

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Perception-driven Trajectory Generation of an Agricultural Robot for Vegetation Cover Maintenance

David Filipe Salazar Brás



Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: André Rodrigues Baltazar

Co-Supervisor: André Silva Pinto de Aguiar

July 31, 2026

Resumo

A crescente escassez de mão de obra agrícola e os requisitos de sustentabilidade das culturas permanentes têm criado fortes incentivos para sistemas robóticos autônomos na viticultura. O controlo do coberto vegetal nos corredores entre linhas apresenta um desafio particular: a largura do actuador de corte é tipicamente inferior à do corredor, pelo que a cobertura total requer uma operação em múltiplas passagens, em que o robô deve localizar as linhas de videira, registar o estado da vegetação e derivar uma referência lateral adaptativa a partir da fronteira entre a zona cortada e a zona por cortar.

Esta dissertação propõe um sistema de percepção, mapeamento e planeamento local para a manutenção autónoma do coberto vegetal em dois passes nos corredores entre linhas de vinhas, utilizando um sensor LiDAR tridimensional como única fonte de percepção exteroceptiva. Três contribuições abordam o problema. A primeira é um modelo preditivo de detecção de linhas com dupla região de interesse, que ajusta planos verticais às estruturas laterais das videiras via RANSAC em regiões próximas e distantes em separado, com fronteiras virtuais adaptativas à dispersão do coberto a delimitar a área navegável. A segunda é um mapa de altura da vegetação acumulado dentro do polígono de corredor derivado da percepção. A terceira é um detetor da linha de corte baseado numa sequência de testes geométricos que só identifica a fronteira entre a zona cortada e a zona por cortar quando existe evidência suficiente, abstendo-se de reportar caso contrário, e que transfere essa estimativa ao planeador como referência de segurança crítica.

O sistema foi avaliado em vinte mundos de simulação Webots abrangendo cinco formas de corredor e quatro famílias de tarefas concebidas para isolar cada modo de operação, sendo esta campanha complementada por um registo adquirido *in situ* numa vinha real do Douro, reproduzido *offline* para verificar indicadores de consistência interna e validar exclusivamente o detetor de linha de corte. A detecção de linhas manteve uma precisão lateral ao nível do centímetro com tempos de processamento compatíveis com operação em tempo real, tendo o seu detetor registado zero falsos positivos em todos os corredores sem linha de corte e localizado a fronteira a poucos centímetros quando uma linha de corte estava presente. O planeador produziu referências G^1 -contínuas que seguiram o desvio lateral pretendido e mantiveram uma referência válida na linha de videira sobrevivente durante uma perda de detecção unilateral induzida. Estes resultados demonstram que um sistema baseado num único LiDAR pode fechar o ciclo percepção-para-referência necessário para a manutenção autónoma do coberto vegetal em corredores de vinhas.

Palavras-chave: Robótica agrícola, viticultura, LiDAR, mapeamento de vegetação, detecção de linhas, planeamento local, corte autónomo.

Abstract

The declining availability of agricultural labour and the sustainability requirements of permanent crops have created strong incentives for autonomous robotic systems in viticulture. Inter-row vegetation control presents a particular challenge: the mowing actuator is typically narrower than the corridor, so complete floor coverage requires multi-pass operation in which the robot must locate the vine rows, record the vegetation state, and derive an adaptive lateral reference from the mown-to-unmown boundary on each successive pass.

This dissertation proposes a perception, mapping, and local-planning stack for autonomous two-pass vegetation maintenance in vineyard inter-row corridors, using a three-dimensional Light Detection and Ranging (LiDAR) sensor as the sole exteroceptive input. Three contributions address the problem. The first is a predictive dual region-of-interest row model that fits vertical planes to the lateral vine structures via Random Sample Consensus (RANSAC) in near- and far-field regions separately, with canopy-dispersion-adaptive virtual boundaries delimiting the navigable corridor area. The second is a vegetation height map accumulated within the perception-derived corridor polygon. The third is an abstaining cut-line cascade that detects the mown-to-unmown boundary only when sufficient geometric evidence is present, transferring that estimate to the local planner as a safety-critical reference.

The stack was evaluated across twenty Webots simulation worlds spanning five corridor shapes and four task families designed to isolate each operating mode. This campaign was complemented by a single continuous recording acquired *in situ* in a real Douro vineyard, replayed offline to verify internal-consistency indicators and validate exclusively the cut-line detector system. Row detection maintained centimetre-level lateral accuracy with per-sweep processing times compatible with on-line operation, with its detector recording zero false positives across all no-cut corridors and localising the mown-to-unmown boundary to within a few centimetres where a cut line was present. The local planner produced G^1 -continuous references that tracked the intended row offset and maintained a valid reference on the surviving vine row throughout an induced one-sided detection loss. These results demonstrate that a single LiDAR-based sensing stack can close the perception-to-reference loop required for autonomous two-pass vegetation maintenance in vineyard corridors.

Keywords: Agricultural robotics, viticulture, LiDAR, vegetation mapping, row detection, local planning, autonomous mowing.

UN Sustainable Development Goals

Adopted in 2015, the 17 UN Sustainable Development Goals (SDGs)¹ set out a shared international agenda for prosperity, environmental protection, and social justice through to 2030. This dissertation was not framed around the SDGs from the outset, but several of its outcomes align with specific goals in sustainable agriculture, innovation, and land stewardship, and this section makes that alignment explicit.

Because the system targets a recurring, labour-intensive vineyard task and, as a by-product of normal operation, produces a record of the vegetation it observes, its relevance extends somewhat beyond the immediate engineering problem. The paragraphs below identify the goals to which the work most plausibly contributes; a summary table linking each to a specific target and indicator follows.

SDG 2 — Zero Hunger: more consistent, automated management of the vineyard corridor floor supports productive and resilient agricultural systems at a time of declining agricultural labour availability.

SDG 8 — Decent Work and Economic Growth: automating a physically demanding and repetitive field task, and removing its dependence on tractor and operator availability, lightens the manual burden of vineyard maintenance and supports the modernisation of viticultural work.

SDG 9 — Industry, Innovation and Infrastructure: the dissertation contributes new sensing, mapping, and planning methods for vineyard robotics, evaluated in simulation and on real field data and submitted for publication, adding to the research base in agricultural automation.

SDG 12 — Responsible Consumption and Production: avoiding redundant mowing of ground already treated, and reducing reliance on chemical herbicides for inter-row vegetation control, supports more efficient use of agricultural inputs and resources.

SDG 13 — Climate Action: fewer tractor-mounted mowing passes mean lower fuel use and less soil compaction, and a compact, low-impact platform supports lower-emission field operations.

SDG 15 — Life on Land: keeping a managed cover of grass and low vegetation, rather than bare or chemically treated soil, reduces erosion and compaction and supports beneficial-insect habitat, in line with the sustainability trends discussed in Chapter 1.

¹<https://sdgs.un.org/goals>

Table 1: Contribution of this dissertation to the relevant UN Sustainable Development Goals, by target.

SDG	Target	Contribution	Performance Indicators and Metrics
2	2.4	Automated, sensor-based management of the corridor floor supports resilient, resource-efficient vineyard production.	Accuracy and consistency of automated corridor coverage.
8	8.2	Automating a physically demanding field task increases technological sophistication and operational efficiency in viticulture.	Reduction in mowing effort required; suitability of the system for real-time field use.
9	9.5	New sensing and mapping methods for vineyard robotics, evaluated in simulation and on real field data and submitted for publication.	Peer-reviewed publication; consistent performance across varied simulated and real-world scenarios.
12	12.2	Avoiding redundant mowing passes and reducing herbicide use for inter-row vegetation control supports more efficient use of agricultural inputs.	Reliability in distinguishing treated from untreated vegetation.
13	13.2	Fewer tractor-mounted mowing passes mean lower fuel use and soil compaction.	Reduction in required mowing passes; efficient use of the corridor area.
15	15.3	A maintained vegetation cover reduces soil erosion and compaction and supports beneficial-insect habitat in vineyard corridors.	Coverage and quality of the vegetation record produced by the system.

Acknowledgements

A realização desta dissertação foi uma etapa exigente e muito enriquecedora, que contou com o apoio de várias pessoas.

Agradeço ao meu orientador, Professor André Baltazar, pela confiança, orientação e espírito crítico ao longo de todo o trabalho. A sua experiência e disponibilidade foram fundamentais para conduzir o projeto nos momentos mais importantes.

Ao meu coorientador, André Aguiar, agradeço o apoio, as sugestões e o conhecimento partilhado, que contribuíram para o desenvolvimento desta dissertação.

Deixo também um agradecimento ao Daniel Queirós, pela ajuda técnica e pela disponibilidade demonstrada em fases cruciais do projeto.

A todos os elementos do TRIBE Lab, agradeço o acolhimento, o ambiente de trabalho e o apoio prestado sempre que necessário.

Por fim, agradeço aos meus amigos e à minha família pela motivação, paciência e apoio incondicional ao longo deste percurso.

A todos, o meu sincero obrigado.

“The time you enjoy wasting is not wasted time.”

Bertrand Russell

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	2
1.3	Problem Statement	2
1.4	Contributions	3
1.5	Document Structure	4
2	Background	5
2.1	Overview of Agricultural Robotics	5
2.2	Classification of Agricultural Robots and Operations	6
2.3	Support Technologies and Foundations	7
2.3.1	Sensors and Operating Principles	7
2.3.2	Software Ecosystem and Tools	8
2.3.3	Simulation Environment	9
2.4	Functional Requirements	9
3	State of the Art	11
3.1	Robotic Platforms for Inter-Row Vegetation Management	11
3.2	Axis 1 — Crop-Row Detection and Cut-Line Detection	12
3.2.1	From Vision to LiDAR — a Motivated Sensor Choice	12
3.2.2	Geometric Row Extraction and Temporal Validation	12
3.2.3	Corridor Delimitation and Separation of Lateral Structures	14
3.2.4	Detection of Mown Areas and Cut Lines	15
3.3	Axis 2 — Mapping and 2.5D Height Mapping	16
3.3.1	Vegetation Height Estimation	16
3.3.2	Grid-Map Frameworks and Real-Time Local Mapping	17
3.4	Axis 3 — Smooth Trajectory Generation and Behaviour Supervision	18
3.4.1	Reactive Row Following	18
3.4.2	Smooth Trajectory Generation with Bézier and Spline Curves	18
3.4.3	Behaviour Supervision with Finite-State Machines	19
3.5	Critical Synthesis and Research Gap	20
4	System Architecture and Overview	21
4.1	System Overview	21
4.2	Processing Nodes	23
4.2.1	Perception Node	23
4.2.2	Local-Planning Node	23
4.2.3	Mapping Node	24

4.3	Software Namespaces and Data Structures	24
4.4	Coordinate Frames and Conventions	25
4.5	Validation Environments	26
5	Perception: Dual-ROI Row Detection and Virtual Boundaries	29
5.1	Overview and Processing Cycle	29
5.2	Dual-ROI Row Model	32
5.3	Region-of-Interest Filtering and Placement	34
5.4	Region-of-Interest Update: Vertical-Plane Fitting and Tangent-Line Extraction . .	40
5.5	FAR-Line Cross-Validation	42
5.6	Temporal Line Filtering: Inlier-Ratio-Weighted FIR with Circular Averaging . .	45
5.7	Per-Sweep Verdict and Accumulated Safety Score	46
5.8	Canopy Dispersion	48
5.9	Virtual Boundaries and the Inter-Row Polygon	49
5.10	Perception Output Message (RowDetectionResult)	54
6	Mapping: Corridor Height Mapping and Cut-Line Detection	56
6.1	Overview and Processing Cycle	56
6.2	Synchronised Intake and Corridor Extraction (CropHull)	58
6.3	Mission-Level Sparse 2.5D Height Map	61
6.4	Robot-Centric Grid Map	65
6.5	Cut-Line Detection — the Acceptance-Gate Cascade	66
6.5.1	Sectioning and 1D Pooled Profiles	67
6.5.2	Derivative-of-Gaussian Matched Filtering	70
6.5.3	Step-Magnitude Gate	72
6.5.4	Geometric-Signature (Boundary-Polarity) Gate	73
6.5.5	Per-Section Temporal Filtering with Directional Saturation	75
6.5.6	Cross-Section Sign Consensus	77
6.5.7	Cross-Section Weighted Least-Squares Fit and Metric Validity Gate . . .	78
6.5.8	Condensation and Publication	79
6.6	Parameter Summary and Calibration	81
7	Local Planning: FSM, Safety Hysteresis and Continuous Trajectories	83
7.1	Node State and Detection Intake	83
7.2	The Supervisory FSM	84
7.3	Hysteretic Safety Scores and On-line Inter-Row Width	86
7.4	Side-Dependent Normals and the Trajectory Endpoint	87
7.5	Cubic Bézier Curves and One Continuous Trajectory	90
7.6	Reference-Side Switching with Complement Offset	92
7.7	Cut-Line Following	93
7.8	Interface to the Trajectory Controller	97
8	Results and Discussion	98
8.1	Experimental Setup	98
8.1.1	Real-Vineyard Rosbag	101
8.2	Perception Results	101
8.2.1	Row-Line Accuracy and Support	102
8.2.2	Availability, Width and Cycle Time	105
8.3	Mapping and Cut-Line Detection Results	107

8.3.1	Point-Cloud Intake	108
8.3.2	Cut-Line Detection and Abstention	109
8.3.3	Cut-Line Geometric Error	110
8.4	Local-Planning Results	112
8.4.1	Continuity, State Occupancy and Cost	112
8.4.2	One-Sided-Loss Robustness	113
8.4.3	Row-Following Trajectory Error	114
8.4.4	Cut-Line-Following Trajectory Error	124
8.5	Height-Map Products	135
8.6	Discussion	137
8.6.1	Synthesis	137
8.6.2	Comparison with the state of the art	138
8.6.3	Limitations and critical scenarios	140
8.6.4	Technological and scientific implications	141
9	Conclusions and Future Work	143
9.1	Main Conclusions	143
9.2	Future Work	144
	References	145

List of Figures

4.1	System architecture: three single-responsibility nodes, the message contracts between them, and the external subsystems with which the system exchanges data. Perception publishes the corridor polygon that configures the mapping node's spatial filter on-line, closing the perception–mapping loop.	22
4.2	The Modular_E robotic platform used for real-vineyard data collection and deployment	27
4.3	Approximate mesh of a single vine plant, the building block concatenated along each row to construct the simulated corridors.	27
4.4	Two representative simulated corridors built by concatenating the single-vine mesh of Figure 4.3 along the row path, with grass pads placed on the corridor floor: (a) a straight corridor and (b) a curved corridor of $R = 20$ m radius.	28
5.1	Per-sweep processing pipeline of the perception node.	30
5.2	Per-side processing, decomposed into a filtering phase and an update phase. The updated centroid of each region of interest defines its crop-box centre on the following sweep.	32
5.3	Overview of the dual-region row model: the robot in the corridor, the two sides, and their CLOSE/FAR regions of interest.	33
5.4	Predictive placement of the CLOSE region across two consecutive sweeps: the box tracks the foot point of the robot on the row and is rotated to the row direction on each sweep.	36
5.5	Bootstrap placement of the two regions before any row has been tracked. The CLOSE box is set by a fixed lateral offset from the robot (a horizontal shift in y); the FAR box is then placed by extending the CLOSE direction forward by the look-ahead distance.	37
5.6	Predictive placement of the FAR region across two consecutive sweeps: the box is centred on the FAR centroid from the previous sweep and re-projected along its direction to hold the fixed separation L from the CLOSE centroid.	38
5.7	Illustration of the fixed separation constraint between the CLOSE and FAR centroids. A circumference of radius L is centred on the CLOSE centroid, and the FAR centroid is moved along its line direction until it lies exactly on the circumference. Among the two possible intersections, the nearest one, t_1 , is selected, while the farther solution, t_2 , is discarded.	39
5.8	Vertical-plane fit and tangent-line extraction for a single region of interest, shown as (a) a side view and (b) a top view. The plane normal is constrained to the horizontal; the extracted line direction is the 90° rotation of the normal's horizontal component.	42

5.9	Geometric cross-validation of the FAR line against the CLOSE line. (a) the FAR heading differs from the CLOSE heading by more than $\theta_{\max}$; (b) the FAR centroid lies more than $d_{\max}$ to the side of the CLOSE line; (c) both are within tolerance and the line is accepted. In (a) and (b) the sweep is recorded as a failed detection. . . .	44
5.10	Effect of the temporal filter on the per-sweep line estimate of the left CLOSE region: heading (top) and lateral offset (bottom), raw per-sweep fit versus FIR-filtered. The inlier-ratio weighting yields adaptive smoothing without an explicit gain schedule.	46
5.11	Life cycle of the per-side safety score, from accumulation and saturation to the at-floor anchor reset, recovery, and publication.	48
5.12	Canopy dispersion. (<i>left</i>) the Webots vine model of a single crop, stacked longitudinally to form a corridor row; (<i>right</i>) the LiDAR point cloud returned by that model, with a Gaussian fitted to the lateral spread of the returns. The standard deviation σ of that spread is the canopy dispersion, used as the inset margin of the corridor boundary.	49
5.13	Construction of a single per-side boundary strip: (<i>a</i>) placement of the four control points, the start point P_0 anchored on the CLOSE region and the end point P_3 on the FAR region, with P_1 and P_2 placed along their respective headings; (<i>b</i>) the resulting cubic Bézier strip, shown in blue.	51
5.14	Estimation of the inter-row width W as the orthogonal distance between the two CLOSE row lines, shown in (<i>left</i>) top view and (<i>right</i>) side view.	52
5.15	Geometric construction of the inter-row polygon $\mathcal{P}$ from the two boundary strips: the left strip sampled along its arc length, the right strip sampled and reversed, and the two sequences concatenated and closed to trace the corridor outline. Figure 5.16 shows the resulting polygon on actual corridor data.	53
5.16	Inter-row polygon $\mathcal{P}$ assembled from the two canopy-dispersion-adaptive boundary strips, on (<i>left</i>) a straight corridor and (<i>right</i>) a curved crop corridor. Returns from the rows fall outside the polygon and are excluded from mapping.	54
5.17	Single-sided fallback, in (<i>left</i>) top view and (<i>right</i>) side view. When one row is lost, its boundary is reconstructed from the surviving row offset by the inter-row width W under the corridor-symmetry assumption.	54
6.1	Per-sweep processing pipeline of the mapping node, from the synchronised intake to the publication of the height-map products and the cut line.	57
6.2	The cut-line detection cascade: per-section profiling, the acceptance gates applied to each candidate, cross-section consensus, and the publication of the resulting cut line.	57
6.3	Effect of the height-gate half-width T on the retained cloud, and the contamination/saturation trade-off that fixes the adopted value.	59
6.4	Construction of the navigable corridor area from the corridor polygon received from perception: the geometric hull used to gate the mapping node's intake. . . .	60
6.5	Corridor extraction. The height gate and the perception-derived CropHull reduce the raw sweep to the vegetation on the corridor floor.	61

6.6	Mission-level sparse height maps exported at the end of a run, shown alongside their respective source environments. (a) Straight inter-row corridor from a real-vineyard rosbag: the field photograph shows the vegetation state at the time of collection, and the height map shows the per-cell maximum vegetation height accumulated during the run. (b) Curved corridor from a Webots simulation: the viewport screenshot shows the simulated environment, and the height map shows the corresponding accumulated result. In both maps, rows are absent because the cloud was restricted to the inter-row corridor before accumulation.	64
6.7	Robot-centric grid map at two points along a vineyard corridor. On the left the cut line is clearly visible as a lateral height step between the higher standing vegetation and the lower, already-mown swath; on the right no such step is present, so no cut line can be read from the map.	66
6.8	The white CropHull-filtered points on the navigable inter-row floor, viewed longitudinally along the corridor from the front. In (a) the (white) points are dispersed and no cut line is discernible; in (b) a cut line is clearly visible as the boundary between the raised vegetation on the left and the lower vegetation on the right. Note that, as the view looks back along the corridor, the point clouds are laterally mirrored, so that the right-hand crop rows appear on the left and the left-hand rows on the right.	67
6.9	Configuration of the sections and the lateral bins within each section. The forward band $[x_{\min}, x_{\max}]$ is divided into N longitudinal sections, each of which pools the retained cells into lateral bins one resolution cell wide. The pooled mean of each bin is the profile value on which the next gate operates.	68
6.10	Sectioned cut-line detection at a glance: the N sections, the per-section points (x'_s, y'_s) , and the curve fitted through them. (a) a valid case, where the per-section points are mutually consistent and a cut line is fitted; (b) a case where the section points are inconsistent and no valid curve is obtained.	69
6.11	Derivative-of-Gaussian matched filtering of a section profile. The response peaks in magnitude at the steepest transition, which the argmax selects as the candidate boundary.	72
6.12	Side view of the inter-row corridor, seen as though looking longitudinally along it. The vine rows bound the corridor on the left and right, with the open inter-row floor between them, where a band of taller, still-standing vegetation adjoins the lower, already-mown vegetation. The two polarities a lateral height step may take are contrasted. At the <i>valid cut</i> (blue), the taller vegetation lies on the row side and the lower vegetation toward the interior of the corridor, which is the signature of a genuine mown/unmown boundary. At the <i>invalid cut</i> (red), the arrangement is reversed, with the taller vegetation toward the interior and the lower vegetation on the row side; this polarity does not correspond to a cut and is rejected whatever its magnitude.	74
6.13	Per-section detection outcome for a single sweep. Each section yields at most one accepted point y'_s ; sections without a valid, correctly oriented step contribute none. Nine of the ten sections are shown here, arranged three by three; the tenth and farthest section is omitted, as it lies beyond the populated range and yielded no candidate.	75
6.14	Temporal filtering for section 4. Isolated jumps are capped at $\Delta_{\max}$ and absorbed by the moving average, whereas a sustained one-directional run of saturations triggers a reset and the estimate snaps to the drifting boundary.	77

