integrated into the deployed method. Instead, image-plane class probabilities and confidence are transferred into the LiDAR-supported BEV grid using point-painting. The fusion stage then combines the FASTC BEV output with the painted camera evidence and applies the selected fusion rule.

This point-painting and fusion logic is retained outside TensorRT. It depends on the LiDAR points, camera intrinsics, camera–LiDAR extrinsics, BEV grid geometry, and NaN-safe fusion rules, and is therefore treated as a geometric post-processing stage rather than a neural-network graph. Its runtime impact is measured separately in Section 5.5.6.

4.3.6 Limitations of full-pipeline conversion

The deployment implementation does not attempt to convert the full Python inference pipelines into TensorRT in their entirety. TensorRT operates on tensor computation graphs and is well suited to neural-network components specifically, but the traversability pipelines also contain algorithmic and geometric operations that cannot naturally be represented as TensorRT graphs, and were therefore left outside the exported boundary.

For FASTC, this means raw point-cloud loading and voxelisation, both data-dependent operations that sit outside the exported neural network. For WVN and GA-Nav, it means the calibration-dependent image-to-BEV projection, a geometric operation whose behaviour depends on the camera's intrinsics, extrinsics, and the selected BEV grid. Each of these excluded stages is retained in Python and included in the full staged latency measurements reported in Chapter 5.5.

This distinction matters for how the deployment numbers should be read. Network-only TensorRT latency describes only the accelerated neural component in isolation, whereas the complete deployment latency also includes preprocessing, post-processing, and projection on top of it. For this reason, the evaluation chapter reports matched full-pipeline measurements instead, in which the PyTorch and TensorRT paths share identical preprocessing, post-processing, and projection stages, and differ only in the exported neural-network boundary itself.

5

Evaluation and results

This chapter presents the results of the experimental evaluation. It begins with the sensor calibration procedure and the predictive evaluation metrics used throughout, followed by evaluation on the custom ROVO dataset, a cross-dataset accuracy comparison across all four domains, and an assessment of computational cost and deployment performance.

5.1 Sensor calibration

Two calibration procedures were performed: intrinsic calibration of the camera, and extrinsic calibration between the camera and the front LiDAR.

5.1.1 Camera intrinsic calibration

After an initial calibration over all detected frames, a small number of frames with anomalous reprojection errors above 1.0 pixel were removed and the calibration was re-solved. The final mean reprojection error was 0.161 px across 245 retained frames.

The calibrated intrinsic matrix was

$$K = \begin{bmatrix} 529.70 & 0 & 485.29 \\ 0 & 530.71 & 302.48 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.1}$$

with distortion coefficients $(k_1, k_2, p_1, p_2, k_3) = (-0.115,\ +0.055,\ +5.7{\times}10^{-4},\ -2.4{\times}10^{-6},\ 0)$. The near-equal focal lengths ($f_x \approx f_y$) confirm square pixels, and the principal point lies within 1.5 px of the geometric image centre, both consistent with a well-aligned modern CMOS camera. The negative k_1 corresponds to mild barrel distortion typical of a moderate wide-angle lens. Tangential coefficients are negligible, indicating that the lens is well-aligned with the sensor plane. The estimated calibration target poses span a range of distances and orientations relative to the camera (Figure 5.2), providing the pose diversity needed to constrain the intrinsic parameters reliably; a narrow range of board poses would leave some parameters under-constrained.

The result was exported to a ROS-compatible `camera_info` YAML file and committed to the project repository under `config/calibration/`. After loading the file via the uEye driver's `camera_info_url`

Table 5.1: Final ros2_calib Extrinsic Transformation Parameters

Parameter	Value
x	0.044067 m
y	0.42941 m
z	−0.33433 m
roll	1.73962 rad
pitch	0.13331 rad
yaw	−1.57403 rad

parameter, the published `/ueye_cam/froody/camera_info` topic was verified to match the calibration values, completing the sensor configuration prior to the subsequent camera-LiDAR extrinsic calibration.

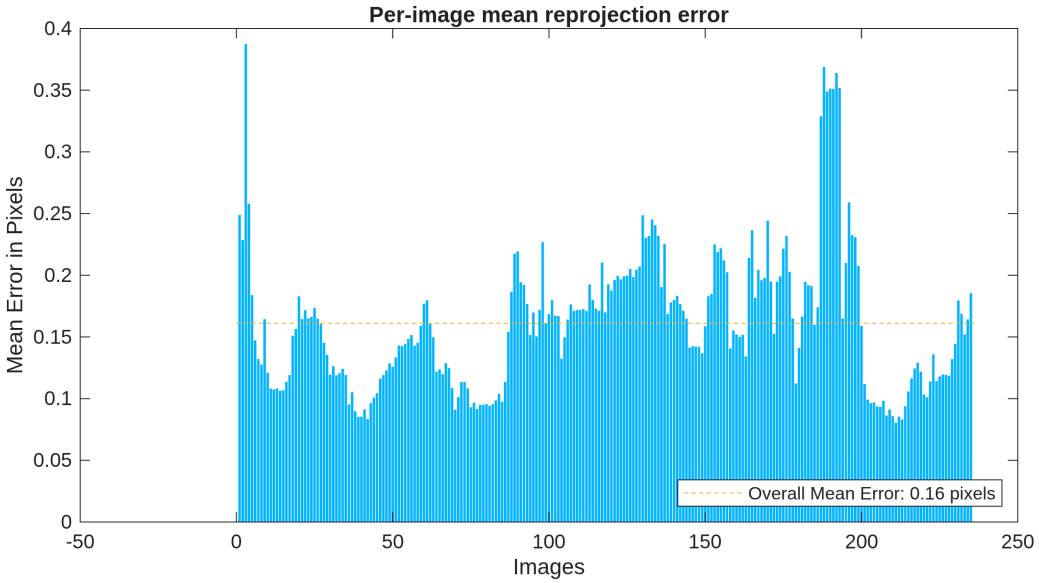


Figure 5.1: Per-image mean reprojection error after refinement. The dashed line indicates the overall mean of 0.161 px across 245 retained frames.

5.1.2 Camera-LiDAR extrinsic calibration

All extrinsic calibration methods used the intrinsic parameters reported in Matrix 5.1.

`ros2_calib`

Result The final estimated parameters are detailed in Table 5.1.

A quantitative comparison against the MATLAB alternative, and the rationale for selecting this transform, are given in Section 5.1.4.

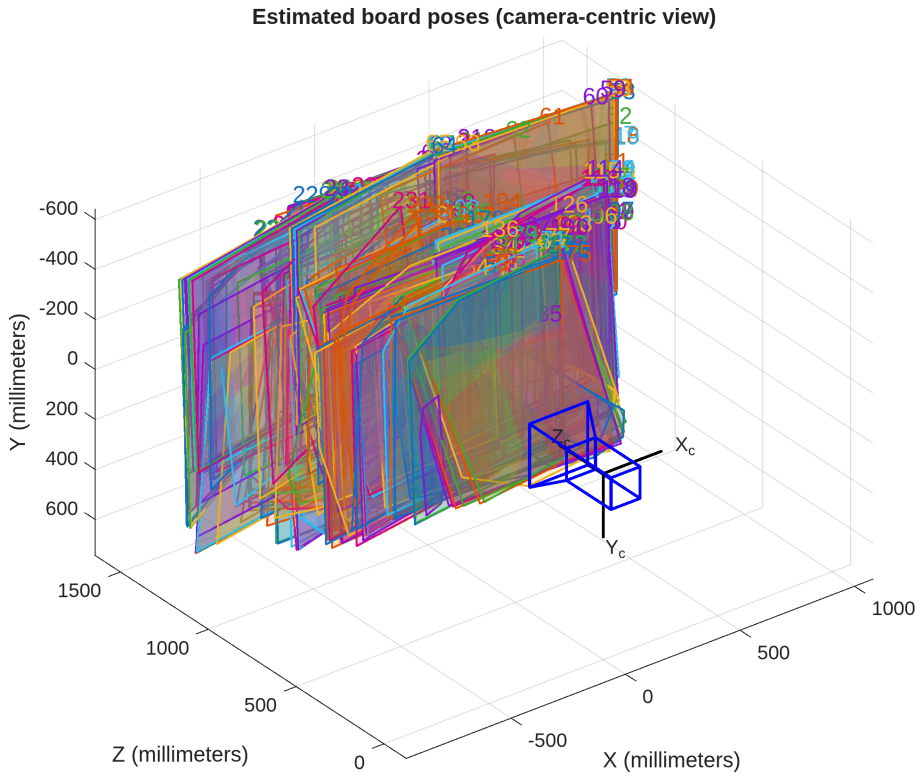


Figure 5.2: Estimated calibration target poses relative to the camera (camera-centric view). The spread of board positions illustrates the pose diversity used to constrain the intrinsic parameters.

MATLAB Lidar Camera Calibrator

Results The target-based extrinsic calibration was successfully computed using the 10 synchronized image-point cloud pairs. The final estimated transformation matrix, which maps coordinates from the LiDAR frame to the camera frame, is detailed below alongside the quantitative error metrics. Furthermore, both the intermediate target detection and the final qualitative validation are presented to confirm the accuracy of the alignment.

$$T_{\text{cam}\leftarrow\text{lidar}} = \begin{bmatrix} 0.0078 & -0.9902 & -0.1395 & 0.4116 \\ -0.1309 & -0.1393 & 0.9816 & 0.5103 \\ -0.9914 & 0.0106 & -0.1307 & -0.0137 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

Per-image calibration errors reported by MATLAB: Below images show how the per image calibrations were reported by the MATLAB method

Table 5.2: Per-image calibration errors reported by MATLAB

Metric	**Min**	**Max**	**Mean**
Translation Error (m)	0.0043	0.0462	0.0237
Rotation Error (°)	0.415	1.703	1.284
Reprojection Error (px)	0.614	7.837	3.701

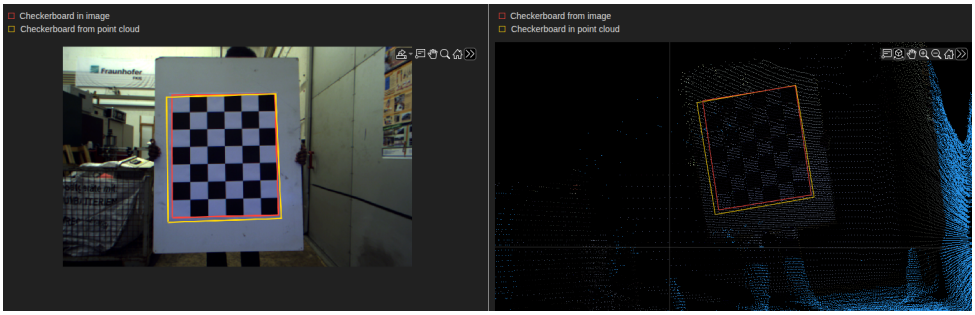


Figure 5.3: Target detection in the MATLAB Lidar Camera Calibrator. The application extracts the checkerboard plane from the camera image (left) and the corresponding 3D region from the LiDAR point cloud (right). The overlaid bounding boxes demonstrate the geometric correspondence used to compute the extrinsic transformation.

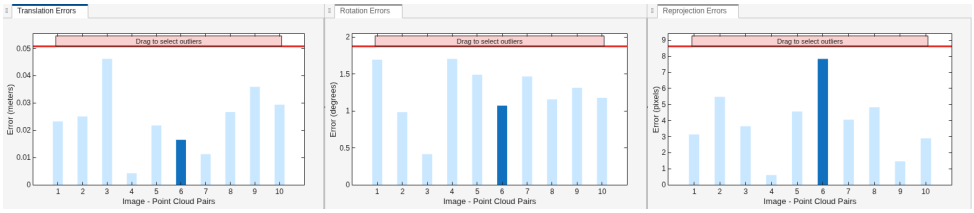


Figure 5.4: Calibration error metrics evaluated across 10 image-point cloud pairs using the MATLAB Lidar Camera Calibrator. The plots display translation errors in meters (left), rotation errors in degrees (center), and reprojection errors in pixels (right). These metrics quantify the accuracy and consistency of the estimated camera-LiDAR extrinsic transformation across different target poses.

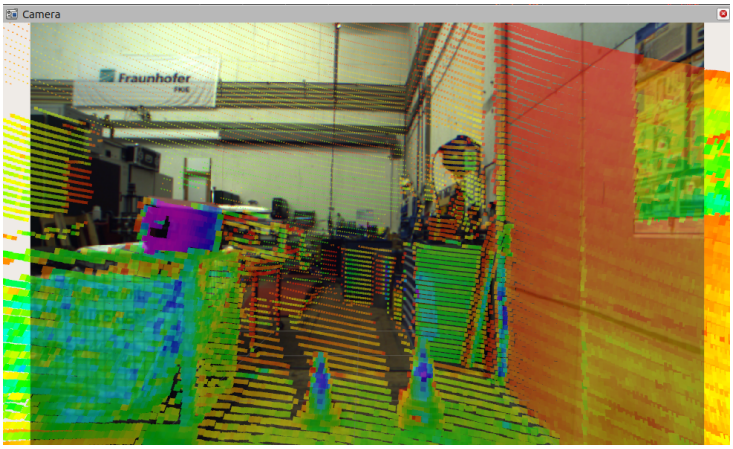


Figure 5.5: Qualitative validation of the camera-LiDAR extrinsic calibration in RViz. The LiDAR point cloud is projected onto the camera image plane using the transformation estimated by the MATLAB target-based method. The close alignment of the projected points with structural edges in the scene, such as the walls and foreground objects, visually confirms the accuracy of the calibration.

Koide targetless calibration

The targetless calibration approach could not produce a reliable camera-LiDAR extrinsic estimate on the available recordings. Two datasets were evaluated, but both failed during the initial correspondence stage due to an insufficient number of valid RANSAC inliers.

Bag 1: Robot in motion In the first recording, the robot was moving while LiDAR scans were accumulated into a dense point cloud using CT-ICP (Section 3.2.3). This produced a scene representation spanning the full trajectory rather than a map corresponding to any single camera view. SuperGlue produced only 2/10 RANSAC inliers, and although the optimizer returned a transform, the projected alignment was visually incorrect.

Bag 2: Stationary robot with AprilGrid motion In the second recording, the robot remained stationary while a person moved the AprilGrid board in front of the sensors. In this case, SuperGlue produced only 1/1 inlier, which was also insufficient for reliable initialization. The resulting data did not permit stable extrinsic estimation.

Evaluation The failure of the Koide method is attributed to two factors. First, the method assumes a static scene during point cloud accumulation; therefore, dynamic content introduces ghosting artifacts that invalidate image-to-map consistency. Second, the repeated AprilGrid pattern does not provide sufficiently unique visual features for reliable SuperGlue matching, which prevents robust 2D-3D correspondence estimation.

These results indicate that the Koide targetless approach was not suitable for the datasets collected in this work without a different recording protocol or a stronger initial pose estimate.

Table 5.3: Comparison of Camera-LiDAR Extrinsic Calibration Methods

Property	**MATLAB**	**Koide targetless**	**`ros2_calib`**
Target required	Plain checkerboard	None	Plain checkerboard
ROS2 Jazzy native	No (MATLAB)	Yes (built from source)	Docker
Bag format supported	PCD + PNG files	sqlite3 (native build)	MCAP
URDF export	No	No	Yes
SuperGlue / deep learning	No	Yes (CUDA required)	No
Sensitivity to scene content	Low	High	Low
Initial guess required	No	Recommended	No
Status	Successful	Unsuccessful on available data	Successful

Table 5.4: Comparison of extrinsic calibration results after converting the MATLAB estimate to the ROS TF convention

Component	**`ros2_calib`**	**MATLAB (converted)**	**Difference**
x (m)	0.0441	0.0500	0.006 m
y (m)	0.4294	0.4788	0.049 m
z (m)	-0.3343	-0.4453	0.111 m
Roll (rad)	1.7396	1.7032	0.036 rad (2.1°)
Pitch (rad)	0.1333	0.1399	0.007 rad (0.4°)
Yaw (rad)	-1.5740	-1.5629	0.011 rad (0.6°)
Overall rotation diff.	-	-	2.3°
Overall translation diff.	-	-	12.2 cm

5.1.3 Method comparison

Table 5.3 compares the three camera-LiDAR extrinsic calibration methods considered in this work in terms of their practical requirements, software compatibility, and expected robustness. The comparison is intended to highlight the trade-offs between target-based and targetless approaches, as well as the additional engineering effort required to integrate each method into the ROS 2 workflow.

As shown in Table 5.3, the two target-based methods required a plain checkerboard and were less sensitive to scene content, whereas the Koide targetless method avoided the need for a calibration target but introduced stronger dependencies on scene structure, feature matching quality, and preprocessing. Among the three approaches, `ros2_calib` offered the most direct integration into the ROS 2 ecosystem through MCAP support and URDF export, while the MATLAB workflow required external file conversion and the Koide pipeline required the greatest implementation effort.

To compare the target-based calibration results on a common basis, the MATLAB estimate was converted to the ROS TF convention by inverting the reported transformation matrix. Table 5.4 summarises the resulting differences between the `ros2_calib` and MATLAB estimates in translation and Euler-angle representation.

The two methods differ by an overall rotational difference of approximately 2.3° and an overall

translational difference of 12.2 cm, with the largest single components being a 2.1° roll difference and an 11.1 cm z-axis difference. At a typical evaluation range of 10 m, a 2.1° angular offset alone corresponds to a lateral positional error of approximately 36 cm, so this disagreement is not negligible for BEV projection accuracy. No independent ground-truth measurement of the camera-LiDAR offset was available to determine which method's estimate is closer to the true transform. The two optimisation pipelines simply converged to different sensor origins, and neither can be confirmed more accurate from the available data.

5.1.4 Selected calibration and system integration

Based on the comparative evaluation, the `ros2_calib` result was selected as the primary camera-LiDAR extrinsic calibration for the remainder of this work. This choice was motivated by three factors: qualitative verification in RViz2 showed good alignment of projected LiDAR points with visible scene structure, the result was already expressed in the ROS TF convention and therefore required no additional frame-convention conversion, and the method jointly optimised the transform over all calibration pairs rather than estimating pose consistency on a per-image basis. The RViz2 check was performed on the `ros2_calib` transform specifically, though the resulting overlay was not retained as a figure; the RViz overlay shown in Figure 5.5 instead visualises the MATLAB transform, included for comparison rather than as evidence for the `ros2_calib` selection.

The selected transform was integrated into the robot description as a fixed joint in `rovoA.urdf.xacro` between `rovo1/lidar/front/sensor` and `rovo1/froody`. Because `ros2_calib` reports the camera pose in the `rovo1/lidar/front/lidar` data frame, the final URDF transform was obtained by composing this estimate with the known 180° rotation about the z-axis between the LiDAR data frame and the corresponding URDF sensor frame.

5.2 Predictive evaluation metrics

All predictive results in this thesis use the full-support convention unless otherwise stated: every GT-valid cell on the shared 200 × 200, 0.1 m, ±10 m sensor-centred BEV grid is scored, and cells for which a method produces no prediction are counted as false negatives rather than excluded. This choice matters because it is not the only convention present in this thesis's working records; other tables and intermediate results elsewhere use coverage-restricted scoring, which is not numerically comparable to full-support figures.

For a class c with true positives TP_c, false positives FP_c, and false negatives FN_c, intersection-over-union is $\mathrm{IoU}_c = \mathrm{TP}_c/(\mathrm{TP}_c + \mathrm{FP}_c + \mathrm{FN}_c)$, and mIoU is the unweighted mean of IoU_c over the reported class set. Precision is $\mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$ and recall is $\mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$; where reported per tier (e.g. obstacle-tier recall), TP, FP, and FN are restricted to that tier's predictions and ground truth. Two mIoU variants are reported throughout: mIoU-4 is the mean IoU across all four traversability tiers (free, low, medium, obstacle); mIoU-3 is the mean IoU across the three tiers excluding free (Section 3.4 discusses a dataset-specific consequence of this distinction).

Obstacle-tier recall is reported under two conventions throughout this thesis. Full support scores every GT-valid cell, so unseen cells count as false negatives; this is the deployment-honest convention used everywhere unless stated otherwise. Valid support scores only the cells a given method actually populates. Obstacle IoU and precision are reported on full support only, unless otherwise noted.

5.3 Custom ROVO dataset evaluation

The custom dataset introduced in Section 3.4.3 is used to evaluate the behaviour of the camera, LiDAR, and fusion methods under real robot operating conditions. This section reports the campus domain specifically; forest results are integrated into the cross-dataset comparison in Section 5.4 alongside the other three evaluation domains.

The campus dataset was collected to provide the fourth evaluation domain, distinct in scene content from RELLIS-3D, GOOSE, and the forest domain. The comparison with the forest domain holds the sensor platform fixed. It comprises five sequences of 30 annotated frames each (150 frames total), captured on the same robotic platform and sensor rig as the forest dataset, on and around a research-campus site. Scene content spans a mixed built and unstructured environment: paved and cobbled surfaces, grass paths and open field, parked vehicles, buildings, fences, bins, and street furniture, under varying daylight conditions across sequences. This composition is deliberate: it places obstacle classes and ground materials outside the vegetated off-road distribution on which both the geometric and the camera methods were trained, while remaining a plausible operating environment for the same platform.

Ground truth follows the same annotation convention as the forest dataset. Each selected LiDAR scan was opened in a bird's-eye-view labelling tool and the dense BEV raster painted directly by the annotator into the four traversability tiers with an explicit *ignore* class; the point-wise labels distributed with the dataset are a sparse export of that raster, not an independent annotation. No multi-scan accumulation, prior map, or image-projected semantic source was used, so ground-truth density reflects annotation coverage and scene observability rather than aggregation. Point-label ground truth (Section 3.4.3) is derived directly from LiDAR returns, so every scored cell has direct point support by construction; this is a different property from the dense manually-painted raster it was exported from, which could label cells with no underlying return. Both conventions and their relationship are described in full in Section 3.4.3.

Because the platform is shared with the forest dataset, the forest camera calibration was used unchanged. Its validity on campus was established in two independent ways: by LiDAR-to-image overlay inspection on all five sequences, and by a quantitative label-agreement perturbation study. The latter is reported here for its negative result: agreement between projected point labels and image-plane labels peaks at non-zero rotational and translational offsets on campus, but applying the identical procedure to RELLIS-3D, whose calibration is independently verified against a calibrated extrinsic quaternion, produces offsets of equal or greater magnitude. The metric is therefore biased by label-boundary softness rather than diagnostic of calibration error, and the shared calibration was retained.

All campus evaluation uses the same 200×200 grid at 0.1 m resolution over a ± 10 m sensor-centred footprint, the same four-tier taxonomy and cost mapping, and the same committed scoring pipeline and checkpoints as the other two domains.

5.4 Cross-dataset accuracy comparison

Before turning to domain-specific analysis, this section collects the headline accuracy of all three conditions (camera-only semantics via UniDepth monocular projection, Condition A, Section 4.1.2; LiDAR-only, Condition B; and fusion, Condition C) across all three evaluation datasets under one metric convention. Three datasets are compared: RELLIS-3D (held-out sequence 00004), GOOSE (validation split), and the FKIE custom dataset, collected on the same ROVO platform, sensor rig, and calibration across two recording sessions. The forest and campus splits of the FKIE dataset are scene groups within that single collection, not separate datasets, and are reported as such throughout this section.

All values in this section use the full-support convention: every GT-valid cell on the shared 200×200, 0.1 m, ± 10 m sensor-centred BEV grid is scored, and cells for which a method produces no prediction are counted as false negatives rather than excluded. This choice matters because it is not the only convention present in this thesis's working records: other tables and intermediate results elsewhere use coverage-restricted scoring, which is not numerically comparable to what follows. The fusion condition uses confidence-weighted fusion on RELLIS-3D, forest, and GOOSE, and fixed $\alpha = 0.7$ fusion on campus, matching the rule established as appropriate for each domain in Chapter 6.

5.4.1 Dataset composition

Table 5.5 gives the ground-truth class balance for each dataset, which is necessary context for reading the mIoU columns below. RELLIS-3D contains no *free*-class ground truth at all. This is a property of the evaluated sequence rather than the class mapping: RELLIS-3D's native taxonomy maps `dirt`, `asphalt`, and `concrete` to *free*, but none of these classes appears in the ground-truth labels of the held-out test sequence (00004) used for evaluation. Its two mIoU columns in Table 5.6 are therefore identical, and the mIoU value that other tables in this thesis label as RELLIS-3D's "4-class" figure has therefore always been computed as a 3-class average over low, medium, and obstacle, since free contributes no cells. Both columns are reported for every dataset rather than only the 4-class figure, because without the 3-class column, RELLIS-3D's mIoU would look directly comparable to the other three datasets' 4-class mIoU when it is not.

Table 5.5: Ground-truth class distribution by dataset (percentage of GT-valid cells). RELLIS-3D has no *free*-class ground truth.

Dataset	Free	Low	Medium	Obstacle
RELLIS-3D (00004)	0.00	71.13	22.35	6.52
FKIE forest (00000)	31.83	17.47	25.99	24.71
FKIE campus (pooled)	19.90	59.05	12.95	8.10
GOOSE (val)	3.82	33.99	32.90	29.30

5.4.2 Headline cross-dataset results

On campus, Table 5.6 additionally reports the best-performing rule (fixed $\alpha = 0.7$) found in the full seventeen-arm sweep, the complete set of fusion rules tested per domain and detailed in Section 6.4 and Appendix A, alongside the common cross-domain reference rule used elsewhere (confidence-weighted fusion). As with every rule- selection result in this thesis, alpha = 0.7 was selected because it produced the best result on the same data being reported here, not because of an independent rationale, and there was no held-out set to test whether it generalizes to new data. This should be read as the best-observed operating point under this evaluation protocol, not a validated generalization.

Table 5.6: Cross-dataset map quality, full support over all GT-valid Path-A cells, for the camera-only, LiDAR-only, and fusion conditions.

Dataset	Method	mIoU (4-class)	mIoU (3-class)	Coverage
RELLIS-3D (00004)	A – GA-Nav (UniDepth)	0.0275	0.0275	6.02%
	A – WVN (UniDepth)	0.0004	0.0004	3.68%
	B – FASTC	0.4923	0.4923	–
	C (conf.-wt., common) – GA-Nav	0.4942	0.4942	–
	C (conf.-wt., common) – WVN	0.4926	0.4926	–
FKIE forest (00000)	A – GA-Nav (UniDepth)	0.0182	0.0242	19.2%
	A – WVN (UniDepth)	0.0205	0.0272	13.8%
	B – FASTC	0.1567	0.2090	–
	C (conf.-wt., common) – GA-Nav	0.1472	0.1963	–
	C (conf.-wt., common) – WVN	0.1573	0.2097	–
FKIE campus (pooled)	A – GA-Nav (UniDepth)	0.0396	0.0525	15.1%
	A – WVN (UniDepth)	0.0268	0.0356	10.3%
	B – FASTC	0.1741	0.2282	–
	C (conf.-wt., common) – GA-Nav	0.1804	0.2363	–
	C (conf.-wt., common) – WVN	0.1745	0.2284	–
	C (best: fixed $\alpha = 0.7$) – GA-Nav	0.1761	0.2307	–
	C (best: fixed $\alpha = 0.7$) – WVN	0.1810	0.2373	–
GOOSE (val)	A – GA-Nav (UniDepth)	0.0164	0.0209	6.02%
	A – WVN (UniDepth)	0.0153	0.0203	3.06%
	B – FASTC	0.2811	0.3531	–
	C (conf.-wt., common) – GA-Nav	0.2808	0.3532	–
	C (conf.-wt., common) – WVN	0.2819	0.3541	–

Absent predictions are scored as false negatives. The table reports both 4-class and 3-class mIoU, and coverage for the camera-only rows; coverage is not repeated for the LiDAR-only and fusion rows. The camera-only condition uses UniDepth monocular depth projection, and its results are weak across all

four datasets as an intentional finding. The point-painted camera evaluation uses LiDAR support and is reported separately as the fusion camera leg in Section 5.4.3; its design rationale is in Section 4.1.3. RELLIS-3D's 4-class and 3-class columns are identical because it has no *free*-class ground truth. The fusion condition is reported using confidence-weighted fusion as the common cross-domain reference rule, chosen for comparability across domains. On RELLIS-3D, forest, and GOOSE, the rule sweep shows fusion is largely rule-insensitive at the aggregate level (no rule changes any metric by more than a small margin), so only the common rule is reported. On campus, fixed $\alpha = 0.7$ is the multi-metric Pareto choice under the aggregate/per-domain criterion, so both the common rule and the best rule are shown.

Table 5.7: Cross-dataset obstacle detection (recall, obstacle IoU, and precision), full support, for the camera-only, LiDAR-only, and fusion conditions.

Dataset	Method	Obstacle IoU	Recall (full → valid)	Precision	Coverage
RELLIS-3D (00004)	A – GA-Nav (UniDepth)	0.0010	0.001 → 0.001	0.5990	6.02%
	A – WVN (UniDepth)	0.0001	0.000 → 0.000	0.5123	3.68%
	B – FASTC	0.6023	0.9449 → 0.945	0.6242	–
	C (conf.-wt., common) – GA-Nav	0.6049	0.9443 → 0.944	0.6272	–
	C (conf.-wt., common) – WVN	0.6026	0.9448 → 0.945	0.6246	–
FKIE forest (00000)	A – GA-Nav (UniDepth)	0.0726	0.127 → 1.000	0.1454	19.2%
	A – WVN (UniDepth)	0.0299	0.033 → 0.596	0.2385	13.8%
	B – FASTC	0.2632	0.4171 → 0.439	0.4163	–
	C (conf.-wt., common) – GA-Nav	0.2673	0.4835 → 0.508	0.3742	–
	C (conf.-wt., common) – WVN	0.2642	0.4165 → 0.438	0.4193	–
FKIE campus (pooled)	A – GA-Nav (UniDepth)	0.0489	0.086 → 0.655	0.1019	15.1%
	A – WVN (UniDepth)	0.0136	0.014 → 0.193	0.4881	10.3%
	B – FASTC	0.0918	0.1070 → 0.108	0.3923	–
	C (conf.-wt., common) – GA-Nav	0.1264	0.1636 → 0.165	0.3572	–
	C (conf.-wt., common) – WVN	0.0907	0.1055 → 0.106	0.3919	–
	C (best: fixed $\alpha = 0.7$) – GA-Nav	0.1333	0.1693 → 0.171	0.3852	–
	C (best: fixed $\alpha = 0.7$) – WVN	0.1057	0.1237 → 0.125	0.4216	–
GOOSE (val)	A – GA-Nav (UniDepth)	0.0184	0.020 → 0.020	0.1922	6.02%
	A – WVN (UniDepth)	0.0019	0.002 → 0.002	0.2576	3.06%
	B – FASTC	0.5037	0.8707 → 0.871	0.5445	–
	C (conf.-wt., common) – GA-Nav	0.5047	0.8713 → 0.872	0.5454	–
	C (conf.-wt., common) – WVN	0.5041	0.8710 → 0.872	0.5448	–

Condition A is camera-only via UniDepth monocular depth projection. Full support scores all GT-valid Path-A cells, with missing predictions counted as false negatives. Camera-only results are weak across all four datasets; this is an intentional finding, not a limitation of the analysis. The point-painted camera evaluation uses LiDAR support and is reported separately as the fusion camera leg in Section 5.4.3 (design rationale: Section 4.1.3). Coverage is shown for the camera-only rows; it is not repeated for the LiDAR-only and fusion rows. Rows and the fusion-rule convention match Table 5.6. Recall is reported as full → camera/LiDAR-valid support throughout: full scores every GT-valid cell (unseen cells count as

false negatives, the deployment-honest convention used everywhere in this thesis); valid scores only the cells each condition actually populates. Obstacle IoU and precision are reported on full support only. The two conventions nearly coincide for the LiDAR-only and fusion conditions, whose coverage is 94–100 % of GT-valid cells on every domain (Section 6.1); they diverge sharply for the camera-only condition, whose coverage stays well below 25% across all domains. The same scoring convention barely moves the number for a fully-covering modality but transforms it for the camera, and this contrast, visible directly in the paired cells above rather than asserted here, is what makes the camera's coverage gap meaningful. GA-Nav and WVN are each self-selected onto different cells within the camera-only condition (their camera-valid supports are not the same cells), so the two backbones' valid-support figures are not scored on identical ground. The fusion condition is reported using confidence-weighted fusion as the common cross-domain reference rule, chosen for comparability across domains. On RELLIS-3D, forest, and GOOSE, the rule sweep shows fusion is largely rule-insensitive at the aggregate level (no rule changes any metric by more than a small margin), so only the common rule is reported. On campus, fixed $\alpha = 0.7$ is the multi-metric Pareto choice under the aggregate/per-domain criterion, so both the common rule and the best rule are shown. Under the common rule, campus fusion still improves on the LiDAR baseline (GA-Nav recall $0.1070 \rightarrow 0.1636$). Camera-valid recall for the campus confidence-weighted arm is not available and is left unpaired above rather than estimated; the campus fixed-$\alpha = 0.7$ row is paired as elsewhere. The rule was selected on the same data it is reported on here, with no held-out selection set.

5.4.3 Point-painted camera leg (fusion camera projection)

These values were previously reported in the cross-dataset tables under the Condition A rows. However, unlike Condition A's UniDepth-based projection, the point-painted camera-leg values shown here use LiDAR support, so they belong to the fusion methodology rather than the camera-only condition. The underlying values are unchanged, only their methodological interpretation differs.

Table 5.8: Point-painted camera-leg obstacle detection used for fusion.

Dataset	Backbone	Obs. IoU	Recall (full → camera-valid)	Precision
RELLIS-3D (00004)	GA-Nav	0.0083	0.008 → 0.639	0.5951
RELLIS-3D (00004)	WVN	0.0022	0.002 → 0.444	0.5737
FKIE forest	GA-Nav	0.0823	0.148 → 1.000	0.1562
FKIE forest	WVN	0.0442	0.048 → 0.774	0.3560
FKIE campus	GA-Nav	0.0689	0.138 → 0.904	0.1206
FKIE campus	WVN	0.0468	0.048 → 0.632	0.6546
GOOSE (val)	GA-Nav	0.0230	0.025 → 0.503	0.2367
GOOSE (val)	WVN	0.0052	0.005 → 0.715	0.3894

Obstacle IoU and precision are full-support values. Recall is shown as full support → camera-valid support; missing predictions count as false negatives for the full-support value. See Section 4.1.3 for the projection design rationale.

Coverage in Table 5.9 follows the same point-painted-versus-Condition-A distinction as Table 5.8. Values are unchanged; coverage is the fraction of GT-valid cells reached by the point-painted leg. GOOSE total coverage was not reported in the current cross-domain record.

Table 5.9: Point-painted camera-leg map quality used for fusion.

Dataset	Backbone	mIoU (4-class)	mIoU (3-class)	Coverage
RELLIS-3D (00004)	GA-Nav	0.0393	0.0393	7.6%
RELLIS-3D (00004)	WVN	0.0227	0.0227	4.7%
FKIE forest	GA-Nav	0.0206	0.0275	20.9%
FKIE forest	WVN	0.0290	0.0385	14.9%
FKIE campus	GA-Nav	0.0519	0.0685	19.0%
FKIE campus	WVN	0.0428	0.0565	12.2%
GOOSE (val)	GA-Nav	0.0273	0.0359	not reported
GOOSE (val)	WVN	0.0161	0.0205	not reported

Read together, the two tables establish five points that recur through the rest of this thesis. First, the genuinely camera-only condition's results are weak across all four domains: full-support mIoU remains between 0.0004 and 0.0396, while the LiDAR baseline is substantially stronger. This is the result of the UniDepth camera-only projection, not evidence about the point-painted fusion leg. The separate point-painted tables report a deliberately different quantity, obstacle recall rather than mIoU, since only recall is meaningful within a support region defined by the camera leg itself: within the cells reached by the LiDAR-supported fusion camera leg, GA-Nav's obstacle recall is 0.639 on RELLIS-3D, 1.000 on forest, 0.904 on campus, and 0.503 on GOOSE. Those conditional recalls describe the camera contribution available to fusion, not standalone camera performance. Chapter 6 develops the resulting coverage–precision trade-off and its consequences for the fusion rules.

Second, fusion's benefit tracks LiDAR-baseline weakness inversely across the four dataset groupings, though less cleanly than the raw baseline ordering alone would suggest. On RELLIS-3D, where FASTC alone already reaches 0.4923 mIoU (Figure 5.7), fusion moves the needle by less than two-thousandths (0.4923 $\rightarrow$ 0.4942 with GA-Nav); on GOOSE, where FASTC alone reaches 0.2811 mIoU, both fusion arms land within 0.001 of the baseline in either direction. On forest, fusion's effect on map quality is mixed rather than uniformly positive: GA-Nav fusion (0.1472 mIoU) sits slightly below the LiDAR-only baseline (0.1567), while WVN fusion (0.1573) is essentially flat against it. Campus, whose LiDAR baseline has the weakest obstacle recall and obstacle IoU of the four domains (0.1070 recall, 0.0918 obstacle IoU) even though its mIoU (0.1741) is not the lowest (forest's 0.1567 is lower), is where fusion produces its clearest map-quality gain: GA-Nav fusion reaches 0.1804 mIoU against FASTC's 0.1741, a gain that narrows under the fixed-blend rule reported as campus's best (0.1761 with fixed $\alpha = 0.7$, still positive but smaller than confidence-weighted here); WVN's campus fusion (0.1745) is again nearly flat.

That obstacle-level picture is the third point: campus is the domain where the point-painted fusion camera leg has the clearest conditional advantage over the LiDAR baseline. On camera-valid support,

GA-Nav reaches 0.904 recall against FASTC's 0.1070 full-support recall. This is not a standalone camera-only result: the UniDepth condition remains weak on full support. It nevertheless explains why campus is the one domain with substantial fusion headroom: the LiDAR-supported camera evidence supplies high-recall information where the LiDAR baseline is weakest, and fusion makes that evidence available in the combined output.

Fourth, the backbone contrast on forest is invisible in mIoU but requires care in the obstacle-detection panel. GA-Nav fusion lifts obstacle recall from FASTC's 0.4171 (Figure 5.6) to 0.4835, alongside a precision drop from 0.4163 to 0.3742: the signature of a camera leg that calls obstacle across nearly all of its point-painted support (98.8% pooled forest, established separately) rather than one that is genuinely selective. The recall gain is real in the sense that it recovers obstacles LiDAR missed, but it should not be read as evidence of accurate camera classification in forest; it is closer to a near-constant-positive vote riding on real LiDAR gaps. WVN fusion, by contrast, leaves recall essentially unchanged (0.4171 without fusion, 0.4165 with it), consistent with a camera leg that is far more conservative in this domain. Table 5.6 alone does not make either behavior legible: forest mIoU under GA-Nav fusion is actually below the LiDAR-only baseline even as its obstacle recall improves, which is consistent with a broad, low-precision obstacle vote that helps the one metric it inflates while costing the aggregate.

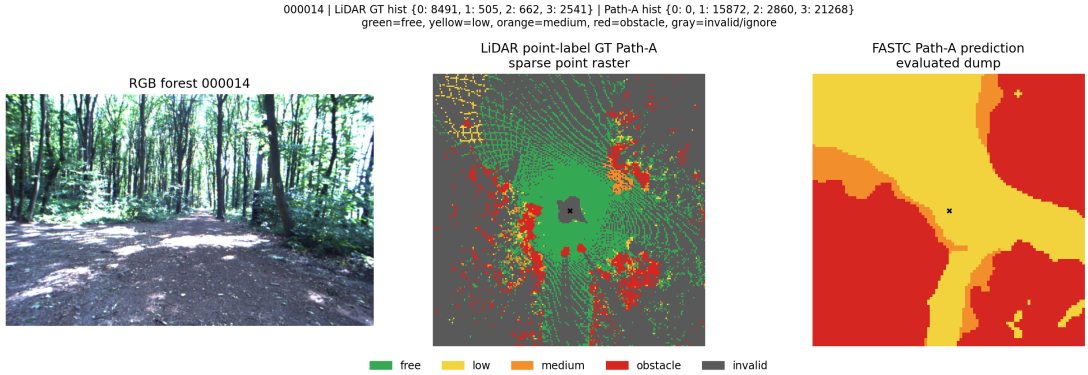


Figure 5.6: Qualitative FASTC prediction on FKIE forest sequence 00000, frame 000014: RGB, LiDAR point-label ground truth (Path-A), and the evaluated FASTC Path-A prediction. Restricted to GT-valid cells (12,199 total), the frame illustrates the free-tier collapse also seen in the camera legs (Section 6.2.2): FASTC predicts zero free-tier cells against a ground-truth free-tier population of 8,491, defaulting bare traversable ground to higher-cost tiers. On the obstacle tier specifically this frame performs well above the forest aggregate (TP=2,196, FP=2,805, FN=345; recall 0.864, IoU 0.411, versus the reported forest aggregate of 0.417 and 0.157 respectively), so it should be read as illustrative of the free-tier mechanism, not as representative of forest obstacle-detection performance. In the panels, free is green, low is yellow, medium is orange, obstacle is red, invalid is grey.

Fifth, the two backbones fail in characteristically different directions, and campus makes the contrast starkest. WVN's camera-only precision on campus remains the highest camera precision recorded anywhere in this comparison (0.4881), but it is achieved on very little coverage: 0.0136 obstacle recall, meaning WVN detects a small fraction of ground-truth obstacle cells within its already-narrow support. GA-Nav shows

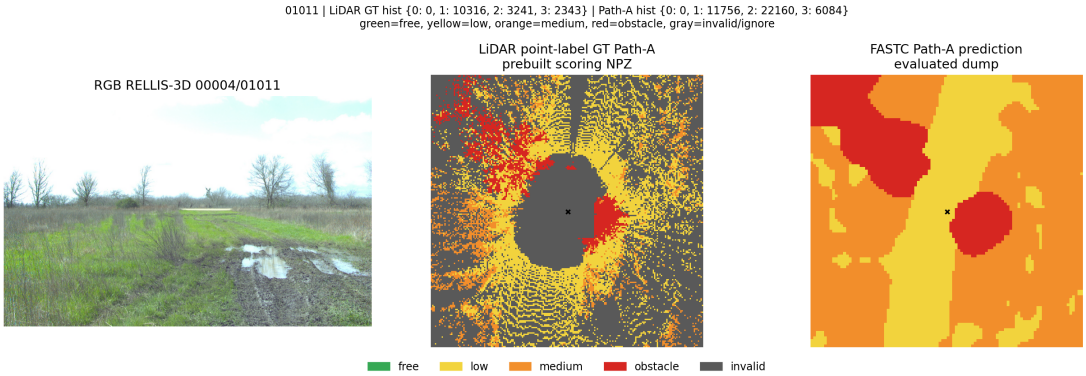


Figure 5.7: Qualitative FASTC prediction on RELLIS-3D sequence 00004, frame 01011: RGB, prebuilt Path-A ground truth, and the evaluated FASTC Path-A prediction. Restricted to GT-valid cells (15,900 total), the frame's obstacle detection closely matches the RELLIS-3D aggregate (TP=2,214, FP=1,463, FN=129; recall 0.945, matching the reported aggregate of 0.945), consistent with RELLIS-3D being near FASTC's training distribution. Compare with Figure 5.6: the same checkpoint, evaluated on off-distribution forest terrain, shows substantially weaker overall tier agreement despite comparable obstacle recall on that particular frame. In the panels, free is green, low is yellow, medium is orange, obstacle is red, invalid is grey.

the opposite trade: the point-painted campus camera-leg reaches 0.1385 recall at 0.1206 precision, broader reach but at a real precision cost. Unlike forest, this is not primarily a near-constant-positive artifact: pooled across all five campus sequences, GA-Nav's camera leg assigns only 49.1% of its valid support to obstacle, with a real spread across free, low, and medium as well, consistent with genuine selectivity rather than blanket obstacle-calling. That pooled figure conceals substantial sequence-level variation, however: the obstacle fraction ranges from 30.2% (sequence 00005) to 81.3% (sequence 00003), so campus's fusion gain likely reflects meaningfully different camera behavior across its five recording sessions rather than one consistent mechanism, and should not be read as uniform across the domain. WVN is conservative and selective; GA-Nav is permissive but domain-dependent in how permissive. Neither failure mode is free, and which one is preferable depends on whether the downstream use of these predictions penalises missed obstacles or false alarms more heavily, a question this thesis returns to in Chapter 6.

5.5 Computational cost and deployment performance

Beyond predictive quality, this thesis also evaluates whether the selected traversability methods are computationally feasible for deployment. A method that performs well offline may still be unsuitable for robotic use if its inference pipeline cannot run at sufficient rate on embedded hardware. The benchmarking follows four stages: first, the original PyTorch implementations are measured on a desktop-class GPU; second, the same method branches are measured on the Jetson AGX Thor without TensorRT; third, the deployable neural-network stages are exported to ONNX, compiled with TensorRT, and validated against the PyTorch outputs; and fourth, matched PyTorch–TensorRT pipelines are benchmarked to isolate the acceleration obtained by replacing only the neural-network stage.

Table 5.10: Benchmark groups used for deployment evaluation. The matched PyTorch–TensorRT benchmark is the final deployment-oriented runtime comparison.

Group	Platform	Backend	Purpose
Desktop/HPC	NVIDIA V100	PyTorch/CUDA	Original stage-wise computational-cost baseline
Thor baseline	Jetson AGX Thor	PyTorch/CUDA	Embedded feasibility without TensorRT
TensorRT parity	Jetson AGX Thor	ONNX → TensorRT FP16	Numerical validation against PyTorch outputs
Matched Thor runtime	Jetson AGX Thor	PyTorch vs. TensorRT	Final deployment-oriented speed-up comparison

All reported measurements use batch size one. Latencies are reported as medians with interquartile ranges where available, because GPU scheduling, thermal behaviour, and Python-level preprocessing can produce long upper tails. Unless explicitly stated otherwise, disk input is excluded from compute latency.

5.5.1 Benchmark protocols

The deployment benchmark is organised into desktop PyTorch measurements, Thor PyTorch measurements, TensorRT parity validation, and matched PyTorch–TensorRT runtime comparison. The desktop GPU benchmark characterises the original method implementations and their dominant stages. The Thor PyTorch benchmark verifies that the methods can run on the embedded target platform using their original runtime stacks. The TensorRT parity benchmark validates that the compiled engines preserve the corresponding PyTorch outputs. Finally, the matched PyTorch–TensorRT benchmark measures the deployment-relevant acceleration obtained when only the neural-network stage is replaced by TensorRT while preprocessing, post-processing, and output projection remain shared within each method.

The Thor benchmarks were executed on the selected 30-frame ROVO forest subset. FASTC and WVN were benchmarked using a shared deployment environment, while GA-Nav required a separate environment because of its legacy MMSegmentation/MMCV dependency stack. This separation is a practical deployment constraint, but it also preserves the working dependency stack of each method.

5.5.2 Desktop GPU stage-wise baseline

The first computational-cost benchmark was performed on a desktop-class NVIDIA V100 GPU. Each method was benchmarked in isolation over the same 30 forest frames at batch size one. Ten warm-up iterations were discarded and 100 timed iterations were recorded. Explicit GPU synchronisation was used around each timed region.

The desktop baseline highlights different bottlenecks for the three methods. FASTC is dominated by voxelisation and BEV conversion, GA-Nav is dominated by the MiT-b0 image forward pass, and WVN is dominated by dense DINO/STEGO feature extraction. The WVN feature extractor also shows the largest variability, with an interquartile range of 20.42 ms and an observed range from 66.95 ms to 129.09 ms.

Table 5.11: Stage-wise latency on the NVIDIA V100 baseline. Values are medians over 100 timed iterations after 10 discarded warm-ups. Geometry stages execute on the CPU regardless of device.

Component	Stage	Median [ms]	IQR [ms]	Min [ms]	Max [ms]
FASTC	Voxelisation	41.92	1.18	40.32	60.00
	Network forward	18.29	1.70	16.28	32.75
	Native $512^2 \rightarrow$ Path-A	8.11	0.25	7.69	10.73
GA-Nav	MiT-b0 forward	26.91	0.76	26.56	31.71
	Image plane $\rightarrow$ BEV	8.02	0.09	7.86	8.04
WVN	DINO/STEGO features	86.59	20.42	66.95	129.09
	MLP head	1.93	0.03	1.63	2.44
	Image plane $\rightarrow$ BEV	7.01	0.10	6.82	7.02
Shared geometry	Point-painting	17.75	3.01	13.97	28.13
	Fusion blend + tier snap	1.35	0.08	1.19	3.54
Optional	Obstacle override $\tau = 0.5$	0.90	0.03	0.86	1.10

Table 5.12: TensorRT parity validation over the 30 selected ROVO frames. FASTC is evaluated at the pre-voxelised LiDAR network boundary, WVN at the STEGO feature-extraction boundary after passing features through the unchanged WVN head, and GA-Nav at the image-logits boundary including downstream softmax and cost computations.

Method	Agreement min	Agreement mean	Main error metric	Maximum error
FASTC	99.0483%	99.5687%	Softmax MAE	0.000925
WVN	99.8047%	99.9270%	Expected-cost MAE	0.000753
GA-Nav	99.9840%	99.9975%	Expected-cost MAE	0.00011184

5.5.3 TensorRT parity validation

Before using the TensorRT engines for runtime comparison, each engine was validated against its corresponding PyTorch implementation on the 30 selected ROVO frames. This step ensures that acceleration does not materially alter the predicted traversability outputs.

All three TensorRT engines preserve their corresponding PyTorch outputs with high agreement. FASTC reaches a thesis-valid four-tier agreement above 99.0%, while WVN and GA-Nav preserve downstream expected-cost maps with maximum mean absolute errors below 10^{-3} and 10^{-4}, respectively. The remaining differences are consistent with FP16 TensorRT execution and do not affect the deployment conclusions.

FASTC produces five output classes, where class 4 denotes the established unknown/no-signal output. In the thesis post-processing path, this class is mapped to invalid cost and zero confidence rather than to a traversability tier. Therefore, the four-tier agreement is computed only over cells where the PyTorch output belongs to classes 0–3. Expected-cost parity is not reported for FASTC because the deployment path uses hard argmax class-to-cost mapping rather than probability-weighted expected cost.

Table 5.13: Matched full-pipeline PyTorch-versus-TensorRT latency on the Jetson AGX Thor. Entries are median/IQR in milliseconds. Disk input is excluded from total compute latency.

Method	Backend	Preprocess	Network	Postprocess	Output/BEV	Total	Rate
FASTC	PyTorch	5.703/1.124	16.559/2.538	4.442/0.147	0.007/0.000	26.682/3.533	37.48 Hz
FASTC	TensorRT	5.703/1.124	5.795/1.058	4.494/0.184	0.007/0.000	16.007/2.194	62.47 Hz
WVN	PyTorch	4.624/6.080	58.165/5.229	3.465/1.778	8.609/8.340	75.849/11.672	13.18 Hz
WVN	TensorRT	4.624/6.080	3.548/2.965	4.219/4.200	8.157/7.674	22.554/8.450	44.34 Hz
GA-Nav	PyTorch	11.747/6.704	46.461/6.904	2.733/3.516	8.846/9.250	71.975/11.980	13.89 Hz
GA-Nav	TensorRT	11.747/6.704	10.819/4.601	3.873/4.489	8.971/9.098	38.173/10.341	26.20 Hz

Table 5.14: Matched full-pipeline speed-up from PyTorch to TensorRT on the Jetson AGX Thor. Matched means both paths use the same input frames, preprocessing, and post-processing; only the neural-network stage is replaced. TensorRT accelerates this stage only, not the surrounding voxelisation, projection, and cost computation, which remain in the reported totals.

Method	PyTorch total	TensorRT total	Speed-up	PyTorch rate	TensorRT rate
FASTC	26.682 ms	16.007 ms	1.667×	37.48 Hz	62.47 Hz
WVN	75.849 ms	22.554 ms	3.363×	13.18 Hz	44.34 Hz
GA-Nav	71.975 ms	38.173 ms	1.885×	13.89 Hz	26.20 Hz

5.5.4 Matched PyTorch–TensorRT pipeline comparison

To isolate the effect of TensorRT acceleration, a matched benchmark was performed for each method. Within each method, the PyTorch and TensorRT paths use the same input frames, preprocessing, post-processing, and output projection. The intended backend difference is therefore the neural-network stage only. All measurements use batch size one on the 30 selected ROVO frames, with 10 warm-up passes and 50 timed passes over all frames.

The table shows that TensorRT reduces complete compute latency for all three methods, with FASTC remaining the fastest final pipeline and WVN obtaining the largest relative speed-up because the STEGO feature extractor is strongly accelerated; exact figures and speed-up factors are analysed below.

The network-stage speed-ups are larger than the full-pipeline speed-ups because TensorRT accelerates only the exported neural-network boundary. FASTC still requires LiDAR voxelisation outside TensorRT, WVN still executes the WVN head, confidence/cost computation, and BEV projection outside TensorRT, and GA-Nav still performs image preprocessing, softmax/cost/confidence computation, and BEV projection outside TensorRT. Therefore, the full-pipeline speed-up is the more deployment-relevant result, while the network-stage speed-up shows how effective TensorRT is on the exported neural graph itself.

The matched PyTorch–TensorRT benchmark confirms that TensorRT improves the full embedded compute pipeline for all three traversability branches. FASTC improves from 26.682 ms to 16.007 ms, WVN from 75.849 ms to 22.554 ms, and GA-Nav from 71.975 ms to 38.173 ms. The corresponding full-pipeline speed-ups are 1.667×, 3.363×, and 1.885×, respectively.

The network-stage gains are larger: 2.857× for FASTC, 16.392× for WVN, and 4.294× for GA-Nav.

Table 5.15: Network-stage speed-up of the exported TensorRT boundaries.

Method	PyTorch network	TensorRT network	Speed-up
FASTC	16.559 ms	5.795 ms	2.857×
WVN	58.165 ms	3.548 ms	16.392×
GA-Nav	46.461 ms	10.819 ms	4.294×

Table 5.16: Matched output-validation sanity checks between PyTorch and TensorRT paths. The checks verify that the matched benchmark uses equivalent preprocessing and post-processing for both backends.

Method	Agreement metric	Value	Error metric	Value
FASTC	Raw 5-class argmax agreement	99.818649%	Logits MAE	0.00637556
FASTC	Thesis-valid 4-tier agreement	99.567655%	Softmax MAE	0.00063178
WVN	Argmax agreement	99.926725%	Expected-cost MAE	0.00025852
WVN	–	–	Softmax MAE	0.00035674
GA-Nav	Argmax agreement	99.997576%	Expected-cost MAE	2.8196×10^{-5}
GA-Nav	–	–	Softmax MAE	3.0428×10^{-5}

This difference between network-stage and full-pipeline speed-up shows that TensorRT acceleration does not remove the cost of the surrounding robotics pipeline. After acceleration, voxelisation, image preprocessing, post-processing, and image-to-BEV projection remain important runtime contributors.

Under the matched TensorRT protocol, FASTC is the fastest complete pipeline at 16.007 ms, corresponding to 62.47 Hz. WVN is second at 22.554 ms, corresponding to 44.34 Hz, benefiting strongly from TensorRT acceleration of the STEGO feature extractor. GA-Nav remains slower at 38.173 ms, corresponding to 26.20 Hz, mainly because image preprocessing and BEV projection remain substantial non-engine stages.

5.5.5 Memory footprint and model size

Table 5.17 summarises the deployment artifacts and process-visible memory usage of the TensorRT pipelines. The memory values are reported per method in isolation. Torch CUDA accounting may not include all TensorRT-internal allocations, so these numbers should be interpreted as process-visible benchmarking diagnostics rather than exact hardware-level memory consumption.

For WVN, the reported parameter count refers only to the trained WVN head because the deployment benchmark loads the STEGO component through the TensorRT engine rather than as a PyTorch module. The STEGO model footprint is nevertheless reflected in the checkpoint, ONNX, and TensorRT engine sizes. The FASTC process-visible memory is higher because the benchmark process also loads the voxelisation and sparse-processing stack; it should not be interpreted as pure neural-network memory.

5.5.6 Corrected Condition C fusion runtime

The single-method runtime measurements do not directly represent the runtime of the fusion condition, because the fused pipeline does not use the camera-only flat-ground BEV projection. In the fusion

Table 5.17: TensorRT deployment artifacts, PyTorch parameter counts, and process-visible memory on the Jetson AGX Thor. Torch CUDA accounting may exclude TensorRT-internal allocations.

Method	Checkpoint	ONNX	Engine	Parameters total/trainable	GPU base/peak/reserved	Host RAM Δ
FASTC	27.34 MB	147.87 MB	76.82 MB	3,371,321 / 3,371,321	571.65 / 590.43 / 610.27 MB	887.28 MB
WVN	346.24 + 0.14 MB	346.23 MB	175.81 MB	head: 34,622 / 34,622	28.58 / 152.11 / 171.97 MB	201.53 MB
GA-Nav	49.90 MB	24.81 MB	16.92 MB	6,215,468 / 6,215,468	24.92 / 71.76 / 77.59 MB	427.98 MB

Table 5.18: Point-painting latency on the Jetson AGX Thor. The benchmark uses real ROVO LiDAR scans and real calibration with synthetic normalised image-plane probabilities for timing.

Quantity	Value
Frames	30
Probability tensor shape	$4 \times 608 \times 968$
LiDAR point count range	128,132–159,943
Mean LiDAR point count	146,845
Warm-up calls	10
Timed calls	100×3 repeats
Repeat medians	13.732, 14.686, 15.469 ms
Median-of-medians	14.686 ms
All-call IQR	6.306 ms
All-call min/max	9.924/36.978 ms

condition, camera image-plane probabilities are transferred into the LiDAR-supported BEV grid through point-painting before applying the fusion rule. Therefore, the camera branch cost used for fusion must exclude the camera-only BEV projection stage and must instead include the LiDAR-assisted point-painting stage. The corrected estimate is compared against the earlier, incorrect one in Table 5.21.

To correct the fusion runtime estimate, the point-painting function `paint_lidar_to_bev()` was benchmarked separately on the Jetson AGX Thor. The benchmark used real selected ROVO LiDAR scans and the real camera–LiDAR calibration, together with synthetic normalised image-plane probability and confidence maps of shape $4 \times 608 \times 968$. The probabilities are synthetic only because this benchmark measures runtime rather than prediction accuracy. The computational structure of point-painting depends on the LiDAR points, calibration, image resolution, and BEV grid geometry, not on the semantic correctness of the probabilities.

Using the measured point-painting latency, the corrected Thor fusion runtime for GA-Nav+FASTC is computed as a composition of the LiDAR branch, the camera image-plane branch, point-painting, and the confidence-weighted fusion step. The camera-only BEV projection is not included.