7.1	State diagram of the supervisory finite-state machine. Edges are the guarded transitions of Table 7.1; in normal operation the machine cycles between <i>Following</i> and <i>ComputeNewCurve</i> , switching reference side or stopping only at curve boundaries.	85
7.2	Construction of the trajectory endpoint. The endpoint $\mathbf{P}_3$ is anchored on the FAR line, offset into the corridor by the desired distance d_{row} along the inward normal $\hat{\mathbf{n}}_F$, and assigned a heading that averages the FAR direction and the platform's current heading.	89
7.3	Construction of one cubic Bézier segment. (a) The interior control points $\mathbf{P}_1, \mathbf{P}_2$ are offset from the endpoints along the start and end headings by $\rho \ \mathbf{P}_3 - \mathbf{P}_0\ $; (b) the resulting curve leaves $\mathbf{P}_0$ along ψ_0 and reaches $\mathbf{P}_3$ along ψ_3	90
7.4	Holding the lateral position through a one-sided loss of detection. While both rows are valid the platform is kept at the desired distance d_{row} from the followed (right) row; when that row is lost, the reference moves to the surviving (left) row at the complementary distance $W - d_{\text{row}}$, where W is the last trusted inter-row width, so the measurement geometry switches without moving the platform.	93
7.5	Endpoint construction for cut-line following. The platform is projected onto the cut line, advanced by the span-based look-ahead ℓ , and offset by half the cutter width toward the followed vine so the cutter reaches the line; a collision guard keeps the endpoint at least the safety distance d_s from the nearest vine row.	95
7.6	The cut-line collision guard. The endpoint is kept at least the safety distance d_s from the FAR line of the nearest vine row: (a) when it is closer than d_s it is displaced toward the corridor interior by the deficit $d_s - \rho_v$; (b) when it already lies beyond d_s it is left in place.	96
8.1	The five corridor shapes used by every family: (a) straight, (b)–(d) three arcs of decreasing radius, and (e) the S-curve.	100
8.2	The real-vineyard inter-row corridor where the rosbag used in this chapter was recorded.	101
8.3	Row-line perception lateral error distribution by side and region, <i>FP</i> worlds.	102
8.4	Row-line perception heading error distribution by side and region, <i>FP</i> worlds.	103
8.5	RANSAC inlier-ratio distribution by ROI, <i>FP</i> worlds; the dashed line marks the acceptance threshold.	104
8.6	RANSAC inlier-ratio distribution by ROI, real-vineyard rosbag; the dashed line marks the acceptance threshold of 0.50.	105
8.7	Inter-row width per sweep over the real-vineyard rosbag run, with the run mean and $\pm 1\sigma$ band.	107
8.8	Cut-line lateral error against the ground-truth boundary: <i>Backward Pass</i> on the left; and <i>Transition Pass</i> on the right.	110
8.9	Cut-line weighted-least-squares fit RMSE on valid detections, real-vineyard rosbag; the dashed line marks the 0.10 m acceptance gate.	111
8.10	Planned row-following reference, <i>Forward Pass</i> family, straight corridor.	115
8.11	Planned row-following reference, <i>Forward Pass</i> family, 20 m arc.	116
8.12	Planned row-following reference, <i>Forward Pass</i> family, 10 m arc.	116
8.13	Planned row-following reference, <i>Forward Pass</i> family, 5 m arc.	117
8.14	Planned row-following reference, <i>Forward Pass</i> family, S-curve.	117
8.15	Planned row-following reference, <i>GAP</i> family, straight corridor; the reference stays on the left row through the missing-row interval.	118
8.16	Planned row-following reference, <i>GAP</i> family, 20 m arc.	118

8.17	Planned row-following reference, <i>GAP</i> family, 10 m arc.	119
8.18	Planned row-following reference, <i>GAP</i> family, 5 m arc.	119
8.19	Planned row-following reference, <i>GAP</i> family, S-curve.	120
8.20	Row-following lateral error over the planning cycle, <i>Forward Pass</i> family, straight corridor.	120
8.21	Row-following lateral error over the planning cycle, <i>Forward Pass</i> family, 20 m arc.	121
8.22	Row-following lateral error over the planning cycle, <i>Forward Pass</i> family, 10 m arc.	121
8.23	Row-following lateral error over the planning cycle, <i>Forward Pass</i> family, 5 m arc.	121
8.24	Row-following lateral error over the planning cycle, <i>Forward Pass</i> family, S-curve.	122
8.25	Row-following lateral error over the planning cycle, <i>GAP</i> family, straight corridor.	122
8.26	Row-following lateral error over the planning cycle, <i>GAP</i> family, 20 m arc.	122
8.27	Row-following lateral error over the planning cycle, <i>GAP</i> family, 10 m arc.	123
8.28	Row-following lateral error over the planning cycle, <i>GAP</i> family, 5 m arc.	123
8.29	Row-following lateral error over the planning cycle, <i>GAP</i> family, S-curve.	123
8.30	Distribution of the row-following lateral error: <i>FP</i> on the left; and <i>GAP</i> on the right.	124
8.31	Planned cut-line reference, <i>Backward Pass</i> family, straight corridor.	125
8.32	Planned cut-line reference, <i>Backward Pass</i> family, 20 m arc.	126
8.33	Planned cut-line reference, <i>Backward Pass</i> family, 10 m arc.	126
8.34	Planned cut-line reference, <i>Backward Pass</i> family, 5 m arc.	127
8.35	Planned cut-line reference, <i>Backward Pass</i> family, S-curve.	127
8.36	Planned cut-line reference, <i>Transition Pass</i> family, straight corridor.	128
8.37	Planned cut-line reference, <i>Transition Pass</i> family, 20 m arc.	128
8.38	Planned cut-line reference, <i>Transition Pass</i> family, 10 m arc.	129
8.39	Planned cut-line reference, <i>Transition Pass</i> family, 5 m arc.	129
8.40	Planned cut-line reference, <i>Transition Pass</i> family, S-curve.	130
8.41	Cut-line-following lateral error over the planning cycle, <i>Backward Pass</i> family, straight corridor.	130
8.42	Cut-line-following lateral error over the planning cycle, <i>Backward Pass</i> family, 20 m arc.	131
8.43	Cut-line-following lateral error over the planning cycle, <i>Backward Pass</i> family, 10 m arc.	131
8.44	Cut-line-following lateral error over the planning cycle, <i>Backward Pass</i> family, 5 m arc.	132
8.45	Cut-line-following lateral error over the planning cycle, <i>Backward Pass</i> family, S-curve.	132
8.46	Cut-line-following lateral error over the planning cycle, <i>Transition Pass</i> family, straight corridor.	133
8.47	Cut-line-following lateral error over the planning cycle, <i>Transition Pass</i> family, 20 m arc.	133
8.48	Cut-line-following lateral error over the planning cycle, <i>Transition Pass</i> family, 10 m arc.	134
8.49	Cut-line-following lateral error over the planning cycle, <i>Transition Pass</i> family, 5 m arc.	134
8.50	Cut-line-following lateral error over the planning cycle, <i>Transition Pass</i> family, S-curve.	135
8.51	Distribution of the cut-line-following lateral error: <i>Backward Pass</i> on the left; and <i>Transition Pass</i> on the right.	135

- 8.52 Mission-level height maps for the transition (*TP*) family, one per corridor shape:
(a) straight, (b)–(d) the three arcs, and (e) the S-curve. 136

List of Tables

1	Contribution of this dissertation to the relevant UN Sustainable Development Goals, by target.	iv
3.1	Representative crop- and vine-row detection approaches.	14
4.1	The three processing nodes and their principal message interfaces.	23
4.2	Principal shared data structures and message contracts, with the namespace or module that owns each and the role it serves.	25
5.1	Parameters of the region-of-interest filtering operations. The chassis-exclusion footprint is 1.0×1.0 m, common to both regions.	35
5.2	Parameters of the vertical-plane fit and its acceptance gates.	41
5.3	Bounds and initial value of the per-side safety score.	47
5.4	Principal fields of the <code>RowDetectionResult</code> message. The listed fields are present for both sides; * stands for the <code>close/far</code> region prefix.	55
6.1	Parameters of the mapping pipeline and the cut-line detection cascade, with the values used. The basis for each is given in the text and in the section where it is introduced.	81
7.1	Transition table of the supervisory finite-state machine. <i>Good</i> and <i>critical</i> are the hysteretic predicates of Section 7.3, read on the current or the opposite side; <i>near segment end</i> marks the approach to the end of the current trajectory curve; <i>settled</i> denotes that the machine has remained in the state for longer than the three-second dwell.	86
7.2	Fields of the published <code>tribe_nav_msgs/Trajectory</code> message that the planner populates each cycle. Nested rows are the fields of each <code>BezierCurve</code> entry.	97
8.1	The four task families and, for each, what it verifies and the expected outcome.	99
8.2	Row-line perception lateral RMSE on the forward-pass (FP) worlds, averaged over the four ROIs (left/right $\times$ CLOSE/FAR). <i>N</i> valid sweeps.	102
8.3	Row-line perception heading RMSE on the forward-pass (FP) worlds, averaged over the four ROIs (left/right $\times$ CLOSE/FAR). <i>N</i> valid sweeps.	102
8.4	RANSAC inlier ratio on the forward-pass (FP) worlds, averaged over the four ROIs. <i>N</i> valid sweeps.	103
8.5	RANSAC inlier ratio per ROI for the real-vineyard rosbag: mean, standard deviation, and the fraction of valid fits with inlier ratio at or above 0.50. <i>N</i> = number of valid sweeps.	104
8.6	Valid-fit rate per side on the forward-pass and gap worlds; each side is the mean of its CLOSE and FAR regions. The right side collapses over the induced gap.	105

8.7	Inter-row width on the forward-pass worlds against the nominal 2.5 m spacing. N sweeps.	106
8.8	Per-sweep perception cycle time on the forward-pass worlds. N sweeps.	106
8.9	Inter-row width W measured per sweep: mean, standard deviation, and observed range. No fixed ground-truth spacing is assumed; these figures assess the internal consistency of the estimator across the run. N sweeps.	106
8.10	Cycle time per subsystem: mean, standard deviation, maximum, and 95th-percentile latency. Times in milliseconds. N frames.	107
8.11	Mapping intake point counts on the cut-bearing (BP, TP) worlds. N frames.	108
8.12	Mapping intake processing time on the cut-bearing (BP, TP) worlds.	108
8.13	Mapping intake point counts: mean raw points and mean surviving points after the height-gate and CropHull filtering stages, with the mean fraction removed at each step. N frames.	109
8.14	Cut-line detection rates on the cut-bearing worlds. TP and FN are percentages of the positive frames; TN and FP of the negative frames. The <i>BP</i> worlds have no negative region, so their abstention columns are not defined.	109
8.15	Cut-line external geometric accuracy on the cut-bearing worlds (positive valid frames).	110
8.16	Cut-line weighted-least-squares fit RMSE on the cut-bearing (<i>BP</i> , <i>TP</i>) worlds.	110
8.17	Cut-line WLS fit RMSE on valid detections: mean, standard deviation, maximum, and fraction within the 0.10 m acceptance gate. N is the number of frames in which a cut line was successfully detected.	111
8.18	Cut-line and mapping-cycle time on the cut-bearing worlds. N frames.	112
8.19	Finite-state-machine occupancy (% of effective cycles, STOP blocks compressed) on the transition and gap worlds, where the planner changes mode or reference side.	113
8.20	Per-cycle planning time (forward-pass worlds, representative). N cycles.	113
8.21	One-sided-loss robustness (GAP worlds). Within the arc-length window where the right row is physically absent, the number of control cycles and the fraction in which the planner kept a usable reference (did not enter STOP), navigating on the remaining left row.	114
8.22	Row-following trajectory lateral error on the forward-pass and gap worlds. N samples.	114
8.23	Row-following trajectory heading error on the forward-pass and gap worlds. N samples.	115
8.24	Cut-line-following trajectory lateral error on the cut-bearing worlds. N samples.	124
8.25	Cut-line-following trajectory heading error on the cut-bearing worlds. N samples.	125
8.26	Quantitative comparison between thesis results and selected state-of-the-art references. Thesis figures are open-loop reference errors against analytic ground truth in Webots simulation. Literature figures are from real-field closed-loop deployments unless otherwise stated.	138

List of Acronyms

DoG	Derivative of Gaussian
FIR	Finite Impulse Response
FOV	Field of View
FSM	Finite-State Machine
GNSS	Global Navigation Satellite System
LiDAR	Light Detection and Ranging
PCL	Point Cloud Library
RANSAC	Random Sample Consensus
RMSE	Root-Mean-Square Error
ROI	Region of Interest
ROS	Robot Operating System
SOR	Statistical Outlier Removal
TF	Transform (ROS transform tree)
WLS	Weighted Least Squares

Nomenclature

$\mathbf{c}$	Row/ROI centroid
$\hat{\mathbf{d}}$	Unit direction of a fitted line
$\hat{\mathbf{n}}$	Unit normal
ρ	RANSAC inlier ratio
δ	Per-sweep detection verdict
S	Accumulated safety score
σ_{canopy}	Canopy dispersion
W	On-line inter-row width
λ	Blend / weighting factor
Δs	Arc-length discretisation step
w_c	Cutter width
K	Minimum valid bins (occupancy gate)
γ	Cross-section consensus fraction
Δh_{min}	Minimum step magnitude
M	Temporal-filter window length
N	Number of profile sections

Chapter 1

Introduction

This chapter situates the work and defines the problem it addresses. It first establishes the agricultural and technological context that motivates autonomous robots in vineyard corridors (Section 1.1) and narrows to the specific operational problem (Section 1.2). The problem is then stated in engineering terms (Section 1.3), followed by the main contributions of the work (Section 1.4) and an outline of the document structure (Section 1.5).

1.1 Context

Autonomous robotic systems have become an increasingly relevant response to the labour shortages, rising production demands, and sustainability requirements of modern agriculture [10]. The global agricultural workforce has been declining in many regions while the costs of manual field operations continue to rise; robotic mechanisation offers the prospect of continuous, repeatable work at consistent quality and, increasingly, of real-time spatial data collection that supports precision farming at scales impractical for human operators.

Row-based permanent crops — vineyards, orchards, and olive groves — are particularly well suited to early robotic intervention. Their regular geometry provides structural cues that perception systems can exploit, while the high value per hectare and the long investment horizon of perennial crops make the economic case for automation compelling [49]. At the same time, the narrow corridors, uneven terrain, proximity to trellis structures, and seasonal variation in canopy density that characterise these environments impose strict constraints on vehicle dimensions, sensing range, and navigation reliability that distinguish this domain from open-field agriculture [38].

Viticulture has attracted sustained robotic research over the past two decades. Early work focused on platform engineering and GPS-guided row guidance, establishing the open architectures and drivetrain designs that underpin long-term field experimentation [5]. More recent efforts have pursued perception-driven navigation in which the vine rows themselves — rather than a satellite fix — serve as the primary reference, making the robot robust to the signal degradation that occurs at canopy height and in hilly terrain. Simultaneously, the maturity of three-dimensional LiDAR

sensors and point-cloud processing libraries has made dense, real-time geometric perception of row structure a practical proposition on commodity embedded hardware.

1.2 Motivation

Among the recurring tasks in vineyard management, vegetation control in the inter-row corridors occupies a prominent place. Maintaining a ground cover of grass and low vegetation reduces soil erosion and compaction, supports beneficial insect populations, and lowers the reliance on chemical herbicides, all of which align with the sustainability objectives of contemporary viticulture and with regulatory trends across the European Union [13]. The cover must nevertheless be kept below a height that would compete with the vine for water and nutrients, so periodic mowing is required throughout the growing season.

Currently, mechanised mowing is usually carried out by tractor-mounted cutters, which are effective but constrained by tractor dimensions and operator availability. Moreover, repeated tractor passes can compact the soil and risk mechanical damage to vine trunks and low-hanging infrastructure. The deployment of compact robotic mowers operating between the vine rows offers several advantages: a smaller footprint, reduced soil compaction, the ability to operate during weather windows that prevent tractor access, and the potential to gather crop-health data simultaneously with the mowing task [49].

In many cases, however, full-width coverage of the inter-row corridor floor cannot be achieved in a single automated pass. On most crop layouts, the width of the mowing actuator is narrower than the corridor itself, and the robot must maintain a safe lateral clearance from the trellis structures flanking each row, so any single traversal inevitably leaves a portion of the floor unmown. Delivering complete vegetation maintenance therefore requires the robot to coordinate its perception of the crop structure, a record of the vegetation state it has already treated, and an adaptive lateral reference that evolves from one pass to the next, so that successive swaths together cover the full corridor width.

The central objective of this work is to develop and validate the perception, mapping, and planning capabilities that enable this on a compact robotic platform equipped with a single 3D LiDAR sensor, operating under the unstructured lighting and terrain conditions of a working vineyard. The specific engineering problems that define the scope are stated in Section 1.3.

1.3 Problem Statement

The central engineering challenge addressed in this dissertation is to enable an autonomous mobile robot to perform two-pass vegetation cover maintenance in the inter-row corridor of a vineyard, using a 3D LiDAR sensor as its primary exteroceptive input. Three coupled sub-problems define the scope.

P1 — Row perception and corridor estimation. From LiDAR sweeps and externally provided localisation data, the two lateral vine rows must be detected and tracked in real time and

represented by a local geometric model. This perception must remain robust to curved rows and to partial canopy occlusion, and must yield an estimate of the robot’s lateral position within the corridor. The same row geometry must support an estimate of the navigable inter-row area, segmenting the lateral vine structures so that only the low ground vegetation at the centre of the corridor is retained for vegetation characterisation.

P2 — Local planning. The detected geometry must be converted into a smooth, continuous path that should serve as a motion reference for the robot to navigate the inter-row corridor, either following a chosen row or the detected cut line at a configurable lateral standoff while remaining well-behaved under intermittent or degraded detections. Because trajectory tracking is delegated to a separate controller, this constraint implies publishing the resulting trajectories to a pre-existing proportional–integral–derivative (PID) controller with feed-forward; the planner must also handle the transition from row following to cut-line following once a reliable cut-line estimate becomes available.

P3 — Vegetation mapping. The vegetation observed within the navigable corridor must be accumulated into a mission-level vegetation-height map, built as a by-product of normal operation and usable by downstream decision-support or crop-management systems.

Two explicit scope boundaries frame these problems. *Global localisation is assumed to be provided externally*: the map-to-robot and map-to-LiDAR transforms are processed and supplied externally via an odometry and GNSS-RTK fusion stack, which achieves centimetre-level accuracy in the open-sky conditions of vineyard corridors, and ego-motion is never estimated. *Trajectory tracking is likewise delegated to a pre-existing controller*, which is otherwise out of scope. Both assumptions are integrated as architectural interfaces and revisited in Chapter 4.

1.4 Contributions

The main contributions of this dissertation are a set of algorithms, organised into a *stack* that addresses the three functional axes of the system, that together enable the two-pass mowing strategy. A *perception* algorithm detects and tracks the lateral vine rows from 3D LiDAR, builds a geometric model of the corridor that remains robust to curved rows and partial occlusion, and estimates the robot’s lateral position within it. A *mapping* algorithm accumulates the inter-row vegetation into a height map and, from it, detects the boundary between the mown and unmown vegetation that guides the return pass. A *planning* algorithm converts the perceived geometry into smooth, continuous motion references and supervises the transitions between the row-following and cut-line-following behaviours. Together, these components address the problem stated in Section 1.3; each is developed in full in the methodology chapters.

The work described in this dissertation has been submitted for publication as a conference paper: [9].

1.5 Document Structure

The remainder of this dissertation is organised as follows.

Chapter 2 — Background provides the conceptual foundations required to read the methodology chapters: the operating context of autonomous agricultural robots, the principles of 3D LiDAR sensing, the software ecosystem, and the functional requirements derived from the operational scenario.

Chapter 3 — State of the Art surveys the related literature on robotic platforms, perception, mapping, and trajectory generation for inter-row navigation, closing with a critical synthesis of the trends and limitations identified.

Chapter 4 — System Architecture and Overview presents the proposed system architecture top-down. It discusses the two bounding scope assumptions and their failure modes, the main data-flow loops that couple the two main operating modes, and the coordinate-frame conventions.