The sequential estimate assumes that the FASTC and GA-Nav branches execute one after another. The theoretical parallel estimate assumes that the LiDAR and camera branches execute concurrently and that the fusion step waits for the slower branch. The two estimates are therefore

$$t_{\text{seq}} = t_{\text{FASTC}} + t_{\text{GA-Nav,img}} + t_{\text{paint}} + t_{\text{fusion}},$$

and

Table 5.19: Corrected Thor Condition C fusion decomposition for GA-Nav+FASTC. The camera flat-ground BEV projection is excluded and replaced by LiDAR-assisted point-painting.

Component	Latency
FASTC full pipeline	28.854 ms
GA-Nav forward only, excluding BEV projection	61.765 ms
LiDAR-assisted point-painting	14.686 ms
Confidence-weighted fusion step	3.016 ms
Sequential total	108.321 ms
Theoretical parallel total	79.467 ms

Table 5.20: Corrected Thor Condition C fusion latency for GA-Nav+FASTC. Corrected replaces an earlier estimate that incorrectly included the camera-only flat-ground projection term with the LiDAR-assisted point-painting term actually used in Condition C. These are computed pipeline-component estimates, not directly measured end-to-end latency; sequential executes the two branches one after another, while theoretical parallel assumes concurrent execution bounded by the slower branch.

Execution assumption	Latency	Rate
Sequential branches	108.321 ms	9.23 Hz
Theoretical parallel branches	79.467 ms	12.58 Hz

$$t_{\text{par}} = \max(t_{\text{FASTC}}, t_{\text{GA-Nav,img}}) + t_{\text{paint}} + t_{\text{fusion}}.$$

With the measured values, this gives

$$t_{\text{seq}} = 28.854 + 61.765 + 14.686 + 3.016 = 108.321\,\text{ms},$$

and

$$t_{\text{par}} = \max(28.854, 61.765) + 14.686 + 3.016 = 79.467\,\text{ms}.$$

Compared with the previously documented estimate (Table 5.21), the corrected fusion runtime is higher because point-painting is more expensive than the camera-only projection term that was incorrectly included before.

A corresponding V100 point-painting measurement was not available. Therefore, the corrected V100 fused latency is not reported as a final result. Any value obtained by substituting the Thor point-painting latency into the V100 formula would only be illustrative and is not used as a benchmark result.

5.5.7 Deployment summary

The deployment experiments show that all three selected traversability branches can run on the Jetson AGX Thor after method-specific engineering. The matched PyTorch–TensorRT benchmark confirms that TensorRT improves the full embedded compute pipeline for all three branches while preserving output

Table 5.21: Effect of correcting the Condition C fusion latency model on Thor.

Execution assumption	Previous total	Corrected total	Difference
Sequential branches	101.52 ms	108.321 ms	+6.80 ms
Theoretical parallel branches	72.69 ms	79.467 ms	+6.78 ms

consistency with the PyTorch implementations; detailed stage-level worker diagnostics are provided in Appendix B.

As established above (Section 5.5.4), FASTC remains the fastest complete single-method pipeline, WVN second, and GA-Nav third, with network-stage speed-ups consistently exceeding full-pipeline speed-ups across all three methods. After neural acceleration, voxelisation, image preprocessing, confidence/cost computation, geometric projection, and BEV fusion remain important optimisation targets.

For the fusion condition, the corrected runtime model shows that LiDAR-assisted point-painting is a non-negligible stage, contributing materially to the sequential and parallel latency estimates reported earlier in this section, which should be interpreted as compute-pipeline estimates, not measured ROS-level end-to-end latency.

The final runtime ranking should not be confused with predictive quality. GA-Nav in particular showed considerable degradation under forest domain-shift in qualitative smoke tests, where the image-plane prediction collapsed toward obstacle on selected forest data. Its runtime result is therefore valid as a deployment measurement, but not evidence of good forest prediction quality.

6

Coverage, competence, and the conditions for camera contribution in fusion

Chapter 5 reported that fusion yields only modest aggregate gains over the LiDAR-only baseline on the first two evaluation domains. Taken alone, this result is easy to misread as evidence that the camera leg makes little contribution. This chapter shows that the modest aggregate is consistent with identifiable and measurable causes, rather than indicating a failure of the camera methods or the fusion design, and that on the campus domain, where those causes align differently, the same frozen components and the same simple fusion rules produce the strongest camera contribution observed in this study, including resolved gains in obstacle recall and obstacle IoU over the LiDAR baseline that hold at both frame level and by leave-one-sequence-out analysis.

The primary fusion-analysis scope rests on three domains spanning a continuum of LiDAR-baseline strength in obstacle detection: RELLIS-3D, near the geometric method's training distribution (LiDAR-only obstacle recall 0.945); the custom forest dataset, off-distribution vegetated terrain (0.417); and the custom campus dataset introduced in Chapter 5, a mixed built/unstructured environment on which the geometric method's obstacle detection largely fails (0.107). Overall map quality (mIoU) is a separate observation and does not share this ordering: RELLIS-3D leads at 0.492, but forest (0.157) sits below campus (0.174) even though forest's obstacle recall is higher. This divergence is itself informative about what makes each domain hard for the geometric baseline: forest's difficulty is map quality (weak segmentation across classes despite comparatively intact obstacle detection), while campus's is obstacle detection specifically (mIoU recovers relative to forest while obstacle recall collapses further). These three primary fusion-analysis domains (excluding GOOSE, which appears in the four-domain cross-dataset and camera-leg summaries) are scored by the identical committed pipeline, with the identical checkpoints, on the identical $\pm 10\,\mathrm{m}$ grid (Figure 6.1). The chapter proceeds from measuring the camera's structural constraint (Section 6.1), through separating coverage from detection competence and characterising the camera as a high-sensitivity, low-specificity detector confined to a mid-range annulus (Section 6.2.1), through quantifying what is achievable within its coverage (Sections 6.2 and 6.3), to identifying the fusion rules that capture the achievable gain (Section 6.4), examining where confidence-driven rules trade against fixed blends rather than dominating them (Section 6.5), and finally pricing the resulting safety trade (Section 6.6).

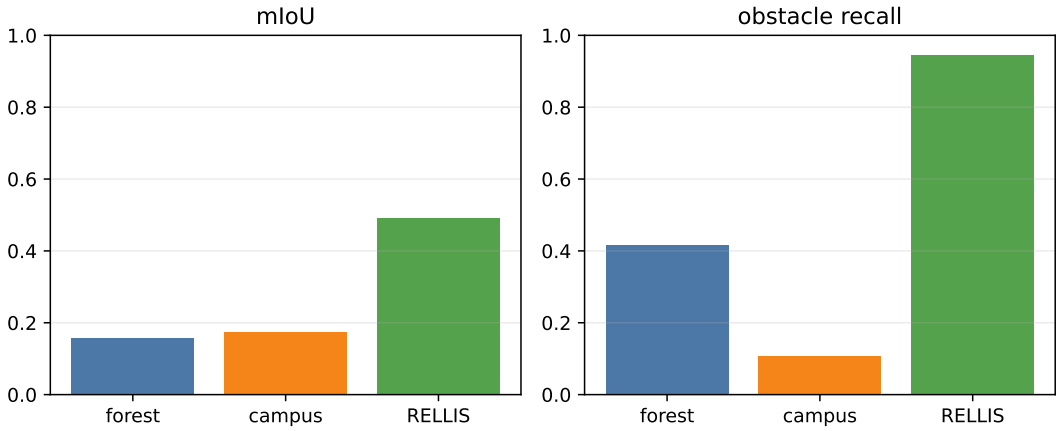


Figure 6.1: The LiDAR-only baseline (Condition B, FASTC) across the three evaluation domains (RELLIS, FKIE forest, FKIE campus), the baseline-strength continuum in obstacle detection discussed above. Map quality (mIoU) does not follow the same order: FKIE forest (0.157) is lower than FKIE campus (0.174), so this continuum should be read as an ordering on obstacle detection, not on overall segmentation quality.

6.1 The coverage wall

Any comparison of camera-based and LiDAR-based traversability estimation on a shared bird's-eye-view (BEV) grid must begin with coverage, because it dominates every downstream number. In this chapter the relevant quantity is the coverage of the point-painted camera leg within the fusion pipeline. Within the ±10 m evaluation crop, those legs reach only a small fraction of ground-truth-valid cells: 20.9 % (GA-Nav) and 15.0 % (WVN) on forest, 19.0 % and 12.2 % on campus, 7.6 % and 4.7 % on RELLIS-3D (Figure 6.2). The cause is primarily geometric rather than algorithmic, and its severity in the near field depends on the mounting geometry of the particular rig. On the GOOSE platform, the relatively high-mounted, upward-facing camera provides negligible point-painted support in the 0–3 m near field, with most of its support occurring beyond 5 m. The ROVO platform used for the forest and campus datasets is less extreme: within 0–3 m its point-painted legs still cover 6.4 % (forest) and 5.46 % (campus) of ground-truth-valid cells, with high obstacle recall inside that coverage (Section 7.3). The near-field cutoff is therefore a property of the point-painted fusion leg on a given rig, while coverage well below 25% is common to the measured fusion legs. The campus domain confirms the wall is geometric in origin while revealing it to be mildly scene-modulated in magnitude: campus coverage is the same order as forest's on the identical rig (pooled ratio 0.65–0.70), with the open-field sequences lowest and the built sequences highest, because open terrain returns fewer LiDAR points within the paintable near-field band. No scene lifts the camera coverage above 25%.

The LiDAR method's coverage also deserves a correction to the common intuition. On RELLIS-3D it is exactly 100 %, but this is a property of that dataset, not of the method: on forest it covers 94.4 % of GT-valid cells and on campus 96.4 %, with the deficit concentrated in the evaluation square's corners beyond the sensor's useful radius. The operative contrast is therefore 94–100 % (LiDAR) against 5–11 %

(camera): two orders of structure, unchanged in force. Any aggregate metric computed without this context invites the misreading raised at the start of this chapter: that the fusion camera leg is simply inaccurate. Its point-painted coverage is small, and coverage, not projection quality, is the wall for the fusion contribution analysed here. The standalone camera-only baseline is even weaker: the UniDepth projection comparison in Chapter 5 found that a genuinely camera-only construction (monocular-depth lifting, UniDepth-v2, no LiDAR return required) covers 6.0 % (GA-Nav) and 3.7 % (WVN) of GT-valid cells, inside the same single-digit band, and in fact below point-painting's LiDAR-gated coverage on the identical sequence. Removing the LiDAR dependency reduces coverage rather than increasing it. These results reinforce that camera-only BEV projection is not viable for the evaluated task and motivate the point-painting choice for the fusion condition; they do not turn the fusion-leg coverage measurements into a standalone camera competence claim.

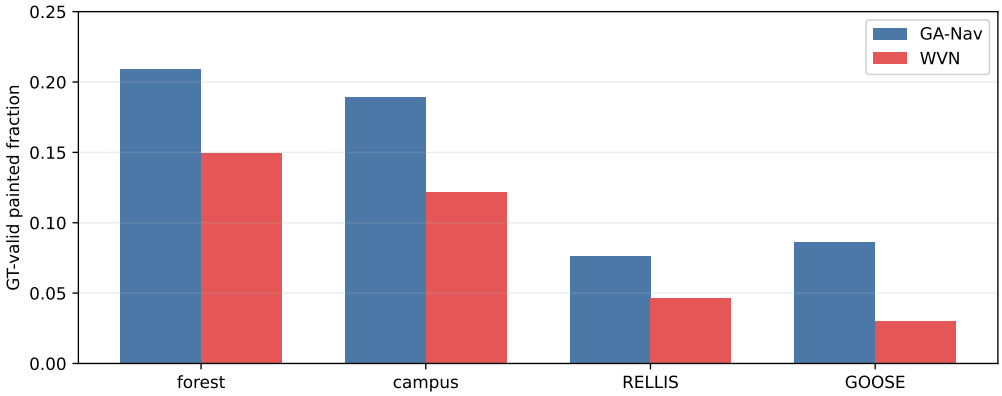


Figure 6.2: Point-painted fusion camera-leg coverage (fraction of GT-valid cells with a painted prediction) by domain and backbone. Coverage stays well below 25% across all domains, geometric in origin, mildly scene-modulated in magnitude.

6.2 Contribution within coverage

Within its coverage, the point-painted fusion camera leg contributes measurably, and the projection mechanism matters. Camera predictions are transferred to the BEV grid by point-painting (attaching each prediction to the LiDAR returns that fall within the camera frustum) rather than by projecting the image onto an assumed ground plane. The distinction is material in unstructured terrain. As established in Section 4.1.3, a flat-ground projection places every prediction at the range implied by a planar world, which mislocates vertical structure and produces spurious support wherever the terrain departs from the plane, whereas point-painting grounds each prediction in a measured return and excludes sky by construction, since the absence of a LiDAR return implies the absence of a point to paint. The cost is coverage: point-painting can only label cells the LiDAR already observes, which is the mechanism behind the coverage figures of Figure 6.2. On RELLIS-3D the point-painted camera leg reaches a full-support obstacle IoU of 0.0083, while covering 7.6 % of GT-valid cells. On forest, where the LiDAR baseline is

weak, the contribution becomes significant. Confidence-weighted fusion with GA-Nav raises obstacle recall by +0.054 (95 % CI [+0.037, +0.074], jackknife and paired bootstrap, no single frame driving the effect).

Table 6.1 summarises the same point-painted fusion camera-leg contribution across all four evaluation domains and both backbones. The paired recall columns make the baseline-conditional pattern explicit: the camera leg has a small spatial coverage everywhere, while its measurable fusion contribution appears primarily where the LiDAR baseline is weak.

Table 6.1: Point-painted fusion camera-leg coverage and contribution across the four evaluation domains. Coverage is the fraction of GT-valid cells reached by the point-painted camera leg; camera-leg obstacle IoU is scored on full support, consistent with B and C. B and C denote LiDAR-only and fusion obstacle recall on full support, respectively. The table shows that point-painted coverage remains well below 25% across domains and backbones, while fusion contribution (Δ recall) is negligible or negative except where the LiDAR baseline is weak: forest GA-Nav and both campus backbones. Campus C values use fixed $\alpha = 0.7$, the domain's best rule; RELLIS-3D, forest, and GOOSE use confidence-weighted fusion as the common rule.

Domain	Backbone	Coverage	Camera-leg Obs IoU	B recall	C recall	Δ recall
RELLIS 00004	GA-Nav	7.61%	0.0083	0.9449	0.9443	−0.0005
RELLIS 00004	WVN	4.68%	0.0022	0.9449	0.9448	−0.0001
FKIE forest	GA-Nav	20.92%	0.0823	0.4171	0.4835	+0.0664
FKIE forest	WVN	14.95%	0.0442	0.4171	0.4165	−0.0006
FKIE campus	GA-Nav	18.95%	0.0689	0.1070	0.1693	+0.0622
FKIE campus	WVN	12.22%	0.0468	0.1070	0.1237	+0.0167
GOOSE val	GA-Nav	8.63%	0.0230	0.8707	0.8713	+0.0006
GOOSE val	WVN	3.03%	0.0052	0.8707	0.8710	+0.0003

6.2.1 Coverage versus competence

The full-support numbers above conflate two distinct properties: how much of the scene the point-painted fusion camera leg reaches, and how well it detects obstacles within that support. Re-scoring on camera-valid support (the same confusion matrices with the no-prediction column dropped, so only cells the camera actually populates are counted) separates them, and the separation is stark for GA-Nav on every domain: obstacle recall rises from 0.008 to 0.639 on RELLIS-3D, 0.119 to 1.000 on forest, 0.091 to 0.933 on campus, and 0.025 to 0.503 on GOOSE. WVN shows the same direction, smaller in magnitude on the first two domains and larger on GOOSE: 0.002 to 0.444, 0.043 to 0.883, 0.030 to 0.736, and 0.005 to 0.715. Both backbones achieve high recall on their selected fusion support, while the limiting factor is the size of that support. These recall figures describe the fusion camera leg, not the camera modality in general; the standalone UniDepth camera-only results in Chapter 5 are weak across all four evaluation domains.

The two frame-level examples in Figures 6.3 and 6.4 make this support-conditioned result explicit. They show that the same point-painted camera leg can be dominated by false positives on open ground in one frame, yet recover most obstacle cells near equipment in another. Frame 000008 is an open grass/dirt

track with zero ground-truth obstacle cells: LiDAR-valid support contains 22,040 cells, camera point-painted-valid support 2,134, with 1,892 overlapping (1,890 GT-valid), and all 875 predicted-obstacle cells are false positives (TP=0, FP=875, FN=0), the cleanest frame-level illustration of the free-tier mechanism described in Section 6.2.2. Frame 000011 is a mixed case: LiDAR-valid support contains 38,904 cells and camera point-painted-valid support 1,794, all overlapping and GT-valid; of 1,109 predicted-obstacle cells, TP=456, FP=653, and FN=106, with correct detections concentrated near the actual equipment and false positives extending over adjacent open grass.

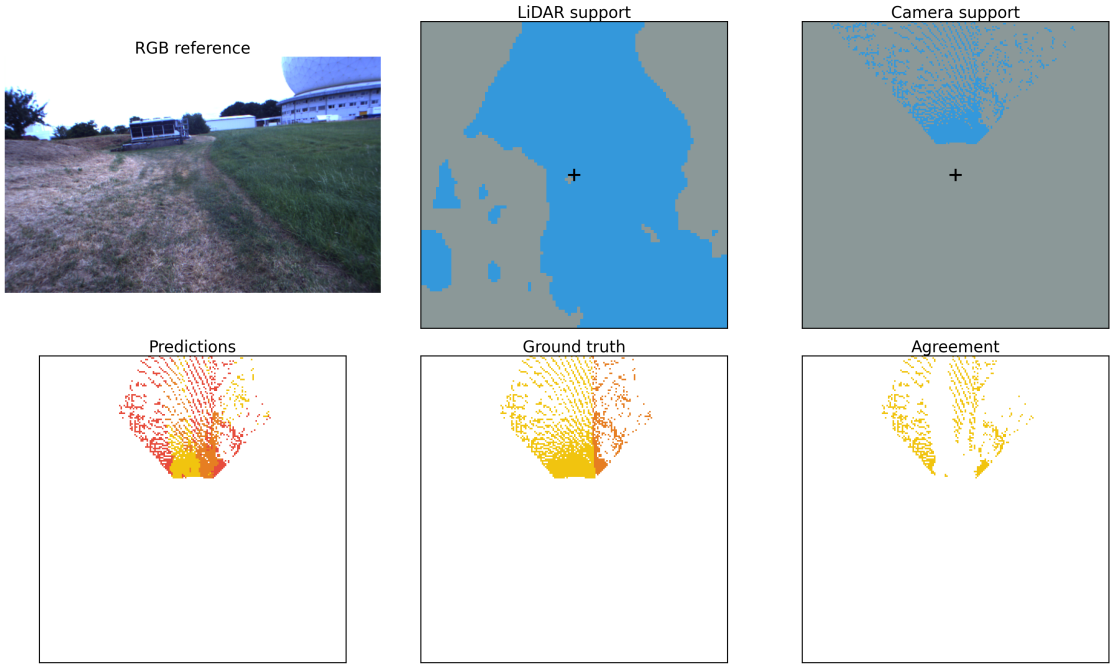


Figure 6.3: Spatial validity and point-painted camera predictions for campus sequence 00002, frame 000008. In the panels, free is green, low is yellow, medium is orange, obstacle is red, invalid is grey, and valid-support-with-no-tier is blue. The Agreement panel shows cells where the camera's predicted tier matches ground truth, coloured by the matching tier; cells where the prediction and ground truth disagree are left uncoloured.

Within the fusion camera leg, these two frames illustrate a high-sensitivity, low-specificity detector. GA-Nav's point-painted camera-valid recall of 0.87–0.98 on every campus sequence (below) is paired with precision of only 0.126–0.248: it rarely misses an obstacle it can see, but it also raises many false positives. This is not a coverage artifact: precision is identical under both the full-support and camera-valid conventions, since a cell the camera never predicts on can never contribute a false positive either way (campus GA-Nav precision is 0.1891 under both). This low precision is also the direct mechanism behind the fusion precision–recall trade characterised in Section 6.4 and Section 6.6.

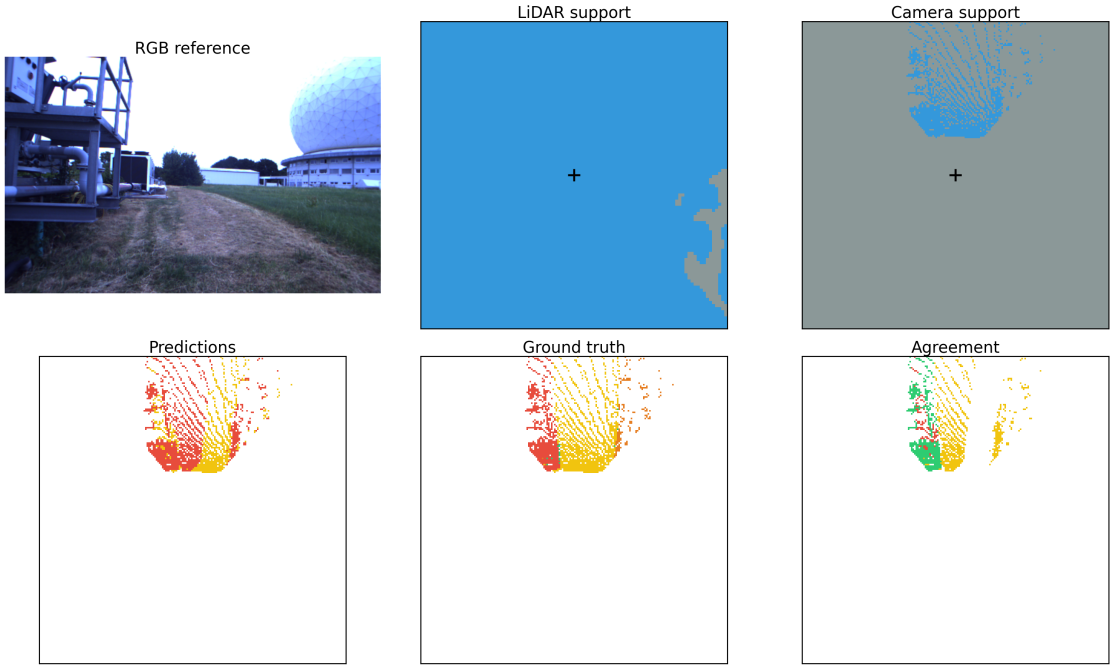


Figure 6.4: Spatial validity and point-painted camera predictions for campus sequence 00002, frame 000011. In the panels, free is green, low is yellow, medium is orange, obstacle is red, invalid is grey, and valid-support-with-no-tier is blue. The Agreement panel shows cells where the camera's predicted tier matches ground truth, coloured by the matching tier; cells where the prediction and ground truth disagree are left uncoloured.

The sparseness visible in these point-painted predictions reflects both camera field-of-view limits and LiDAR return density in the paintable near-field band (Section 6.1); this work did not explore gap-filling techniques such as virtual-point densification to recover additional coverage. Any rule that propagates camera evidence into the fused output inherits both its sensitivity and its false positives, which is why every fusion rule that improves recall on campus does so at a measurable precision cost, and why ordinal max-pooling and confidence-gated max-pooling (the rules that trust the camera most directly) pay for it most.

One caveat is required here for accuracy: camera-valid support is self-selected, not a fixed evaluation set. On campus, per-sequence coverage ranges 17.90–19.45 % for GA-Nav and 11.74–13.20 % for WVN, and because the two backbones paint different pixels, these are not the same cells; the two backbones' camera-valid figures are not scored on identical ground, and neither figure should be read as an estimate of standalone system performance. A point-painted detector with 0.933 recall over 19 % of the map is not a standalone traversability system; full support remains the deployment-honest number precisely because it is scored on the cells the robot actually needs an answer for. The gap between the two conventions is itself the finding: it is the coverage wall expressed in performance terms.

Campus escalates this from contribution to compensation. Within its painted coverage, GA-Nav achieves obstacle recall 0.904 on the same cells where the LiDAR baseline achieves 0.107, and even

on full support, carrying its ∼81 % no-coverage handicap, the point-painted fusion camera leg's recall (0.091) exceeds the LiDAR baseline's, the first such case in this study. The two camera backbones fail in opposite directions on campus obstacles: GA-Nav detects nearly everything at low precision (recall 0.933, precision 0.189 in-domain), while WVN detects less but more selectively (recall 0.736, precision 0.607). The compensation result must be read with its cause: the LiDAR model, trained on off-road terrain, misclassifies 84 % of campus ground-truth obstacle cells as low- or medium-cost, a severity-compression failure under domain shift that mirrors, on the geometric side, the vocabulary-limited severity compression documented qualitatively for the camera (vehicles mapped to *medium*). Neither modality is the transfer-robust one; they are differently fragile, which is precisely the circumstance under which fusion has something to offer.

The fusion camera-leg contribution just established is consistent across the five evaluated campus sequences, which matters for how the campus result should be read. Per-sequence camera-valid obstacle recall against FASTC on the same sequence is: 00001 (parking lot) 0.901 GA-Nav / 0.746 WVN vs. 0.035; 00002 (field/structures) 0.954 / 0.574 vs. 0.157; 00003 (urban street) 0.982 / 0.686 vs. 0.061; 00004 (paved path/park) 0.913 / 0.738 vs. 0.132; 00005 (field path) 0.873 / 0.902 vs. 0.029. GA-Nav's camera-valid recall holds in the high 0.87–0.98 band on both the most structured sequence (00003, urban street) and the most unstructured one (00005, field path), while FASTC's recall varies by more than 5× across the same sequences (0.029–0.157). The campus advantage is therefore not a built-environment effect: it does not depend on scene structure, which means the fusion headroom established in Chapter 5 and exploited in Section 6.4 is available uniformly across campus's structured and unstructured sequences alike, not concentrated in the built ones.

6.2.2 The effect of class-imbalance on GA-Nav's precision

The preceding section establishes that the point-painted fusion camera leg's low precision is real rather than a coverage artifact; this section identifies a candidate mechanism. GA-Nav's four-tier retraining set is heavily imbalanced at the class supervised least often: the *free* tier received only 2.66 % of supervised pixels (208.5M of ∼7.85G), against 51.2 % for *low*. The trained model's predictions are consistent with this imbalance: across camera-valid cells, GA-Nav predicts *free* at 0.0000 % on RELLIS-3D, 0.0000 % on forest, 0.63 % on campus, and 0.55 % on GOOSE. A model that was almost never shown the *free* class during training rarely predicts it at inference, though this association has not been isolated by a controlled ablation (see limitations below). This is an upstream classifier property rather than a point-painting effect, so the same mechanism applies in principle to the standalone UniDepth camera-only path as well, although its much smaller camera-only support makes the consequence correspondingly limited in the reported evaluation.

Figures 6.3 and 6.4 show this mechanism at the frame level; the same pattern is measurable across the full campus dataset.

Across all 150 campus frames, GA-Nav's image-plane obstacle fraction correlates with the frame's ground-truth free fraction at $r = 0.649$, rising monotonically by quartile: mean GT-free of 0.10 %, 12.66 %, 33.41 %, and 58.60 % corresponds to 35.5 %, 49.0 %, 69.3 %, and 82.5 % obstacle prediction respectively:

frames with more true free space produce *more* obstacle predictions from GA-Nav, not fewer. Restricted to camera-valid cells that are ground-truth free, the pattern holds per sequence: GA-Nav predicts obstacle on 49.9 %, 67.2 %, 77.6 %, 86.6 %, and 28.4 % of such cells on sequences 00001–00005 respectively. This is not a paved-surface artifact: on forest, whose traversable ground is dirt rather than pavement, GA-Nav assigns 97.9 % of camera-valid GT-free cells to obstacle, worse than any campus sequence. The failure tracks the *free* tier itself, not a particular surface type.

WVN, trained on a different objective and without the same class imbalance, serves as a control. Over the same campus frames its image-plane obstacle fraction stays flat at 40–51 % with $r = 0.315$, a far weaker relationship, and it emits *free* at 0.9 % on campus and 3.7 % on GOOSE against GA-Nav's 0.63 % and 0.55 %. This mechanism is therefore specific to the RELLIS-retrained GA-Nav head rather than a shared camera limitation, which is also consistent with, and offers a candidate explanation for, the two backbones' precision gap on campus (0.189 for GA-Nav against 0.607 for WVN, Section 6.2.1).

This mechanism connects directly to the fusion results in Section 6.4: it is a candidate explanation for both GA-Nav's low obstacle precision and the precision cost paid by every fusion rule that propagates camera evidence into the fused output. The campus gate-τ=0.7 arm's precision cost relative to the best fixed blend (-0.163) and the ordinal-max-pooling arm's precision cost relative to fixed α=0.9 (-0.0263, both Section 6.4) are consistent with rules that trust the camera most directly importing GA-Nav's systematic free-classified-as-obstacle errors onto traversable ground, alongside its genuine obstacle detections.

Three limits on this mechanistic explanation should be stated plainly. First, this is an observational association between training-set class balance, inference-time class frequency, and prediction error, not a controlled ablation; no retrained, rebalanced GA-Nav head was evaluated to isolate the effect. Second, the class-imbalance explanation is consistent with every pattern reported above (the RELLIS/forest zero, the campus quartile trend, the forest surface-independence check, and the WVN control), but it is not established as the sole cause; other factors correlated with training-set composition could contribute. Third, the campus image-plane obstacle-fraction figures in this section are drawn from cached per-frame predictions rather than the committed scoring pipeline used elsewhere in this thesis, and should be read with that provenance in mind.

6.2.3 Range-dependent usability of the point-painted fusion leg

Coverage of the point-painted fusion camera leg is not concentrated near the robot; a forward-facing camera looks out rather than down, so the near field falls below its field of view while a fixed-height LiDAR continues to return points underneath it. The physically measured platform placements in Figure 3.1 provide the geometric basis for this effect: the camera mounting heights are 0.83 m (RELLIS-3D), 1.59 m (GOOSE), and approximately 0.45 m (FKIE/ROVO). On pooled campus GA-Nav, coverage by range band is 5.46 % at 0–3 m, rising to 16.52 % at 3–5 m, and falling back to 7.45 % at 5–10 m. These fractions use the full 360° $\pm$10 m grid as the denominator for every band, so they understate coverage within the forward camera field of view; no forward-restricted fraction was computed. Because that denominator is applied equally across the bands, the relative shape is the meaningful result: coverage is lowest at 0–3 m,

peaks at 3–5 m, and falls beyond that range. RELLIS-3D and GOOSE are near-blind within 3 m (0.03 % coverage), consistent with the rig-geometry analysis of Section 6.1. Point-painted fusion-leg coverage is therefore neither uniform with range nor concentrated where a ground robot's most urgent obstacles typically lie; it peaks in a mid-range band.

Within that point-painted fusion support, the camera leg's obstacle prediction quality does not degrade with range and precision actually improves with it: campus GA-Nav within-band recall is 0.907, 0.916, and 0.922 for the 0–3, 3–5, and 5–10 m bands respectively, while precision rises from 0.018 to 0.072 to 0.215 over the same bands. The false positives responsible for the camera's low aggregate precision therefore concentrate near the robot, consistent with the near-field failure-mode analysis in Section 7.3.1. Taken together, the point-painted fusion camera leg's reliably usable region is a mid-range annulus of roughly 3–10 m: it is neither near-blind nor uniformly capable, but complementary to the LiDAR by range as well as by modality. This characterises where the camera leg itself is confident and accurate, not where fusion's overall benefit over LiDAR is largest; Section 7.3.1 shows those two questions have different answers, because fusion's marginal value tracks where the LiDAR baseline weakens, not where the camera happens to reach. Practically, this means the fusion camera leg can supply early warning of obstacles the LiDAR will only later resolve at closer range, but it cannot serve immediate obstacle avoidance in the 0–3 m band where a deploying system most needs a low-latency, low-false-positive signal; that band remains the LiDAR's alone. This early-warning value holds regardless of the camera leg's particular accuracy. Correct classification of far-range cells, which LiDAR alone cannot resolve due to its lack of geometric context at range, can produce disproportionate behavioral gains, for example enabling a different path choice when low traversability is flagged ahead, even where the raw number of correctly classified cells is small.

6.2.4 Robustness of the fusion gain to evaluation-frame selection

The forest GA-Nav gain reported above is a single paired estimate, which raises the question of how much of it depends on which frames happened to be in the evaluation set. Because the pooled metric is a sum of per-frame confusion matrices, this is arithmetic, not re-inference: a per-frame jackknife (leave-one-out) plus paired bootstrap over frozen predictions, run on every domain×backbone combination rather than forest-GA-Nav alone (Table 6.2). Two distinctions matter for reading it. First, this is evaluation-set stability, not training cross-validation; the checkpoints were never trained on these frames. Second, because both the fusion arm and the LiDAR-only baseline are scored on the same frames, the correct test is the *paired* difference: the frame-to-frame variance common to both arms cancels, so overlapping marginal confidence intervals do not by themselves imply no gain.

Only forest-GA-Nav clears both bars. Every other paired gain is negligible in magnitude ($|\Delta| \leq 0.0005$) even on the rows where the confidence interval technically excludes zero: a consistently tiny difference can register as statistically significant purely because its variance is minuscule, which is exactly why significance and effect size have to be checked separately rather than reading "CI excludes zero" as "this matters." The forest-GA-Nav gain itself survives this scrutiny on both counts: no single held-out frame carries it (maximum per-frame leverage 0.007, two orders of magnitude below the +0.054 gain), which is what justifies treating it, rather than any of the other five rows, as a reportable finding elsewhere in this

Table 6.2: Paired leave-one-out / bootstrap stability of the fusion gain (Condition C − Condition B, obstacle recall) across every evaluated domain and camera backbone. "Meaningful" requires both a 95 % CI that excludes zero *and* an effect size that is not negligible.

Dataset	Backbone	Δ obstacle recall (C−B)	Paired 95 % CI	Meaningful
Forest	GA-Nav	+0.0539	[+0.036, +0.074]	**yes**
Forest	WVN	−0.0002	[−0.0004, −0.0001]	no (negligible)
RELLIS	GA-Nav	−0.0005	[−0.0007, −0.0004]	no
RELLIS	WVN	−0.0001	[−0.0001, −0.00004]	no
GOOSE	GA-Nav	+0.0006	[−0.0004, +0.0015]	no
GOOSE	WVN	+0.0003	[+0.0001, +0.0006]	no

chapter.

6.3 An oracle ceiling on camera contribution

An oracle analysis quantifies the maximum camera contribution available to *any* fusion rule, by granting the camera every cell where it is correct and the LiDAR is wrong. On forest, GA-Nav's ceiling is +6.80 pp of obstacle recall, of which the realized confidence-weighted gain captures 79.2 %. On RELLIS-3D and GOOSE, the ceilings collapse to ≤ 0.6 pp: where the LiDAR baseline is already strong, there is almost nothing for any camera, projection, or rule to add. The realized fraction is backbone-specific (WVN's forest ceiling of +2.79 pp is essentially uncaptured by confidence weighting), which motivates the rule study of Section 6.4. Campus then supplies the continuum's far end: with the baseline near failure, headroom is the largest of any domain, and Section 6.4 shows a substantial fraction of it is realized. The ceiling result reframes the fusion literature's implicit promise in both directions: fusion architecture cannot manufacture headroom that the modalities' error structure does not provide; where the error structure provides it, simple rules suffice to collect it.

6.4 Fusion rule selection by backbone

A sweep of 17 fusion rules per camera backbone (fixed blends $T = \alpha\, t_{\text{cam}} + (1 - \alpha)\, t_{\text{lidar}}$ with $\alpha \in \{0.1, \ldots, 0.9\}$, confidence weighting, confidence sharpening ($\gamma \in \{1, 2, 4\}$), ordinal max-pooling, and confidence-gated max-pooling) was scored on the committed evaluation pipeline for forest and RELLIS-3D. Campus was initially evaluated on a claim-bearing subset of this space and subsequently extended to the full 17-arm sweep under matched semantics, both backbones, pooled over all 150 frames. Every weighted rule, including the fixed blends, is normalised by the sum of valid weights and therefore only acts as a tie-break in cells where *both* the camera and LiDAR legs are valid: where only the LiDAR leg is valid the fused cost equals the LiDAR cost exactly, and camera-only cells pass through unchanged; α is never a global camera/LiDAR mixing ratio over the whole grid. The conclusion has three parts, one per regime of baseline strength.

Where the baseline is strong (RELLIS-3D), no rule in the sweep moves any metric by more than 0.25 pp in either direction. This exhaustive null is the empirical counterpart of the ceiling: fusion architecture is irrelevant where fusion headroom is absent.

Where the baseline is weak (forest), the best rule differs by camera backbone, but more simply than intermediate results suggested. For GA-Nav, confidence weighting reaches the highest obstacle recall of any arm (0.4190). A direct paired comparison against the fixed blends resolves this into a trade rather than a win: confidence weighting versus fixed $\alpha = 0.6$ is statistically indistinguishable on recall, obstacle IoU, and mIoU (all bootstrap CIs cross zero), and the same holds against fixed $\alpha = 0.7$; the one resolved difference in both comparisons is precision, which is measurably lower under confidence weighting (-0.0061, CI $[-0.0117, -0.0017]$ vs. $\alpha = 0.6$; -0.0049, $[-0.0101, -0.0004]$ vs. $\alpha = 0.7$). Sharpening at $\gamma = 4$ reaches comparable recall within noise of confidence weighting and without this precision cost. For WVN, a fixed camera-heavy blend ($\alpha = 0.9$, applied only in cells where both the camera and LiDAR legs are valid) is the only rule for which obstacle recall ($+0.023$, CI $[+0.010, +0.037]$), obstacle IoU ($+0.013$, $[+0.005, +0.023]$), and mIoU ($+0.0047$, $[+0.0024, +0.0076]$) all improve significantly over the LiDAR baseline at no measurable precision cost, and the neighbouring $\alpha = 0.8$ reproduces its gains, indicating a robust camera-heavy regime rather than a tuned point. A direct paired comparison against the initially promising ordinal max-pooling arm ($N = 30$ forest frames, 10,000 bootstrap resamples, seed 20260724, ordinal $-$ fixed $\alpha = 0.9$) resolves into a trade-off rather than a displacement: fixed $\alpha = 0.9$ is ahead on mIoU (-0.0027, CI $[-0.0040, -0.0015]$), obstacle IoU (-0.0021, $[-0.0042, -0.0001]$), and obstacle precision (-0.0127, $[-0.0177, -0.0089]$), while ordinal max-pooling is ahead on obstacle recall ($+0.0052$, $[+0.0034, +0.0074]$). All four differences are statistically resolved (every CI excludes zero) but small, the largest under 1.3 pp, and neither arm dominates the other; the metric on which ordinal wins is the safety-relevant one. The campus equivalent of this comparison (Section 6.7.2) is likewise only marginal on mIoU, so neither custom-platform domain supports a stronger claim than this.

Where the baseline fails (campus), the same rules produce the study's strongest result (Figure 6.5), though not a strict Pareto improvement. The fixed blend at $\alpha = 0.7$ with GA-Nav improves on the LiDAR baseline in obstacle recall ($+0.062$, CI $[+0.045, +0.082]$) and obstacle IoU ($+0.041$, $[+0.028, +0.057]$), both resolved effects validated at frame level and by leave-one-sequence-out analysis with no sequence carrying the effect. The remaining two metrics do not resolve in either direction: the mIoU change ($+0.002$) and the precision change (-0.007) both have confidence intervals that cross zero, so neither can be reported as a gain or a cost. This is a real, substantial improvement on the two metrics most directly tied to obstacle detection, not the strict, all-metric Pareto improvement claimed in an earlier analysis of this comparison. The class-conditional obstacle override at $\tau = 0.5$, relative to its fixed-$\alpha = 0.6$ baseline, trades a $+0.064$ obstacle-recall gain for a -0.023 change in obstacle IoU and a -0.223 precision change. For WVN, the gate at $\tau \geq 0.7$ is degenerate on campus (see Section 6.5), but ordinal max-pooling is not simply tied with fixed $\alpha = 0.9$: its obstacle-recall and obstacle-IoU gains over that fixed arm are positive and resolved. One registered expectation was refuted in an informative direction: GA-Nav's blend optimum was predicted to shift toward LiDAR on campus because of the camera's out-of-vocabulary handicap; instead camera-heavy blends win decisively. In particular, fixed $\alpha = 0.7$ raises obstacle recall by $+0.062$ over LiDAR-only (95 % CI $[+0.045, +0.082]$), because the LiDAR model's own domain failure is the larger effect.

The subsequent full-sweep extension refines this picture: on campus, two rules exceed the fixed $\alpha = 0.7$ blend on individual metrics, though neither dominates it once precision is accounted for. GA-Nav's

confidence-gated max-pooling at $\tau = 0.7$ improves on fixed $\alpha = 0.7$ in obstacle recall (+0.040, CI $[+0.028, +0.052]$) and obstacle IoU is worse, not better, in this comparison (-0.012, CI $[-0.023, -0.002]$, both resolved), at a substantial precision cost (-0.163, CI $[-0.213, -0.117]$); this is a real trade with a cost larger than earlier reported, not a Pareto improvement over the fixed blend. WVN's ordinal max-pooling improves on fixed $\alpha = 0.9$ in obstacle recall (+0.011, CI $[+0.008, +0.014]$) and obstacle IoU (+0.008, CI $[+0.006, +0.011]$), both resolved, with no resolved precision cost (CI $[-0.005, +0.011]$, crossing zero) and no resolved mIoU change. Unlike the GA-Nav comparison, this WVN result is close to a genuine free improvement on the metrics that resolve. Both comparisons use the same paired-bootstrap protocol as the forest ordinal-vs-fixed comparison above (10,000 resamples, seed 20260724), with leave-one-sequence-out agreement.

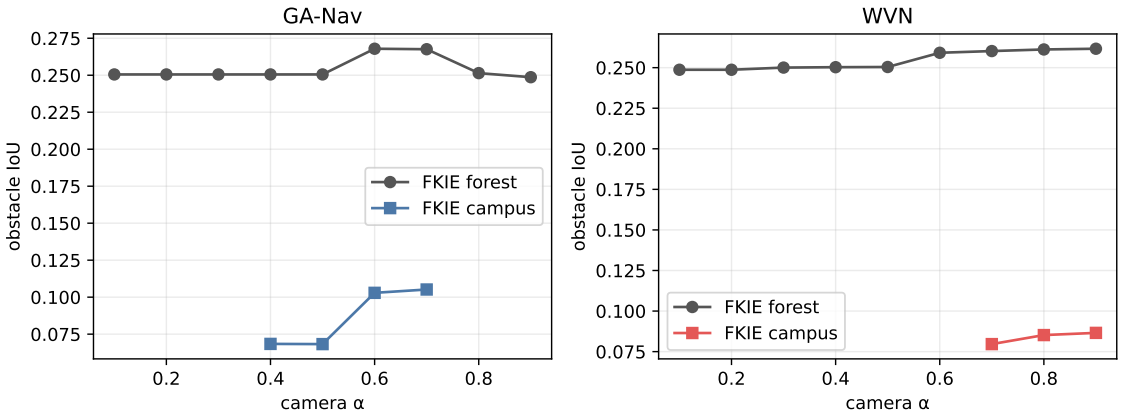


Figure 6.5: Obstacle IoU across the fixed-blend sweep, forest and campus, per backbone, on full support. The camera-weighted regime ($\alpha \geq 0.6$) is the region where gains are consistent on both weak-baseline domains; at lower α on campus, several blend points (both backbones near $\alpha = 0.1$ and $\alpha = 0.5$) fall slightly below the LiDAR baseline rather than exceeding it.

The practical conclusion strengthens with the campus domain, with one qualification added by the full campus sweep: per-backbone fixed blends remain strong and near-optimal across the entire baseline-strength continuum, capturing essentially all realizable camera contribution, and no rule dominates them across every metric on any domain. On campus, however, a confidence-gated or ordinal rule can exceed the fixed blend on individual metrics (obstacle recall, and for the gated arm, mIoU) at a substantial precision cost, so the choice among rules there is a genuine trade rather than a foregone conclusion in the fixed blend's favour. The appropriate α remains stable in the camera-weighted regime rather than needing per-domain retuning. The complete sweep and campus arm tables are given in Appendix A.

6.5 Reliability of confidence signals across domains

Direct reliability measurement of all four available confidence signals against per-cell correctness (ten-bin expected calibration error, ECE [60], globally pooled on GT-valid, prediction-valid support) across the three primary fusion-analysis domains shows that no signal is trustworthy in every domain, and

Table 6.3: Ten-bin expected calibration error (ECE) for the four confidence signals on method-valid support. The dash denotes the unavailable RELLIS-3D WVN reconstruction-familiarity value.

Signal	RELLIS-3D	Forest	Campus
GA-Nav entropy	0.017	0.645	0.367
FASTC maximum-softmax	0.299	0.479	0.333
WVN classification-head entropy	0.034	0.594	0.495
WVN reconstruction-familiarity	—	0.099	0.205

that the two signal families fail in complementary regimes (Figure 6.6). The softmax-derived signals track distance from their training distribution monotonically: GA-Nav's entropy confidence runs ECE 0.017 on RELLIS-3D, 0.367 on campus, 0.645 on forest: a clean three-point ordering confirming that softmax-family calibration degrades with domain shift. FASTC's maximum-softmax and WVN's classification-head entropy follow the same family pattern, with the additional caution that bin-level correlation can badly misrepresent cell-level behaviour: on campus, WVN's head entropy shows bin-level $r = 0.74$ while the raw-cell correlation is *negative* ($r = -0.11$), which is why both statistics are reported throughout (Figure 6.7).

Table 6.3 reports the complete set of 10-bin ECE values used in this comparison. The RELLIS-3D value for WVN's reconstruction-familiarity signal was not available in the committed comparison record and is therefore left unreported. The campus WVN reconstruction signal also illustrates why calibration and ranking must be distinguished: its ECE is 0.205, while its raw-cell correlation is $r = 0.61$ and its accuracy span is 0.68. ECE measures calibration of the confidence values; correlation and accuracy span measure ranking or discrimination, so both properties are relevant to the claim that no signal is trustworthy everywhere.

This characterisation is specific to domain shift, and should not be read as a general insensitivity of softmax confidence to input quality. Analysed along a photometric axis rather than a domain axis (Section 7.3), GA-Nav's entropy confidence declines monotonically as image adversity increases and its calibration error *improves* correspondingly, while the reconstruction-familiarity signal shows no such response. The two signal families therefore respond to two different kinds of novelty: the softmax family registers degradation of the input image but not displacement of the data distribution, and the familiarity signal registers the latter but not the former. Neither covers both.

Campus, where both signal families are directly comparable on the same evaluation frames, makes this complementary failure concrete. Figure 6.7 shows reliability diagrams for all four signals on camera-valid, GT-valid campus cells: the reconstruction signal is under-confident but strongly rank-informative, while the head-entropy signal illustrates a bin-versus-cell divergence, a rising bin-level calibration curve that conceals a negative cell-level correlation.

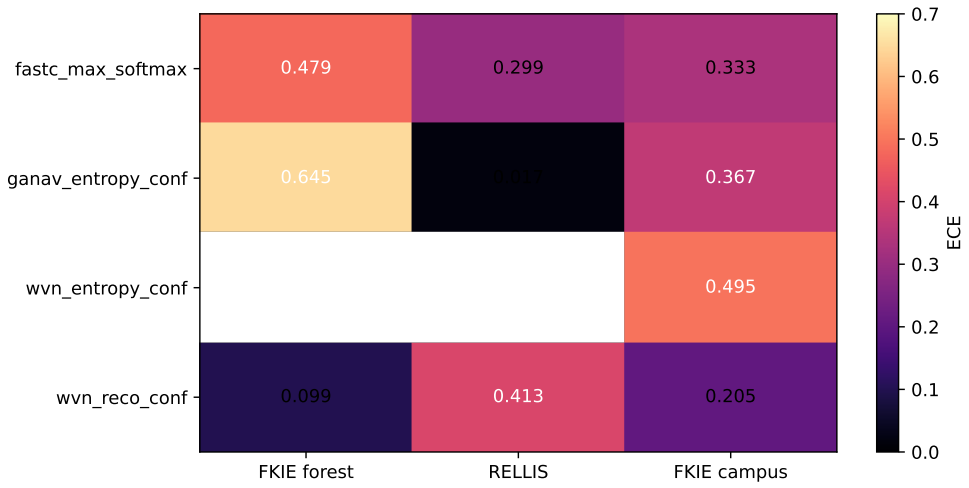


Figure 6.6: ECE by confidence signal and domain, each signal computed on its own method-valid support. This is a calibration diagnostic, not a Condition-A accuracy result. No column is uniformly dark: every signal is substantially miscalibrated in at least one domain, and the two signal families (softmax-derived vs. reconstruction-familiarity) fail in complementary regimes.

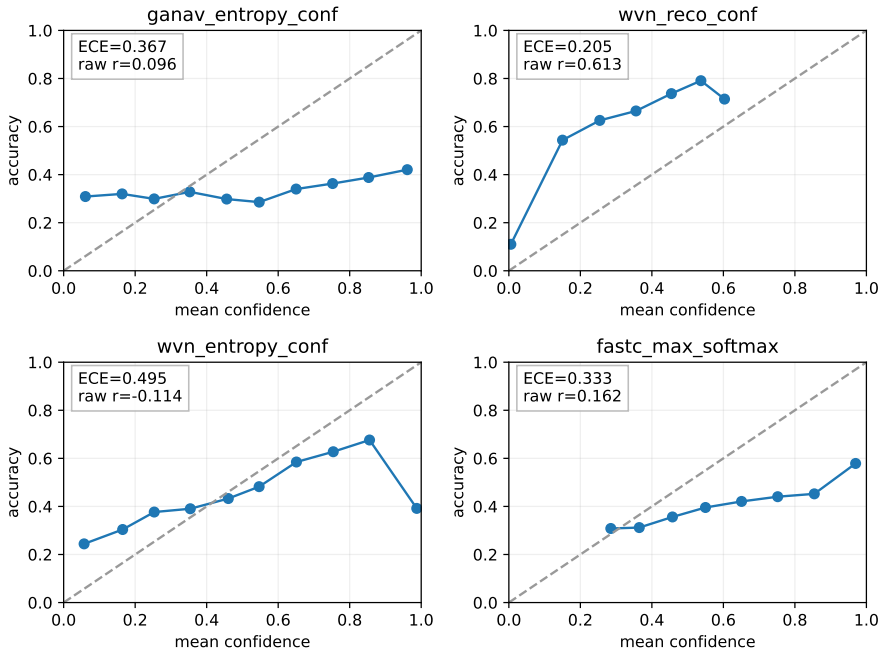


Figure 6.7: Campus reliability diagrams for the four signals, computed on camera-valid ∩ GT-valid cells only, pooled globally, a calibration diagnostic, not a Condition-A accuracy result. Annotated with ECE and raw-cell correlation.

The reconstruction-familiarity signal, however, refutes the simple distribution-distance story. On the initial two domains it appeared to invert the softmax pattern: well calibrated off-distribution (forest, ECE 0.099), broken on-distribution (RELLIS-3D, ECE 0.413, table-level $r = 0.07$), and the mechanism identified was dynamic-range compression under frozen normalization: the signal applies a sigmoid with mean and standard deviation fixed at training time ($\mu = 0.0399$, $\sigma_s = 0.0694$), and on RELLIS-3D 96.2 % of cells fall within half a standard deviation of the training mean, leaving no usable range (no cell on either domain exceeds confidence 0.7, which is why every confidence-gated fusion arm at $\tau \geq 0.7$ degenerates exactly to the LiDAR baseline). Campus completes and corrects the picture: there the reconstruction signal is the *best* measured anywhere in this study, with an accuracy span of 0.68 and raw-cell $r = 0.61$, and with incorrect cells showing an order-of-magnitude higher median reconstruction error than correct ones. Improved rank-correlation does not translate into a wider dynamic range, however: WVN's confidence still never exceeds 0.6079 across the 196,104 valid camera cells on campus, so confidence-gated fusion at $\tau = 0.7$ or $\tau = 0.9$ collapses to the LiDAR baseline exactly for WVN there as well. Even $\tau = 0.5$ changes campus recall by only +0.0013 and precision by only +0.0020 relative to the LiDAR-only baseline, a negligible effect. Confidence gating is therefore effectively inert across the tested threshold range for this backbone, not only at $\tau \geq 0.7$. This is a further symptom of the same frozen-normalization ceiling rather than evidence that gating fails as a mechanism: the signal is informative in its ranking but never reaches the higher operating points, which is why tables and passages reporting WVN gate $\tau \geq 0.7$ arms should read them as inert (identical to the LiDAR-only baseline) rather than as a tested and failing configuration. The revised mechanism, supported by the error distributions across the three primary fusion-analysis domains (Figure 6.8), is that the signal's utility tracks not distribution distance but whether the reconstruction-error distribution *spans the frozen sigmoid's dynamic range*: pure near-distribution content (RELLIS-3D) compresses below the operating point; pure far-OOD content (forest) partially saturates above it; campus's mixture of familiar terrain and alien structure straddles it, and the signal separates the two cleanly. A measurement caveat applies: 33 % of campus cells sit at the low clip of the inversion used to recover raw errors, so campus's upper quantiles are saturated rather than measured.

Post-hoc recalibration was tested and functions as a diagnostic rather than a treatment. Temperature scaling of GA-Nav's painted probabilities finds no finite optimum on forest. The likelihood is maximized in the limit $T \to \infty$, i.e. by flattening predictions to uniform, meaning the off-distribution signal is broken rather than merely miscalibrated. On RELLIS-3D, by contrast, the fitted interior $T = 1.66$ *worsens* held-out ECE, because the signal was already near-calibrated. The boundary behaviour of temperature scaling thus cleanly separates the fixable case from the unfixable one; in this system, every signal falls on one side or the other depending on domain, and the practical consequence closes the loop with Section 6.4: fixed blends, requiring no confidence signal at all, sidestep a problem that recalibration cannot solve.

6.6 The recall–precision trade across three safety regimes

The safety-relevant question for fusion is not whether it improves mean accuracy but whether it misses fewer real obstacles. Earlier analysis identified a safety inversion: fusion arms that win on mIoU while missing more obstacle cells than the LiDAR alone. Across the thresholded fusion interventions evaluated

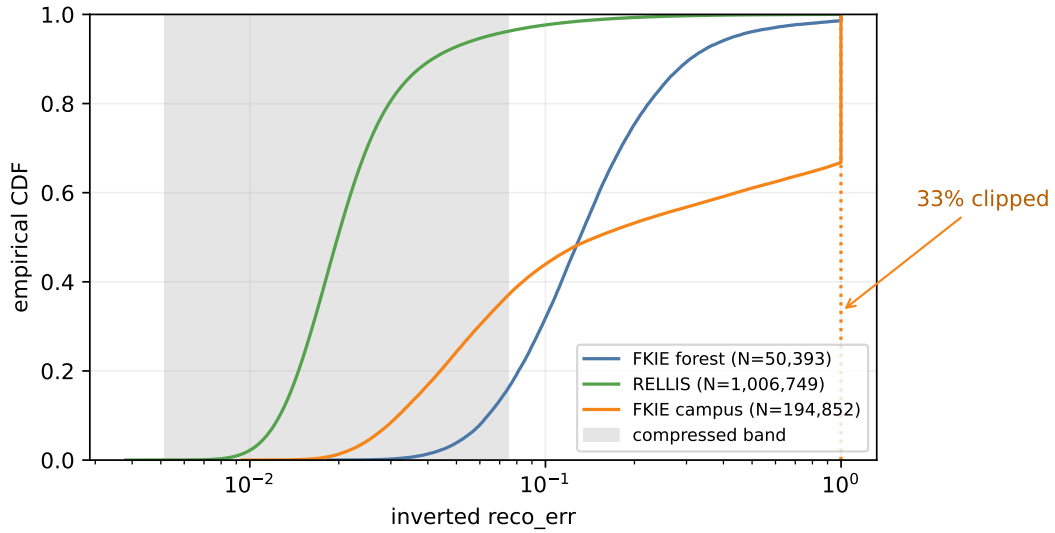


Figure 6.8: Reconstruction-error distributions across the three primary fusion-analysis domains (empirical CDFs, log scale), computed on camera-valid support, a distribution diagnostic, not a Condition-A accuracy result. Shown with the frozen normalization's compressed band shaded and campus's clip saturation marked. RELLIS-3D compresses inside the band; forest spreads beyond it; campus straddles it, the regime in which the familiarity signal is most informative.

here, the safety trade has three baseline-dependent regimes.

Where the baseline is strong (RELLIS-3D), thresholded interventions are inert: the camera confidently contradicts the LiDAR on at most 0.04 % of cells, and no setting moves any metric meaningfully. Where the baseline is weak (forest), the forest class-conditional obstacle-override sweep trades recall against obstacle IoU and precision; no free-safety region exists. Where the baseline fails (campus), the campus GA-Nav confidence-gated arm at $\tau = 0.7$, compared with fixed $\alpha = 0.7$, raises obstacle recall by $+0.040$ (95 % CI $[+0.028, +0.052]$). This comparison could not be reproduced from a preserved original script and was reconstructed from the documented bootstrap method, so the values here should be read as the current best estimate rather than a verified replication. At this operating point, the precision cost is -0.163 (95 % CI $[-0.213, -0.117]$) and the obstacle-IoU difference is resolved and negative, -0.012 (95 % CI $[-0.023, -0.002]$). The intervention therefore does not convert precision into additional obstacle detection at no obstacle-IoU cost; it trades precision and obstacle IoU for recall simultaneously. The trade remains baseline-conditional and priced: a deployment selects an operating point on a measured curve, with LiDAR-only behaviour preserved outside camera coverage.

The forest override sweep also closes the chapter's internal loop: GA-Nav's override-changed fraction barely responds to the threshold (5.7 % at $\tau = 0.3$, 4.7 % at $\tau = 0.9$), because its painted obstacle probabilities are saturated near one, the fusion-level signature of the off-distribution overconfidence measured in Section 6.5. Figure 6.9 shows the qualitative counterpart: the override corrects wall, bin, and vegetation-edge cells that the geometric method reached but assigned the wrong cost tier, at a

visible false-positive cost in the near field. Because the point-painted fusion camera leg exists only on LiDAR-supported cells (Section 5.4.3), the override can act only where the LiDAR already produced a (wrong) tier; it does not add coverage the LiDAR never reached.