Chapter 5 — Perception describes the perception pipeline that detects and models the vine rows and delimits the inter-row corridor.

Chapter 6 — Mapping and Cut-Line Detection describes the vegetation mapping, built within the perceived corridor, and the detection of the cut line that guides the return pass.

Chapter 7 — Local Planning describes the local planning layer that generates the motion references and supervises the transitions between behaviours.

Chapter 8 — Results and Discussion presents the experimental evaluation, conducted primarily across twenty simulated Webots scenarios and complemented by a single real-vineyard rosbag used to check internal-consistency indicators rather than to validate the complete closed-loop system, organised by component and by scenario.

Chapter 9 — Conclusions and Future Work revisits the contributions, states the limitations of the current system, and proposes concrete directions for future research.

Chapter 2

Background

This chapter provides the conceptual foundations required to follow the methodology and results of this dissertation. It opens with a brief overview of agricultural robotics and of the agronomic practice of inter-row vegetation-cover management (Section 2.1), then introduces a working taxonomy of agricultural robots and identifies where the proposed system fits within it (Section 2.2). The supporting technologies are reviewed at the level of operating principle in Section 2.3: the exteroceptive sensors that observe the crop and corridor (Section 2.3.1), and the software ecosystem in which the system is implemented (Section 2.3.2). The chapter closes by stating the functional requirements the proposed system must satisfy (Section 2.4), which serve as the criteria against which the state of the art is assessed in Chapter 3. Throughout, the distinction is maintained between *background* — the established concepts a non-specialist reader needs in order to follow the work — and the critical *review* of prior art, which is deferred to Chapter 3.

2.1 Overview of Agricultural Robotics

Robotics has been applied across the range of recurring field operations carried out in permanent row crops such as vineyards, olive groves, and orchards, including tillage, weeding, harvesting, and mowing [10]. Regardless of the task, these systems typically combine a mobile platform, one or more task-specific actuators, and a perception system that converts sensor data into information usable for navigation and control [10]; the taxonomy of platforms and actuation paradigms that follows this section situates the system studied here within that broader family (Section 2.2). The remainder of this section introduces the specific recurring operation this dissertation automates: the maintenance of an inter-row vegetation cover.

Soil health is a central concern in permanent-crop management. The improper or excessive use of intensive field operations — heavy machinery, pesticides, and chemical inputs — can degrade soil through compaction, erosion, salinisation, and water contamination [13]. Nature-based interventions that protect soil quality without compromising productivity are accordingly of considerable agronomic interest [13].

One such intervention is **cover cropping**: the deliberate covering of bare soil with herbs, legumes, or grasses grown primarily for their environmental benefit rather than for harvest. In row-based crops the cover is typically established in the inter-row corridor, where it improves soil quality and supports ecosystem balance without competing directly with the main crop for water and nutrients. Sustained ground cover has been shown to reduce soil and water loss, increase the availability of organic substrate, and regulate nutrient cycles by promoting microbial biomass, thereby reducing the need for chemical fertilisers [13]. The same benefits extend to *spontaneous* cover that emerges without being sown, which is why managing such growth is generally preferred over eliminating it with herbicides [13].

The benefits of a vegetation cover are realised only if the cover is itself *managed*: depending on the species present, it must be periodically trimmed to prevent it from competing with the crop for water and nutrients, which in practice requires recurrent monitoring and intervention [13]. This recurring maintenance task is the agronomic operation that the system developed in this dissertation automates.

2.2 Classification of Agricultural Robots and Operations

Despite the diversity of their applications, autonomous agricultural robots share a common high-level architecture. It comprises a robust mobile platform that traverses the field; one or more task-specific mechanical or chemical actuators; a perception system built from exteroceptive sensors that converts the surroundings into data usable for localisation and task execution; and a control system that coordinates these components and handles decision-making [10]. A further, increasingly important function is field data acquisition: continuous environmental sensing enables data-driven agronomic decisions, so that accurate perception has become as important as the physical operation itself [10].

Robots in this domain may be classified along two complementary axes. By **field of operation**, the literature distinguishes field robots, fruit-and-vegetable robots, and animal-husbandry robots, with most research concentrated on the first two [10]. By **task**, the same platforms are differentiated by the operations they perform — tillage, seeding, mowing, harvesting, transport, or chemical spraying — and accordingly vary in size, sensing payload, and actuator mechanism [10]. For inter-row vegetation management, the most useful classification is by **actuation paradigm**, which determines the perception and control requirements directly. Three paradigms recur in the literature.

Mechanical and physical intervention. Ground-based platforms equipped with physical tools — mowers, cutters, hoes, or weeding implements — remove or manage vegetation directly [5, 42]. They are well suited to tasks requiring close, repeatable interaction with the crop, where accurate lateral positioning and safe actuation are critical. Within this paradigm, two design concerns dominate: the *platform* itself — drivetrain configuration, steering geometry, and an open software architecture — and the *end-effector* — the choice of cutter type and the means by which the cutting height is regulated. The robot considered in this dissertation belongs to this family.

Precision chemical application. A second paradigm reduces rather than eliminates chemical input by applying herbicide selectively. Ground-based spot-spraying systems detect weeds and dose them locally, achieving weed suppression comparable to broadcast spraying while substantially reducing herbicide volume [4]. Aerial platforms extend the same principle to large or hard-to-access plantations [43]. These approaches retain a chemical footprint and a fundamentally different actuation principle, and are therefore peripheral to a mechanical mowing platform.

Non-chemical alternatives. A third, emerging paradigm replaces both blade and chemical with directed energy. Laser-based weeding couples vision-based detection with precise energy delivery to damage individual weeds without herbicides [52]. Such systems are demonstrated chiefly for discrete weed removal rather than continuous cover management, but they reinforce a broader trend: perception is increasingly used not only to *navigate* but also to *trigger* fine-grained intervention based on local vegetation state [35].

Across all three paradigms, ground robots with mechanical actuators emerge as the natural foundation for cover-crop maintenance in permanent crops, offering a favourable balance of selectivity, robustness, and operational safety while providing a platform on which perception-driven mowing can be built [5, 38, 42]. The narrow corridors and proximity to trellis structures that characterise vineyards have motivated a dedicated line of research on compact autonomous platforms specifically for these environments [38].

2.3 Support Technologies and Foundations

The chain from raw measurement to motion reference rests on a small set of supporting technologies: the exteroceptive sensors that observe the crop and corridor, and the software ecosystem in which perception, planning, and mapping are implemented. This section reviews each at the level of operating principle; the critical comparison of sensing modalities and the rationale for the sensor choice made in this work are provided in Section 3.2.1.

2.3.1 Sensors and Operating Principles

RGB cameras. Passive RGB cameras are the most widely used exteroceptive sensors in agricultural robots, owing to their low cost, high resolution, and rich appearance cues. They capture colour and texture information that supports vegetation detection, boundary identification, and semantic classification wherever appearance is informative [35]. Their principal limitations are an absence of native metric depth and a sensitivity to illumination conditions: variable lighting, shadow, and dust degrade colour-based perception.

RGB-D cameras. RGB-D devices add a dense, short-range depth channel co-registered with the colour image. Depth is recovered by one of three operating principles: *structured light*, in which a known pattern is projected and its deformation triangulated; *time-of-flight* (ToF), in which the round-trip time of emitted light is measured per pixel; and *stereoscopy*, in which disparity between two calibrated viewpoints yields range. The resulting per-pixel colour-plus-depth data

are well matched to vineyard inter-rows at short range, where the vegetation lies close to the robot and canopy shape can be observed directly [23]. Performance degrades under strong sunlight and at longer ranges, which limits RGB-D sensing to short-range tasks.

LiDAR. Light Detection and Ranging (LiDAR) is a technology based on the emission of laser pulses and the measuring of the time of their return to reconstruct three-dimensional space, producing a point cloud that captures the geometric structure of the environment with high precision [40]. Because it provides its own illumination, LiDAR sensors yield accurate, lighting-independent range over longer distances than RGB-D cameras. These properties have made LiDAR the sensor of choice for grass and crop height measurement and for elevation mapping over uneven terrain [21, 40]. The sensor used in this work is a three-dimensional rotating LiDAR sensor, from which both row detection and vegetation mapping are derived as a single modality.

GNSS and proprioception. Global Navigation Satellite Systems (GNSS), frequently fused with odometry and inertial sensing, provide a global positioning reference that supports mapping consistency and repeatable corridor coverage. In open-sky vineyards, real-time kinematic correction (RTK) brings GNSS positioning to centimetre-level accuracy, making it reliable for mission-level localisation on platforms that operate outside the canopy shadow. The present work assumes that this external localisation is available and delivered through the ROS 2 TF tree; ego-motion estimation is not addressed.

2.3.2 Software Ecosystem and Tools

ROS 2. The proposed system is implemented in **ROS 2 (Humble Hawksbill)**, the second generation of the Robot Operating System. ROS 2 is an open-source framework that provides libraries, tools, and conventions for robotics software development; built on the Data Distribution Service (DDS) middleware, it targets real-time and safety-critical operation and supports reliable inter-process communication across heterogeneous platforms. Its *node-and-topic* abstraction encourages a modular decomposition of the system into cooperating processes. The *TF2* subsystem maintains the time-varying geometric relationships between coordinate frames on which all perception and planning operations depend.

Point Cloud Library (PCL). PCL supplies the point-cloud processing primitives used throughout the perception pipeline: spatial cropping (`CropBox`, `CropHull`), voxel-grid downsampling, statistical outlier removal (SOR), and model fitting by Random Sample Consensus (RANSAC).

Eigen and OpenCV. **Eigen** provides the linear-algebra substrate — vectors, matrices, and the least-squares and eigen-decomposition routines on which line and plane fitting rely. **OpenCV** contributes image-processing and matrix operations, used for optional projection of detected geometry onto the camera image and for colour-mapped export of height maps.

grid_map. The `grid_map` library [16] provides a well-established, multi-layered, robot-centric grid-map abstraction with an *efficiently re-positionable* circular-buffer storage scheme: the map is re-centred by a buffer-index shift rather than a data copy as the robot moves, keeping maintenance cost constant regardless of distance travelled. This property makes it well suited to maintaining a rolling local window updated at sensor rate. Multiple statistical layers — count, mean height, minimum, maximum — are maintained per cell, which the cut-line detector exploits to build one-dimensional height profiles of the inter-row corridor.

Together, ROS 2 and these libraries allow the system to be structured as thin orchestration nodes (subscriptions, TF lookups, publishers, timers) wrapping framework-agnostic algorithmic namespaces — a separation that improves testability and enables the geometry and mathematics to be unit-tested in isolation from the robotics middleware.

2.3.3 Simulation Environment

Development and testing of the system are supported by the **Webots** robotics simulator [36], an open-source environment that provides physics-based simulation of mobile robots together with their sensors and actuators. Webots is used here chiefly to exercise the local planner: candidate reference trajectories can be generated and tracked in a controlled, repeatable setting, and specific situations — among them the curved corridors and the transition from row following to cut-line following — can be reproduced on demand before and alongside field trials. To keep this representative, the simulated world is modelled with a structure similar to a real vineyard, with parallel rows of vine-like obstacles forming inter-row corridors of both straight and curved layout. The simulator exposes the same interfaces as the physical platform, a 3D LiDAR stream and the localisation transforms, so that the perception, mapping, and planning nodes run unchanged across simulation and hardware.

2.4 Functional Requirements

The following requirements state the criteria the proposed system must satisfy. Requirements R1–R6 consolidate the baseline expectations for an inter-row mowing robot; requirements R7–R9 are specific to the cut-line-following scope adopted in this dissertation and are not fully addressed by any single prior system, as established in Section 3.5. Each is phrased as a verifiable engineering goal.

R1 — Vegetation characterisation and height estimation. The system shall identify and map the inter-row vegetation cover and provide an estimate of its canopy height, with sufficient resolution to differentiate stages of growth and density across the corridor.

R2 — Row detection and corridor geometry. The system shall detect and track the crop rows flanking the robot from LiDAR data and recover the geometry of the inter-row corridor as the reference for navigation.

R3 — Dynamic actuator control. The system shall adjust the cutting height of the mechanical actuator in real time, synchronised with the robot's motion and driven by perceived vegetation height, to deliver a terrain-adapted cut.

R4 — Operational area discrimination. The system shall distinguish previously maintained zones from untreated areas and exploit this distinction to maximise newly covered area while minimising redundant passes.

R5 — Operational safety and boundary constraints. All motion shall remain strictly within the physical boundaries of the inter-row space, keeping a safe distance from the crop rows, trellis structures and any obstacles that bound the corridor, and the platform shall be brought safely to rest whenever the corridor can no longer be reliably perceived.

R6 — Autonomous corridor navigation. The system shall navigate the inter-row corridor autonomously from on-board LiDAR sensor and the localisation transforms alone, without reliance on GNSS, producing a continuous reference trajectory that follows both straight and curved rows.

R7 — Cut-line detection. The system shall detect the cut line, the lateral boundary between the already-mown swath and the adjacent standing vegetation, from the accumulated vegetation-height map, and report its position, orientation and validity at each traversal.

R8 — Cut-line following. The system shall follow a detected cut line, offsetting the reference path by half the mechanical actuator width so that the cutter meets the standing vegetation at the line, in order to mow the strip adjacent to the previously cut swath.

R9 — Integrated two-pass operation. The system shall combine row following and cut-line following into a single continuous mission, a forward pass guided by the rows and a return pass guided by the cut line, so as to maximise the inter-row area covered while preserving trajectory continuity across the transition and tolerating the temporary loss of detection on one side.

Requirements R1–R6 are addressed, to varying degrees, by existing inter-row platforms, whereas R7–R9 are the specific contribution of this dissertation. Chapter 3 assesses the state of the art against these requirements and locates the gap that R7–R9 are intended to fill.

Chapter 3

State of the Art

This chapter critically reviews the body of work on which the perception, mapping, and local-planning components of this dissertation are built, and identifies the gap that the work addresses. The review is organised thematically around the three technical axes that mirror the architecture of the proposed system. A brief survey of robotic platforms for inter-row vegetation management first situates the work within the wider field (Section 3.1). The first axis treats the detection of crop rows from LiDAR and the detection of the cut line that separates mown from unmown vegetation (Section 3.2); the second treats the mapping representations — in particular sparse 2.5D height maps — from which both navigation and the cut line are derived (Section 3.3); and the third treats the generation of smooth, continuous trajectories and the supervisory logic that arbitrates between behaviours (Section 3.4). The chapter closes with a critical synthesis of the principal trends and limitations identified across these axes (Section 3.5). Prior work is reviewed in cycles — an approach is described, then its limitation is identified. Depth is proportional to relevance: works most directly underpinning the proposed system are analysed at the level of algorithm and filter, while the remainder are treated more concisely.

3.1 Robotic Platforms for Inter-Row Vegetation Management

As established in Chapter 2, mechanical-intervention ground robots are the appropriate platform class for continuous cover-crop mowing in permanent crops. Bakker *et al.* [5] exemplify the platform-centric strand of this work, emphasising the systematic engineering of a robust weeding platform — drivetrain, steering, and open software architecture — as the enabler of long-term experimentation with perception and actuation; their work established modular design practices that remain foundational. Reiser *et al.* [42] exemplify the application-centric strand, integrating an autonomous electric carrier with a mechanical intra-row weeding tool for vineyards and validating it through extensive field trials. Reviews of robotic operations in orchards and vineyards confirm that ground robots dominate tasks requiring precise proximity to the crop [38].

Beyond mechanical removal, vegetation management has been pursued through precision chemical application — ground-based spot-spraying [4] and aerial spraying for large plantations [43] — and through non-chemical actuation such as laser weeding [52]. These differ fundamentally from continuous mechanical mowing and lie outside the scope of this work. The remainder of the chapter concentrates on the perception, mapping, and planning that a mechanical inter-row mower requires.

3.2 Axis 1 — Crop-Row Detection and Cut-Line Detection

Detecting and tracking the crop rows is the perceptual foundation of inter-row navigation; detecting the cut line is the perceptual foundation of the return-pass behaviour central to the proposed system. This axis reviews four intertwined strands: the sensing modality for row detection (Section 3.2.1), the geometric algorithms that extract and validate row lines (Section 3.2.2), the delimitation of the corridor and separation of lateral structures for mapping (Section 3.2.3), and the detection of the mown/unmown boundary (Section 3.2.4).

3.2.1 From Vision to LiDAR — a Motivated Sensor Choice

Camera-based approaches to crop-row detection have a long history and are effective in early growth stages under good illumination. Section 1.2 argued, on robustness grounds, that LiDAR is better suited to the specific conditions of open-sky vineyard navigation; this section provides the cited evidence that underlies that argument.

The transition from vision to geometry is well documented in the literature. A recent survey of crop-row detection methods confirms that vision-based detectors perform well under controlled conditions but degrade under variable lighting, dust, and heavy canopy occlusion, which has driven a substantial migration towards LiDAR-based detection [32]. The argument for geometry over appearance recurs in cut-edge detection as well: a method based on the three-dimensional profile of the scene is largely independent of crop colour and texture and far less sensitive to shadow and daylight change than an appearance-based one. An additional practical consideration favours LiDAR: camera-based systems are sensitive to intrinsic and extrinsic calibration parameters, and even small calibration errors can translate into significant lateral displacement errors in the estimated row position; an active ranging sensor avoids this dependency, as range is recovered from the time of flight rather than from geometric triangulation across calibrated viewpoints [7, 18]. This dissertation adopts a purely geometric, LiDAR-based detector for these reasons.

3.2.2 Geometric Row Extraction and Temporal Validation

LiDAR-only row detection. Malavazi *et al.* [34] proposed a LiDAR-only navigation algorithm that extracts row lines from two-dimensional point clouds without prior knowledge of the row

spacing. The core is an energy-minimisation model-fitting procedure, **PEARL**, in which candidate line models compete to explain the points, penalising both fitting error and spatial inconsistency. The refined algorithm, **RUBY**, eliminates model pairs that are not mutually parallel, while the unconstrained variant, **OPAL**, is shown to be markedly less robust. A subsequent validation stage checks that the retained models are parallel and equidistant before driving any control decision, and temporal filtering is applied to stabilise the estimates across successive scans. The hard parallelism constraint is, however, also the principal limitation: by construction it assumes the two flanking rows are straight and mutually parallel — an assumption that holds in regularly planted straight vineyards but is violated in curved rows, on headland approaches, or where one side is missing.

Higuti *et al.* [18] achieved reliable under-canopy navigation from a single two-dimensional LiDAR sensor. The pipeline cycles through *Initialise* → *Update* → *Estimate* → *Validate* → *Control*: the Estimate stage restricts the scan to a rectangular region of interest per side, fits a row line by least squares, and the Validate stage grades the result against the previous estimate, rejecting temporally inconsistent fits. This work demonstrates that explicit, reactive temporal validation is both necessary and sufficient for reliable row following. Its limitation, shared with Malavazi *et al.* [34] and with most reactive LiDAR row detectors, is the *single-horizon formulation*: a fixed region of interest per side yields a single look-ahead estimate, and the row is modelled as locally straight. This is precisely the formulation least robust to curved or undulating rows, where the corridor direction changes within the sensor’s range.

Three-dimensional and map-free detection. In structured permanent crops the regular geometry of the rows can be exploited explicitly. Bertoglio *et al.* [7] presented a map-free LiDAR system for vineyards that performs in-row, turn, and end-of-row navigation by fitting the parallel-row structure directly, reporting centimetric in-row displacement errors in simulation; its map-free, reactive philosophy — navigating from the rows perceived at the current instant rather than from a pre-built global map — is representative of a broader trend away from prior maps and towards on-line perception. Biglia *et al.* [8] detected and localised vine rows in three-dimensional point clouds through key-point extraction and density-based clustering, handling arbitrary row orientations, curvilinear layouts, and missing plants, but only as an offline process. In orchards, Zhou *et al.* [51] reconstructed dense clouds with a rolling two-dimensional LiDAR sensor and extracted tree-row lines from trunk geometric features by RANSAC, achieving low distance and angular errors but depending on clearly visible trunks — a cue absent in low cover-crop vineyards. Liu *et al.* [29] addressed the over-canopy case with a crop-agnostic detector robust to crop type, growth stage, weeds, curved rows, and discontinuities, navigating an entire field without GNSS, at the cost of assuming the sensor looks down onto the canopy from above. Shi *et al.* [45] combined RGB vision and two-dimensional LiDAR sensor to extract navigation lines in pomegranate orchards, improving robustness in dense plantations but introducing the calibration and fusion overhead inherent to multi-sensor systems. Two trends are visible across this group: a movement from offline towards on-line, map-free detection, and a persistent dependence on assumptions —

Table 3.1: Representative crop- and vine-row detection approaches.