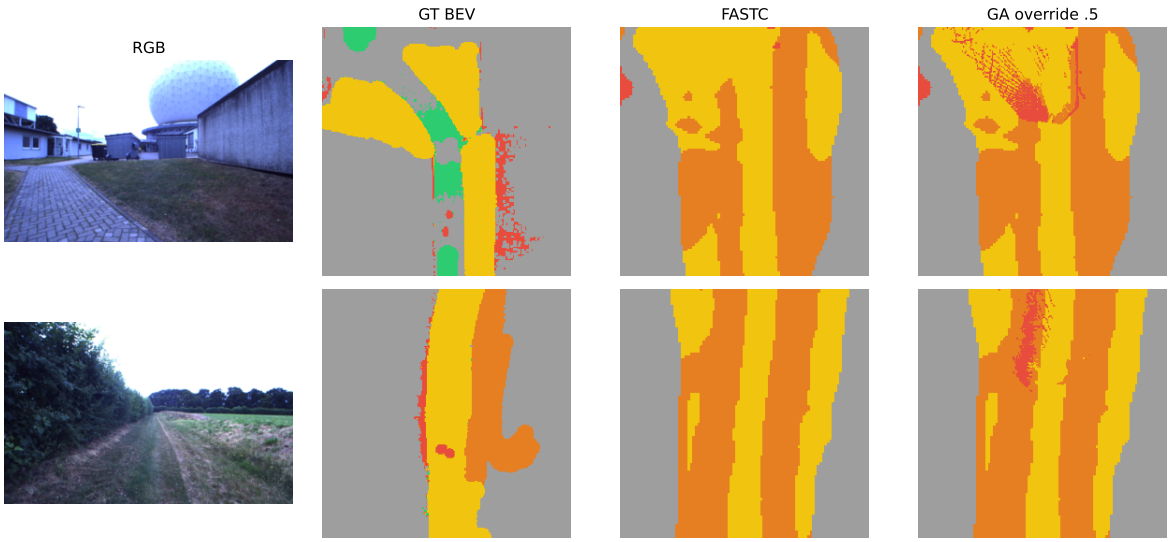


Figure 6.9: Qualitative campus examples (built scene, top; grass path, bottom): RGB, ground truth, LiDAR-only prediction, and the obstacle-override fusion. The override corrects wall, bin, and vegetation-edge cells that the geometric method reached but assigned the wrong cost tier, at a visible false-positive cost in the near field.

6.7 Domain shift and spatial coverage in campus fusion

The campus experiments make the interaction between camera coverage, classifier reliability, and fusion trust explicit. A 48-configuration sweep was evaluated on sequences 00004 and 00005, using fixed camera weights $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, confidence weighting, optional cross-modal obstacle promotion, and sparse or flat BEV densification. The grass sequences are treated as the clean campus test set for this sweep; the urban sequences are analysed separately as a domain-shift probe. (The full 150-frame, five-sequence pooled results reported earlier in this chapter remain the primary evaluation; this two-sequence sweep is a confirmatory, higher-control follow-up.)

6.7.1 Effect of BEV densification alone

Nearest-neighbour BEV densification follows the sparse BEV completion operation described by Qiu et al. (ECCV 2022): an occupied camera-painted cell propagates its prediction to nearby empty cells using an Euclidean distance transform. The flat variant used here applies no confidence decay, isolating spatial coverage from confidence attenuation.

Applied by itself, densification is not a meaningful source of improvement. Under confidence-weighted fusion, the change on RELLIS-3D is only $+0.0003$ mIoU and -0.0006 obstacle recall. On campus the

corresponding changes are +0.0004 mIoU and +0.004 obstacle recall, within evaluation noise. This is consistent with the validity-aware pass-through already used by the pipeline: cells without valid camera evidence are not allowed to inherit an unsupported camera decision. Spatial completion therefore cannot create useful camera evidence when the fusion rule does not trust it.

6.7.2 Fusion-rule dependence of camera contribution

The fixed-weight sweep reveals a safety–accuracy Pareto front rather than a universally optimal setting (Figure 6.10). The FASTC-only baseline on campus sequences 00004 and 00005 is mIoU 0.240 with obstacle recall 0.082. With sparse painting, GA-Nav reaches recalls 0.082, 0.082, and 0.080 at $\alpha = 0.1$, 0.3, and 0.5; recall rises to 0.147 at $\alpha = 0.7$ and 0.181 at $\alpha = 0.9$. Confidence weighting gives 0.141. No rule therefore simultaneously maximises aggregate accuracy and obstacle recall: α is a deployment-time safety dial.

This campus trade generalises the safety inversion observed on RELLIS-3D: fusion can improve aggregate accuracy while missing more obstacles than LiDAR-only. On RELLIS there is effectively no camera headroom, whereas campus exposes the same trade as a controllable parameter. Forest lies between these regimes and remains backbone-conditional: GA-Nav benefits from confidence-weighted fusion, whereas for WVN the rule choice barely matters; ordinal max-pooling and a camera-dominant fixed blend trade off rather than one dominating (Section 6.4).

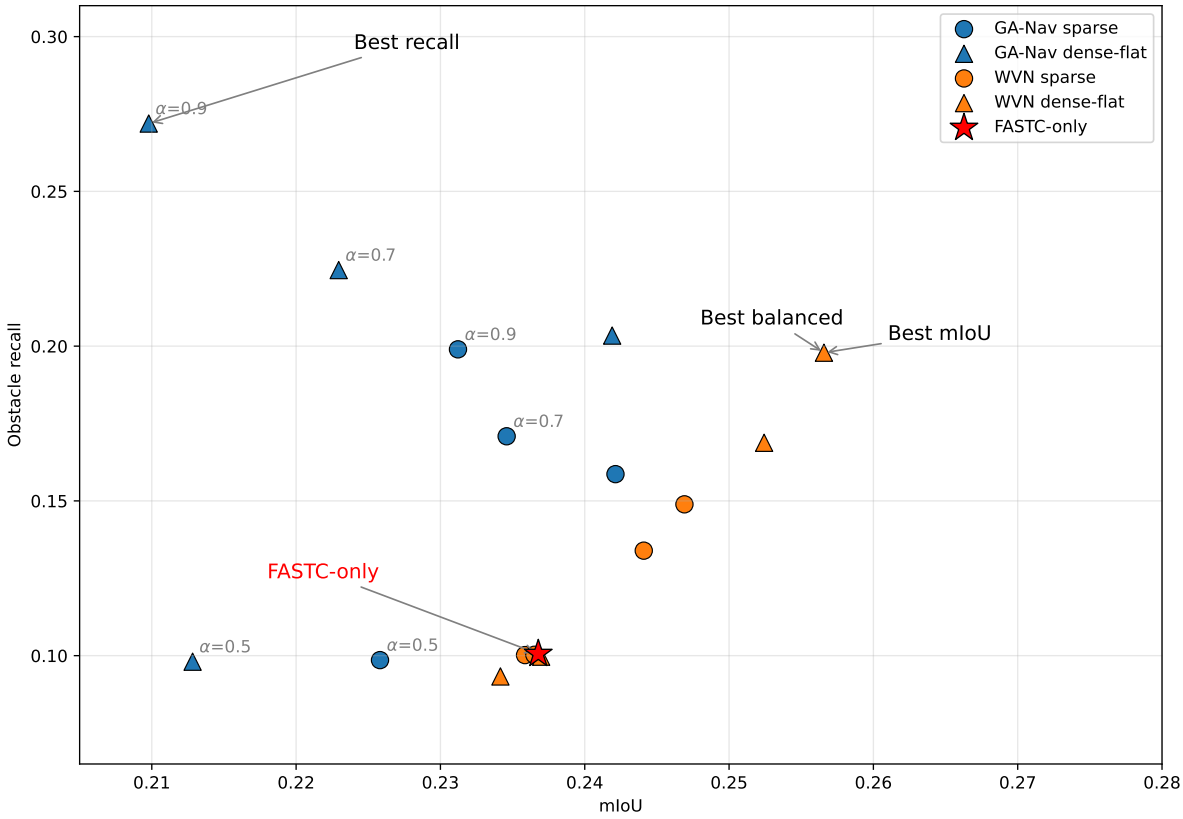


Figure 6.10: Campus safety–accuracy Pareto front. Camera-dominant weights improve obstacle recall while trading against aggregate accuracy.

6.7.3 Coupling between densification and fusion trust

Densification becomes useful only when the fusion rule gives the camera enough authority. For GA-Nav, flat densification changes obstacle recall by only +0.002 at $\alpha = 0.1$, has no measurable effect at 0.3 or 0.5, but increases recall by +0.051 at 0.7 and +0.105 at 0.9. Under confidence weighting, the increase is +0.058. Dense recalls at $\alpha = 0.7$ and 0.9 are 0.198 and 0.286, respectively (Figure 6.11). WVN shows zero effect across the tested rules, consistent with its broken confidence signal.

The interaction is multiplicative in operational terms: neither spatial coverage nor camera trust is sufficient alone; camera-dominant weighting and useful BEV support are both required.

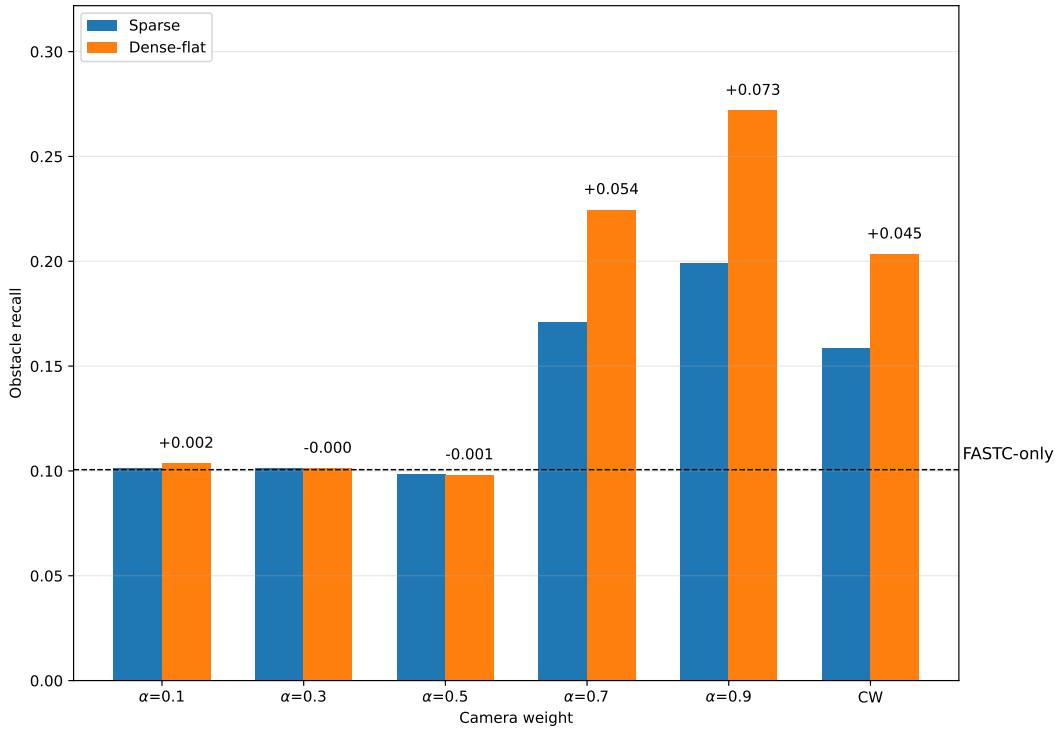


Figure 6.11: Coupling between flat BEV densification and camera weight on campus. Densification matters primarily in the camera-dominant regime.

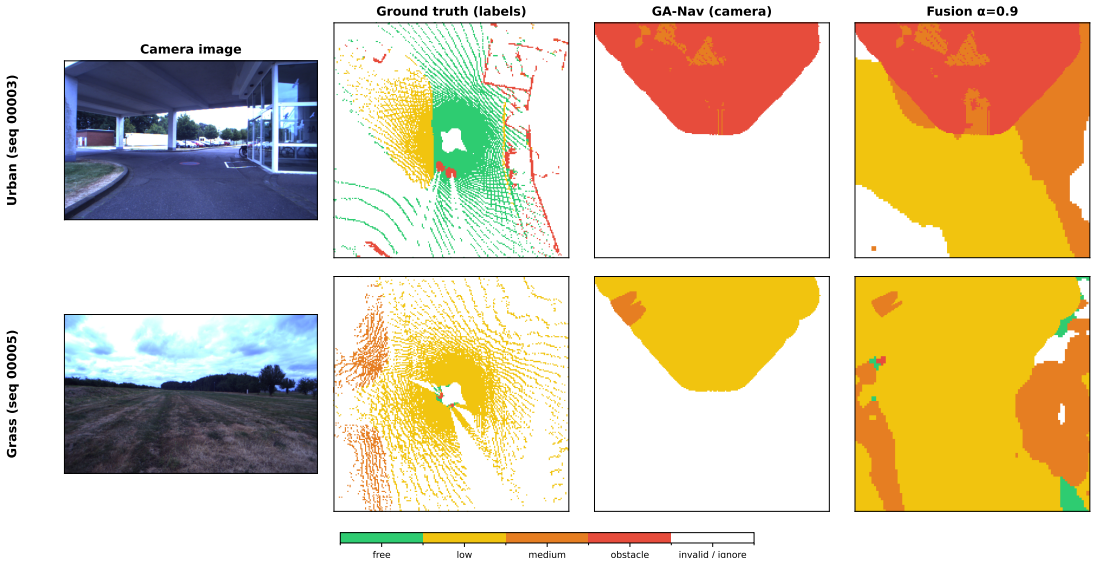


Figure 6.12: Qualitative domain-shift comparison between urban and grass campus sequences. The camera contribution is bounded by the reliability of the underlying semantic predictions.

6.7.4 Domain-shift limits on camera contribution

The Pareto front is bounded by camera-classifier reliability. GA-Nav predicts approximately 78 % of annotated obstacles as obstacle on urban sequence 00003, compared with approximately 24 % on grass sequence 00005 (Figure 6.12). This reflects the domain shift between the built campus scenes and GA-Nav's RELLIS forest training distribution. Fusion and densification can redistribute camera evidence, but cannot repair a classifier whose semantic predictions are wrong; camera-trusting fusion is consequently destructive in such scenes.

The frontier is perceptual, not architectural. The frozen backbones and simple fusion family produce useful gains on the clean campus sequences because camera and geometric errors are complementary, but the gain is bounded when classifier reliability collapses.

6.8 Summary

This chapter has traced a single mechanistic chain from a geometric constraint to a deployment-level recommendation, and tested it across a continuum defined by the three primary fusion-analysis domains. Its headline result is that the point-painted fusion camera leg is a reliable but spatially narrow contributor: re-scored on the cells it actually populates, obstacle recall reaches 0.50–1.00 across all four evaluation domains for GA-Nav, a high-sensitivity, low-specificity regime paired with precision as low as 0.13–0.25, confined to a coverage footprint that is a mid-range annulus rather than a uniform fraction of the scene (Section 6.2.1). The camera's contribution is limited first by coverage: well below 25% of the scene, geometric in origin for the fusion leg and accompanied by the separately reported weak UniDepth camera-only baseline (Section 6.1). Within that point-painted coverage its contribution ranges from statistically robust improvement where the LiDAR is weak to outright compensation where the LiDAR fails (Section 6.2), with the achievable amount bounded in advance by an oracle ceiling that tracks baseline strength (Section 6.3). Simple per-backbone fixed blends capture essentially all of the achievable gain in every regime, including resolved gains in obstacle recall and obstacle IoU over the LiDAR baseline where the geometric method fails, though not a strict improvement across every metric, and remain near-optimal with no rule dominating them across every metric on any domain, though on campus a confidence-gated rule and ordinal max-pooling each exceed the fixed blend on individual metrics (obstacle recall, and for the gated arm, mIoU), at precision costs of -0.163 for the confidence-gated arm and -0.0263 for ordinal max-pooling (Section 6.4); consistent with direct measurement showing that no confidence signal is trustworthy in every domain, with the softmax family degrading monotonically under domain shift and the reconstruction-familiarity signal governed by whether the error distribution spans its frozen dynamic range (Section 6.5). Finally, the safety trade between precision and obstacle recall is not a fixed property of fusion but a baseline-conditional, single-parameter curve: inert, costly, or nearly free depending on where the geometric method stands, selectable at deployment time without retraining any component (Section 6.6). Chapter 7 returns to the scope of these findings, including what the campus result implies for the assumption that geometric methods are the transfer-robust modality.

7

Discussion

Chapter 6 established a mechanistic account of the camera modality's contribution to fusion and tested it across the three primary fusion-analysis domains spanning the full observed range of LiDAR-baseline strength. This chapter steps back from that account to consider its scope: what the campus domain changed, what generalises beyond the domains studied, what the study's design choices leave open, how the findings relate to the broader fusion literature, and what they imply for a system that must actually be deployed.

7.1 Generalisability of the findings

The addition of the campus domain converts several two-point contrasts into orderings across the three primary fusion-analysis domains, and in doing so changes which claims are strongly supported, which are strengthened but still bounded, and which were corrected. Separating them by evidential status:

7.1.1 Findings by evidential status

- **Strongly supported across the three primary fusion-analysis domains.** The coverage wall is a geometric limitation, mildly scene-modulated in magnitude, and the camera-only UniDepth projection reaches only a small fraction of GT-valid cells across all four evaluation domains; its expected coverage can be estimated in advance from the extrinsics of a given rig, although this transfer to untested hardware is not validated here. The point-painting comparison in Section 4.1.3 is a separate fusion-methodology decision, not evidence for the camera-only coverage wall. Fusion headroom tracks LiDAR-baseline weakness across the three primary fusion-analysis domains, and realized gains broadly track the headroom, from an exhaustive null (RELLIS-3D) through a partial, mechanism-limited gain (forest, where the camera leg's contribution is difficult to separate from broad, low-precision obstacle-calling; see Chapter 6) to the largest aggregate contribution in the study (campus, though this aggregate itself conceals a genuinely mixed sequence-level picture, detailed below). The softmax-family calibration ordering: ECE degrades monotonically with distance from the training distribution across the three primary fusion-analysis domains ($0.017 \rightarrow 0.367 \rightarrow 0.645$). Fixed blends remain strong and near-optimal in practice, and no rule dominates them across every metric on any domain, though on campus a confidence-gated rule (resolved gains in obstacle recall

and mIoU, but a resolved obstacle-IoU cost) and ordinal max-pooling (resolved gains in obstacle recall and obstacle IoU for WVN) each exceed the best fixed blend on some metrics at a substantial precision cost.

- **Newly established across all four evaluation domains.** Coverage and competence are separable, and full-support scoring conflates them. Re-scored on camera-valid support, the point-painted fusion camera leg's GA-Nav obstacle recall is 0.50–1.00 across RELLIS-3D, forest, campus, and GOOSE, and holds in the 0.87–0.98 band on every campus sequence regardless of scene structure (Section 6.2.1). The point-painted fusion camera leg is therefore a spatially narrow contributor: on the three off-distribution domains (forest, campus, GOOSE) it is a high-sensitivity, low-specificity detector (precision 0.12–0.24 for GA-Nav, identical under full-support and camera-valid scoring since a cell with no camera prediction cannot contribute a false positive either way), while on RELLIS-3D, near the training distribution, precision is substantially higher (0.5951); its usable region is a mid-range annulus of roughly 3–10 m rather than a uniform fraction of the scene (Section 6.2.3). This is the mechanism behind the fusion precision–recall trade characterised throughout Chapter 6: any rule that trusts the point-painted fusion camera leg inherits both its sensitivity and its false positives.
- **Strengthened but still bounded.** The backbone-conditional rule structure (GA-Nav benefits from moderate camera weighting, WVN from heavy camera weighting or ordinal pooling) now replicates on two weak-baseline domains, but both use the same rig, the same checkpoints, and the same tier-cost mapping; the specific optimal α values should not be assumed to transfer to a different cost formulation without re-sweeping.
- **Corrected by the campus domain.** The confidence-complementarity claim in its original form (softmax signals fail off-distribution, the reconstruction-familiarity signal fails on-distribution, in mirror image) does not survive campus, where the reconstruction signal is the best measured anywhere. The corrected account (Section 6.5) replaces distribution distance with dynamic-range spanning as the governing variable for the familiarity signal. This is a stronger mechanism precisely because it was reached by refutation; it remains, however, a observation across the three primary fusion-analysis domains about one frozen normalization, not a law.
- **Newly established, and the largest single correction.** The implicit assumption, common in the fusion literature and held by this study until the campus evaluation, that the geometric modality is the transfer-robust one, was refuted. The same LiDAR checkpoint spans obstacle recall 0.945, 0.417, and 0.107 across the three primary fusion-analysis domains, with 84 % of campus ground-truth obstacle cells misclassified as traversable-at-cost even where direct LiDAR evidence exists (annotation-density effects ruled out by support-restricted scoring). Geometry trained off-road does not transfer to built environments; it fails by severity compression, the same failure mode the camera exhibits for out-of-vocabulary objects. This is an existence proof from a single site and season (Section 7.2), not a calibrated estimate of how badly the failure generalises to built environments broadly. Neither modality is robust; they are differently fragile.

Table 7.1: Pre-registered campus predictions and outcomes, registered before the campus results existed.

#	Topic	Outcome (summary)	Verdict
P1	Baseline mIoU mid-range	Near forest level; obstacle recall collapsed to 0.107	refuted
P2	Coverage ≈ forest	Ratio 0.65–0.70, within registered bound	confirmed
P3	GA-Nav ECE ordering	$0.017 < 0.367 < 0.645$	confirmed
P4	Reco-conf intermediate	Band fraction interpolates; signal is best of all domains	partial
P5	GA-Nav optimum toward LiDAR	Camera-heavy blends win; LiDAR failure dominates	refuted
P6	Fusion gain between forest/RELLIS	Largest gain of any domain; resolved recall and obstacle-IoU gains, not a strict Pareto win	refuted
P7	Urban scenes worst for GA-Nav	Sequence ranking driven by LiDAR failure profile	refuted

7.1.2 Scoring the pre-registered predictions

The campus evaluation was designed with seven falsifiable predictions registered before any result existed (Table 7.1). Two were confirmed, one partially, and four were refuted. The refutations carry most of the value: P6's refutation produced the strongest fusion result in the study, and P1, P5, and P7 jointly overturn the geometric-robustness assumption discussed above; P4's partial result independently produced the revised dynamic-range mechanism. Reporting the misses alongside the hits is deliberate: it demonstrates that the evaluation design constrained the conclusions rather than merely decorating them.

7.2 Limitations

Several design choices and dataset properties constrain how the results should be read.

Sample sizes and independence structure. The forest domain comprises 30 evaluation frames and campus 150 (five sequences of 30). Every claim-bearing difference was validated by jackknife leave-one-out and paired bootstrap resampling, and on campus additionally by leave-one-sequence-out analysis, because consecutive frames within a sequence are temporally correlated and the sequence, not the frame, is the honest independence unit. All headline campus effects survive both levels with no single sequence carrying them; nevertheless, five sequences bound how finely sequence-level effects can be resolved, and frame-level confidence intervals should be read as optimistic wherever the sequence-level range is the wider of the two.

Source-recording overlap between the OOD probe and the campus evaluation. The qualitative out-of-distribution probe (Section 7.5) and the campus evaluation draw on the same source recordings, sequences 00001–00003. The frames are disjoint (no frame used in the probe appears in the labelled evaluation set, or vice versa), but the recordings are not, and frame-level disjointness should not be read as full independence: frames drawn from the same recording share scene content, lighting, and platform state. The probe's role in this thesis is illustrative; the quantitative claims in Chapter 6 and elsewhere in this chapter rest on the labelled campus evaluation, not on the probe.

Ground-truth provenance. The forest and campus ground truth was annotated by the author, directly in the BEV plane on single selected scans, after the methods under evaluation were already developed. Point-label ground truth (Section 3.4.3) is derived directly from LiDAR returns, so every scored cell has direct point support by construction; this is a different property from the dense manually-painted raster it was exported from, which could label cells with no underlying return. Both conventions and their relationship are described in full in Section 3.4.3. Per-tier IoU values on thin-support classes should nevertheless be read with their support counts.

The flat-ground projection construction was re-established as a sourced diagnostic on RELLIS-3D sequence 00004, where it produced zero obstacle IoU for both backbones (Section 4.1.3). It was not re-established as a full evaluation across the forest, campus, and GOOSE domains; the final cross-domain camera-only results instead use UniDepth monocular-depth projection throughout.

Clip saturation in the reconstruction-error analysis. Raw reconstruction errors were recovered from cached confidences by inverting the frozen sigmoid; on campus 33 % of cells sit at the inversion's low clip, so the upper quantiles of the campus error distribution are saturated rather than measured. The qualitative three-regime structure is unaffected; the exact campus tail statistics are not reportable.

Temporal block splits in calibration. Temperature scaling used contiguous calibration/evaluation blocks to avoid overstating generalisation under temporal correlation; a different split boundary could yield a different fitted temperature, particularly where the fitted value was already found to be split-sensitive.

Residual extrinsic calibration uncertainty. The two calibration methods retained for comparison (Section 3) agree to within 2.3° in rotation but disagree by 12.2 cm in translation, particularly along the z-axis. The selected calibration was chosen on qualitative and integration grounds, not because its accuracy was independently verified against the alternative; this study did not measure how much of the residual translation disagreement propagates into point-painting accuracy or fusion results.

Unconfirmed training details. The GA-Nav and WVN training hyperparameters were not independently reconfirmed during this study, though checkpoint provenance (training-data exposure, absence of leakage, and consistent checkpoint identity across domains) was.

Domain scope. The three primary fusion-analysis domains were captured in temperate European or North American outdoor conditions with two sensor rigs. The campus domain in particular is a single site captured in a single season; its dramatic baseline failure should be read as an existence proof that geometric traversability does not transfer to built environments, not as a calibrated estimate of how badly it fails in built environments generally.

Multi-scan LiDAR configurations. FASTC-S was evaluated only in its single-scan configuration on the custom forest and campus datasets; no multi-scan-accumulated configuration was tested there. This is a limitation of this work's use of the data, not of the data itself: the raw recordings retain the pose information a multi-scan configuration would need (odometry, dynamic TF, and static rig transforms are all present), but the curated evaluation dataset used in this study does not carry that information forward. A prior attempt to build registered multi-scan windows produced point clouds that were less well aligned than the unregistered single scans (median nearest-neighbour distance rose from 0.78 m to 2.48 m; occupancy IoU fell from 0.197 to 0.051), indicating an unresolved inter-scan registration convention on

this project's side rather than absent or unusable source data. Multi-scan configurations were therefore not evaluated on the custom datasets because that registration convention was not resolved within the project timeline.

Cross-dataset taxonomy collapse. RELLIS-3D's native 20-class ontology [14] and GOOSE's native 64-class ontology [16] are each mapped onto this thesis's shared four-tier scheme (Section 3), but the specific class-to-tier crosswalk is not published in this thesis. Collapsing a 20- or 64-class native taxonomy into four tiers necessarily discards distinctions the original annotators drew; the extent to which this affects cross-dataset comparability, particularly for classes near tier boundaries, is not quantified here.

7.3 Failure-mode analysis

The evaluation so far has compared methods in aggregate and by domain. This section examines *where* each method fails along five structured axes: range, terrain, lighting, obstacle geometry, and sensor density. Four of these were committed to at the proposal stage; the fifth was added because sensor density proved measurable on the campus data and because it repairs a claim made earlier in this project. Each axis was analysed with pre-registered predictions recorded before the results existed. Most of those predictions were refuted, and the refutations are reported alongside the confirmations, since several of them corrected assumptions that would otherwise have propagated into the conclusions.

7.3.1 Range

The evaluation footprint was partitioned into four concentric bands (0–3, 3–5, 5–8, 8–10 m) defined purely by grid geometry, plus a fifth audit region covering the square crop's diagonal corners (10–14.07 m), which is scored in the aggregate metrics but is not a range band: its angular coverage is a function of the square footprint rather than of sensor range. The audit region is not negligible: on forest it holds 36,067 of 68,591 ground-truth-valid obstacle cells, so it is reported separately rather than merged into the outermost band. Band-wise confusion matrices sum exactly to the committed full-crop confusion on the three primary fusion-analysis domains, confirming the partition.

Three registered predictions were refuted. First, coverage of the point-painted fusion camera leg in the near field is *not* negligible on this platform: within 0–3 m the point-painted camera legs cover 6.4 % (forest) and 5.46 % (campus) of ground-truth-valid cells, with near-perfect obstacle recall inside that coverage. The frequently cited observation that the fusion camera leg contributes nothing below 3 m derives from the GOOSE platform's mounting geometry and does not transfer to the ROVO rig; only RELLIS-3D (0.03 %) reproduces it. Second, LiDAR obstacle recall *increases* with range rather than decreasing — forest rises from 0.303 (0–3 m) to 0.377 (8–10 m), and the other domains show the same direction. The near field is the harder region, plausibly because close obstacles are partially observed, clipped by the sensor's blind cone or self-occluded, whereas mid-range obstacles present complete profiles. Third, and most consequentially for deployment, the fusion advantage on campus does not concentrate in a near-field annulus where camera coverage is greatest; it grows monotonically with range. The obstacle-override arm's gain over the LiDAR baseline rises across bands from +0.010 to +0.100, while point-painted fusion-camera-leg coverage *peaks* at 3–5 m and declines thereafter. Fusion-camera-leg benefit and its coverage are therefore decoupled: the

point-painted leg contributes most where it covers least, because that is where the geometric method's evidence thins. This is compatible with, not a correction of, Section 6.2.3's characterisation of the camera leg's own "usable region" (roughly 3–10 m): that describes the camera's own competence within its support, whereas the range-dependence of fusion benefit described here tracks the LiDAR baseline's weakness, which need not coincide with where the camera happens to cover most.

7.3.2 Terrain

The terrain axis is addressed by the structure of the three primary fusion-analysis domains (Chapter 6) and, within campus, by its five sequences spanning parking areas, built streets, paved paths and open field. No additional computation was required for this axis; it is reported as a collation of already-committed results, with the individual camera-only UniDepth score artifacts for forest and RELLIS-3D committed and sourced, as used for the Chapter 5 results.