Work	Sensor	Method	Environment	Main limitation
Malavazi <i>et al.</i> [34]	2D LiDAR	PEARL/RUBY energy minimisation + parallelism	Row crops	Mostly simulated; straight rows only
Higuti <i>et al.</i> [18]	2D LiDAR	Least-squares ROI + temporal validation	Maize/sorghum	Under-canopy; straight rows
Liu <i>et al.</i> [29]	3D LiDAR	Crop-agnostic over-canopy detection	Arable fields	Over-canopy assumption
Bertoglio <i>et al.</i> [7]	3D LiDAR	Parallel-row fitting; end-row detection	Vineyards	Larger error on real data
Biglia <i>et al.</i> [8]	3D point cloud	Key points + density clustering	Vineyards	Offline map processing
Zhou <i>et al.</i> [51]	Rolling 2D LiDAR	Trunk features + RANSAC	Orchards	Requires visible trunks
Shi <i>et al.</i> [45]	RGB + 2D LiDAR	Vision-LiDAR fusion	Orchards	Calibration/fusion overhead

straight or parallel rows, visible trunks, over-canopy viewpoints — that narrow each method’s range of applicability.

RANSAC conditioning. A methodological element recurs across these geometric pipelines: robust model fitting by RANSAC [7, 34, 51] is used to extract row lines or the supporting planar structure of the canopy while rejecting foliage and ground outliers. Standard RANSAC is, however, prone to fitting spurious planes in noisy, sparse clouds where random minimal samples can support geometrically valid but semantically meaningless surfaces. Li *et al.* [27] addressed this with an improved RANSAC for three-dimensional plane segmentation that samples within normal-distribution-transform cells, biasing the hypotheses towards locally coherent surfaces and reducing spurious fits. The reliability of geometric row extraction therefore depends heavily on conditioning the fitting process with prior structure, rather than applying generic model fitting to raw returns.

Table 3.1 summarises the representative row-detection approaches along the dimensions most relevant to inter-row mowing.

3.2.3 Corridor Delimitation and Separation of Lateral Structures

Detecting the rows serves a second purpose beyond steering: it makes it possible to decide *which* LiDAR returns belong to the inter-row corridor and which belong to the lateral structures flanking it. This distinction is critical for the mapping component (Section 3.3), because the height map and the cut-line detector are meaningful only over the ground vegetation. If raw returns were inserted indiscriminately, the tall returns from the vine canopy, trunks, and trellis would inflate the per-cell height statistics and produce a map polluted by the very structures the robot must avoid, requiring post-hoc cleaning before it could support cut-line detection or serve as a vegetation

product. Cleanly separating grass from the lateral vine structures *before* accumulation is therefore a prerequisite for a reliable 2.5D map, not a cosmetic refinement.

The literature addresses this filtering problem in three broad ways. The most common is a *fixed, axis-aligned region of interest* or pass-through window: Nehme *et al.* [39] crop the cloud to a fixed lateral band before tracking vineyard structure, and Zhou *et al.* [51] use a fixed rectangular pass-through aligned to a row centreline to remove the vehicle body and adjacent rows. A second family *thresholds by height or fits a ground plane*: Andújar *et al.* [2] discriminate crop, weeds, and soil using height and reflectance thresholds; Liu *et al.* [29] raise an estimated ground plane to the cloud centroid and discard everything below it to isolate canopy tops; and Kateris *et al.* [23] remove vine and soil returns with a simple height filter that exploits the fact that weeds lie between the soil and the vine canopy. A third, less common family uses an *explicit polygon*: Popić *et al.* [41] combine voxel reduction, RANSAC ground removal, and a fixed trapezoidal polygon filter, but the trapezoid is a pre-designed navigation mask rather than a region derived from the perceived crop, and it is not used to construct a vegetation map. Notably, systems that do recover rich row geometry — such as the density-based segmentation of Biglia *et al.* [8] and the line extraction of Malavazi *et al.* [34] — stop short of re-using that geometry as a mask for downstream mapping.

What these approaches share is that the delimiting region is decoupled from the perceived crop geometry: it is a fixed box, a height or plane threshold, or a pre-designed polygon, none of which adapts to the actual shape of the corridor ahead. A delimiting region that adapts to the perceived, possibly curved, structure of the corridor — rather than a rigid template — remains an underexplored design choice, and is the gap this work addresses through the canopy-dispersion-adaptive virtual boundary (contribution C1, Section 5.9).

3.2.4 Detection of Mown Areas and Cut Lines

A defining feature of the proposed system is that, after a forward mowing pass, it detects and follows the boundary between the mown swath and the uncut remainder. This connects two strands of the literature: the segmentation of mown from unmown vegetation, and the detection of the cut/uncut crop edge for the automatic guidance of combine harvesters — a problem geometrically equivalent to the mown/unmown one.

Appearance-based mowing boundary detection. Fukukawa *et al.* [17] detected mowing boundaries for an autonomous lawn mower using Gabor-filter texture features. Schepelmann *et al.* [44] segmented grass from non-grass regions for a robotic lawnmower. These confirm that the mown/unmown distinction is perceptible from appearance, but the reliance on colour and texture inherits the illumination sensitivity that motivated the move to geometry throughout this work.

Harvester cut-edge following. The combine-harvester literature provides the closest analogues for cut-edge following as a navigation strategy. Benson *et al.* [6] established the paradigm with a machine-vision guidance system that located the cut/uncut edge from a head-mounted camera and produced a lateral steering signal — an early demonstration that the crop edge itself can serve

as the navigation reference. Cho *et al.* [11] fused a laser range finder with GNSS to build a three-dimensional terrain map and detect the uncut crop edge by RANSAC, recovering both the lateral offset and the crop height — demonstrating the feasibility of a steering signal from a three-dimensional geometric representation. Luo *et al.* [31] detected multi-crop harvesting edges from a crop elevation image obtained by stereo vision, illustrating the geometry-from-elevation principle with a different sensor.

Kneip *et al.* [26] combine a stereo elevation map of the terrain ahead with Expectation–Maximisation on a Gaussian mixture of cell heights to segment crop from ground cells, stabilised over time by a Bayesian filter. Edge detection then proceeds row-wise: each row of the grid is correlated against a step function to locate a candidate edge cell, and the full set of candidates is fitted by robust linear regression to produce a single crop-edge line. The pipeline is notable for combining a height-map representation, a one-dimensional profile-based edge response, and temporal stabilisation — elements useful when designing an analogous detector for LiDAR on a mowing robot. The proposed system addresses a different setting (a mobile ground robot rather than a harvester, a robot-centric `grid_map` populated by LiDAR) and adopts different processing choices; the relevant comparison is carried out in Chapter 6.

Kim *et al.* [25] localised the uncut crop region with weak supervision and little annotation effort, after which the edge can be recovered by classical image processing — a data-driven route to the same boundary, at the cost of training data and on-board inference. The cut-edge literature confirms the feasibility of edge-following as a navigation strategy and illustrates the spectrum of sensor and algorithmic choices available, from monocular vision through stereo and LiDAR to learning-based detectors.

3.3 Axis 2 — Mapping and 2.5D Height Mapping

To close the loop between perception and planning, vegetation must be represented spatially. The proposed system requires two distinct mapping products with fundamentally different properties. The first is a **mission-level sparse map**: a compact, long-lived representation of inter-row vegetation height that accumulates evidence across the full mission trajectory without knowing the corridor extent in advance. Memory efficiency is essential; dense grid maps with fixed cell sizes are impractical at mission scale without prior knowledge of the field extent, which motivates a sparse, hash-indexed structure. The second is a **real-time local map**: a small, robot-centric rolling window maintained at navigation frequency and used exclusively to detect the cut line; it need only cover the immediate surroundings of the robot and can afford to be dense within that window. These two products are distinct both in purpose and in implementation, and the literature reviewed below should be read with that distinction in mind.

3.3.1 Vegetation Height Estimation

Vegetation height is the physical quantity these maps encode and a direct indicator of growth stage, need for intervention, and treatment effectiveness (requirement R1). Continuous estimation is pre-

dominantly addressed through three-dimensional reconstruction, computing height as the vertical difference between the canopy and the underlying ground. Obanawa *et al.* [40] demonstrated accurate, robust grass-height measurement with a portable LiDAR under real field conditions, validated against structure-from-motion photogrammetry — direct support for the choice of LiDAR as the height sensor. Jimenez-Berni *et al.* [21] estimated plant height, ground cover, and above-ground biomass in wheat from dense LiDAR, establishing that the quantity the mission-level map reports is a well-founded, reliably extractable descriptor. At shorter range, Kateris *et al.* [23] generated spatial weed-height maps along vineyard inter-rows from RGB-D, providing a close functional comparator for the 2.5D vegetation product proposed here.

Vision- and learning-based pipelines extend these capabilities. Machine vision has been applied to plant-height estimation in protected crop facilities [20]; multimodal LiDAR-and-vision estimation of cover-crop biomass implicitly relies on an accurate height-and-structure representation [22]; and broader phenotyping reviews map the convergence of 3D reconstruction, learning, and spatial mapping [35]. Where precise metric height is unnecessary, discrete estimation suffices: random-forest classification of lawn-grass length from mower sensor features [53] and visual grass/non-grass segmentation [44] classify vegetation into a small number of state categories, sufficient to flag already-treated areas (requirement R4).

3.3.2 Grid-Map Frameworks and Real-Time Local Mapping

The **universal grid-map library of Fankhauser and Hutter** [16] is the substrate for the local, real-time map used in cut-line detection. Their library provides a multi-layered, robot-centric grid map with an *efficiently re-positionable* circular-buffer storage scheme in which the map is re-centred by a buffer-index shift rather than a data copy as the robot moves, keeping maintenance cost constant regardless of distance travelled. This property makes it well suited to a small rolling window updated at sensor rate. The multi-layer abstraction allows several statistics — count, mean height, minimum, maximum — to be maintained per cell within the local window, which the cut-line detector exploits to build one-dimensional lateral height profiles. The library does not address long-range, unbounded accumulation; for the mission-level sparse map, a different indexing strategy is required and `grid_map` is not used.

The feasibility of maintaining dense local maps at navigation rate is established by Miki *et al.* [37], who present a GPU-accelerated elevation-mapping pipeline that moves the most expensive stages — point-cloud registration and ray-casting — onto the GPU, enabling high-frequency updates for fast locomotion. The local map for cut-line detection operates on a far smaller window and does not itself require GPU acceleration, but the existence of that head-room is relevant to the real-time requirement R6.

At the mission-level end, efficiency of representation becomes the dominant concern. Shi-raishi *et al.* [46] aggregated raw point clouds into 3D Gaussian clusters projected into 2.5D elevation maps, reducing memory and computation while preserving geometric fidelity — a concern directly relevant to long-mission sparse accumulation. Dergachev *et al.* [14] demonstrated that a 2.5D elevation map is a sufficient world model for closed-loop navigation of a differential-drive

robot in uneven terrain, validating the choice of height-map-based representations over full 3D models. Keen and Berns [24] addressed the generation of elevation maps specifically for planning and navigation in rough natural terrain. Collectively, these works support the two-tier mapping design: a sparse, hash-indexed height map for mission-level vegetation characterisation, and a dense rolling `grid_map` window for real-time cut-line detection.

3.4 Axis 3 — Smooth Trajectory Generation and Behaviour Supervision

Local planning bridges the high-level navigation objective and the tracking controller: it must produce safe, smooth, and continuous motion references that follow the rows while reacting to perception on-line. This axis reviews reactive row-following (Section 3.4.1), the parametric-curve representations that guarantee continuity and bounded curvature (Section 3.4.2), and the supervisory logic that arbitrates between behaviours (Section 3.4.3). Because trajectory *tracking* is delegated to a pre-existing controller in this work, optimisation- and deformation-based tracking strategies are reviewed only as context; the contribution lies in the geometric construction and supervision of the reference.

3.4.1 Reactive Row Following

A large and effective class of agricultural navigators is reactive: assuming a nominal trajectory aligned with the inter-row centre, they maintain row-following from local perception cues. Aghi *et al.* [1] used RGB-D depth to identify the inter-row centre and steer through proportional control, achieving reliable row-following across vineyard layouts. Liu *et al.* [28] estimated traversability heatmaps from RGB-D images with deep learning and converted them into preferred paths, enabling smooth navigation and row switching from local perception alone. Maini *et al.* [33] addressed autonomous weed mowing with an on-line coverage-planning algorithm that adapts the trajectory to weed detections acquired during operation, prioritising untreated regions and reducing redundant coverage — a coverage-aware behaviour conceptually related to the two-pass mowing strategy and to requirement R4.

A common limitation of purely reactive planners is that the trajectory is treated as *fixed*: it is maintained rather than reshaped according to task-specific objectives or the corridor structure ahead. The proposed system departs from this by treating the reference as a continuous curve whose geometric anchor can switch between rows without the robot physically deviating from the corridor.

3.4.2 Smooth Trajectory Generation with Bézier and Spline Curves

For a non-holonomic robot, the reference must be continuous in position, heading, and ideally curvature, so that the tracking controller is never presented with a discontinuity (requirement R8). Parametric curves are the standard tool. Elbanhawi *et al.* [15] generated continuous-steering paths

for car-like robots using **B-spline** curves, exploiting two properties decisive for confined corridors: the curve respects a maximum-curvature constraint compatible with the vehicle’s minimum turning radius, and the *local support* of the basis means that modifying one control point perturbs only a bounded portion of the path. These properties — curvature-bounded, locally editable smooth paths — motivate a control-point-based parametric reference for a corridor-confined mower.

Simba *et al.* [47] specialise this to real-time operation with **piecewise Bézier** curves. Their algorithm generates trajectories that are C^2 -continuous in curvature from a small set of operator-specified via points, using the *de Casteljau* subdivision scheme and inserting a higher-order (quintic) Bézier segment into high-curvature regions so that a maximum-curvature bound is respected exactly where it binds. The construction is incremental: new via points can be incorporated in real time without re-planning the whole path. Simba *et al.* [47] inform — rather than dictate — the trajectory representation adopted here, which is likewise based on Bézier curves but is deliberately simpler: cubic segments whose control points are derived from the perceived row geometry, with only heading continuity (G^1) guaranteed by construction rather than full C^2 curvature continuity, and with via points derived autonomously from perception rather than specified by an operator. Liu *et al.* [30] recently developed a curvature-continuous (G^2) path-smoothing method based on piecewise quadratic Bézier curves — a higher-continuity variant of the same family.

Two further families address trajectory *adjustment* rather than generation. Optimisation-based control re-plans the reference subject to dynamic constraints: nonlinear model predictive control (NMPC) has been applied to trajectory planning for skid-steer robots in agriculture [3] and to GPS-free vineyard navigation [48]. Trajectory-deformation methods such as the improved timed-elastic-band approach of Wu *et al.* [50] continuously reshape a pre-defined path around obstacles. Because tracking is delegated to an existing controller here, these strategies are out of scope; they delimit rather than ground the present contribution.

3.4.3 Behaviour Supervision with Finite-State Machines

The proposed planner must switch between several behaviours — following the desired row, switching the geometric reference to the opposite row when the desired one is temporarily unobservable, following the cut line, and halting safely — and therefore requires an explicit supervisory layer. Finite-state machines (FSMs) are a long-established formalism for such coordination. Conner and Willis [12] proposed a flexible navigation architecture that separates global planning from local planning and control and coordinates the behaviours through a hierarchical FSM integrated with the ROS navigation stack. The separation of *stateless computation* from *stateful decision-making* is a principle the proposed system adopts: row detection and cut-line perception are organised as stateless, framework-agnostic computations, while a single planning node owns the FSM and all stateful planning decisions.

The choice of an FSM over the increasingly popular behaviour trees is supported by Iovino *et al.* [19], whose comparison shows that, while behaviour trees scale better as task complexity grows, FSMs remain simpler, more transparent, and well suited to coordination problems with a small, well-defined set of states and transitions — precisely the regime of the proposed

system, whose behaviours number a handful. The supervisory FSM developed in this work adds **hysteretic, saturating evidence accumulators** that debounce switching decisions against transient perception failures, operationalising the temporal-validation principle of Higuti *et al.* [18] at the level of behaviour arbitration and realising the graceful-degradation requirement R9: when per-side safety scores indicate that the bilateral-corridor assumption no longer holds, the FSM transitions to a controlled stop rather than committing to a spurious reference.

3.5 Critical Synthesis and Research Gap

The literature reviewed in this chapter establishes mature foundations for each individual component of an autonomous inter-row mowing robot. Ground robots with mechanical actuators are well suited to vegetation management in permanent crops [5, 38, 42]; LiDAR-based detectors recover crop and vine rows under canopy and varying illumination [7, 8, 18, 34, 51]; vegetation height is robustly estimated and cast into 2.5D representations [16, 21, 23, 40, 46]; the cut/uncut boundary can be detected from both appearance and geometry [6, 11, 17, 26]; and smooth references can be generated with parametric curves [15, 47] under finite-state-machine supervision [12, 19]. Considered in isolation, each sub-problem can fairly be regarded as well studied.

A recurring limitation emerges, however, when these methods are weighed against the operating conditions set out in Section 1.3. Many of the most reliable row detectors assume straight, mutually parallel rows [18, 34], or particular viewpoints and clearly visible structures [29, 51] — assumptions that weaken in curved corridors, under partial occlusion, or where one row is only intermittently observed. The separation of the corridor from the lateral crop structures is likewise handled predominantly by fixed regions of interest or static height thresholds [2, 23, 39, 41] that do not adapt to the perceived shape of the corridor ahead. These choices are reasonable in their original settings, but sit uneasily with a narrow, possibly curved vineyard inter-row in which the perceived geometry must remain trustworthy throughout the traversal.

Cut-edge following has been demonstrated as a viable navigation strategy, but largely for harvesters relying on vision or stereo and pre-built elevation maps [6, 11, 26, 31], and rarely with an explicit, interpretable measure of when a detection should be trusted. On the planning side, reactive row-followers tend to treat the reference as fixed rather than reshaping it according to the task and the structure of the corridor ahead [1, 28]. Taken together, these trends suggest that the principal difficulty in this domain lies less in the individual components than in their integration under the specific constraints of a vineyard inter-row — a setting that stresses the robustness, adaptivity, and continuity assumptions on which much of the surveyed work rests. It is within this space that the present work is positioned.

Chapter 4

System Architecture and Overview

This chapter presents the proposed system using a top-down approach, so that the reader has an overview of the system before any single module is detailed. It first describes the overall architecture and the way data flow between its parts (Section 4.1), then examines each processing node in turn, stating its responsibility and its interfaces with the external subsystems on which it depends (Section 4.2). The software namespaces and the data structures shared across the system are then enumerated (Section 4.3), and the chapter closes with the coordinate frames and notation conventions used throughout the remainder of the dissertation (Section 4.4).

4.1 System Overview

The proposed system is organised as three single-responsibility processing ROS2 nodes — a *perception* node, a *local-planning* node, and a *mapping* node — that cooperate to realise the two-pass mowing strategy introduced in Chapter 1. Each node owns one concern: the perception node converts sensor data into a geometric model of the corridor, the planning node turns that model into a motion reference, and the mapping node accumulates the vegetation observed within the corridor and detects the boundary that guides the return pass. The decomposition into three nodes is a deliberate design choice: it lets each concern run at its own rate, isolates the stateful decision-making in a single place (the planner), and allows each node to be replayed and tested independently of the others. The overall architecture and the messages exchanged between the nodes are shown in Figure 4.1.

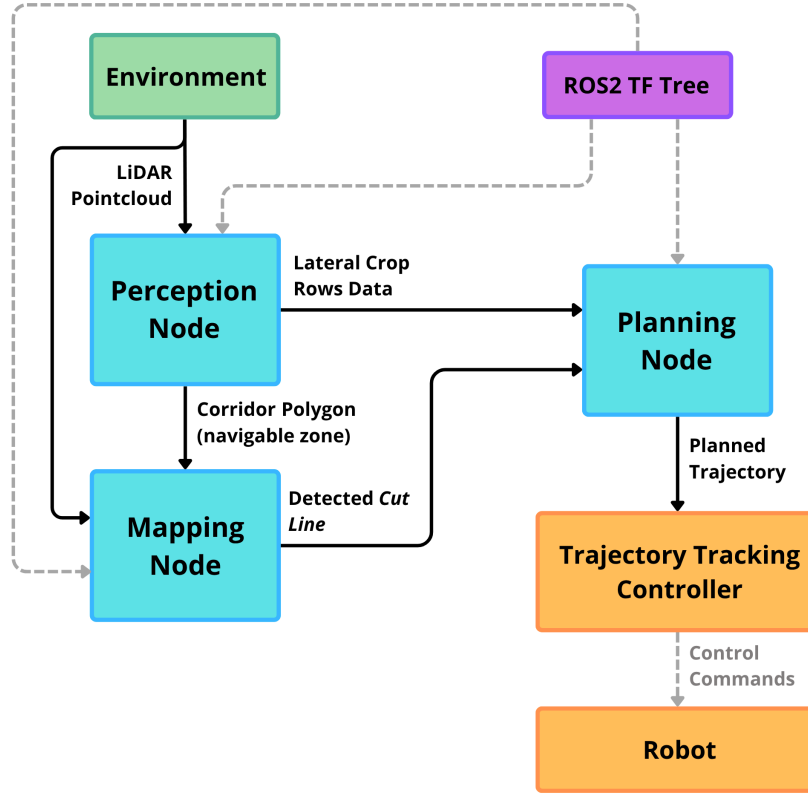


Figure 4.1: System architecture: three single-responsibility nodes, the message contracts between them, and the external subsystems with which the system exchanges data. Perception publishes the corridor polygon that configures the mapping node’s spatial filter on-line, closing the perception–mapping loop.