7.3.3 Lighting

No lighting metadata was recorded during capture, so photometric conditions were derived from the images themselves: mean and median luminance, 5th and 95th percentiles and their difference, the fraction of pixels at or near saturation, the fraction near black, and RMS local contrast, computed for all 180 custom-dataset frames. Frames were then ranked by a composite adversity score and split into quartiles, and a pre-registered confounding check confirmed that no quartile drew disproportionately from a single sequence.

That check was insufficient. The LiDAR method, which never observes an image and cannot be affected by illumination, nevertheless varies two-fold in obstacle recall across the quartiles (0.100, 0.137, 0.202, 0.101). Since a lighting-blind method cannot respond to lighting, this variation demonstrates that photometric adversity correlates with scene content within sequences as well as between them. Per-quartile camera performance therefore cannot be attributed to illumination, and the axis is reported as confounded. Isolating it would require capture designed for the purpose (the same scene under differing conditions), which this dataset does not provide.

Two observations survive the confound, because both concern signal behaviour rather than attribution of accuracy. First, highlight saturation is the dominant photometric property of the custom data: between 9.6 % (forest) and 27.5 % (campus, one sequence) of pixels sit at or near maximum intensity, an order of magnitude above the threshold registered as noteworthy. Second, and more interesting, the two confidence signals respond to photometric adversity in opposite ways, and opposite to the prediction. GA-Nav's entropy confidence declines monotonically across the adversity quartiles (0.799 to 0.692) while its expected calibration error *improves* (0.479 to 0.298): the signal becomes appropriately less certain as images degrade. WVN's reconstruction error, by contrast, shows no monotonic response. This refines the account in Section 6.5: softmax-derived confidence fails under *domain* shift but tracks *photometric* degradation correctly, whereas the reconstruction-familiarity signal behaves inversely. The two signal families respond to two different kinds of novelty, and neither covers both.

7.3.4 Obstacle geometry

The proposal committed to a failure analysis by obstacle semantics. The four-tier annotation retains no per-object class labels, so semantics cannot be recovered; a geometric stratification is substituted and is labelled a proxy throughout. Connected components were extracted from the ground-truth obstacle mask in each frame and characterised by area, elongation, range, and the height distribution of the LiDAR points falling within them.

The pre-registered four-way stratification by area and height did not hold up. Stratum membership and the resulting ordering are unstable across height thresholds of 1.0, 1.5 and 2.0 m, and neither intended contrast is statistically resolved (forest +0.030, 95 % CI $[-0.230, +0.282]$; campus +0.007, $[-0.046, +0.077]$). The component-level data nonetheless reveal a cleaner structure on RELLIS-3D, where the geometric method works well enough for its behaviour to be characterised: detectability is governed principally by height, with a 50 % detection crossover at -0.599 m relative to the sensor plane and a plateau above 0.95 for components exceeding +1 m. The crossover is robust to using the 90th-percentile height in place of the maximum (a shift of 0.018 m). Area is not irrelevant, as an initial reading of the component scatter suggested: below the crossover, detection declines with component area rank, from approximately 0.60 to 0.11 in the largest rank. Height is the clearer boundary; area modulates it.

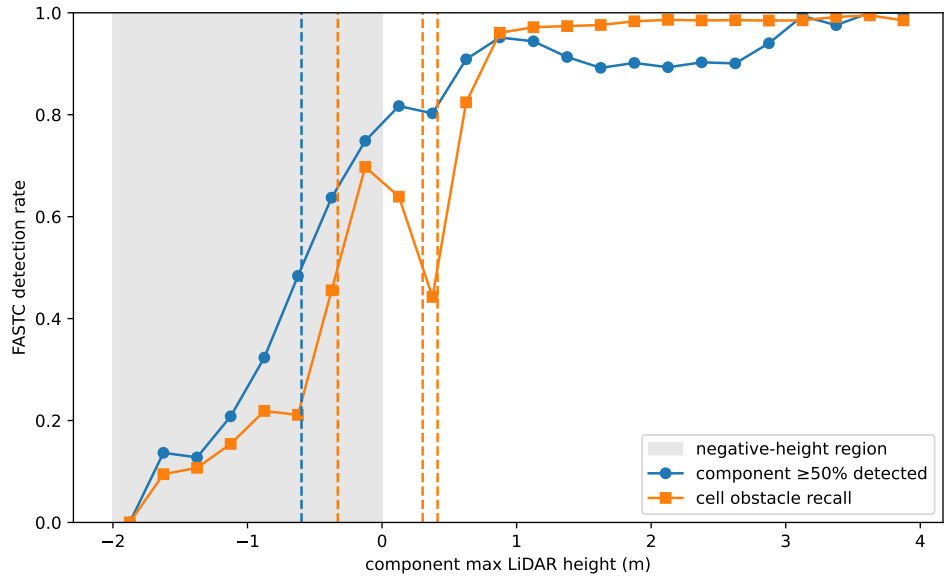
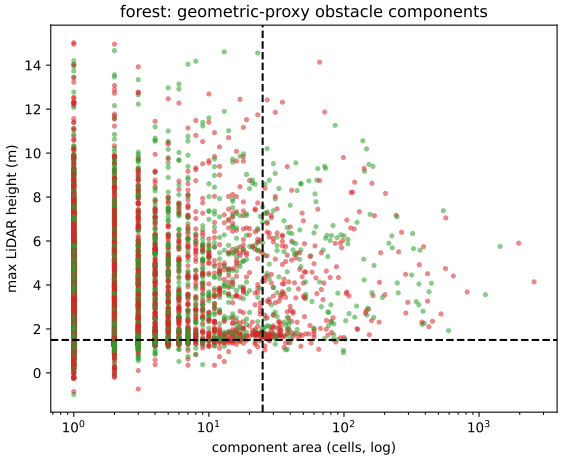
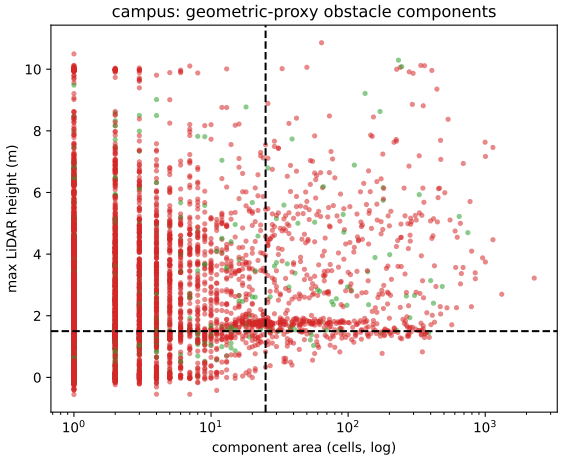


Figure 7.1: RELLIS-3D obstacle-component detection as a function of relative component height. The plot supports the height-based crossover and plateau reported in the obstacle-geometry analysis.

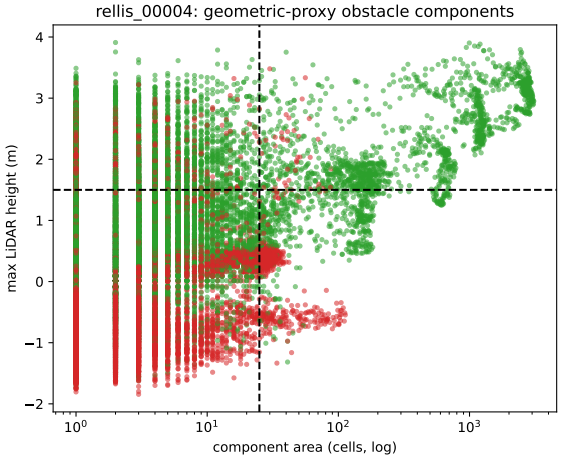
The component-level distributions for the three analysed collections are shown in Figure 7.2. They are presented as a geometric proxy because the four-tier annotations do not retain object identities or semantic obstacle classes.



(a) FKIE forest.



(b) FKIE campus.



(c) RELLIS-3D.

Figure 7.2: Obstacle-component geometry for the custom forest, custom campus, and RELLIS-3D evaluation collections. Area, elongation, range, and height are used as proxy descriptors rather than recovered object semantics.

Table 7.2: Beam-density downsampling on the custom forest recording (FKIE test site, $N = 11$ single-scan frames). Ring-exact decimation from the LiDAR's native 128 beams; FASTC-S (`exp9`) was trained at 64-beam density.

Beams	mIoU	Obstacle IoU	Obstacle recall	Coverage	No-pred
128	0.0425	0.0829	0.1114	0.779	0.221
64	**0.0553**	**0.1243**	**0.2528**	0.645	0.355
32	0.0421	0.0686	0.1037	0.263	0.737
16	0.0367	0.0409	0.0501	0.108	0.893

The analysis also isolates a blind spot shared by both modalities. 17,507 annotated obstacle components on RELLIS-3D (44,949 cells, 3.2 % of that domain's obstacle ground truth) lie entirely below the sensor plane. Inspection of sampled components confirms these are genuine negative structures such as ditches and embankment drop-offs, with their full point distributions below zero rather than a single outlier depressing the maximum. The LiDAR method recovers 32.4 % of these cells; the camera legs recover essentially none (GA-Nav 0.0002, WVN 0.000), and fusion adds nothing measurable (0.3238 against the LiDAR's 0.3238). Obstacles defined by the *absence* of ground rather than the presence of structure are therefore invisible to the point-painted fusion camera leg in this evaluation and only partially visible to the geometric one, a safety-relevant limitation that no fusion rule in this study addresses.

On campus, by contrast, the component scatter shows undetected obstacles distributed almost uniformly across the entire area–height plane. No geometric boundary exists to be found, which is independent visual confirmation that the campus failure documented in Chapter 6 is semantic rather than structural.

7.3.5 Sensor density and configuration

An earlier experiment in this project found that the geometric method's accuracy peaked at 64 beams rather than at full sensor resolution, and interpreted this as a train/test density mismatch: performance is best when the evaluation density matches the training density, a mechanism consistent with reports that aligning source and target point-cloud density is essential for cross-sensor transfer [61]. That original result, run on the custom forest recording ($N = 11$ hand-labelled single-scan frames, the same FASTC checkpoint as Appendix Table C.1, one shared ground-truth footprint across every beam arm) before the campus data existed, is summarised in Table 7.2: physical-ring decimation (exact per-point `ring` index) from the sensor's native 128 beams down to 16 shows accuracy peaking exactly at FASTC-S's 64-beam training density and falling off on both sides, a dose-response shape rather than a monotonic sparsity trend, so it is not simply a coverage artefact (coverage instead falls monotonically across the same arms, $0.779 \rightarrow 0.645 \rightarrow 0.263 \rightarrow 0.108$, a different curve entirely). A ring-parity control (scoring the even- and odd-indexed 64-beam halves separately) confirmed the peak did not depend on which half of the rings were kept: both halves outperformed the full 128-beam cloud (obstacle IoU 0.124 and 0.104 respectively, versus 0.083 at 128 beams).

That finding rested on this single dataset and could not be replicated at the time, because the forest

export discarded per-point beam indices. The campus data retains them, and its point coordinates are element-wise identical to the committed clouds, so beam selection is exact rather than estimated. Decimation follows the beam-missing protocol used in robustness benchmarking [62].

Two structural facts had to be established first. The campus stream is two merged 128-beam sensors, not a single 256-beam unit: elevation rises monotonically across the first ring block, resets, and rises again, in every sequence, and the two blocks differ in azimuth sector and timestamp offset. And the two sensors cover different regions, so any comparison between them confounds beam count with field of view. Density arms were therefore drawn from both blocks with endpoint anchoring, so that only angular spacing varies; robust spatial extents confirm the field of view is preserved at every rung.

The peak did not replicate on the merged stream. All confidence intervals for halving the beams include zero, so the effect is not statistically distinguishable from full resolution, and it is carried by a single sequence whose removal reverses its sign. On a single sensor, which eliminates the merging confound and is the closer analogue to the original experiment's platform, halving the beams is *significantly harmful*: obstacle recall -0.059 (95 % CI $[-0.090, -0.032]$) on full support and -0.086 on point-supported cells, with every leave-one-sequence-out estimate negative. The earlier claim is accordingly narrowed. Accuracy peaked at training density on one dataset, and that behaviour did not generalise. Given that the geometric method's obstacle class fails on campus regardless of density (Chapter 6), the reasonable interpretation is that domain content dominates sensor density in this setting: the density-alignment mechanism is real but is not the binding constraint here.

Two further results follow. Below roughly 64 beams per sensor, performance does not merely degrade but collapses: predicted coverage falls from 0.74 to 0.22 to 0.02 across successive halvings, far faster than point count, indicating that the completion stage falls below an operating threshold and ceases to produce predictions rather than producing worse ones. This differs from the graceful degradation reported for point-wise segmentation architectures [62] and appears attributable to the completion-based BEV architecture evaluated here. Separately, a single sensor *outperforms* the merged pair on obstacle detection where sensor evidence exists (point-supported recall $+0.074$, CI $[+0.029, +0.123]$; obstacle IoU $+0.049$, $[+0.014, +0.085]$), suggesting that merging two differently posed sensors introduces structure the model was not trained to interpret, a configuration finding with direct deployment relevance.

7.3.6 Summary of the failure-mode analysis

Of the five axes, three yielded usable characterisations, one was completed but confounded, and one produced a documented non-replication that narrows a prior claim. The camera's benefit grows with range rather than concentrating near the platform; obstacle detectability by the geometric method is governed principally by height above the sensor plane, with a measurable crossover; negative obstacles are a blind spot shared by both modalities and unaddressed by fusion; lighting could not be isolated from scene content with the data available; and sensor density, contrary to an earlier result in this project, does not exhibit a training-matched optimum on a second dataset. Six of the eight pre-registered predictions across these axes were refuted, a rate worth reporting rather than concealing.

7.4 Relation to the broader fusion literature

The rule study tested confidence weighting, sharpening, and gating using static per-modality confidence signals, found all of them unreliable in at least one domain, and found none that dominates a fixed blend across every metric, though on campus a confidence-gated rule and ordinal max-pooling each exceed the best fixed blend on individual metrics (obstacle recall, and for the gated arm, mIoU) at a substantial precision cost. Read across all three datasets and the full 17-arm rule space run on each, this is more than a null result about one signal: no confidence-driven rule dominates a per-backbone fixed blend on any domain; the two exceptions on campus trade recall for a real precision cost rather than improving jointly; and the underlying signal is itself demonstrably miscalibrated: WVN's painted confidence never exceeds 0.6079 on campus, so any gate threshold above that is inert by construction rather than merely conservative. Taken together, this is evidence against the assumption shared by UP-Fuse, UFO, and evidential fusion frameworks alike: that modelling per-modality uncertainty and using it to weight or gate fusion earns its added complexity. UP-Fuse trains a dedicated head to regress feature instability under input degradation rather than reading confidence off the task head [63]; UFO performs uncertainty-aware LiDAR-image fusion in BEV space for off-road terrain [35]; evidential frameworks learn per-class, per-modality reliability coefficients through contextual discounting [64]. The scope of this challenge should be stated precisely, however: it is evidence about *cost-space late fusion using confidence read directly off the task head* – the setting this study and UFO both operate in. It does not speak to feature-level fusion, where uncertainty can inform representation learning before any late-stage weighting decision, nor to dedicated uncertainty heads trained specifically to regress reliability, which is what UP-Fuse does and what this study's frozen, off-the-shelf confidence signals were not. This study supplies what that literature largely assumes: direct measurement, on real off-road and built-environment data, of exactly how and where static, task-head confidence fails, with a mechanism for each failure, and, within that specific setting, a reason to doubt that learning it is necessary.

The campus result adds a second, sharper point of contact. The per-class, per-modality structure that evidential discounting proposes to *learn* is precisely the structure this study *measured*: a geometric modality reliable for obstacles only within its training domain, camera modalities reliable only within a narrow coverage window and subject to vocabulary limits, and, critically, both modalities failing by the same mechanism, severity compression, when pushed off distribution. A learned reliability module for this system would therefore not need to discover an arbitrary function; it would need to represent domain membership, which the reliability tables for the three primary fusion-analysis domains suggest is largely recoverable from the signals this study already characterised (softmax entropy for the near-distribution case, spanning of the reconstruction-error range for the mixed case). Building such a module was outside this study's frozen-component scope and is the natural next step.

7.5 From qualitative probe to quantitative finding

An early out-of-distribution probe on the campus dataset's unlabelled urban sequences (00001–00003, analysed separately from the clean grass test set in Section 6.7) produced two qualitative observations: the camera segmenter mapped vehicles to *medium*, severity compression under vocabulary mismatch,

while the geometric method appeared structurally coherent. The campus evaluation, with ground truth, quantified both and reversed their apparent importance. The camera's vocabulary limit is real, but it is second-order. Camera-heavy fusion still delivers the study's largest gains on exactly these scenes. The geometric method's apparent structural coherence concealed the dominant failure: predictions that *look* sensible in BEV while misclassifying 84 % of annotated obstacle cells as traversable-at-cost. This is a methodological caution worth stating generally: plausibility inspection of unlabelled out-of-distribution output systematically favours geometric methods, whose failure mode (wrong severity, right shape) is invisible without labels, over semantic methods, whose failure mode (wrong class, visible category) is obvious. The lighting and obstacle-semantics failure axes identified earlier remain only partially addressed: campus varies both, but with five sequences the honest treatment is the per-sequence description given in Chapter 6, not a fitted effect.

The fusion analysis across RELLIS, forest, and campus reveals a three-factor coupling: classifier quality, spatial coverage, and fusion trust. On RELLIS, fusion headroom is absent because the LiDAR baseline leaves almost nothing for the camera to add. On forest, headroom exists but is backbone- conditional: the confidence-weighted GA-Nav arm improves over the LiDAR baseline on forest, without outperforming the fixed blends; for WVN the rule choice barely matters: ordinal max-pooling and a camera-dominant fixed blend trade off rather than one dominating (Section 6.4). On campus, all three factors interact measurably: camera-dominant weighting ($\alpha = 0.7$, GA-Nav; applied only in cells where both modalities are valid, an overlap-region tie-break rather than a global camera/LiDAR mixing ratio) and flat BEV densification are both necessary and neither sufficient, together raising obstacle recall over LiDAR-only by roughly 1.6× (pooled across all 150 campus frames: obstacle recall 0.107037 → 0.169251) at a cost to overall accuracy. The gain is not uniform across sequences, and its costs are not confined to sequence 00003. Under point-label scoring, three of five sequences show a loss on at least one metric: sequence 00001 loses mIoU-4 by a small margin (0.0867 → 0.0858); sequence 00003 trades a substantial precision loss (0.688 → 0.466) for gains in recall (0.083 → 0.161) and mIoU-4 (0.064 → 0.075); sequence 00004 loses both precision (0.378 → 0.348) and mIoU-4 (0.265 → 0.244) while gaining recall; and sequence 00005 loses precision substantially (0.498 → 0.413) while gaining recall and mIoU-4. Only sequence 00002 improves on all three reported metrics simultaneously. A post-hoc, descriptive stratification of the five campus sequences by scene structure (structured: 00001, 00003; mixed: 00002; unstructured: 00004, 00005; $n = 2$ per endpoint stratum, no inferential test meaningful at this size and none run) shows a mixed picture rather than a monotonic one: B_FASTC mIoU is lower for structured sequences than unstructured (0.0796 → 0.2368, using the same pooled point-label reduction), and GA-Nav fixed $\alpha = 0.7$ mIoU follows the same direction (0.0827 structured vs. 0.2344 unstructured), but the unstructured group's own fusion delta is slightly negative rather than positive, the reverse of what a simple "campus is where fusion helps most" narrative would predict at the sequence level. Nor does obstacle-detection failure separate cleanly by structure: the worst obstacle recall of all five sequences occurs on unstructured sequence 00005 (0.0324 under point-label scoring). The 00003 precision cost, and the broader sequence-level mix of gains and losses, are therefore not attributable to a simple built-environment or training-domain-divergence effect. The campus Pareto front therefore generalises the RELLIS safety inversion: the precision–recall trade is

not merely observed after the fact, but exposed as a deployment-time choice, and it operates unevenly across the five recording sessions that make up this domain.

7.6 Future work

Ranked by the leverage the three primary fusion-analysis results suggest:

- **Domain-adapted or built-environment-inclusive training for the geometric method**, to test whether the campus collapse is recoverable and improve the camera-side signal available to fusion (discussed further in Chapter 8).
- **A learned, per-class reliability module** of the contextual-discounting form, trained on the degradation patterns this study characterised, evaluated against the fixed-blend baseline this study established as the bar to clear.
- **Test-time-augmentation confidence for the camera leg** (corruption-consistency variance as a non-softmax signal), already supported by the committed per-tier probability dumps; a cheap test of whether the one signal family not yet measured escapes the domain-dependence of the other two.
- **Additional built-environment capture** across sites and seasons, to convert the campus existence proof into a calibrated transfer characterisation.
- **Embedded benchmarking** of the conditional-camera deployment posture on target hardware.

7.7 Deployment implications

The three-regime characterisation of the obstacle override (Section 6.6) upgrades the deployment story from a single trade-off curve to a small decision procedure, though the campus regime's cost is now known to be substantially higher than earlier reported. Where the geometric baseline is known to be strong, in-domain and validated, fusion can be disabled at negligible cost, saving the camera inference budget entirely. Where it is weak (forest), the operating point on the measured recall-precision curve is a risk-tolerance choice with a real, moderate cost. Where the platform enters environments resembling the campus regime, built structure and vocabulary-alien obstacles, the override buys a resolved obstacle-recall gain ($+0.040$) at a resolved obstacle-IoU cost (-0.012) and a substantial precision cost (-0.163); it is not close to free on the obstacle class, and enabling it by default should be treated as a deliberate risk-tolerance decision rather than a default-safe posture. Because the override acts only within point-painted fusion-camera-leg coverage and leaves LiDAR-only behaviour untouched elsewhere, switching between these postures requires no retraining.

On a desktop-class GPU the recommended configuration runs at 87.4 ms per frame when the two sensing chains execute concurrently, or 11.4 Hz, marginally viable against a 10 Hz LiDAR, and only on that class of hardware. The obstacle-override safety layer adds 0.9 ms, under one percent of the pipeline: in computational terms the safety dial is free, and its only cost is the precision priced in Section 6.6. The camera backbone recommended on accuracy grounds in Chapter 6 is also substantially the cheaper of the

two: 87.4 ms against 117.4 ms concurrent, a 34 % difference driven almost entirely by the alternative's frozen self-supervised backbone, so no accuracy–cost trade arises here.

These standalone camera-only timing figures are stage-sum estimates for a flat-ground camera branch, not measurements of the deployable configuration evaluated for accuracy: the projection comparison in Chapter 5 shows zero obstacle detections for this flat-ground projection on both backbones on RELLIS-3D, whereas the committed forest analysis reports severely degraded full-support GA-Nav obstacle recall of 0.1437 rather than zero. A point-painted fusion-camera-leg latency was not measured under matched conditions; point-painting was timed only in the isolated V100 geometry benchmark and in the Thor fusion decomposition, which are different benchmark generations. Because point-painting exceeds the flat projection stage it replaces, a point-painted camera branch would be slower; no corrected standalone camera-only latency is claimed here.

The stage breakdown locates the bottleneck somewhere other than where it is usually assumed. Roughly half of the concurrent path is CPU-bound geometry rather than neural inference: LiDAR voxelisation alone costs 41.9 ms, more than double the network forward it feeds (18.3 ms), and point-painting a further 17.8 ms, against 26.9 ms for the entire camera network. Neither figure reflects model capacity; both are data-preparation steps that admit standard optimisation, vectorisation, GPU voxelisation, parallelised painting, without modifying any trained component. A second observation points the same way: WVN's trained head is 34 622 parameters against an 86.5-million-parameter frozen backbone, and costs 1.9 ms against 86.6 ms: over 97 % of that method's computational cost lies in a component never trained for this task, the same pattern seen in GA-Nav's MiT-b0 backbone above. The practical implication is that the accuracy findings of Chapter 6 and the cost of acting on them are largely independent: a deployment can pursue either without renegotiating the other.

The roughly 51% CPU-geometry share comes from the sole available concurrent-execution trace, a single-run generation superseded elsewhere by canonical V2-final per-stage medians. The provenance does not change the underlying comparison: canonical V2 voxelisation alone (40.7 ms) remains comparable in magnitude to GA-Nav's network forward (55.5 ms) regardless of the exact fraction.

Three constraints emerged during benchmarking that are worth reporting as findings rather than incidental difficulties. The three methods' dependency requirements are mutually incompatible: the camera segmentation stack pins an older framework and CUDA release, while the self-supervised backbone requires a newer Python and framework, so no single runtime can host all three and deployment is necessarily multi-process. One camera model retains synchronised batch normalisation from its distributed training configuration, which rejects CPU tensors outright and requires model surgery before it will run without a GPU. And the inter-process message-passing latency that a multi-process deployment necessarily pays could not be measured in the available environments, so the reported figures exclude it and should be read as lower bounds. Taken together, these indicate that the engineering distance between a working research pipeline and a deployable one lies less in the models than in their packaging, a cost rarely reported in comparative studies of this kind.

Embedded latency was measured on the Jetson AGX Thor (Section 5.5). These stage-sum values exclude ROS 2 inter-process communication and are lower bounds on end-to-end latency.

8

Conclusion

This thesis set out to answer a comparative question: how do learning-based camera, LiDAR, and fusion approaches compare for terrain traversability estimation, and under what conditions does each approach excel or fail? The literature offered these three paradigms largely in isolation, evaluated under incompatible protocols, with fusion generally presented as an improvement over its constituent modalities but rarely characterised in terms of *where*, *why*, or *by how much*. Answering the question therefore required first constructing a common evaluation footing, then, once the comparison produced a result more nuanced than a ranking, explaining it, and finally testing that explanation on a domain deliberately chosen to stress it.

8.1 Summary of contributions

The three contributions introduced in the Introduction, a calibrated camera-LiDAR rig, a custom-annotated terrain dataset, and a controlled cross-modal comparison built on both, stand as delivered. The six findings below are the substantive results that comparison produced: the specific answer to what its promised discussion of practical trade-offs actually found.

A unified cross-modal evaluation protocol. Three representative methods (WVN and GA-Nav for camera, FASTC for LiDAR, and a cost-level late fusion of the two) were evaluated as frozen, independently verifiable components on a single shared bird's-eye-view grid, with an identical class taxonomy, an identical scoring footprint, and a single pooled confusion matrix per condition, across four evaluation domains drawn from three datasets: RELLIS-3D, GOOSE, and the custom FKIE dataset, whose forest and campus recordings were collected in separate sessions with identical platform, sensor rig, and calibration. Within each domain, method identity was the only variable; the same checkpoints were used across domains, so cross-domain comparisons additionally reflect the effect of domain shift on a fixed method. This protocol is the precondition for every subsequent finding, and its construction surfaced and corrected several evaluation-geometry issues that would otherwise have silently distorted the comparison, including a scoring footprint that excluded over a third of valid ground truth on one dataset: on RELLIS-3D, where the previously used $[3, 8]$ m radial scoring band retained only 63.6 % of valid ground-truth cells overall and only 38 % of obstacle-tier GT, before being replaced by the common ± 10 m Path-A crop used throughout this thesis.

Separating coverage from competence: the fusion camera leg is narrow but high-recall within valid support. The headline account in this thesis of what the point-painted fusion camera leg contributes. Every committed camera-only metric elsewhere in this work is reported on full support, scoring every GT-valid cell and counting cells the camera cannot see as false negatives, a deployment-honest convention, since it counts unseen cells as failures rather than excluding them, but one that conflates how much of the scene the camera reaches with how well it performs where it does. Re-scoring the point-painted fusion camera leg on camera-valid support (the same confusion matrices with the no-prediction column dropped) separates the two, consistent with the dedicated RELLIS-3D projection comparison indicating that the shortfall is not attributable to the projection methods tested there. GA-Nav's camera-valid obstacle recall is 0.50–1.00 across all four evaluation domains, holding in the 0.87–0.98 band on every campus sequence regardless of scene structure, so the effect is neither a single-domain artifact nor a built-environment one. Within the fusion construction, the point-painted camera leg is therefore a narrow, high-sensitivity detector whose specificity is domain-dependent rather than uniform: GA-Nav precision within coverage ranges from 0.12 (campus) to 0.60 (RELLIS-3D), tracking distance from the training distribution in the same direction as the LiDAR baseline's own recall degradation, and is structurally identical between full-support and camera-valid scoring (dropping only no-prediction cells does not alter predicted-positive precision counts), so the domain-dependence is not explained by this particular coverage-convention difference. Its usable region is further shown to be a mid-range annulus of roughly 3–10 m rather than a uniform fraction of the scene, complementary to the LiDAR by range as well as by modality. This distinction is the mechanism behind the fusion precision–recall trade documented throughout Chapter 6: every rule that propagates camera evidence inherits both its sensitivity and its false positives. The self-selection caveat is stated explicitly where this result is developed: camera-valid support differs by backbone (on campus, per-sequence coverage ranges 17.90–19.45 % for GA-Nav and 11.74–13.20 % for WVN, not the same cells for either), so neither figure is a standalone system estimate, and full support remains the number that should govern deployment decisions.

A quantitative account of the fusion camera-leg contribution. Rather than reporting that fusion yields modest aggregate gains and stopping there, this thesis identifies the leading explanations. The fusion camera-leg contribution is bounded first by coverage (a geometric consequence of mounting height and pitch that limits its evaluable footprint to well below 25% of the scene on every domain tested, now confirmed directly by a camera-only projection comparison rather than inferred from the point-painted figures alone) and second by an oracle ceiling that tracks the LiDAR baseline's weakness. These two bounds explain the aggregate result. Coverage can be estimated in advance for a new dataset from sensor geometry alone, though this has not been validated on hardware beyond the rig used here; the oracle-ceiling pattern relating fusion benefit to LiDAR-baseline weakness, observed consistently across the three domains studied here, offers a diagnostic heuristic rather than an established predictive rule, testable from a baseline evaluation before any fusion rule is designed.

Fusion is a baseline-conditional complement, not a universal improvement. The fusion study finds an exhaustive null where the geometric baseline is strong, a statistically robust partial gain where it is weak, and, where it fails, the study's strongest result, resolved gains in obstacle recall and

obstacle IoU over the LiDAR baseline from a plain fixed blend, though not a strict Pareto improvement, since the corresponding mIoU and precision changes are not resolved from zero. Seventeen rules per camera backbone were evaluated, but no confidence-driven rule outperforms a fixed blend across every reported metric on any domain. The reliability analysis explains why. The available confidence signals are not trustworthy across domains. This is a supporting result about rule selection, not the thesis's central modality comparison.

A priced, baseline-conditional safety trade. The safety-relevant failure mode, winning on mean accuracy while missing more real obstacles, was addressed through a class-conditional obstacle override swept on the three primary fusion-analysis domains (RELLIS, forest, and campus). The recall–precision trade exhibits three regimes in the tested override sweep, inert where the baseline is strong, genuinely costly where it is weak, and, where the baseline fails (campus), a substantial rather than nearly-free trade, obstacle recall improves by $+0.064$ at a cost of -0.023 obstacle IoU and -0.223 precision, relative to the fixed blend. In every regime it is characterised as a single-parameter curve on which a deployment selects its own operating point, with LiDAR-only behaviour preserved outside camera coverage.

Evidence that the geometric modality is not the transfer-robust one. Across the RELLIS, forest, and campus analysis domains, the same LiDAR checkpoint yields obstacle recall 0.945, 0.417, and 0.107, misclassifying 84 % of annotated obstacle cells in the built environment as traversable-at-cost even where direct sensor evidence exists; this is an existence proof from a single site and season, not a calibrated estimate of how badly the failure generalises to built environments broadly. Its failure mode, severity compression under domain shift, is the same one the point-painted fusion camera leg exhibits for objects outside its training vocabulary. Neither modality is robust; they are differently fragile, which is both the reason fusion has something to offer and the reason its benefit cannot be assumed without measuring the baseline first.

8.2 Answering the research questions

This section answers the thesis's research questions in turn, first comparing the modalities and identifying where fusion helps, then examining what fusion costs.

8.2.1 Comparing the modalities and where fusion helps

Under a unified protocol, which method performs best depends on the evaluation domain rather than any fixed ranking of the modalities. The LiDAR method leads on aggregate quality where its training distribution holds, and the reason is coverage rather than per-prediction accuracy: it predicts almost everywhere, while the point-painted fusion camera leg predicts on a narrow, geometrically determined near-to-mid-field band. Off its training distribution the same method degrades, and in a built environment it fails on the obstacle class outright, at which point the point-painted fusion camera leg, despite covering under a tenth of the scene and lacking vocabulary for urban objects, becomes the better obstacle detector: a simple fixed blend delivers resolved gains in obstacle recall and obstacle IoU over the geometric baseline, though not an improvement in every reported metric, since the corresponding mIoU and precision effects are not resolved from zero.

Each approach fails differently. The LiDAR method fails where scene geometry and semantics diverge from its training terrain, compressing obstacle severity rather than producing visibly wrong structure. The point-painted fusion camera leg faces two distinct limitations. Coverage is lowest in the near field and peaks at mid-range (3–5 m), because a forward-facing camera looks out rather than down: point-painted coverage is small but non-zero (5.46 % on campus and 6.4 % on forest) within 0–3 m. Obstacle recall is high within the near-field cells the camera does cover, so this is a coverage limitation rather than a competence one. The second limitation is semantic: systematic misclassification of out-of-vocabulary objects onto the nearest available class. Fusion inherits the point-painted camera leg's coverage limit exactly, since the point-painting mechanism connecting the modalities is gated on LiDAR returns falling within the camera frustum, and its benefit tracks an oracle ceiling relating to the geometric baseline's weakness, a pattern observed across this study's domains rather than an established predictive rule.

8.2.2 What fusion costs

The third question asked what this fusion costs, in calibration effort, computational load, and implementation complexity, and how those costs weigh against the documented accuracy gains. The costs are real and only partially quantified. Calibration selection was qualitative rather than measured: `ros2_calib` was chosen for its ROS 2 integration and joint optimisation over per-pair MATLAB fitting, but the two successful methods disagree by 12.2 cm in translation and 2.3° in rotation, and this study did not measure how that residual uncertainty propagates into fusion accuracy. Computational cost is substantial in isolation. The fused pipeline's estimated sequential latency of 108.321 ms (9.23 Hz) is markedly slower than any single-method pipeline (26.20–62.47 Hz under TensorRT), though parallel execution recovers some of this gap (79.467 ms, 12.58 Hz; both estimates exclude unmeasured inter-process communication). Implementation complexity is likewise real: the three methods' mutually incompatible dependencies require a multi-process deployment under the current implementation, and one backbone required model surgery to run without a GPU. Weighed against this, the accuracy gain is neither free nor universal. Where the LiDAR baseline is strong (RELLIS-3D), fusion earns none of this cost, a null result across the tested rule sweep. Where it is weak (campus), the gain is real and resolved on obstacle recall (+0.062) and obstacle IoU (+0.041), metrics most directly tied to safety, though not on mIoU or precision. The deployment architecture partially resolves this tension: because fusion's posture is selectable at runtime without retraining, its computational and implementation cost can be limited to domains where the geometric baseline is known to be weak, provided that baseline strength is established in advance, and largely avoided where it is strong (Section 7.7). The calibration cost, by contrast, is paid once regardless of deployment posture, and its downstream effect on accuracy remains an open question this thesis does not resolve.

8.3 Reflection on the original deliverables

This project set out to implement a baseline fixed-weight late fusion and, as an optional extension, a confidence-aware variant adapting weights by per-modality reliability. Both were implemented, and the confidence-aware variant was extended well beyond the original scope: into a full rule sweep, direct

calibration measurement of every available confidence signal across the three primary fusion-analysis domains (RELLIS, forest, and campus), and a mechanistic account of their failure modes. The outcome inverts the implicit expectation behind the original deliverable in two ways. The confidence-aware variant does not dominate the fixed baseline on any domain: it exceeds the fixed blend on individual metrics only on campus, and only at a substantial precision cost, so it is a trade rather than a displacement; and the fixed baseline itself performed better than expected, delivering resolved gains in obstacle recall and obstacle IoU over LiDAR-only where the geometric method fails, though not a strict improvement across every metric.

The transferable lesson is about the assumption rather than the mechanism. Confidence-aware fusion is proposed on the intuition that a model's own uncertainty estimate is a usable reliability signal. Measured directly on real data across the three primary fusion-analysis domains, that intuition holds only within the distribution each signal was fitted to, which is precisely the regime in which fusion has least to offer, since the modality producing the confident signal is already performing well there. The signals are least trustworthy exactly where they would be most useful.

8.4 Outlook

Two directions follow most directly. The first is to address the geometric method's transfer failure at its source. It is a training-data problem before it is an architecture problem. A variant trained to include built-environment obstacles would test whether the collapse documented here is recoverable. If so, fusion would combine with a substantially stronger camera-side signal rather than continuing to compensate for a persistently weak one. The second is to stop repurposing task-head uncertainty as a reliability estimate and to learn reliability directly, per class and per modality, trained on the degradation patterns this study characterises and evaluated against the fixed-blend baseline this study establishes as surprisingly hard to beat.

More broadly, the methodological point may outlast the specific numbers. Fusion results reported as aggregate deltas are difficult to interpret and easy to over-claim. Reporting them alongside a coverage figure and an oracle ceiling (both cheap, and both computable before any fusion rule is designed) makes clear whether an observed improvement represents most of what was available or a small fraction of it, and whether a null result reflects a poor rule or an absent opportunity. Reporting them alongside a per-domain baseline characterisation makes clear something further: whether the modality being fused into is working at all.

A

Full fusion rule sweep results

This appendix tabulates the complete fusion rule sweep summarised in Section 6.4 (17 rules × 2 camera backbones × 3 datasets: RELLIS-3D and forest in Tables A.1–A.4, and campus (initially evaluated on a subset and subsequently extended to the same full 17-arm sweep) in Tables A.9–A.10) and the complete τ-override sweep summarised in Section 6.6 (7 thresholds × 2 camera backbones × 2 datasets, Tables A.5–A.8). All rules are anchored to the LiDAR-only baseline (FASTC-B), reproduced in every table for reference. Rules are defined as follows: *fixed* α blends fused cost as $T = \alpha\, t_{\text{cam}} + (1 - \alpha)\, t_{\text{lidar}}$; *confw* weights each leg by its own confidence signal; *ordinal max* takes the more pessimistic tier class of the two legs; *gate* τ applies ordinal max only where camera confidence $\geq \tau$, else falls back to LiDAR-only; *sharp* γ raises each leg's confidence weight to the power γ before confidence weighting. The correctness gates for all tables (bit-exact reproduction of the $\alpha = 0.5$, confidence-weighted, and ordinal-max arms against the independently committed evaluation pipeline) passed prior to acceptance of any result in this appendix.

A.1 Fusion rule sweep (forest and RELLIS-3D)

A.2 Obstacle-override threshold sweep

Each table also reports the *override-changed fraction*: the proportion of GT-valid cells whose fused decision was altered by the τ-override relative to the base fixed blend. This is distinct from the override *firing* on a cell that already agreed with the base decision.

A.3 Campus third-domain results

Campus was initially evaluated on a claim-bearing subset of the rule space rather than the full 17-arm sweep, since the rule search was closed on the first two domains and campus was intended only to test generalisation of its outcome. It was subsequently extended to the same full 17-arm sweep used on forest and RELLIS-3D, under campus fusion-rule semantics matched across both camera backbones, pooled over all 150 frames. This extension is what surfaced the campus confidence-gated and ordinal results reported in Section 6.4: on full support, GA-Nav's confidence-gated max-pooling at $\tau = 0.7$ and ordinal max-pooling each exceed their respective best fixed blend on obstacle recall (and, for the gated arm, mIoU), at a substantial precision cost. The FASTC-B baseline for campus (full support, pooled 150

Table A.1: Forest, GA-Nav camera leg: full 17-arm sweep.

Arm	mIoU	Obstacle IoU	Recall	Precision
FASTC-B	0.156725	0.263165	0.417065	0.416287
fixed 0.1	0.156734	0.263201	0.417520	0.415924
fixed 0.2	0.156734	0.263201	0.417520	0.415924
fixed 0.3	0.148427	0.263201	0.417520	0.415924
fixed 0.4	0.148130	0.263201	0.417520	0.415924
fixed 0.5	0.148042	0.263217	0.417520	0.415965
fixed 0.6	0.146296	0.271453	0.477389	0.386228
fixed 0.7	0.145784	0.270483	0.477941	0.383908
fixed 0.8	0.142291	0.228071	0.501235	0.295027
fixed 0.9	0.142125	0.223251	0.501837	0.286813
confw	0.147228	0.267338	0.483501	0.374204
ordinal max	0.141320	0.220898	0.502666	0.282678
gate 0.5	0.142035	0.222326	0.502227	0.285163
gate 0.7	0.144370	0.227560	0.500699	0.294356
gate 0.9	0.150283	0.242222	0.490052	0.323851
sharp 1	0.147787	0.271093	0.465636	0.393519
sharp 2	0.147951	0.271638	0.475731	0.387696
sharp 4	0.148076	0.268436	0.482493	0.376972

frames) is reproduced in Table 5.6 and Table 5.7; it is not repeated per-row here. Every value below uses full support (all GT-valid cells).

Table A.2: Forest, WVN camera leg: full 17-arm sweep.

Arm	mIoU	Obstacle IoU	Recall	Precision
FASTC-B	0.156725	0.263165	0.417065	0.416287
fixed 0.1	0.156805	0.263112	0.417032	0.416187
fixed 0.2	0.156805	0.263112	0.417032	0.416187
fixed 0.3	0.158214	0.266659	0.416187	0.426014
fixed 0.4	0.158843	0.267358	0.415553	0.428473
fixed 0.5	0.159222	0.267737	0.414562	0.430510
fixed 0.6	0.162614	0.274160	0.425323	0.435474
fixed 0.7	0.163090	0.275669	0.428102	0.436366
fixed 0.8	0.164153	0.277634	0.433678	0.435540
fixed 0.9	0.164014	0.278086	0.437010	0.433326
confw	0.157337	0.264163	0.416512	0.419348
ordinal max	0.157172	0.271680	0.450080	0.406674
gate 0.5	0.156725	0.263211	0.417146	0.416321
gate 0.7	0.156725	0.263165	0.417065	0.416287
gate 0.9	0.156725	0.263165	0.417065	0.416287
sharp 1	0.157788	0.265084	0.416431	0.421758
sharp 2	0.156930	0.263594	0.416837	0.417591
sharp 4	0.156858	0.263404	0.416935	0.417016

Table A.3: RELLIS-3D, GA-Nav camera leg: full 17-arm sweep. No arm moves any metric by more than 0.25 pp from FASTC-B, consistent with the camera-contribution ceiling of Section 6.3.

Arm	mIoU	Obstacle IoU	Recall	Precision
FASTC-B	0.492280	0.602313	0.944876	0.624248
fixed 0.1	0.492280	0.602313	0.944876	0.624248
fixed 0.2	0.492280	0.602313	0.944876	0.624248
fixed 0.3	0.492689	0.603773	0.944341	0.626052
fixed 0.4	0.492802	0.604203	0.944262	0.626549
fixed 0.5	0.493066	0.605072	0.943859	0.627662
fixed 0.6	0.494322	0.605154	0.943678	0.627830
fixed 0.7	0.494426	0.605110	0.943550	0.627839
fixed 0.8	0.494676	0.605242	0.943770	0.627884
fixed 0.9	0.494622	0.605098	0.943626	0.627793
confw	0.494199	0.604864	0.944329	0.627230
ordinal max	0.491507	0.601306	0.945418	0.622931
gate 0.5	0.491670	0.601419	0.945417	0.623053
gate 0.7	0.492048	0.601842	0.945410	0.623510
gate 0.9	0.492298	0.602385	0.945399	0.624097
sharp 1	0.493940	0.604900	0.944058	0.627388
sharp 2	0.494006	0.604756	0.944200	0.627172
sharp 4	0.494114	0.604769	0.944386	0.627103

Table A.4: RELLIS-3D, WVN camera leg: full 17-arm sweep. As with GA-Nav, no arm meaningfully departs from FASTC-B.

Arm	mIoU	Obstacle IoU	Recall	Precision
FASTC-B	0.492280	0.602313	0.944876	0.624248
fixed 0.1	0.492280	0.602313	0.944876	0.624248
fixed 0.2	0.492280	0.602313	0.944876	0.624248
fixed 0.3	0.492361	0.602619	0.944652	0.624675
fixed 0.4	0.492376	0.602696	0.944599	0.624781
fixed 0.5	0.492400	0.602825	0.944526	0.624952
fixed 0.6	0.492794	0.602817	0.944470	0.624967
fixed 0.7	0.492804	0.602751	0.944402	0.624927
fixed 0.8	0.492892	0.602904	0.944704	0.624959
fixed 0.9	0.492848	0.602833	0.944666	0.624899
confw	0.492562	0.602620	0.944816	0.624605
ordinal max	0.491787	0.602038	0.945294	0.623770
gate 0.5	0.491909	0.602083	0.944954	0.623967
gate 0.7	0.492280	0.602313	0.944876	0.624248
gate 0.9	0.492280	0.602313	0.944876	0.624248
sharp 1	0.492453	0.602755	0.944620	0.624835
sharp 2	0.492449	0.602680	0.944682	0.624728
sharp 4	0.492440	0.602580	0.944800	0.624568

Table A.5: Forest, GA-Nav camera leg: τ-override sweep, anchored to the base fixed blend ($\alpha = 0.6$).

Arm	mIoU	Obstacle IoU	Recall	Precision
FASTC-B	0.156725	0.263165	0.417065	0.416287
base ($\alpha = 0.6$)	0.146296	0.271453	0.477389	0.386228
$\tau = 0.3$	0.141232	0.220042	0.502666	0.281279
$\tau = 0.4$	0.141272	0.220496	0.502666	0.282021
$\tau = 0.5$	0.141336	0.221031	0.502650	0.282902
$\tau = 0.6$	0.141415	0.221730	0.502585	0.284068
$\tau = 0.7$	0.141591	0.222767	0.502292	0.285868
$\tau = 0.8$	0.142089	0.224657	0.501495	0.289252
$\tau = 0.9$	0.142773	0.229158	0.500439	0.297128

Table A.6: Forest, WVN camera leg: τ-override sweep, anchored to the base fixed blend ($\alpha = 0.9$). The override rate collapses toward zero as τ rises, the decision-level signature of the compressed reconstruction confidence discussed in Section 6.5.

Arm	mIoU	Obstacle IoU	Recall	Precision
FASTC-B	0.156725	0.263165	0.417065	0.416287
base ($\alpha = 0.9$)	0.164014	0.278086	0.437010	0.433326
$\tau = 0.3$	0.160865	0.275464	0.449072	0.416073
$\tau = 0.4$	0.161834	0.276321	0.446130	0.420613
$\tau = 0.5$	0.162608	0.276779	0.442879	0.424622
$\tau = 0.6$	0.163253	0.277052	0.440050	0.427905
$\tau = 0.7$	0.163808	0.277759	0.438229	0.431344
$\tau = 0.8$	0.164006	0.278022	0.437108	0.433074
$\tau = 0.9$	0.164014	0.278086	0.437010	0.433326

Table A.7: RELLIS-3D, GA-Nav camera leg: τ-override sweep. The override changes at most 0.04 % of cells at any threshold, quantifying the camera-contribution ceiling (Section 6.3) at the level of individual fusion decisions.

Arm	mIoU	Obstacle IoU	Recall	Precision	Changed fraction
FASTC-B	0.492280	0.602313	0.944876	0.624248	—
base ($\alpha = 0.6$)	0.494322	0.605154	0.943678	0.627830	0.000000
$\tau = 0.3$	0.493620	0.603374	0.944308	0.625638	0.000409
$\tau = 0.4$	0.493885	0.604070	0.944279	0.626399	0.000286
$\tau = 0.5$	0.494077	0.604574	0.944219	0.626967	0.000191
$\tau = 0.6$	0.494205	0.604905	0.944131	0.627362	0.000120
$\tau = 0.7$	0.494299	0.605134	0.944101	0.627621	0.0000766
$\tau = 0.8$	0.494381	0.605331	0.944082	0.627841	0.0000403
$\tau = 0.9$	0.494406	0.605389	0.944053	0.627916	0.0000255

Table A.8: RELLIS-3D, WVN camera leg: τ-override sweep. As with GA-Nav, the override changes a negligible fraction of decisions.

Arm	mIoU	Obstacle IoU	Recall	Precision	Changed fraction
FASTC-B	0.492280	0.602313	0.944876	0.624248	—
base ($\alpha = 0.9$)	0.492848	0.602833	0.944666	0.624899	0.000000
$\tau = 0.3$	0.492641	0.602291	0.945001	0.624170	0.000150
$\tau = 0.4$	0.492754	0.602585	0.944954	0.624507	0.0000920
$\tau = 0.5$	0.492831	0.602785	0.944895	0.624747	0.0000480
$\tau = 0.6$	0.492859	0.602861	0.944812	0.624865	0.0000206
$\tau = 0.7$	0.492851	0.602841	0.944704	0.624891	0.0000053
$\tau = 0.8$	0.492848	0.602835	0.944670	0.624900	0.0000003
$\tau = 0.9$	0.492848	0.602833	0.944666	0.624899	0.000000

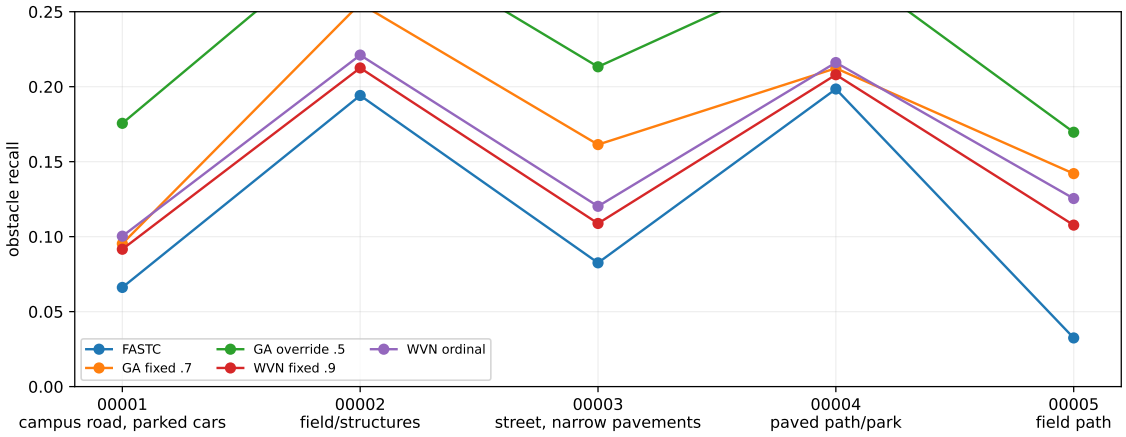


Figure A.1: Obstacle recall per campus sequence for the LiDAR baseline and the principal fusion arms. The fusion arms do not improve on the baseline in every sequence: three of five sequences show a loss on at least one reported metric even where recall improves, and the structured/ unstructured split does not predict which sequences gain or lose (see the sequence-level discussion in Chapter 7). Scene labels provisional.

Table A.9: Campus, GA-Nav camera leg: full 17-arm sweep, pooled results over 150 frames, matched campus fusion-rule semantics, full support. The historical `campus_eval/g5` record is retained separately for provenance and differs materially from this table only for the confidence-weighted arm.

Arm	mIoU (4-class)	Obst. IoU	Recall	Precision
fixed 0.1	0.174561	0.090640	0.107430	0.367076
fixed 0.2	0.174426	0.090640	0.107430	0.367076
fixed 0.3	0.168098	0.090792	0.107386	0.370098
fixed 0.4	0.166761	0.090520	0.106879	0.371620
fixed 0.5	0.165955	0.090182	0.106329	0.372592
fixed 0.6	0.176065	0.129366	0.162041	0.390819
fixed 0.7	0.176092	0.133253	0.169251	0.385191
fixed 0.8	0.176362	0.117032	0.200606	0.219308
fixed 0.9	0.176664	0.111112	0.212212	0.189118
confw	0.180433	0.126411	0.163631	0.357219
ordinal max	0.176984	0.105006	0.227741	0.163071
gate 0.5	0.181634	0.114214	0.223966	0.189017
gate 0.7	0.183600	0.120754	0.208917	0.222483
gate 0.9	0.180514	0.111659	0.159611	0.270955
sharp 1	0.174421	0.111771	0.136252	0.383505
sharp 2	0.177178	0.117897	0.145935	0.380283
sharp 4	0.180168	0.124295	0.158038	0.367942

Table A.10: Campus, WVN camera leg: full 17-arm sweep, pooled results over 150 frames, matched campus fusion-rule semantics, full support. The historical `campus_eval/g5` record is retained separately for provenance and differs materially from this table only for the confidence-weighted arm. Gate $\tau = 0.7$ and $\tau = 0.9$ are marked inert†: WVN's campus confidence never exceeds 0.6079 across the 196,104 valid camera cells, so neither threshold admits any camera evidence and both arms collapse exactly to the LiDAR-only baseline; their identical values below are that collapse, not two independently tested rules.

Arm	mIoU (4-class)	Obst. IoU	Recall	Precision
fixed 0.1	0.174655	0.091677	0.106871	0.392043
fixed 0.2	0.174658	0.091677	0.106871	0.392043
fixed 0.3	0.174224	0.090583	0.105490	0.390622
fixed 0.4	0.174086	0.090227	0.105018	0.390467
fixed 0.5	0.174253	0.090193	0.104782	0.393128
fixed 0.6	0.179973	0.102006	0.119062	0.415911
fixed 0.7	0.181036	0.105748	0.123694	0.421589
fixed 0.8	0.183268	0.113844	0.133910	0.431731
fixed 0.9	0.184029	0.116654	0.137545	0.434407
confw	0.174542	0.090652	0.105490	0.391903
ordinal max	0.182933	0.124696	0.148460	0.437891
gate 0.5	0.174380	0.092655	0.108129	0.393006
gate 0.7†	0.174139	0.091814	0.107037	0.392313
gate 0.9†	0.174139	0.091814	0.107037	0.392313
sharp 1	0.175053	0.091687	0.106801	0.393173
sharp 2	0.175096	0.091698	0.106836	0.392897
sharp 4	0.175116	0.091735	0.106888	0.392863

†Inert: collapses exactly to the LiDAR-only (FASTC-B) baseline; not an independently tested rule.

Table A.11: Campus, Condition B (FASTC) per sequence. Sequence labels are provisional pending confirmation. The baseline fails on the obstacle class in every sequence, but not uniformly or by scene type: sequence 00001 is broadly weak across tiers, sequence 00005 is specifically obstacle-blind despite strong low-tier performance, and sequence 00003 combines a medium-tier collapse with conservative but comparatively precise obstacle firing (see Chapter 6 for the full analysis).

Seq	Scene (provisional)	mIoU	Obst. IoU	Recall	Precision
00001	campus road, parked cars	0.086684	0.053218	0.066129	0.214201
00002	field/structures	0.205411	0.150301	0.194145	0.399599
00003	street, narrow pavements	0.063543	0.079556	0.082531	0.688249
00004	paved path/park	0.264680	0.149635	0.198452	0.378227
00005	field path	0.215253	0.031418	0.032448	0.497521

Table A.12: Campus stability analysis: paired differences against the LiDAR-only baseline, 10,000-resample paired bootstrap (frame level, $N = 150$) with leave-one-sequence-out ranges ($N = 5$). Only comparisons carrying claims in the main text were tested.

Comparison	Metric	Point	Frame 95 % CI	Seq. LOO range
GA override 0.5 − B[†]	recall	+0.118878	[+0.098322, +0.140520]	[+0.113706, +0.122916]
	obst. IoU	+0.014629	[−0.001439, +0.031583]	[+0.004725, +0.023438]
	mIoU	+0.005794	[+0.001143, +0.010619]	[+0.004141, +0.008012]
	precision	−0.224759	[−0.320743, −0.142275]	[−0.269258, −0.176462]
GA fixed 0.7 − B	recall	+0.062214	[+0.044912, +0.082107]	[+0.053048, +0.071235]
	obst. IoU	+0.041439	[+0.027799, +0.057258]	[+0.034041, +0.047308]
	mIoU	+0.001953	[−0.003431, +0.007658]	[−0.000534, +0.004902]
	precision	−0.007122	[−0.063560, +0.038531]	[−0.032989, +0.013039]
GA confw − B	recall	+0.023552	[+0.007134, +0.041795]	[+0.016355, +0.029276]
	obst. IoU	+0.013952	[+0.000467, +0.029105]	[+0.008527, +0.018195]
	mIoU	+0.007846	[+0.002071, +0.013690]	[+0.006089, +0.009296]
GA override 0.5 − fixed 0.6[†]	recall	+0.063874	[+0.050336, +0.078561]	[+0.058408, +0.069695]
	obst. IoU	−0.022923	[−0.035818, −0.009837]	[−0.030694, −0.018651]
	precision	−0.223265	[−0.282499, −0.167070]	[−0.242847, −0.192757]
WVN ordinal − B	recall	+0.041423	[+0.030525, +0.054629]	[+0.031447, +0.045192]
	obst. IoU	+0.032882	[+0.023672, +0.043956]	[+0.024560, +0.037046]
	mIoU	+0.008794	[+0.006198, +0.011889]	[+0.006950, +0.009575]
WVN fixed 0.9 − B	recall	+0.030508	[+0.021408, +0.041450]	[+0.021851, +0.033671]
	obst. IoU	+0.024840	[+0.017015, +0.034242]	[+0.017483, +0.028292]
	mIoU	+0.009758	[+0.007043, +0.012824]	[+0.008510, +0.011022]
WVN ordinal − fixed 0.9	recall	+0.010915	[+0.008027, +0.014162]	[+0.009596, +0.011521]
	obst. IoU	+0.008042	[+0.005727, +0.010674]	[+0.007078, +0.008754]
	mIoU	−0.001096	[−0.002711, +0.000379]	[−0.001703, +0.000485]

[†]The obstacle-IoU effect of GA override 0.5 depends on the comparison baseline. Versus FASTC directly, it is unresolved ([−0.001439, +0.031583] crosses zero); versus fixed $\alpha = 0.6$, it is resolved and negative (−0.022923, [−0.035818, −0.009837]). Readers should not treat either interval as representative of the other.

B

Deployment benchmark details

This appendix reports detailed deployment benchmark results that support the summary tables in Chapter 5.5. The main text reports the matched PyTorch–TensorRT comparison as the final deployment result. The standalone TensorRT worker benchmarks are retained here as supplementary diagnostics because they show the detailed stage breakdown of the TensorRT-only execution paths.

B.1 FASTC ONNX export compatibility adaptations

The original FASTC implementation could not initially be compiled directly by TensorRT because several PyTorch operations exported to ONNX operators that were ambiguous under dynamic input shapes or unsupported by the TensorRT ONNX parser. Two source-level adaptations were introduced. These changes preserve the trained parameters and mathematical behaviour of the network while improving ONNX and TensorRT compatibility.