The system exchanges data with three external subsystems, which appear at the boundary of Figure 4.1 and which are best understood as the interfaces of the architecture rather than as separate assumptions. A 3D LiDAR provides the only exteroceptive measurements that enter the system; an upstream localisation stack supplies the robot pose through the transform (TF) tree; and a trajectory tracking controller consumes the references the planner produces. Each interface is described, together with the node that owns it, in Section 4.2. The figure also encodes, by colour, which parts of the system were developed in this work: the blocks drawn in blue — the perception, mapping, and planning nodes — are the contribution of this dissertation, whereas the blocks drawn in orange, the trajectory tracking controller and the robot platform, are pre-existing and out of scope. The ROS 2 TF tree, drawn in purple, is likewise a pre-existing interface, used throughout as the main pose reference.

Internally, two data-flow loops couple the nodes and correspond to the two operating modes of the mission. In the *row-following mode*, the LiDAR sweep drives the perception node, which detects the lateral rows and publishes a geometric model from which the planner derives the forward-pass reference. In the *cut-line-following mode*, a second, longer loop becomes active: the corridor polygon produced by perception bounds the point cloud accumulated by the mapping

Table 4.1: The three processing nodes and their principal message interfaces.

Node	Consumes	Produces
<code>perception_node_v2</code>	LiDAR point cloud; robot pose (TF)	<code>RowDetectionResult</code> ; corridor-boundary polygon
<code>local_planning_node</code>	<code>RowDetectionResult</code> ; <code>CutLine</code> ; robot pose (TF)	Reference trajectory to the tracking controller
<code>mapping_node</code>	LiDAR point cloud; corridor polygon; robot pose (TF)	<code>CutLine</code> ; 2.5D vegetation-height map

node, the mapping node detects the cut line between mown and unmown vegetation within that boundary, and the planner follows the detected line. The coupling between perception and mapping is bidirectional in effect — perception dictates, on-line, where the mapping node is allowed to accumulate evidence, and the resulting map feeds the cut line back to the planner — and is the mechanism by which a single sensing stack serves navigation, the mowing strategy, and vegetation mapping at once.

4.2 Processing Nodes

This section describes the three nodes individually. For each node it states the single responsibility the node carries and the interface it maintains with the relevant external subsystem; the parameters that configure each node are introduced where the corresponding module is described in detail, in the chapters that follow. The subscriptions and publications of the three nodes are summarised in Table 4.1.

4.2.1 Perception Node

The perception node (`perception_node_v2`) is responsible for detecting and tracking the two lateral vine rows and for delimiting the navigable inter-row corridor. Its sole exteroceptive input is the 3D LiDAR point cloud: no camera or complementary range sensor participates, so every geometric estimate the system forms is LiDAR-derived. This is a deliberate design decision, motivated in Section 1.2 by the robustness of geometric range sensing to the variable illumination, dust, and calibration drift of the vineyard, and it makes the LiDAR sensor the single sensing interface of the architecture. The robot pose is not computed inside the node; it is read from the TF tree, so that the node behaves as a client of an external localisation service (odometry and GNSS-RTK fused upstream) and never estimates ego-motion. From each sweep the node produces a per-side row model and a closed corridor polygon, published respectively as a `RowDetectionResult` message and as a boundary point cloud.

4.2.2 Local-Planning Node

The local-planning node (`local_planning_node`) is responsible for converting the perceived geometry into a smooth motion reference and for supervising the transitions between behaviours

through a finite-state machine. It consumes the `RowDetectionResult` from perception and the `CutLine` from mapping, together with the robot pose obtained from the TF tree. The trajectories it generates are not executed within the node: they are handed to an external trajectory tracking controller — a previously tuned PID scheme with feed-forward — which acts as the receptacle of the planned references. This controller is the actuation interface of the architecture and lies outside the scope of this work, so the navigation results reported later are system-level results that include, but do not isolate, the tracking controller’s contribution.

4.2.3 Mapping Node

The mapping node (`mapping_node`) is responsible for accumulating the inter-row vegetation into a 2.5D height map and for detecting the cut line that the planner follows on the return pass. Like the perception node, it draws on the same single LiDAR modality and on the robot pose from the TF tree; what distinguishes its input is that the incoming point cloud is spatially gated by the corridor polygon supplied by perception, which is reused as the node’s spatial filter so that only returns from inside the navigable corridor are accumulated. This on-line configuration of the mapping filter by perception is the concrete realisation of the perception–mapping coupling described in Section 4.1. The node publishes the detected cut line as a `CutLine` message and maintains the mission-level height map as a by-product of normal operation.

4.3 Software Namespaces and Data Structures

The nodes are thin orchestration layers built on top of a set of framework-agnostic software components, organised into three namespaces. The `vineyard_nav` namespace (abbreviated `vnav`) holds the ROS-free geometric and navigation primitives shared by perception and planning. The `vineyard_mapping` namespace (`vmap`) holds the mapping and cut-line-detection structures used by the mapping node. The `vnav_analysis` namespace provides instrumentation — scoped timers and point-cloud statistics — used to profile the pipeline. Keeping the algorithmic structures free of ROS dependencies allows the geometry and mathematics to be unit-tested in isolation from the middleware. The main data structures are enumerated below and summarised in Table 4.2; their internal definitions, fields, and governing equations are deferred to the chapters in which they are used.

Within `vineyard_nav`, the `Line2D` structure represents a single 2D line extracted from the point cloud (its centroid and direction) and is the atomic row-segment estimate. The `Waypoint` and `ParametricCurve` structures represent, respectively, a planar pose and the cubic Bézier curve from which smooth references are built, and a `Trajectory` chains a bounded history of such curves to preserve continuity between planning cycles. A `RowRegion` represents one oriented search region (`CLOSE` or `FAR`) on a given side of the crop rows, and a `Side` aggregates the two regions of a row together with its safety score, forming the complete per-side row model. A `VirtualBoundary` represents the inter-row corridor boundary and is the structure through

Table 4.2: Principal shared data structures and message contracts, with the namespace or module that owns each and the role it serves.

Structure	Namespace	Role
Line2D	vineyard_nav	Atomic 2D row-segment estimate (centroid and direction)
ParametricCurve	vineyard_nav	Cubic Bézier curve used to build smooth references
Trajectory	vineyard_nav	Bounded chain of curves preserving inter-cycle continuity
RowRegion	vineyard_nav	One oriented search region (CLOSE or FAR) on a Side
Side	vineyard_nav	Complete per-side row model (two Regions, safety score)
VirtualBoundary	vineyard_nav	Inter-row corridor boundary; perception–mapping link
SparseHeightMap	vineyard_mapping	Hash-indexed mission-level 2.5D vegetation-height map
GridMapSection	vineyard_mapping	Rolling-window longitudinal section of the grid map, used for cut-line detection
RowDetectionResult	message	Perception → planner: row models and safety verdict
CutLine	message	Mapping → planner: detected cut-line geometry

which perception and mapping are coupled. A small set of enumerations and the finite-state machine type complete the namespace.

Within `vineyard_mapping`, the `SparseHeightMap` (with its `HeightCell` and `CellIndex` helpers) is a hash-indexed, memory-efficient 2.5D height map that accumulates vegetation evidence across the mission, while the `GridMapSection` holds the per-section state of the rolling local grid map on which cut-line detection operates.

Finally, the two nodes communicate through two message contracts: `RowDetectionResult`, carrying the per-side row models, canopy-dispersion measures, and per-sweep safety verdict from perception to the planner; and `CutLine`, carrying the detected line’s anchor point, direction, span, and validity flag from mapping to the planner.

4.4 Coordinate Frames and Conventions

All geometric reasoning in the system is expressed with respect to three coordinate frames arranged in a short transform (TF) tree. The `map_sn1` frame is the global, fixed reference in which row models, trajectory endpoints, and the height map are accumulated, and it forms the root of the tree. The `base_footprint` frame is attached to the robot at ground level and is registered to `map_sn1` by the localisation estimate, so this edge is dynamic and updates as the robot moves. The `rslidar` frame is attached to the sensor by a fixed calibration transform, a static offset of 0.92 m

above ground. Because the sensor is mounted facing the rear of the platform on the experimental data, a configuration flag signals this backward mounting so that headings are disambiguated consistently. Working in the global frame keeps successive sweeps mutually registered, while the region-of-interest cropping is performed in the LiDAR frame — after transforming only the region pose rather than the whole cloud — for efficiency.

The notation used throughout the methodology chapters follows a single convention. Two-dimensional vectors are written in bold lower case, unit-length directions carry a hat, and scalar quantities are written in italic; operators such as $\sin$ and $\exp$ are kept upright. Symbols are introduced at first use and collected in the Nomenclature.

4.5 Validation Environments

The complete architecture was exercised in two complementary settings. The first is a set of purpose-built simulated vineyards in Webots, in which the robot, its drivetrain, and its 3D LiDAR are modelled so that the same nodes run without modification. The second is a collection of raw LiDAR recordings captured in a real vineyard and replayed offline, which subjects perception and mapping to genuine sensor noise, occlusion, and row geometry. A formal evaluation across both settings, with quantitative metrics, is presented and discussed in Chapter 8; the present section introduces only the simulated worlds, so that the figures of the following chapters can be read against the scene that produced them.

The experimental work was conducted within the TRIBE centre at INESC TEC/CRIIS, whose activity focuses on robotics and IoT solutions for precision agriculture and forestry. The robotic platform used for both the real-vineyard data collection and deployment is the Modular-E (Figure 4.2), equipped with a Livox Mid-360 3D LiDAR as the primary exteroceptive sensor, an OAK-D RGB camera for visual data acquisition, an IMU for inertial measurements, and a GNSS-RTK receiver for centimetre-level localisation support. The sensing and navigation stack was implemented in ROS 2 on a Linux-based system, enabling modular communication between the perception, mapping, localisation, and local-planning components.



Figure 4.2: The Modular_E robotic platform used for real-vineyard data collection and deployment

Each simulated vineyard is built from a single repeating element: an approximate mesh of one vine plant (Figure 4.3), concatenated along each row to the desired corridor length and shape, with grass pads placed on the corridor floor to model the standing inter-row vegetation. Figure 4.4 shows two representative worlds built this way, a straight corridor and a curved corridor of $R = 20$ m radius; the complete set of corridor shapes and task families used in the experimental campaign is presented in Chapter 8.

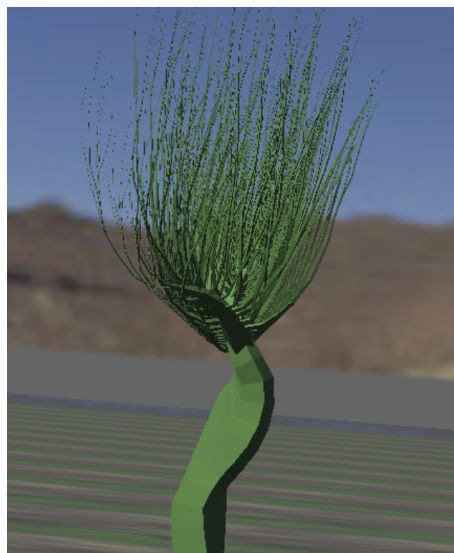
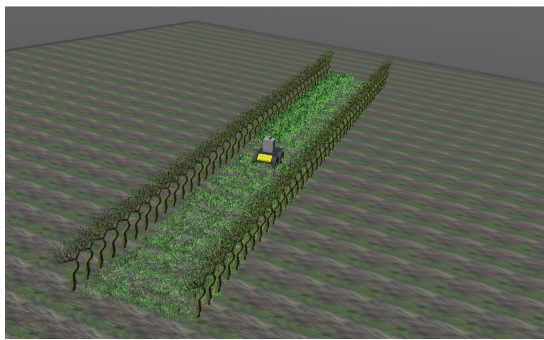


Figure 4.3: Approximate mesh of a single vine plant, the building block concatenated along each row to construct the simulated corridors.



(a) straight

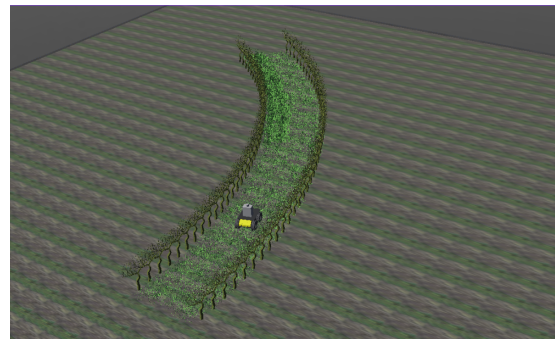
(b) arc $R = 20$ m

Figure 4.4: Two representative simulated corridors built by concatenating the single-vine mesh of Figure 4.3 along the row path, with grass pads placed on the corridor floor: (a) a straight corridor and (b) a curved corridor of $R = 20$ m radius.

With the overall architecture, its interfaces, its shared structures, its frames, and the environments in which it is validated now established, each of the next three chapters details one node: perception in Chapter 5, mapping with cut-line detection in Chapter 6, and local planning in Chapter 7.

Chapter 5

Perception: Dual-ROI Row Detection and Virtual Boundaries

This chapter describes the perception node in full: how the two lateral vine rows are detected and tracked from raw LiDAR sweeps, and how the polygonal model of the navigable corridor — the structure that couples perception to mapping — is constructed. The presentation proceeds from the processing cycle to its constituent stages. Section 5.1 sets out the sweep-driven processing cycle at a glance. Section 5.2 introduces the dual-region (CLOSE/FAR) row model and the predictive placement of its regions of interest, and Sections 5.3 to 5.5 detail the point-cloud filtering chain, the RANSAC vertical-plane fit from which each row line is extracted, and the initialisation and validation of the FAR region. Section 5.6 describes the temporal filter that stabilises the line estimates, Section 5.7 the per-sweep detection verdict and its accumulation into a safety score, and Section 5.8 the canopy-dispersion measure. Section 5.9 assembles the virtual boundaries into the closed inter-row polygon, and Section 5.10 specifies the output message.

5.1 Overview and Processing Cycle

The perception node (`perception_node_v2`) is *sweep-driven*: each incoming LiDAR point cloud triggers exactly one processing cycle, at the end of which the node has refreshed its estimate of both vine rows, updated the corridor polygon, and published a single message summarising the geometric state of the corridor. The node is a pure producer of geometry; the stateful navigation decisions are deferred to the planner (Chapter 7). Its only exteroceptive input is the raw LiDAR cloud, and the robot pose is read from the TF tree rather than estimated internally (Section 4.1). This section gives the pipeline at a glance; each stage, together with its parameters, is then developed in the sections that follow.

The figures in this chapter are of two kinds, chosen for whichever makes a given construction clearest. Purely geometric constructions, such as the region placement, the separation constraint and the inter-row width, are drawn as clean vector diagrams. Quantities that are easier to read on real or simulated data, such as the fitted plane, the corridor polygon and the single-sided fallback,

are instead shown as screenshots of the RViz visualisation. The simulated vineyards used for the latter are the Webots worlds introduced in Section 4.5 and shown in Figure 4.4.

Every side of the corridor is sensed through two independent regions of interest: a `CLOSE` region, centred abreast of the robot, which measures the row segment alongside the vehicle and therefore its current lateral placement; and a `FAR` region, centred a lookahead distance ahead along the row, which anticipates the upcoming geometry and anchors the trajectory endpoint. The two sides are processed independently and identically.

At the level of the node, one cycle comprises the five steps shown in Figure 5.1:

1. **Transform look-up.** The map-to-LiDAR and map-to-base transforms are read from the TF tree; the sweep is skipped if either is unavailable. The robot heading, taken from the base yaw, is carried through the pipeline to resolve the sign ambiguity of the fitted row directions.
2. **Per-side processing.** The right and then the left side are processed independently, one after the other, each yielding an updated `CLOSE` and `FAR` line, a per-side detection verdict, and a canopy-dispersion measure.
3. **Inter-row width update.** The corridor width is re-estimated from the two `CLOSE`-line centroids and kept as the fallback width used when only one side is reliable.
4. **Virtual-boundary update.** The per-side boundaries are rebuilt from both row models and the width estimate; when valid, the closed corridor polygon is published as a point cloud in the map frame (Section 5.9).
5. **Message publication.** A single `RowDetectionResult` is published, carrying, for each side, the `CLOSE` and `FAR` line estimates and validity, the canopy dispersion, and the per-sweep safety verdict (Sections 5.7 and 5.10).

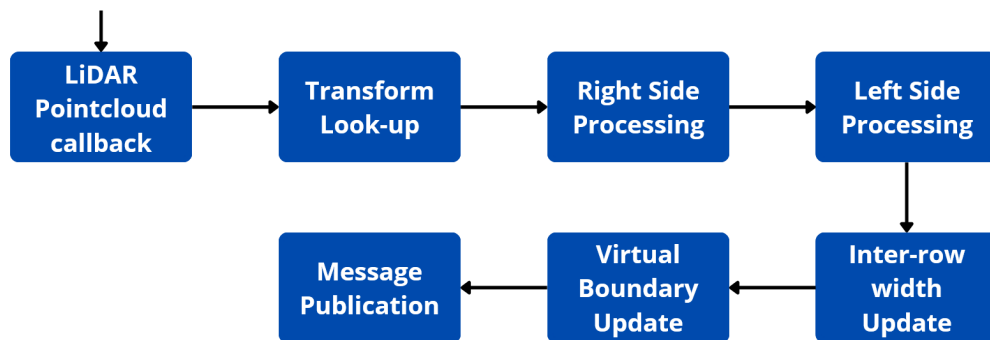


Figure 5.1: Per-sweep processing pipeline of the perception node.

The per-side processing of Step 2 is the substantive stage of the cycle. It is applied independently to each side and, within a side, to each of its two regions of interest, and it decomposes into two phases: a *filtering* phase, which selects the LiDAR points belonging to a region of interest, and

an *update* phase, which fits a line to those points and refreshes the region’s geometric estimate. The two phases are shown in Figure 5.2.

The filtering phase is carried out entirely in the LiDAR frame and comprises three operations:

1. **Chassis exclusion.** A negative crop box of the robot footprint, centred at the sensor, removes the platform’s own returns.
2. **Region-of-interest crop.** The oriented crop box associated with the region of interest — `CLOSE` or `FAR` — is applied to the raw cloud, retaining the points taken to belong to that region. The box is placed from the region’s estimate on the previous sweep, or from a fixed prior on the first sweep, and is oriented to the row direction; because the crop is performed in the LiDAR frame, only the box pose is transformed rather than the full cloud (Section 5.3).
3. **Voxel downsampling.** The retained cloud is reduced on a voxel grid, bounding the cost of the subsequent stages independently of the raw point density.

The update phase then operates on the reduced, per-region cloud produced by the filtering phase:

1. **Transform to the map frame.** Only the reduced cloud is transformed, so the full raw sweep is never carried into the map frame.
2. **Statistical outlier removal.** Remaining sparse returns are stripped to spare the fit from spurious outliers.
3. **Vertical-plane fit, line extraction, and centroid.** The row is modelled as a vertical plane, fitted by RANSAC; the plane is collapsed to a 2D tangent line, and the region centroid is computed from the inlier points (Section 5.4).
4. **Acceptance gating.** The fit is accepted only if it is supported by enough points, is sufficiently vertical, and exceeds a per-region inlier-ratio threshold; the line direction is anchored to the robot heading to prevent 180° reversals.
5. **FAR cross-validation.** A `FAR` fit is additionally checked against the `CLOSE` line and downgraded when the two are inconsistent (Section 5.5).
6. **Temporal filtering.** The accepted line is fused with the most recent estimates to yield the stabilised line reported for the region (Section 5.6).

The update phase produces the region’s new centroid, vertical plane, and tangent direction. The centroid, in turn, defines the centre of the region’s crop box on the following sweep, closing the per-region tracking loop formalised in Section 5.2.

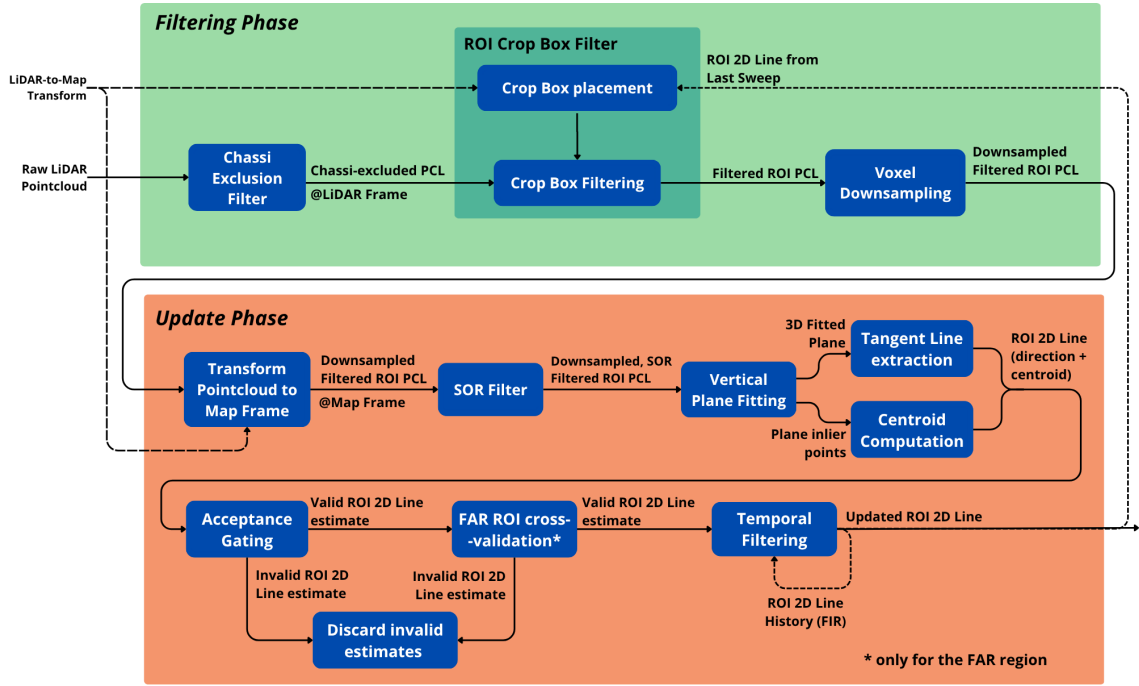


Figure 5.2: Per-side processing, decomposed into a filtering phase and an update phase. The updated centroid of each region of interest defines its crop-box centre on the following sweep.

Two further per-side quantities are maintained alongside the line estimates. The per-sweep verdict records whether the side was sensed consistently and is accumulated, inside the perception node, into a saturating safety score that the planner later consumes already integrated (Section 5.7). The canopy dispersion, the lateral spread of foliage about the row axis, is computed per region and used as a physically meaningful inset margin when the virtual boundaries are built (Sections 5.8 and 5.9). With the pipeline established at this level, the remaining sections develop each stage — and the parameters and equations behind it — in the order in which it executes.

5.2 Dual-ROI Row Model

A single region of interest per side, fitted by one line, implicitly models the vine row as straight over the sensor range; the estimate is then accurate only where that assumption holds and degrades on curved rows, at headland approaches, and where one side of the corridor is sparsely populated [18, 34]. The model adopted here retains the reactive, map-free formulation [7] but represents each side of the corridor by two regions of interest placed at distinct longitudinal horizons. The present section defines this model in purely geometric terms; the operations that populate it — filtering, fitting, and temporal smoothing — are deferred to Sections 5.3 to 5.6; Figure 5.3 previews the resulting geometry.

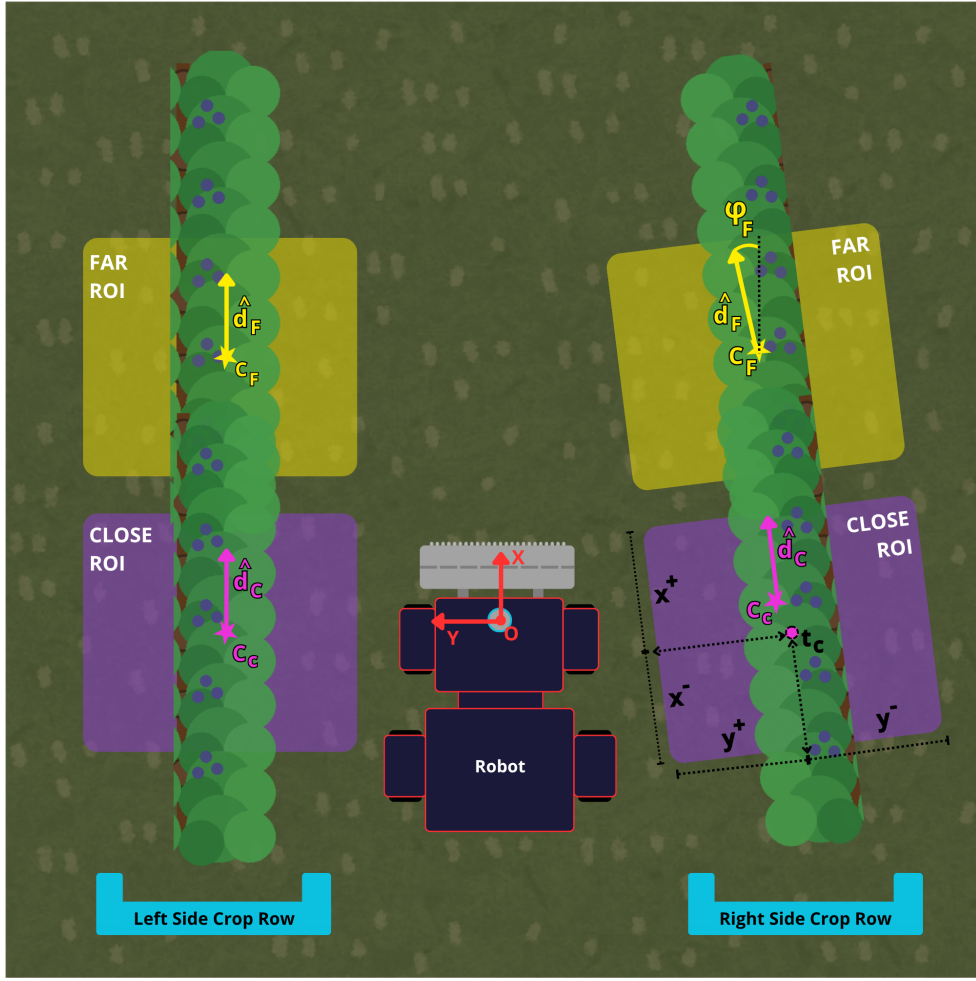


Figure 5.3: Overview of the dual-region row model: the robot in the corridor, the two sides, and their CLOSE/FAR regions of interest.

The inter-row corridor is modelled by two sides, S_L and S_R , one for each flanking crop row. Each side aggregates two regions of interest, a CLOSE region and a FAR region, together with a scalar safety score σ that summarises the recent detection quality of the side and is defined in Section 5.7:

$$S = \{ \mathcal{R}_{\text{CLOSE}}, \mathcal{R}_{\text{FAR}}, \sigma \}. \quad (5.1)$$

Each region of interest $\mathcal{R}$ is described, at sweep k , by two coupled geometric objects: the tangent line it estimates and the oriented crop box from which that estimate is drawn. The tangent line is

$$\mathcal{L}^k = (\mathbf{c}^k, \hat{\mathbf{d}}^k), \quad \mathbf{c}^k \in \mathbb{R}^2, \quad \hat{\mathbf{d}}^k \in \mathbb{R}^2, \quad \|\hat{\mathbf{d}}^k\| = 1, \quad (5.2)$$

where $\mathbf{c}^k$ is the centroid of the points assigned to the region on sweep k and $\hat{\mathbf{d}}^k$ the estimated row direction. The crop box is the oriented box $\mathcal{B}(\mathbf{t}^k, \phi^k)$ of fixed extents, with planar centre $\mathbf{t}^k \in \mathbb{R}^2$ and orientation ϕ^k . A region of interest at sweep k is therefore the tuple

$$\mathcal{R}^k = (\mathbf{c}^k, \hat{\mathbf{d}}^k, \mathbf{t}^k, \phi^k). \quad (5.3)$$

The centroid $\mathbf{c}^k$ and the crop-box centre $\mathbf{t}^k$ are distinct quantities. The centroid is an *output* of sweep k , namely the location of the points the region captured, whereas the crop-box centre is an *input* that determines which points are captured. The two are coupled across consecutive sweeps: the crop-box centre is derived from the centroid estimated on the preceding sweep,

$$\mathbf{t}^k = g(\mathbf{c}^{k-1}, \hat{\mathbf{d}}^{k-1}), \quad k \geq 1, \quad (5.4)$$

so that each region remains anchored to the segment it last measured, with the orientation φ^k set from the preceding direction estimate in the same manner. The only exception is the initialisation, $k = 0$, for which $\mathbf{t}^0$ and φ^0 are set from a fixed prior. The explicit form of the map g , which differs between the `CLOSE` and `FAR` regions, and of the initial prior, is given in Section 5.3.

The model is used recursively. On every sweep, the points lying within the crop box of a region are filtered and fitted to yield the updated line $(\mathbf{c}^k, \hat{\mathbf{d}}^k)$; the centroid then defines the crop-box centre of the following sweep through (5.4), closing a per-region recursion that requires no external state estimator. The two sides, each carrying its `CLOSE` and `FAR` estimates and its safety score, jointly constitute the geometric model of the corridor used downstream: the centroids and directions delimit the navigable inter-row area and anchor the trajectory references (Section 5.9 and Chapter 7), while the safety scores arbitrate the supervisory logic of the planner.

The lateral and heading accuracy of the resulting `CLOSE/FAR` line estimates is reported in Section 8.2.1.

5.3 Region-of-Interest Filtering and Placement

The filtering phase introduced in Section 5.1 selects, for each region of interest, the points from the raw LiDAR point cloud that the update phase will use for the plane and line fitting operations, and places the oriented crop box from which those points are drawn. This section specifies both: the filtering operations applied to every region of interest, and the rules that place each region's crop box — the bootstrap prior, the predictive tracking of the `CLOSE` region, the seeding and tracking of the `FAR` region, and the fixed-separation constraint that keeps the two horizons distinct. All operations described here are performed in the LiDAR frame.

Filtering operations. Three operations are applied to each region of interest, in order. The first is chassis exclusion: a negative crop box of the robot footprint, centred at the sensor and unbounded in height, discards every point inside it, removing the platform's own returns before any region is processed. The second is the region's oriented crop box. A LiDAR-frame point $\mathbf{q} = (q_x, q_y, q_z)$ is retained when its planar coordinates, expressed in the box frame, lie within the fixed longitudinal and lateral extents and its height lies within the fixed window,

$$\mathbf{q} \in \mathcal{B}(\mathbf{t}, \varphi) \iff R(\varphi)^\top ((q_x, q_y)^\top - \mathbf{t}) \in [x^-, x^+] \times [y^-, y^+] \wedge q_z \in [z^-, z^+], \quad (5.5)$$

Table 5.1: Parameters of the region-of-interest filtering operations. The chassis-exclusion footprint is 1.0×1.0 m, common to both regions.

Parameter	CLOSE	FAR
Crop-box longitudinal extent $[x^-, x^+]$	$[-1, 1]$ m	$[-1, 1]$ m
Crop-box lateral extent $[y^-, y^+]$	$[-1, 1]$ m	$[-1.25, 1.25]$ m
Crop-box height window $[z^-, z^+]$	$[-0.25, 0.50]$ m	$[-0.25, 0.50]$ m
Voxel leaf size	0.05 m	0.04 m

where $\mathbf{t} \in \mathbb{R}^2$ is the planar box centre, φ its orientation, and $R(\varphi) \in \text{SO}(2)$ the corresponding rotation. Performing the crop in the LiDAR frame requires only the box pose $(\mathbf{t}, \varphi)$ to be transformed into that frame, rather than the full raw cloud into the map frame. The third operation is voxel-grid downsampling, which replaces the points within each occupied voxel by their centroid and bounds the cost of the fit independently of the raw point density; the FAR region uses a finer leaf, since its returns are sparser. The parameters of the three operations are summarised in Table 5.1. These point-cloud operations — the chassis-exclusion and region-of-interest crop boxes and the voxel-grid downsampling, together with the frame transformation, the statistical outlier removal, and the RANSAC plane fit applied in the update phase (Sections 5.4 and 5.6) — are implemented with the standard primitives of the Point Cloud Library (PCL).

Bootstrap placement. Before a region has produced a valid estimate there is nothing to predict from, so its crop box is placed from a fixed default parameter. For the CLOSE region the box centre is offset laterally to the relevant side by a constant amount and its orientation is set to zero, leaving the box aligned with the sensor axes,

$$\mathbf{t}_C^{(0)} = \begin{bmatrix} 0 \\ \pm y_{\text{off}} \end{bmatrix}, \quad \varphi_C^{(0)} = 0, \quad (5.6)$$

where the sign of the lateral offset y_{off} is fixed by the side and by the backward-mounted-LiDAR flag of Section 4.4. This prior encodes only the expectation that a row lies approximately at the nominal lateral offset. Once a fit is accepted, the region acquires a history and placement proceeds according to the tracking rule.

Tracking placement of the CLOSE region. When a history is available, that is, when at least one sweep has been completely processed and marked as *valid*, the CLOSE box is centred on the row rather than on the vehicle. Let $\mathcal{L}_C^k = (\mathbf{c}_C^k, \hat{\mathbf{d}}_C^k)$ denote the CLOSE line estimate updated on sweep k and $\mathbf{p}_0^k$ the robot position obtained from the TF tree at the same sweep. The region's crop-box centre for sweep $k+1$ is the orthogonal projection of $\mathbf{p}_0^k$ onto the line $\mathcal{L}_C^k$, the *foot point*, given by the scalar coordinate

$$\lambda^k = (\mathbf{p}_0^k - \mathbf{c}_C^k) \cdot \hat{\mathbf{d}}_C^k, \quad \mathbf{t}_C^{k+1} = \mathbf{c}_C^k + \lambda^k \hat{\mathbf{d}}_C^k. \quad (5.7)$$

With this choice, a lateral displacement of the vehicle within the corridor moves the robot relative to a box that remains attached to the row. The box orientation is obtained from the `CLOSE` line direction expressed in the LiDAR frame, whose heading defines the box yaw,

$$\varphi_C^{k+1} = \text{atan2}(e_y^k, e_x^k), \quad (e_x^k, e_y^k)^\top = R_{m \rightarrow \ell}^k (\hat{\mathbf{d}}_C^{k\top})^\top, \quad (5.8)$$

where $R_{m \rightarrow \ell}^k$ is the rotational part of the map-to-LiDAR transform. The centre $\mathbf{t}_C^{k+1}$ is likewise transformed into the LiDAR frame and the crop-box filter is applied there, so that a single pose is transformed per region rather than the complete point cloud. Because the line estimate is updated once per sweep (Section 5.4), the foot point computed on one sweep determines the box placement on the next, and the placement is in this sense predictive.

Figure 5.4 shows the resulting placement of the `CLOSE` region across two consecutive sweeps.

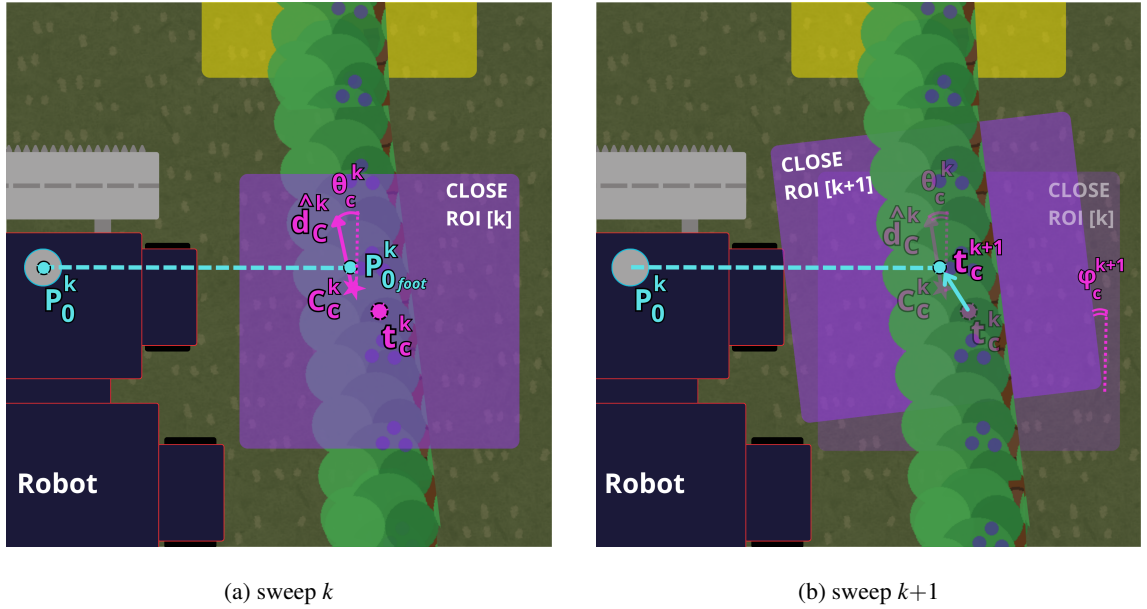


Figure 5.4: Predictive placement of the `CLOSE` region across two consecutive sweeps: the box tracks the foot point of the robot on the row and is rotated to the row direction on each sweep.

Seeding and tracking of the `FAR` region. The `FAR` region is initialised from the `CLOSE` estimate rather than from a default lateral offset. On first acquisition its crop-box centre $\mathbf{t}_F^{(0)}$ is seeded by projecting the `CLOSE` centroid forward along the `CLOSE` direction,

$$\mathbf{t}_F^{(0)} = \mathbf{c}_C^{(0)} + \ell \hat{\mathbf{d}}_C^{(0)}, \quad (5.9)$$

inheriting the `CLOSE` direction and height, where ℓ is the `FAR` longitudinal offset, equal to the lookahead distance. This parameter was tuned to balance the competing requirements of anticipating the upcoming geometry of the crop row and maintaining an adequate signal-to-noise ratio in the `FAR` region, which degrades with distance owing to the decreasing density of the LiDAR returns; its final value was set to 3.0 m. A guard reverses the projection if it would place the seed

behind the CLOSE centroid, ensuring that the seed lies ahead of the robot. The FAR orientation is seeded from the CLOSE line as well, so that both boxes are aligned with the row from the outset,

$$\varphi_F^{(0)} = \varphi_C^{(0)}. \quad (5.10)$$

Figure 5.5 shows the bootstrap placement of the two regions on first acquisition.

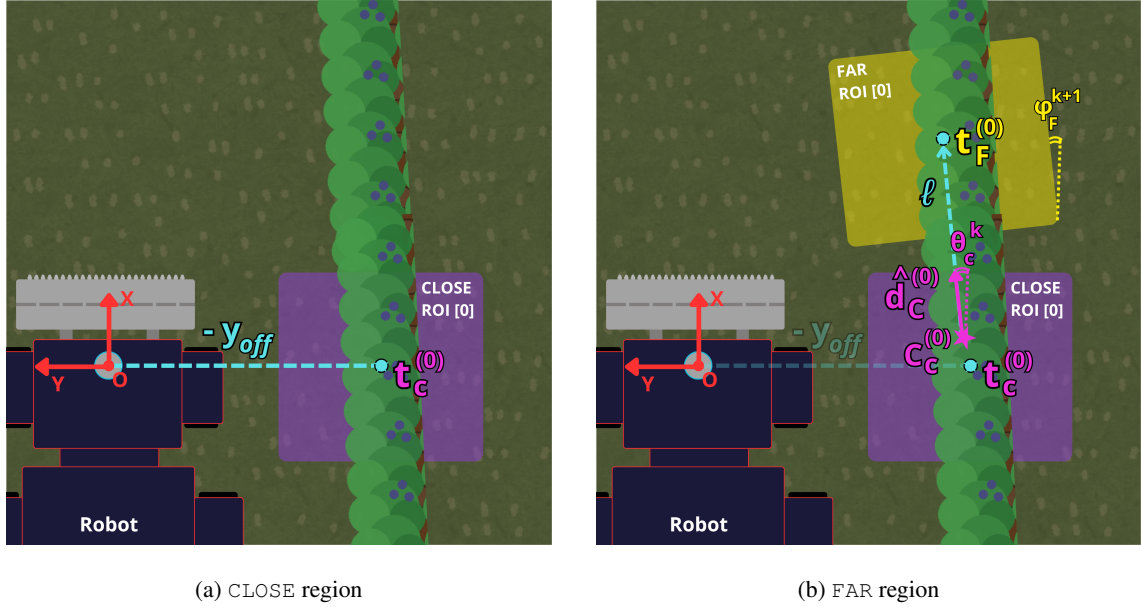


Figure 5.5: Bootstrap placement of the two regions before any row has been tracked. The CLOSE box is set by a fixed lateral offset from the robot (a horizontal shift in y); the FAR box is then placed by extending the CLOSE direction forward by the look-ahead distance.

On subsequent sweeps the box is centred on the FAR centroid estimated on the previous sweep, so that the region remains associated with the segment it last measured rather than being re-derived from the CLOSE estimate,

$$\mathbf{t}_F^{k+1} = \mathbf{c}_F^k, \quad (5.11)$$

$$\varphi_F^{k+1} = \text{atan2}(e_y^k, e_x^k), \quad (e_x^k, e_y^k)^\top = R_{m \rightarrow \ell}^k (\hat{\mathbf{d}}_F^{k\top})^\top. \quad (5.12)$$

Figure 5.6 shows the resulting placement of the FAR region across two consecutive sweeps.