Explicit squeeze dimension The PointPillars feature encoder originally used an unspecified squeeze operation:

```
features.squeeze()
```

With a dynamic number of input voxels, this exported to an ONNX `Squeeze` operator for which TensorRT could not reliably infer the dimension to remove. Since only the singleton point dimension had to be removed, the operation was changed to

```
features.squeeze(1)
```

This preserves the intended tensor transformation while making the exported graph unambiguous.

Replacement of sequence-based channel splitting The RegSeg backbone contains dilated convolution blocks that split the channel dimension into equally sized groups. The original implementation used

```
torch.tensor_split(x, self.num_splits, dim=1)
```

During ONNX export, this operation generated sequence operators such as `SplitToSequence` and `SequenceAt`, which were not supported by the TensorRT parser on the target platform. Since the implementation already ensures that the channel count is divisible by the number of splits, the split width is

$$C_{\text{group}} = \frac{C}{N_{\text{split}}}. \tag{B.1}$$

The sequence operation was replaced with explicit slicing:

```
start = i * self.split_channels
end = start + self.split_channels
x_part = x[:, start:end, :, :]
```

This exports as standard ONNX `Slice` operations and produces the same channel partitions as the original implementation.

B.2 GA-Nav TensorRT pipeline benchmark

B.3 WVN TensorRT pipeline benchmark

B.4 FASTC TensorRT pipeline benchmark

B.5 TensorRT parity validation details

FASTC produces five output classes, where class 4 denotes the established unknown/no-signal output. In the thesis post-processing path, this class is mapped to invalid cost and zero confidence rather than to a traversability tier. Therefore, the four-tier FASTC agreement is computed only over cells where the PyTorch output belongs to classes 0–3. Expected-cost parity is not reported for FASTC because the deployment path uses hard argmax class-to-cost mapping rather than probability-weighted expected cost.

B.6 Network-only and image-plane TensorRT benchmarks

These partial benchmarks are retained to show the effect of TensorRT on the exported neural-network boundaries. They are not used as the main deployment comparison because each method has a different TensorRT boundary. The main deployment comparison therefore uses the matched PyTorch–TensorRT pipeline results reported in Section 5.5.4.

B.7 Desktop GPU memory and model-size baseline

B.8 Fusion-stage timing details

This section records the shared geometry and fusion-stage timings from the earlier desktop/HPC computational-cost benchmark. These values are retained for traceability, but the corrected Thor fusion estimate in Section 5.5.6 uses the separately measured Thor point-painting latency.

Table B.1: Detailed staged GA-Nav TensorRT pipeline latency on the Jetson AGX Thor over 30 selected ROVO images. The benchmark uses 10 warm-up passes and 50 timed passes over all frames, resulting in 1500 timed samples. Disk input is reported separately and excluded from total compute latency.

Stage	Median [ms]	IQR [ms]	Min [ms]	Max [ms]
Input load	10.972	0.208	9.838	11.881
Preprocessing	10.845	6.599	3.636	38.675
TensorRT inference	11.081	2.981	9.263	30.848
Post-processing	1.932	3.242	1.030	15.657
BEV output processing	8.372	9.007	1.954	23.863
Total compute	35.735	11.865	16.663	64.987
Total end-to-end	46.542	11.882	26.674	75.782

Table B.2: GA-Nav TensorRT deployment memory and model-size summary on the Jetson AGX Thor. The memory values are process-visible measurements from the benchmark process.

Metric	Value
GPU baseline memory	24.92 MB
GPU peak allocated memory	71.76 MB
GPU peak reserved memory	77.59 MB
Host RAM delta after model and engine load	427.98 MB
Checkpoint size	49.90 MB
ONNX model size	24.81 MB
TensorRT engine size	16.92 MB
Total parameters	6,215,468
Trainable parameters	6,215,468

Table B.3: Detailed staged WVN TensorRT pipeline latency on the Jetson AGX Thor over 30 selected ROVO images. The benchmark uses 10 warm-up passes and 50 timed passes over all frames, resulting in 1500 timed samples. Disk input is reported separately and excluded from total compute latency.

Stage	Median [ms]	IQR [ms]	Min [ms]	Max [ms]
Input load	10.985	0.193	9.757	11.640
Preprocessing	2.235	5.337	0.944	25.887
TensorRT STEGO	3.266	2.050	2.721	18.668
WVN head	2.439	1.061	2.174	23.161
Confidence/cost	0.745	0.732	0.599	14.250
Combined post-processing	3.480	2.461	2.798	24.851
BEV output processing	5.293	7.249	1.859	30.442
Total compute	19.818	10.081	8.503	54.595
Total end-to-end	30.664	10.116	18.869	65.412

Table B.4: WVN TensorRT deployment memory and model-size summary on the Jetson AGX Thor. The STEGO component is loaded through the TensorRT engine in this deployment path; therefore, PyTorch parameter counts are reported for the WVN head only.

Metric	Value
GPU baseline memory	28.58 MB
GPU peak allocated memory	152.11 MB
GPU peak reserved memory	171.97 MB
Host RAM delta after model and engine load	201.53 MB
STEGO checkpoint size	346.24 MB
WVN head checkpoint size	0.14 MB
ONNX model size	346.23 MB
TensorRT engine size	175.81 MB
WVN head total parameters	34,622
WVN head trainable parameters	34,622
STEGO PyTorch parameters	Not measured in this deployment path

Table B.5: Detailed staged FASTC TensorRT pipeline latency on the Jetson AGX Thor over 30 selected ROVO LiDAR frames. The benchmark uses 10 warm-up passes and 50 timed passes over all frames, resulting in 1500 timed samples. Disk input is reported separately and excluded from total compute latency.

Stage	Median [ms]	IQR [ms]	Min [ms]	Max [ms]
Input load	0.834	0.134	0.343	1.007
Voxelisation/preprocessing	5.969	0.324	5.445	9.215
TensorRT inference	5.903	0.964	4.783	10.561
Post-processing	4.464	0.129	4.260	6.333
Native BEV output packaging	0.007	0.000	0.006	0.017
Total compute	16.515	1.238	14.770	21.744
Total end-to-end	17.336	1.225	15.540	22.657

Table B.6: FASTC TensorRT deployment memory and model-size summary on the Jetson AGX Thor. The process-visible memory includes the FASTC voxelisation and sparse-processing stack and should not be interpreted as pure neural-network memory.

Metric	Value
GPU baseline memory	571.65 MB
GPU peak allocated memory	590.43 MB
GPU peak reserved memory	610.27 MB
Host RAM delta after model and engine load	887.28 MB
Checkpoint size	27.34 MB
ONNX model size	147.87 MB
TensorRT engine size	76.82 MB
Total parameters	3,371,321
Trainable parameters	3,371,321

Table B.7: Detailed network-level parity between the PyTorch and TensorRT FASTC paths over all 30 selected ROVO LiDAR frames. The comparison is performed on pre-voxelised input tensors and excludes voxelisation, post-processing, and BEV projection.

Metric	Min	Mean	Median	Max
Raw 5-class argmax agreement (%)	99.7108	99.8184	99.8028	99.9371
Thesis-valid 4-tier agreement (%)	99.0483	99.5687	99.5920	99.8686
Logits MAE	0.004739	0.006380	0.006276	0.009057
Softmax MAE	0.000398	0.000635	0.000620	0.000925

Table B.8: Detailed real-image parity between the PyTorch and TensorRT WVN STEGO feature-extraction paths over all 30 selected ROVO frames. The comparison is performed after passing both feature tensors through the unchanged WVN head.

Metric	Min	Mean	Median	Max
Argmax class agreement (%)	99.8047	99.9270	99.9283	99.9701
Expected-cost MAE	0.000131	0.000259	0.000243	0.000753
Softmax MAE	0.000191	0.000357	0.000339	0.000855

Table B.9: Detailed real-image parity between the PyTorch and TensorRT GA-Nav image-logits paths over all 30 selected ROVO frames. The comparison includes downstream softmax, expected-cost, entropy-confidence, and argmax computations.

Metric	Min	Mean	Median	Max
Argmax class agreement (%)	99.9840	99.9975	100.0000	100.0000
Expected-cost MAE	0.00000121	0.00002895	0.00002607	0.00011184
Softmax MAE	0.00000129	0.00003128	0.00003046	0.00009123
Entropy-confidence MAE	0.00001179	0.00011145	0.00012437	0.00022353
Logits MAE	0.00141486	0.00189897	0.00177668	0.00398264

Table B.10: Network-only performance of the FASTC TensorRT FP16 engine on the Jetson AGX Thor. The benchmark uses pre-voxelised input tensors and excludes point-cloud loading, voxelisation, post-processing, and output packaging.

Metric	Value
TensorRT precision	FP16
Mean host latency	4.63 ms
Median host latency	4.59 ms
95th-percentile host latency	4.84 ms
Mean GPU compute time	4.55 ms
Throughput	$215.8\,\mathrm{s}^{-1}$

Table B.11: Runtime of the TensorRT-backed WVN image-plane path on the Jetson AGX Thor. The benchmark uses 30 real ROVO images, excludes disk input, and excludes the final image-to-BEV projection.

Stage	Mean [ms]	Median [ms]	P95 [ms]
Preprocessing	0.933	0.832	1.602
TensorRT STEGO feature extraction	2.904	2.729	3.616
WVN head	2.407	2.302	2.957
Confidence and expected cost	0.458	0.421	0.770
Total measured image-plane path	6.702	6.343	8.009

Table B.12: Network-only performance of the GA-Nav TensorRT FP16 logits engine on the Jetson AGX Thor. The benchmark uses preprocessed image tensors and excludes image loading, preprocessing, softmax, cost/confidence computation, and BEV projection.

Metric	Value
TensorRT precision	FP16
Mean host latency	9.91 ms
Median host latency	9.35 ms
95th-percentile host latency	11.93 ms
Mean GPU compute time	9.59 ms
Throughput	104.16 s^{-1}

Table B.13: Peak memory and model size for the original desktop/HPC benchmark. Memory is measured per method in isolation. WVN is split into its frozen feature extractor and the trained head.

Component	CUDA alloc. [MB]	CUDA res. [MB]	Peak RSS [MB]	Parameters
FASTC	741	1206	1566	3 371 321
GA-Nav	893	965	2992	6 215 468
WVN total	787	925	1641	—
feature extractor, frozen	—	—	—	86 541 771
head, trained	—	—	—	34 622
Shared geometry	0	0	727	—

Table B.14: Shared geometry and fusion-stage latency from the earlier desktop/HPC computational-cost benchmark. These values are retained for traceability and are not the corrected Thor Condition C fusion estimate.

Stage	Median [ms]	IQR [ms]	Min [ms]	Max [ms]
Point-painting	17.75	3.01	13.97	28.13
Fusion blend + tier snap	1.35	0.08	1.19	3.54
Obstacle override $\tau = 0.5$	0.90	0.03	0.86	1.10

B.9 Matched PyTorch–TensorRT memory diagnostics

Torch CUDA memory accounting may omit TensorRT-internal allocations. Therefore, the matched-worker memory values are used only as process-level diagnostics. The isolated resource table in Section 5.5.5 remains the main model-footprint comparison.

Table B.15: Shared-process memory accounting for the matched PyTorch–TensorRT workers. These values include both backend objects in the same process and should not be interpreted as isolated PyTorch-only or TensorRT-only memory.

Method	Baseline [MB]	Peak allocated [MB]	Peak reserved [MB]	Caveat
FASTC	591.22	1254.68	1767.90	PyTorch and TensorRT objects coexist; voxelisation stack loaded.
WVN	393.44	501.86	645.92	PyTorch STEGO and TensorRT engine coexist.
GA-Nav	40.79	891.38	952.11	PyTorch model and TensorRT engine coexist; model frozen for inference.

C

Checkpoint provenance

Table C.1 records the checkpoints used for the reported conditions. Checkpoints are identified in the main text only where needed for readability; full paths and available SHA-256 values are retained here.

Table C.1: Checkpoint provenance for the reported conditions.

Condition	Method	Checkpoint identifier	SHA-256
B	FASTC-S	`exp9_fw1/model.pth.600`	`3f2d1a6061e073369e7456ed19ea56668aafa230699c554e20dba7a93f9a7537`
A/C	GA-Nav	`ganav_4tier/iter_38000.pth`	`0c02b3da8a2cafdad3740fb95864a5c556aabd1fafce5fab8926e2915`
A/C	WVN	`wvn_head.pt`	`33ca0dd52e0c96bd7df7b96adf991684c64c0350e0785affe65389144b4c934c`

The following checkpoints were excluded: `ganav_rellis_6.pth`, FASTC `exp7`, `exp3/best.pth.129`, and `fusion_exp2`. Each was excluded because RELLIS evaluation sequence 00004 entered its training or model-selection path. None contributed to any reported result.

D

Description of AI-generated content

AI assistants were used to help write this report. Claude, accessed through claude.ai, was used for online research and verification of information such as dataset documentation and hardware specifications, and for analysis and reasoning support when interpreting and describing experimental results. Codex was used to edit and format the LaTeX source, and to check report text and figures against the underlying experimental data for consistency. Grammarly was used to identify improvements in writing style and to correct grammar. The experimental design, implementation, methodological decisions, interpretation of the results, and final claims in this thesis are the author's own.

References

[1] A. E. Carvalho, D. Portugal, and P. Peixoto, "On terrain traversability analysis in unstructured environments: recent advances in forest applications," *Intelligent Service Robotics*, vol. 18, no. 2, pp. 195–213, 2025. 1, 6

[2] P. V. K. Borges, T. Peynot, S. Liang, B. Arain, M. Wildie, M. G. Minareci, S. Lichman, G. Samvedi, I. Sa, N. Hudson, M. Milford, P. Moghadam, and P. Corke, "A Survey on Terrain Traversability Analysis for Autonomous Ground Vehicles: Methods, Sensors, and Challenges," *Field Robotics*, vol. 2, pp. 1567–1627, Jul. 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10876033 1, 6

[3] C. Sevastopoulos and S. Konstantopoulos, "A Survey of Traversability Estimation for Mobile Robots," *IEEE Access*, vol. 10, pp. 96 331–96 347, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9869644 1, 6

[4] Y. Shu, L. Dong, J. Liu, C. Liu, and W. Wei, "Overview of Terrain Traversability Evaluation for Autonomous Robots," *Journal of Field Robotics*, vol. 42, no. 5, pp. 1724–1765, 2025, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.22461. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22461 1, 6

[5] T. Guan, D. Kothandaraman, R. Chandra, A. J. Sathyamoorthy, K. Weerakoon, and D. Manocha, "GANav: Efficient Terrain Segmentation for Robot Navigation in Unstructured Outdoor Environments," Jun. 2022, arXiv:2103.04233 [cs]. [Online]. Available: http://arxiv.org/abs/2103.04233 2, 6

[6] J. Seo, S. Sim, and I. Shim, "Learning Off-Road Terrain Traversability With Self-Supervisions Only," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4617–4624, Aug. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10146445 2, 7

[7] I. Cho and W. Chung, "Learning Self-Supervised Traversability With Navigation Experiences of Mobile Robots: A Risk-Aware Self-Training Approach," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4122–4129, May 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10468651 2, 7

[8] M. Mattamala, J. Frey, P. Libera, N. Chebrolu, G. Martius, C. Cadena, M. Hutter, and M. Fallon, "Wild visual navigation: fast traversability learning via pre-trained models and online self-supervision," *Autonomous Robots*, vol. 49, no. 3, p. 19, Jul. 2025. [Online]. Available: https://doi.org/10.1007/s10514-025-10202-x 2, 4, 7, 29, 37

[9] J. Sock, J. Kim, J. Min, and K. Kwak, "Probabilistic traversability map generation using 3D-LIDAR and camera," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 5631–5637. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7487782 2, 8, 10

[10] L. Zhou, J. Wang, S. Lin, and Z. Chen, "Terrain Traversability Mapping Based on LiDAR and Camera Fusion," in *2022 8th International Conference on Automation, Robotics and Applications (ICARA)*, Feb. 2022, pp. 217–222, iSSN: 2767-7745. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9738526 2, 8, 10

[11] W. Zhang, S. Lyu, C. Yao, F. Xue, Z. Zhu, and Z. Jia, "Analysis of Robot Traversability over Unstructured Terrain using Information Fusion," in *2022 International Conference on Advanced Robotics and Mechatronics (ICARM)*, Jul. 2022, pp. 413–418. [Online]. Available: https://ieeexplore.ieee.org/document/9959694 2, 8, 10

[12] A. V. Malawade, T. Mortlock, and M. A. Al Faruque, "HydraFusion: Context-Aware Selective Sensor Fusion for Robust and Efficient Autonomous Vehicle Perception," in *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*, May 2022, pp. 68–79. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9797530 2, 8

[13] L. Wijayathunga, A. Rassau, and D. Chai, "Challenges and Solutions for Autonomous Ground Robot Scene Understanding and Navigation in Unstructured Outdoor Environments: A Review," *Applied Sciences*, vol. 13, no. 17, p. 9877, Jan. 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/17/9877 2, 6

[14] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, "RELLIS-3D Dataset: Data, Benchmarks and Analysis," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 1110–1116, iSSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9561251 3, 9, 10, 20, 86

[15] M. Wigness, S. Eum, J. G. Rogers, D. Han, and H. Kwon, "A RUGD Dataset for Autonomous Navigation and Visual Perception in Unstructured Outdoor Environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 5000–5007, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8968283 3, 9, 10

[16] P. Mortimer, R. Hagmanns, M. Granero, T. Luettel, J. Petereit, and H.-J. Wuensche, "The GOOSE Dataset for Perception in Unstructured Environments," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 14 838–14 844. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10611298 3, 9, 10, 20, 86

[17] M. Benrabah, C. Orou Mousse, E. Randriamiarintsoa, R. Chapuis, and R. Aufrère, "A Review on Traversability Risk Assessments for Autonomous Ground Vehicles: Methods and Metrics," *Sensors*, vol. 24, no. 6, p. 1909, Jan. 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/6/1909 3, 6

[18] L. Piccinelli, C. Sakaridis, Y.-H. Yang, M. Segu, S. Li, W. Abbeloos, and L. Van Gool, "UniDepthV2: Universal monocular metric depth estimation made simpler," 2025, arXiv:2502.20110. [Online]. Available: https://arxiv.org/abs/2502.20110 4, 7, 29

[19] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4604–4612. 4, 9, 33

[20] F. Atas, G. Cielniak, and L. Grimstad, "From Simulation to Field: Learning Terrain Traversability for Real-World Deployment," Jan. 2025, arXiv:2501.06904 [cs]. [Online]. Available: http://arxiv.org/abs/2501.06904 4, 8

[21] T. Arafin, A. Hosen, Z. Najdovski, and M. Johnstone, "Advances and Trends in Terrain Classification Methods for Off-Road Perception," *Journal of Field Robotics*, 2025. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/rob.22586 6

[22] M. G. Müller, W. Boerdijk, M. Durner, R. Giubilato, A. Gawel, W. Stürzl, R. Siegwart, and R. Triebel, "Trinity: Unifying Class-Agnostic Terrain and Semantic Segmentation for Unstructured Outdoor Environments by Leveraging Synthetic Data," 2026. [Online]. Available: https://arxiv.org/abs/2605.27644 7

[23] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, "Fast Traversability Estimation for Wild Visual Navigation," May 2023, arXiv:2305.08510 [cs]. [Online]. Available: http://arxiv.org/abs/2305.08510 7, 9

[24] S. Jung, J. Lee, X. Meng, B. Boots, and A. Lambert, "V-STRONG: Visual Self-Supervised Traversability Learning for Off-road Navigation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 1766–1773. [Online]. Available: https://ieeexplore.ieee.org/document/10611227 7

[25] M. V. Gasparino, A. N. Sivakumar, and G. Chowdhary, "WayFASTER: a Self-Supervised Traversability Prediction for Increased Navigation Awareness," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 8486–8492. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10610436 7

[26] J.-H. Hwang, J. Bae, D.-W. Kim, Y. Lee, and S.-W. Seo, "From General Vision to Reliable Traversability Estimation: Adapting Vision Foundation Models for Unstructured Outdoor Environments," 2026. [Online]. Available: https://arxiv.org/html/2605.29565v1 7

[27] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, "Depth anything v2," in *Advances in Neural Information Processing Systems*, vol. 37, 2024, arXiv:2406.09414. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/hash/26cfdcd8fe6fd75cc53e92963a656c58-Abstract-Conference.html 7, 30

[28] M. Hu, W. Yin, C. Zhang, Z. Cai, X. Long, K. Wang, H. Chen, G. Yu, C. Shen, and S. Shen, "Metric3D v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation," 2024, arXiv:2404.15506. [Online]. Available: https://arxiv.org/abs/2404.15506 7, 30

[29] Z. Jian, Z. Lu, X. Zhou, B. Lan, A. Xiao, X. Wang, and B. Liang, "PUTN: A Plane-fitting based Uneven Terrain Navigation Framework," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 7160–7166, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9981038 7

[30] Y. Chen, P. Wei, Z. Liu, B. Wang, J. Yang, and W. Liu, "FASTC: A Fast Attentional Framework for Semantic Traversability Classification Using Point Cloud," Jun. 2024, arXiv:2406.16564 [cs]. [Online]. Available: http://arxiv.org/abs/2406.16564 7, 9

[31] A. Santo, E. Heredia, C. Viegas, D. Valiente, and A. Gil, "Ground Segmentation for LiDAR Point Clouds in Structured and Unstructured Environments Using a Hybrid Neural–Geometric Approach," *Technologies*, vol. 13, no. 4, Apr. 2025. [Online]. Available: https://www.mdpi.com/2227-7080/13/4/162 7

[32] A. Santo, J. J. Cabrera, D. Valiente, C. Viegas, and A. Gil, "TE-NeXt: A LiDAR-Based 3D Sparse Convolutional Network for Traversability Estimation," 2024. [Online]. Available: https://arxiv.org/abs/2406.01395 7

[33] M. A. V. Saucedo, A. Patel, C. Kanellakis, and G. Nikolakopoulos, "EAT: Environment Agnostic Traversability for reactive navigation," *Expert Systems with Applications*, vol. 244, p. 122919, Jun. 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423034218 7

[34] J. Guo, B. Qian, J. Cheng, and X. Dong, "Terrain Traversability Analysis in Uneven Environments," in *2025 6th International Conference on Mechatronics Technology and Intelligent Manufacturing (ICMTIM)*, Apr. 2025, pp. 354–358. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/11041520 8, 9

[35] O. Kim, J. Seo, S. Ahn, and C. H. Kim, "UFO: Uncertainty-aware LiDAR-image Fusion for Off-road Semantic Terrain Map Estimation," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2024, pp. 192–199, iSSN: 2642-7214. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10588787 8, 10, 92

[36] S. Beycimen, D. Ignatyev, and A. Zolotas, "Real-time terrain traversability analysis and mapping for autonomous robotics in dynamic environments: Fusing appearance- and geometry-based approaches," *Journal of Field Robotics*, 2026. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/rob.70097 8, 10

[37] N. Wang, X. Li, Z. Suo, J. Fan, J. Wang, and D. Xie, "Traversability Analysis and Path Planning for Autonomous Wheeled Vehicles on Rigid Terrains," *Drones*, vol. 8, no. 9, p. 419, 2024, num Pages: 419. [Online]. Available: https://www.proquest.com/docview/3110455836/abstract/5614C7E8BE864A91PQ/1 8

[38] J. Frey, M. Patel, D. Atha, J. Nubert, D. Fan, A. Agha, C. Padgett, P. Spieler, M. Hutter, and S. Khattak, "RoadRunner—Learning Traversability Estimation for Autonomous Off-Road

Driving," *IEEE Transactions on Field Robotics*, vol. 1, pp. 192–212, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10684236 8

[39] M. Patel, J. Frey, D. Atha, P. Spieler, M. Hutter, and S. Khattak, "RoadRunner M&M: Learning Multi-range Multi-resolution Traversability Maps for Autonomous Off-road Navigation," 2024. [Online]. Available: https://arxiv.org/abs/2409.10940 8

[40] M. Patel, F. Yang, Y. Qiu, C. Cadena, S. Scherer, M. Hutter, and W. Wang, "TartanGround: A Large-Scale Dataset for Ground Robot Perception and Navigation," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2025, pp. 20 524–20 531, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/11246002 9, 10

[41] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/Cordts_The_Cityscapes_Dataset_CVPR_2016_paper.html 9

[42] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4 9

[43] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "ViNT: A Foundation Model for Visual Navigation," Oct. 2023, arXiv:2306.14846 [cs]. [Online]. Available: http://arxiv.org/abs/2306.14846 9

[44] A. Sridhar, D. Shah, C. Glossop, and S. Levine, "NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration," Oct. 2023, arXiv:2310.07896 [cs]. [Online]. Available: http://arxiv.org/abs/2310.07896 9

[45] D. Shah, B. Osinski, B. Ichter, and S. Levine, "LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action," Jul. 2022, arXiv:2207.04429 [cs]. [Online]. Available: http://arxiv.org/abs/2207.04429 9

[46] M. Elnoor, K. Weerakoon, G. Seneviratne, R. Xian, T. Guan, M. K. M. Jaffar, V. Rajagopal, and D. Manocha, "Robot Navigation Using Physically Grounded Vision-Language Models in Outdoor Environments," Sep. 2024, arXiv:2409.20445 [cs]. [Online]. Available: http://arxiv.org/abs/2409.20445 9

[47] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke,

Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," Jul. 2023, arXiv:2307.15818 [cs]. [Online]. Available: http://arxiv.org/abs/2307.15818 9

[48] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, "PaLM-E: An Embodied Multimodal Language Model," Mar. 2023, arXiv:2303.03378 [cs]. [Online]. Available: http://arxiv.org/abs/2303.03378 9

[49] Y. Lee, Y. Cheong, and J. An, "GPT-4off: On-Board Traversability Probability Estimation for Off-Road Driving via GPT Knowledge Distillation," *Applied Sciences*, vol. 15, no. 4, p. 2130, 2025. 9

[50] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, B. Ichter, D. Driess, J. Wu, C. Lu, and M. Schwager, "Foundation Models in Robotics: Applications, Challenges, and the Future," Dec. 2023, arXiv:2312.07843 [cs]. [Online]. Available: http://arxiv.org/abs/2312.07843 9

[51] J. A. R. Marshall and A. B. Barron, "Are Transformers Truly Foundational for Robotics?" Feb. 2025, arXiv:2411.16917 [cs]. [Online]. Available: http://arxiv.org/abs/2411.16917 9

[52] The MathWorks, Inc., "Camera and lidar toolbox documentation," 2026. [Online]. Available: https://www.mathworks.com/help/lidar/camera-lidar-calibration.html 15

[53] ETH Zurich Autonomous Systems Lab, "Kalibr visual-inertial calibration toolbox documentation," 2022. [Online]. Available: https://github.com/ethz-asl/kalibr/wiki 15

[54] T. Beemelmanns, "ros2_calib: A multi-sensor calibration tool for ros2," 2025. [Online]. Available: https://github.com/ika-rwth-aachen/ros2_calib 16

[55] K. Koide, S. Oishi, M. Yokozuka, and A. Banno, "General, single-shot, target-less, and automatic lidar-camera extrinsic calibration toolbox," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023. [Online]. Available: https://arxiv.org/abs/2302.05094 18

[56] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. [Online]. Available: https://arxiv.org/abs/1911.11763 18

[57] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, arXiv:2105.15203. [Online]. Available: https://arxiv.org/abs/2105.15203 29

[58] W.-F. Lin, D.-Y. Tsai, L. Tang, C.-T. Hsieh, C.-Y. Chou, P.-H. Chang, and L. Hsu, "Onnc: A compilation framework connecting onnx to proprietary deep learning accelerators," in *2019 IEEE*

International Conference on Artificial Intelligence Circuits and Systems (AICAS). IEEE, 2019, pp. 214–218. 35

[59] Y. Zhou and K. Yang, "Exploring tensorrt to improve real-time inference for deep learning," in *2022 IEEE 24th International Conference on High Performance Computing & Communications; 8th International Conference on Data Science & Systems; 20th International Conference on Smart City; 8th International Conference on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 2022, pp. 2011–2018. 35

[60] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International conference on machine learning*. PMLR, 2017, pp. 1321–1330. 72

[61] Y. Wei, Z. Wei, Y. Rao, J. Li, J. Zhou, and J. Lu, "LiDAR Distillation: Bridging the Beam-Induced Domain Gap for 3D Object Detection," in *European Conference on Computer Vision (ECCV)*, Mar. 2022, arXiv:2203.14956 [cs]. [Online]. Available: http://arxiv.org/abs/2203.14956 90

[62] L. Kong, Y. Liu, X. Li, R. Chen, W. Zhang, J. Ren, L. Pan, K. Chen, and Z. Liu, "Robo3D: Towards Robust and Reliable 3D Perception against Corruptions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 19 994–20 006. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Kong_Robo3D_Towards_Robust_and_Reliable_3D_Perception_against_Corruptions_ICCV_2023_paper.html 91

[63] R. Mohan, F. Drews, Y. Miron, D. Cattaneo, and A. Valada, "UP-Fuse: Uncertainty-guided LiDAR-camera fusion for 3D panoptic segmentation," *arXiv preprint arXiv:2602.19349*, 2026. 92

[64] L. Huang, S. Ruan, P. Decazes, and T. Denœux, "Deep evidential fusion with uncertainty quantification and reliability learning for multimodal medical image segmentation," *Information Fusion*, vol. 113, p. 102648, 2024. 92