Figure 5.6: Predictive placement of the FAR region across two consecutive sweeps: the box is centred on the FAR centroid from the previous sweep and re-projected along its direction to hold the fixed separation L from the CLOSE centroid.

Fixed-separation constraint. Re-using the previous FAR centroid without an additional constraint is unstable: the FAR centroid was observed to migrate towards the CLOSE centroid over successive sweeps, until the two regions coincided and the two-horizon representation reduced to a single near-field model.

This behaviour is caused by the loss of point density of the LiDAR sensor with distance: because the FAR region is inherently sparser than the CLOSE region, its centroid naturally drifts towards the region of space where more points were found on each LiDAR sweep. This is a form of degeneracy: the FAR region is intended to anticipate the upcoming geometry of the row, but if it is placed too close to the CLOSE region it loses its predictive value and the two regions become redundant.

To prevent this from happening, a fixed-separation constraint is imposed. After each accepted FAR fit, the initial estimate of the FAR centroid, c'_F , is re-projected onto the circle of fixed radius L centred on the CLOSE centroid, so that the inter-centroid separation is held at *exactly* $L = 3.0$ m on every sweep. The centroid is displaced along its own direction $\hat{\mathbf{d}}_F$ by the amount t for which

$$\|\mathbf{m} + t \hat{\mathbf{d}}_F\| = L \quad (5.13)$$

where $\mathbf{m} = \mathbf{c}'_F - \mathbf{c}_C$ is the current offset of the FAR centroid from the CLOSE centroid. Expanding the norm yields the quadratic equation in t ,

$$t^2 + 2(\mathbf{m} \cdot \hat{\mathbf{d}}_F)t + (\|\mathbf{m}\|^2 - L^2) = 0 \quad \implies \quad t = -\mathbf{m} \cdot \hat{\mathbf{d}}_F \pm \sqrt{\Delta} \quad (5.14)$$

where

$$\Delta = (\mathbf{m} \cdot \hat{\mathbf{d}}_F)^2 - \|\mathbf{m}\|^2 + L^2 \quad (5.15)$$

Assuming $\Delta \geq 0$, the equation yields two solutions, t_1 and t_2 . The root of smaller magnitude, that is, the one corresponding to the minimal displacement, is selected and applied: $\mathbf{c}_F \leftarrow \mathbf{c}'_F + t \hat{\mathbf{d}}_F$.

$$t = \begin{cases} t_1, & |t_1| \leq |t_2|, \\ t_2, & |t_2| < |t_1|. \end{cases} \quad (5.16)$$

Figure 5.7 illustrates this geometric construction.

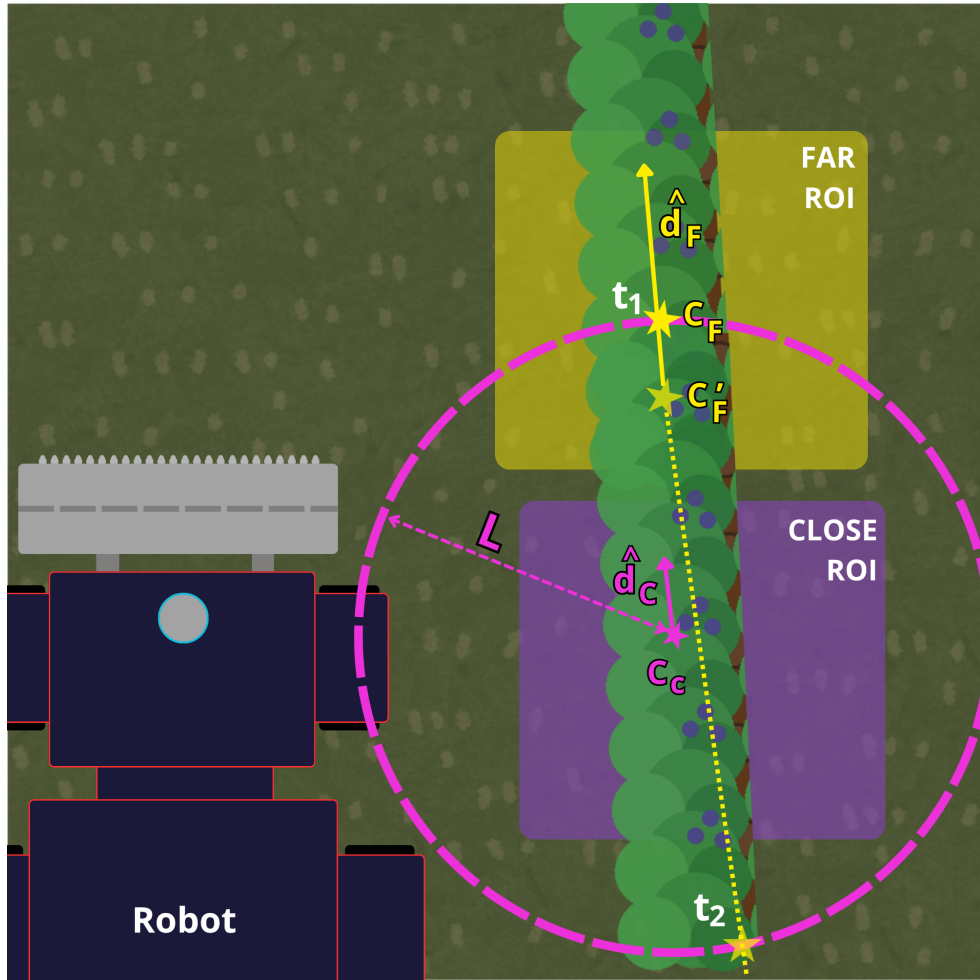


Figure 5.7: Illustration of the fixed separation constraint between the CLOSE and FAR centroids. A circumference of radius L is centred on the CLOSE centroid, and the FAR centroid is moved along its line direction until it lies exactly on the circumference. Among the two possible intersections, the nearest one, t_1 , is selected, while the farther solution, t_2 , is discarded.

When the discriminant Δ is negative the line through $\mathbf{c}_F$ along $\hat{\mathbf{d}}_F$ does not meet the circle because the direction is close to tangent, and Δ is therefore clamped to zero, yielding the point of closest approach. This closed-form projection fixes the separation to a constant value, providing

a stable baseline between the two horizons and a well-conditioned endpoint geometry for the planner.

Properties of the placement scheme. The placement rules define a feedback loop in which each accepted line determines the position and orientation of the regions of interest cropped on the following sweep, without a covariance estimate or a motion model. Two properties follow. First, points distant from the predicted row are excluded by the crop, so returns from the adjacent row observed through a gap, from trellis posts, and from stray vegetation do not enter the fit. Second, because the regions translate along the row and rotate with it, the detector follows a curved corridor without representing the curvature explicitly, which removes the straight-row assumption of the single-horizon detectors noted above. A mis-placed region produces a poor crop; the effect is local and bounded, since such a crop fails the acceptance gates of Section 5.4 or the FAR cross-validation of Section 5.5 on that sweep, and the accumulated safety score of Section 5.7 accommodates intermittent failures.

The resulting inter-row width tracking is evaluated in Section 8.2.2.

5.4 Region-of-Interest Update: Vertical-Plane Fitting and Tangent-Line Extraction

The update phase, introduced in Section 5.1, refreshes the geometric estimate of a region of interest from the points retained by the filtering phase. It comprises the vertical-plane fit and the tangent-line extraction described in this section, together with their acceptance gates; the FAR cross-validation (Section 5.5) and the temporal filtering (Section 5.6) complete the phase. The update opens by transforming the reduced cloud of the region into the map frame — only this small cloud is transformed, not the full sweep — and applying a statistical outlier removal step to discard the sparse returns that survive the crop. The remaining points are then fitted as described below.

A vine row, observed from the inter-row corridor, forms an approximately vertical sheet of foliage and trellis; the row is therefore modelled by a vertical plane and recovered by RANSAC. Let $\mathcal{P} \subset \mathbb{R}^3$ denote the filtered point set of the region in the map frame, of cardinality $N = |\mathcal{P}|$. A candidate plane π is parameterised by its unit normal $\mathbf{n} = (n_x, n_y, n_z)$ and offset D ,

$$\pi = \{ \mathbf{x} \in \mathbb{R}^3 : \mathbf{n} \cdot \mathbf{x} + D = 0 \}, \quad \|\mathbf{n}\| = 1. \quad (5.17)$$

The fit constrains the plane to be vertical: its normal is required to be orthogonal to the world vertical $\hat{\mathbf{z}}$ within an angular tolerance ε , so that the vertical direction lies in the plane. Subject to this constraint, RANSAC seeks the plane that maximises the inlier set

$$\mathcal{I}(\pi) = \{ \mathbf{p} \in \mathcal{P} : |\mathbf{n} \cdot \mathbf{p} + D| \leq \tau \}, \quad (5.18)$$

Table 5.2: Parameters of the vertical-plane fit and its acceptance gates.

Parameter	Symbol	Value
Inlier distance threshold	τ	0.15 m
Maximum RANSAC iterations	N_{it}	250
Plane-normal angular tolerance	ε	0.26 rad ($\approx 15^\circ$)
Minimum supporting points	N_{min}	10
Verticality guard on $ n_z $	η	0.3
Inlier-ratio threshold (CLOSE / FAR)	ρ_{thr}	0.46 / 0.42

where τ is the inlier distance threshold, sampling minimal point sets for at most N_{it} iterations. Restricting the normal to the horizontal confines the search to vertical sheets and suppresses the dominant ground plane, which would otherwise attract the fit.

The fitted plane is reduced to the two-dimensional tangent line of Equation (5.2). The row direction is the horizontal direction contained in the plane, obtained by rotating the horizontal component of the normal by 90° ,

$$\hat{\mathbf{d}} = \frac{(-n_y, n_x)}{\|(-n_y, n_x)\|}, \quad (5.19)$$

and the centroid is the planar part of the centroid of the inlier set,

$$\mathbf{c} = \frac{1}{|\mathcal{J}|} \sum_{\mathbf{p} \in \mathcal{J}} (p_x, p_y), \quad (5.20)$$

with the corresponding mean height retained for region placement and visualisation.

The fitted plane is accepted as a valid row estimate only if it satisfies three conditions. First, the region must retain enough support after filtering, $N \geq N_{min}$. Second, the plane must be sufficiently vertical; a guard on the vertical component of the normal,

$$|n_z| \leq \eta, \quad (5.21)$$

rejects near-horizontal fits that pass within the RANSAC tolerance, with η a conservative bound. Third, the fit must be supported by a sufficient fraction of the points, quantified by the inlier ratio

$$\rho = \frac{|\mathcal{J}(\pi)|}{N}, \quad (5.22)$$

which must exceed a per-region threshold ρ_{thr} . This threshold is lower for the FAR region of interest, whose cloud is sparser and whose inlier ratio is correspondingly lower at equal fit quality. If any of the three conditions fails, the region returns a negative verdict for the sweep (Section 5.7) and its estimate is left unchanged. The parameters of the fit and its gates are listed in Table 5.2.

A line direction is defined only up to sign, so the same plane admits both $\hat{\mathbf{d}}$ and $-\hat{\mathbf{d}}$. To prevent the estimated direction from alternating between consecutive sweeps, its sign is anchored to the robot heading $\hat{\mathbf{h}}$ (Section 5.1),

$$\hat{\mathbf{d}} \leftarrow -\hat{\mathbf{d}} \quad \text{if} \quad \hat{\mathbf{d}} \cdot \hat{\mathbf{h}} < 0. \quad (5.23)$$

After this correction every row direction points in the robot’s direction of travel, eliminating the 180° reversals that would otherwise corrupt the temporal filter (Section 5.6) and the orientation of the regions of interest on the following sweep (Section 5.3). The resulting line $(\mathbf{c}, \hat{\mathbf{d}})$ is the region’s per-sweep measurement, which is passed to the FAR cross-validation and then to the temporal filter. Figure 5.8 shows a fitted plane and the extracted tangent line on a row segment.

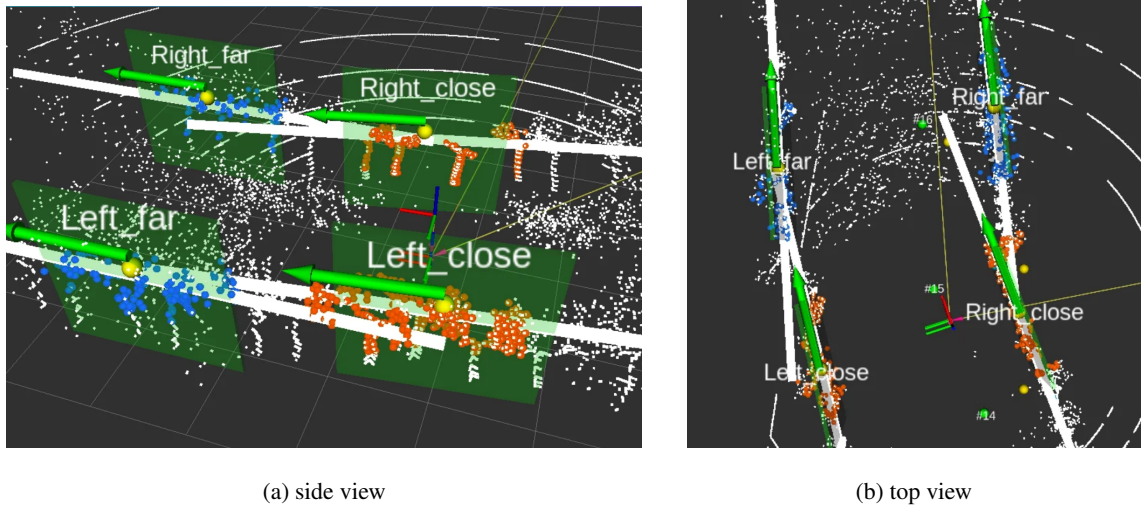


Figure 5.8: Vertical-plane fit and tangent-line extraction for a single region of interest, shown as (a) a side view and (b) a top view. The plane normal is constrained to the horizontal; the extracted line direction is the 90° rotation of the normal’s horizontal component.

The RANSAC inlier ratio and the resulting row-line accuracy are reported in Section 8.2.1.

5.5 FAR-Line Cross-Validation

The vertical-plane fit of Section 5.4 is applied to each region of interest independently. For the FAR region this independence is a liability: because the region looks several metres ahead, where the cloud is sparser, it can return a geometrically valid fit that belongs not to the followed row but to the adjacent row, seen through a gap in the canopy. To detect this case the FAR line is cross-validated against the CLOSE line before its detection is accepted. The CLOSE region, which measures the row abreast of the robot from a denser cloud, is treated as the more reliable reference, and the two lines are required to be mutually consistent in orientation and in lateral position.

Let $\mathcal{L}_C = (\mathbf{c}_C, \hat{\mathbf{d}}_C)$ and $\mathcal{L}_F = (\mathbf{c}_F, \hat{\mathbf{d}}_F)$ be the CLOSE and FAR lines on the current sweep. The first check concerns orientation. Writing $\angle \hat{\mathbf{d}} = \text{atan2}(d_y, d_x)$ for the heading of a unit direction, the angular discrepancy between the two lines is the wrapped absolute difference of their headings,

$$\Delta\theta = \left| \text{wrap}_{(-\pi, \pi]}(\angle \hat{\mathbf{d}}_F - \angle \hat{\mathbf{d}}_C) \right|, \quad (5.24)$$

where $\text{wrap}_{(-\pi, \pi]}$ reduces an angle to the interval $(-\pi, \pi]$. Because both directions are anchored to the robot heading (Section 5.4), two lines belonging to the same, locally straight or gently curved

corridor differ in heading only slightly; a large discrepancy indicates that the FAR fit has latched onto a differently oriented structure. The line is rejected when $\Delta\theta$ exceeds a tolerance $\theta_{\max}$.

The second check concerns lateral position. The orthogonal distance from the FAR centroid to the CLOSE line,

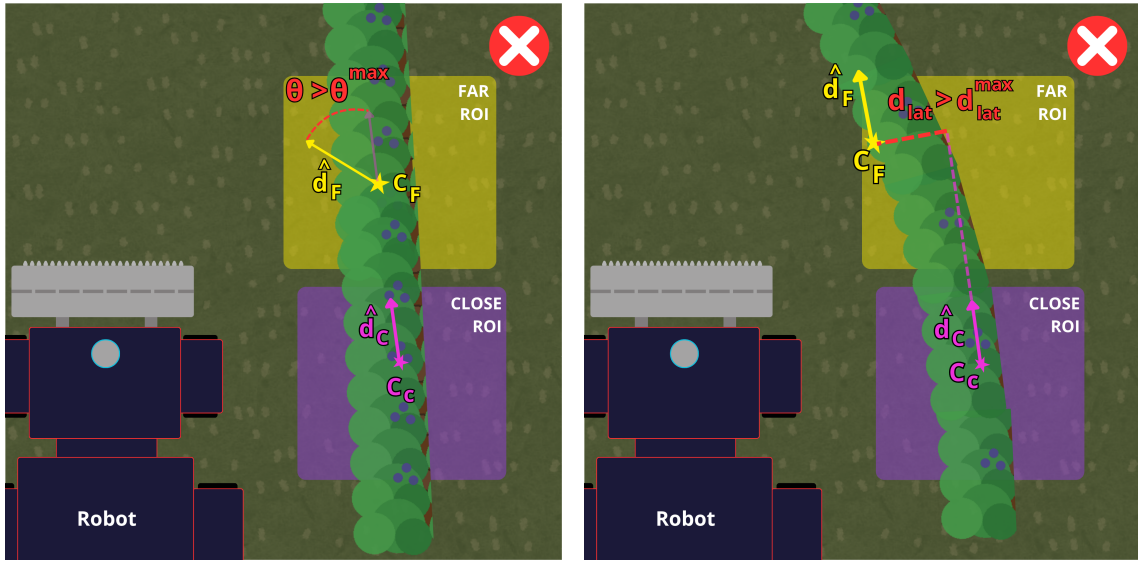
$$d_{\perp} = \|\mathbf{u} - (\mathbf{u} \cdot \hat{\mathbf{d}}_C) \hat{\mathbf{d}}_C\|, \quad \mathbf{u} = \mathbf{c}_F - \mathbf{c}_C, \quad (5.25)$$

measures how far the FAR estimate lies to the side of the line the robot is currently following. A FAR line that genuinely continues the followed row keeps a small $d_{\perp}$, of the order of the row's own deviation over the lookahead distance, whereas a fit on the next row across a gap is displaced laterally by approximately the inter-row spacing. The line is rejected when $d_{\perp}$ exceeds a tolerance $d_{\max}$, which is set well below the inter-row spacing so that the adjacent row is excluded.

The FAR detection is accepted only if both gates are satisfied,

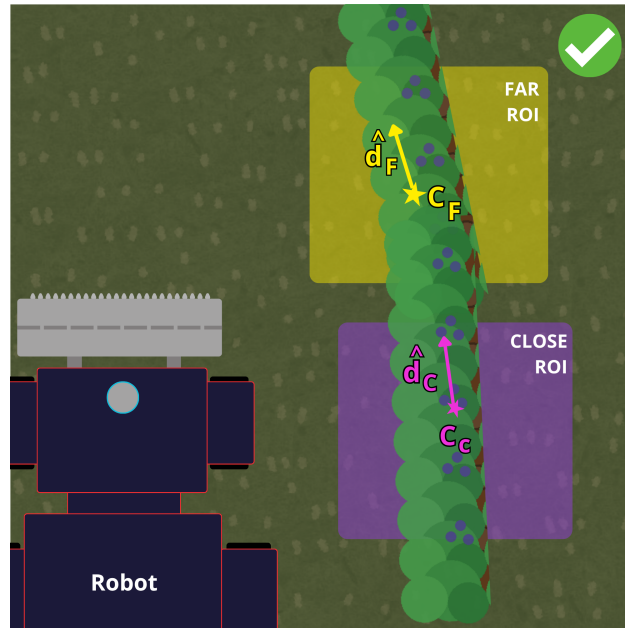
$$\text{valid}_F = (\Delta\theta \leq \theta_{\max}) \wedge (d_{\perp} \leq d_{\max}), \quad (5.26)$$

with $\theta_{\max} = 45^\circ$ and $d_{\max} = 2.0$ m. A FAR fit that satisfies the acceptance gates of Section 5.4 but fails this cross-validation has its per-sweep verdict downgraded to -1 : the geometric estimate is still produced, but the sweep is recorded as a failed detection for the side and contributes negatively to its accumulated safety score (Section 5.7). The cross-validation therefore acts on the *verdict* rather than on the geometry, leaving the smoothing of the accepted estimate to the temporal filter of Section 5.6. Figure 5.9 shows the two failure modes, an excessive heading discrepancy and an excessive lateral offset, alongside an accepted FAR line.



(a) rejected: heading

(b) rejected: lateral offset



(c) accepted

Figure 5.9: Geometric cross-validation of the FAR line against the CLOSE line. (a) the FAR heading differs from the CLOSE heading by more than $\theta_{\max}$; (b) the FAR centroid lies more than $d_{\max}$ to the side of the CLOSE line; (c) both are within tolerance and the line is accepted. In (a) and (b) the sweep is recorded as a failed detection.

The effect of this cross-validation on the per-side detection availability is quantified in Section 8.2.2.

5.6 Temporal Line Filtering: Inlier-Ratio-Weighted FIR with Circular Averaging

Despite the applied gating conditions, the per-sweep line produced by the fit of Section 5.4 is still noisy: the cropped cloud changes from one sweep to the next, and the RANSAC fit varies with it, so the raw estimate of a region jitters even when the robot and the row are stationary. Before a line is reported it is therefore smoothed over time by a finite-impulse-response (FIR) filter that averages the last M accepted estimates of the region, weighting each by the confidence of its fit. This is a standard signal-processing step rather than a novel component; its purpose is to stabilise the geometry passed on to region placement (Section 5.3) and to the virtual-boundary construction (Section 5.9).

The filter maintains a sliding window of the last M accepted line estimates of the region, with $M = 5$; when a new estimate is accepted, the oldest is discarded. On the first valid detection the window is filled with that single estimate, a warm start that lets the filter produce a sensible output from the first sweep rather than converging over M sweeps.

Each estimate i in the window carries the inlier ratio ρ_i of its fit (Equation (5.22)), which is used directly as its weight $w_i = \rho_i$. Confident fits, supported by a large fraction of the points, dominate the average, whereas marginal fits contribute little; the smoothing is in this way adaptive, without an explicit gain schedule, since a low-confidence sweep barely moves the reported estimate. Writing $W = \sum_{i=1}^M w_i$ for the total weight, the centroid is smoothed by the linear weighted mean

$$\bar{\mathbf{c}} = \frac{1}{W} \sum_{i=1}^M w_i \mathbf{c}_i. \quad (5.27)$$

The direction angle φ cannot be averaged in the same way, because it is an angular quantity: a linear mean of angles would be corrupted by wrap-around near $\pm\pi$. It is instead averaged circularly, as the normalised weighted vector sum of the unit directions,

$$\bar{\mathbf{d}} = \frac{\sum_{i=1}^M w_i \hat{\mathbf{d}}_i}{\left\| \sum_{i=1}^M w_i \hat{\mathbf{d}}_i \right\|}, \quad (5.28)$$

which is equivalent to the weighted circular mean of the heading angles. The sign anchoring of Section 5.4 keeps every direction vector $\hat{\mathbf{d}}_i$ within the half-plane of the travel direction, so the weighted sum cannot cancel and the mean is always well defined; the normalisation then guarantees that the resulting direction has unit norm. The filtered line $(\bar{\mathbf{c}}, \bar{\mathbf{d}})$ is the estimate reported for the region and used to place its crop box on the following sweep (Section 5.3). Figure 5.10 compares the raw and filtered line parameters along a traversal.

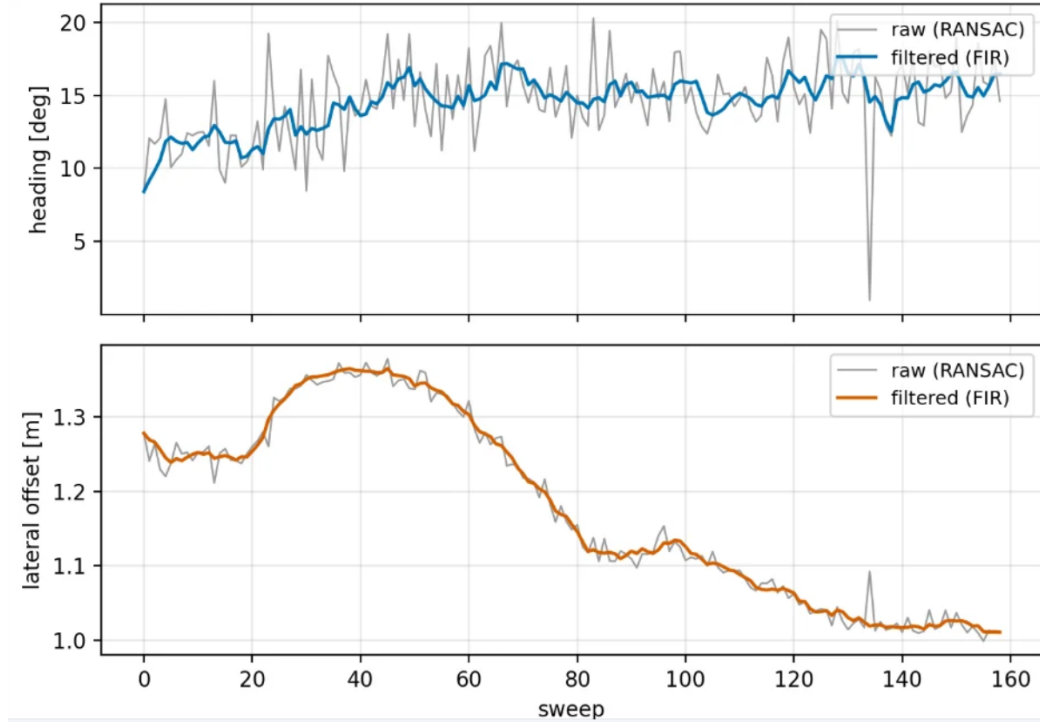


Figure 5.10: Effect of the temporal filter on the per-sweep line estimate of the left `CLOSE` region: heading (top) and lateral offset (bottom), raw per-sweep fit versus FIR-filtered. The inlier-ratio weighting yields adaptive smoothing without an explicit gain schedule.

Its contribution to the reported row-line accuracy is reflected in Section 8.2.1.

5.7 Per-Sweep Verdict and Accumulated Safety Score

Each side condenses the outcome of a sweep into a single binary verdict. A region of interest is deemed to have succeeded on the sweep when its fit passes the acceptance gates of Section 5.4 and, for the `FAR` region, the cross-validation of Section 5.5; otherwise it has failed. Denoting the `CLOSE` and `FAR` outcomes by $\delta_C, \delta_F \in \{+1, -1\}$, the per-sweep verdict of the side is their conjunction,

$$\delta = \min(\delta_C, \delta_F), \quad (5.29)$$

which is $+1$ only when both regions succeed and -1 as soon as either fails. A side is therefore considered well observed on a sweep only if both of its horizons are.

A single verdict is too brittle to drive behaviour, since isolated failures occur routinely under occlusion or on a momentarily sparse sweep. The verdicts are accordingly integrated over time into a bounded *safety score* $S \in [S_{\min}, S_{\max}]$. On each sweep the previous score is incremented by

Table 5.3: Bounds and initial value of the per-side safety score.

Symbol	Meaning	Value
$S_{\max}$	Ceiling; score of a reliably tracked side	0
$S_{\min}$	Floor; score at which the side is treated as lost	-20
S^0	Initial score at start-up (one step below the ceiling)	-1

the verdict and saturated at the two bounds,

$$S^k = \begin{cases} S_{\max}, & \text{if } S^{k-1} + \delta^k > S_{\max}, \\ S_{\min}, & \text{if } S^{k-1} + \delta^k < S_{\min}, \\ S^{k-1} + \delta^k, & \text{otherwise.} \end{cases} \quad (5.30)$$

The bounds $S_{\min}$ and $S_{\max}$ and the initial value S^0 are listed in Table 5.3.

The score is a saturating integrator: each successful sweep increments it towards the ceiling $S_{\max}$ and each failed sweep decrements it towards the floor $S_{\min}$, the saturation in (5.30) bounding it on both sides. This saturation gives the score hysteresis: a side that has been tracked reliably rests at the ceiling and tolerates a run of failures before the score falls appreciably, whereas a side that has been lost must accumulate a run of successes before it is trusted again. The score is initialised one step below the ceiling, at $S^0 = -1$, so that a side reaches full confidence only after recording at least one valid sweep; a side whose first sweep fails drops immediately below the initial value.

When the score reaches the floor, $S = S_{\min}$, the side is treated as lost: the line histories of both its regions of interest are discarded, so that on the next sweep their crop boxes are placed from the bootstrap prior of Section 5.3 rather than from a stale estimate, and the side re-acquires the row from scratch. A single successful sweep then lifts the score off the floor and history-based tracking resumes. The safety score thus couples detection health back to region placement, closing the loop between the verdict and the predictive crop of Section 5.3. The complete life cycle of the score is summarised in Figure 5.11.

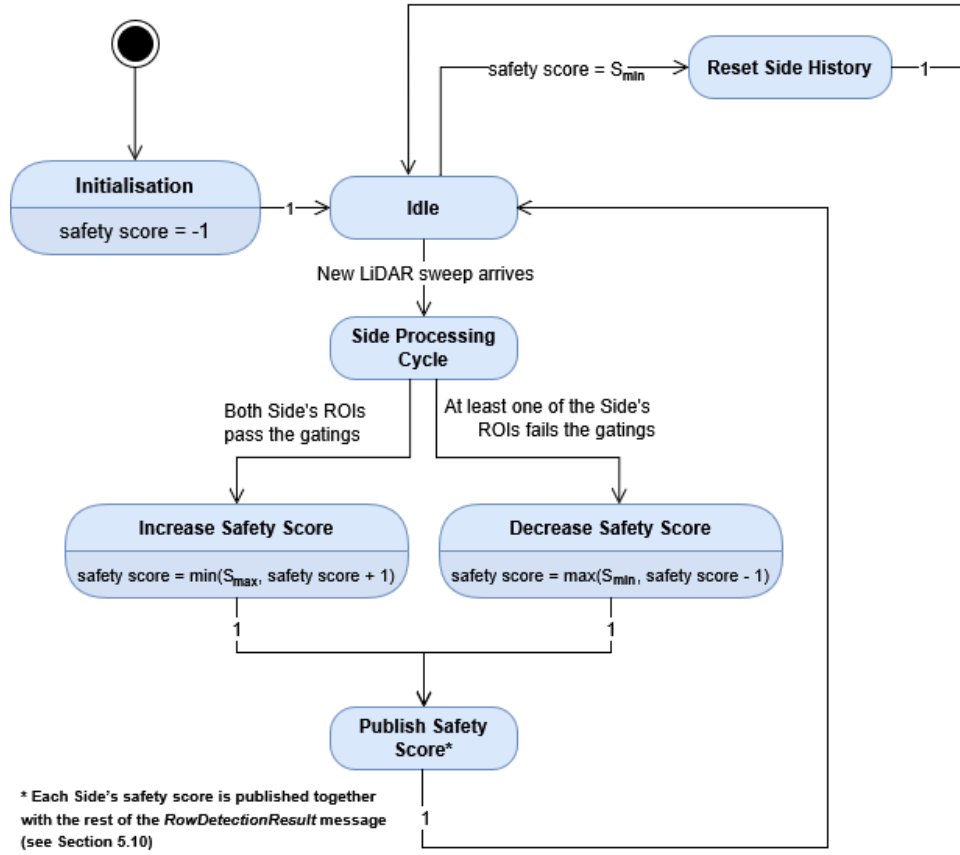


Figure 5.11: Life cycle of the per-side safety score, from accumulation and saturation to the at-floor anchor reset, recovery, and publication.

The accumulation is performed inside the perception node, and it is the integrated score, not the per-sweep verdict, that each `RowDetectionResult` carries for the side (Section 5.10). This is a deliberate producer–decider separation: perception maintains the evidence of detection health, while the planner consumes the already-integrated score and applies its own thresholds to arbitrate between following the row, switching the reference side, and halting (Chapter 7). Beyond the anchor reset described above, the perception node takes no navigation decision from the score. The downstream effect of this scoring, the per-side detection availability under sustained loss, is reported in Section 8.2.2.

5.8 Canopy Dispersion

A region of interest captures not only the row line but the lateral spread of the foliage about it. This spread — the *canopy dispersion* — quantifies how thick the row is on a given sweep, and is used as a physically meaningful margin by which the navigable corridor is inset from the row: a denser, wider canopy pushes the boundary further from the vines (Section 5.9).

For a region with filtered cloud $\{\mathbf{p}_i\}_{i=1}^N$ and line $\mathcal{L} = (\mathbf{c}, \hat{\mathbf{d}})$, let $\mathbf{n} = (-\hat{d}_y, \hat{d}_x)$ be the unit normal to the row. The signed lateral offset of a point is its projection onto this normal, $\rho_i =$

$(\mathbf{p}_i - \mathbf{c}) \cdot \mathbf{n}$, and the dispersion is the standard deviation of these offsets, clamped to a maximum,

$$\sigma = \min\left(\sqrt{\frac{1}{N} \sum_{i=1}^N (\rho_i - \bar{\rho})^2}, \sigma_{\max}\right), \quad \bar{\rho} = \frac{1}{N} \sum_{i=1}^N \rho_i, \quad (5.31)$$

where $\bar{\rho}$ is the mean offset, N the number of points, and $\sigma_{\max} = 0.5$ m bounds the estimate against severe outliers. A region's dispersion is regarded as valid only when σ exceeds a small floor of 0.05 m, below which the measured spread is indistinguishable from sensor noise.

Because ρ_i is an orthogonal projection onto the row's own normal, σ depends only on the relative geometry of the points and the fitted line and is therefore invariant to the frame in which the cloud is expressed. It is computed independently for the CLOSE and FAR regions of each side, carried per region in the `RowDetectionResult` (Section 5.10), and consumed by the virtual-boundary construction of the next section, where it sets the inset of the corridor boundary from the detected row. Figure 5.12 illustrates the measure.

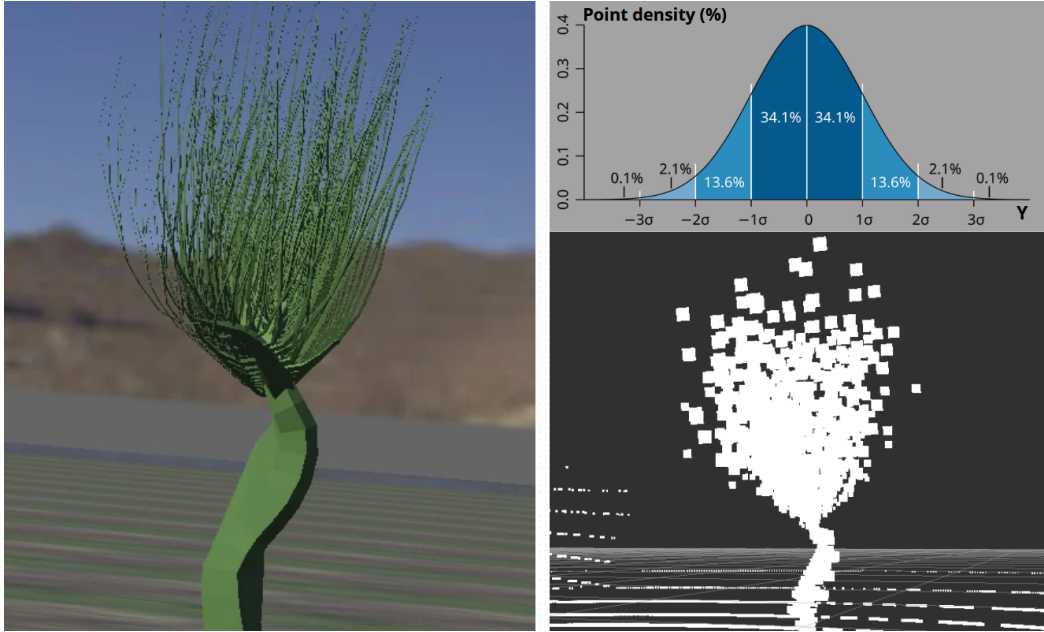


Figure 5.12: Canopy dispersion. (*left*) the Webots vine model of a single crop, stacked longitudinally to form a corridor row; (*right*) the LiDAR point cloud returned by that model, with a Gaussian fitted to the lateral spread of the returns. The standard deviation σ of that spread is the canopy dispersion, used as the inset margin of the corridor boundary.

5.9 Virtual Boundaries and the Inter-Row Polygon

The perception cycle closes by converting the two row models into a closed polygon that delimits the navigable inter-row area. This polygon is one of the central structures of the architecture: besides bounding the corridor for navigation, it is handed to the mapping node as the spatial mask within which vegetation is accumulated and the cut line is sought (Chapter 6). Perception therefore

does not merely feed geometry to the planner; it dictates, on every sweep k , the region in which the mapping node is permitted to act. Binding the two subsystems through a single perception-derived boundary is a fundamental contribution of this work, and it is one of the elements least represented in the surveyed literature (Section 3.5). The construction has three parts: a per-side boundary strip, the closed polygon assembled from the two strips, and a fallback that keeps the polygon well formed when one row is lost.

Per-side boundary strip. Each side contributes one boundary strip, modelled as a cubic Bézier curve that runs alongside its row and is inset towards the corridor interior by a margin set by the canopy dispersion of Section 7.5. A cubic Bézier is adopted because, over the short look-ahead stretch considered here, the vine rows are not expected to curve so sharply that a single cubic segment becomes insufficient. This provides a compact but expressive representation for the boundary.

A second motivation is implementation consistency. The trajectory reference produced by the planner is also a cubic Bézier built from the `CLOSE` and `FAR` row directions (Chapter 7). The boundary strips therefore reuse the same curve representation, together with the associated data structures, functions, and namespaces already implemented in the planning module. Only the parameters that define the lateral offsets of the control points and the tangent imposed at the end point are changed, avoiding a separate curve-generation implementation and reducing code duplication between planning and mapping.

Concretely, each strip is defined by four control points. The start point is anchored on the `CLOSE` region, the end point on the `FAR` region, and the two intermediate control points are obtained from these endpoints and their headings so that the strip leaves and arrives tangentially, yielding a continuous curve. This four-point construction is the same one formalised in Section 7.5 for generating the trajectory reference from the row geometry. Figure 5.13 shows the construction of a single strip, while Figure 5.16 shows how two discretised strips are later joined into the closed corridor polygon.

The strip is recomputed on every sweep k from that side's current row models.

Let $\hat{\mathbf{n}}_C^k$ and $\hat{\mathbf{n}}_F^k$ be the inward unit normals of the `CLOSE` and `FAR` regions, each the 90° rotation of its region's direction towards the corridor centre. For a region on side s with unit direction $\hat{\mathbf{d}} = (d_x, d_y)$, the two possible planar perpendicular directions are $(d_y, -d_x)$ and $(-d_y, d_x)$; the one pointing into the corridor is selected by the side,

$$\hat{\mathbf{n}}_s = \begin{cases} (d_y, -d_x), & s = \text{Left}, \\ (-d_y, d_x), & s = \text{Right}, \end{cases} \quad (5.32)$$

with $\hat{\mathbf{n}}_C^k$ and $\hat{\mathbf{n}}_F^k$ obtained from the `CLOSE` and `FAR` directions, respectively. This side-dependent sign convention, restated for the planner in Chapter 7, guarantees that the inset always moves the boundary towards the interior of the corridor, whichever side is processed. Each endpoint is then offset along its own region's normal direction by a certain distance: the start point P_0^k is the *foot*

point of the robot on the CLOSE line displaced by the CLOSE dispersion, and the end point P_3^k is the FAR centroid displaced by the FAR dispersion,

$$P_0^k = \mathbf{p}_{\text{foot}}^k + k_\sigma \sigma_C^k \hat{\mathbf{n}}_C^k, \quad P_3^k = \mathbf{c}_F^k + k_\sigma \sigma_F^k \hat{\mathbf{n}}_F^k, \quad (5.33)$$

where $\mathbf{p}_{\text{foot}}^k$ is the foot point on the CLOSE line, $\mathbf{c}_F^k$ the FAR centroid, σ_C^k, σ_F^k the CLOSE and FAR canopy dispersions, and $k_\sigma = 0.5$ a fixed dispersion gain. Because the inset scales with the measured dispersion, a bushier row pushes its boundary further into the corridor, on every sweep.

The two interior control points of the cubic Bézier follow from the headings at its endpoints. With θ_0^k the heading at the start point, taken as the CLOSE direction, and θ_3^k the heading at the end point, taken as the FAR direction, the interior points are placed along those headings at a fraction λ of the chord length $\|P_3^k - P_0^k\|$,

$$P_1^k = P_0^k + \lambda \|P_3^k - P_0^k\| \begin{pmatrix} \cos \theta_0^k \\ \sin \theta_0^k \end{pmatrix}, \quad P_2^k = P_3^k - \lambda \|P_3^k - P_0^k\| \begin{pmatrix} \cos \theta_3^k \\ \sin \theta_3^k \end{pmatrix}, \quad (5.34)$$

where λ is the control-point distance ratio (set to 0.3). This construction makes the strip leave P_0^k tangent to the CLOSE direction and arrive at P_3^k tangent to the FAR direction, so that it follows the near geometry where the robot is and turns onto the look-ahead geometry it is about to enter. The strip is then the cubic Bézier $\mathbf{B}^k(u)$, $u \in [0, 1]$, with control points $(P_0^k, P_1^k, P_2^k, P_3^k)$.

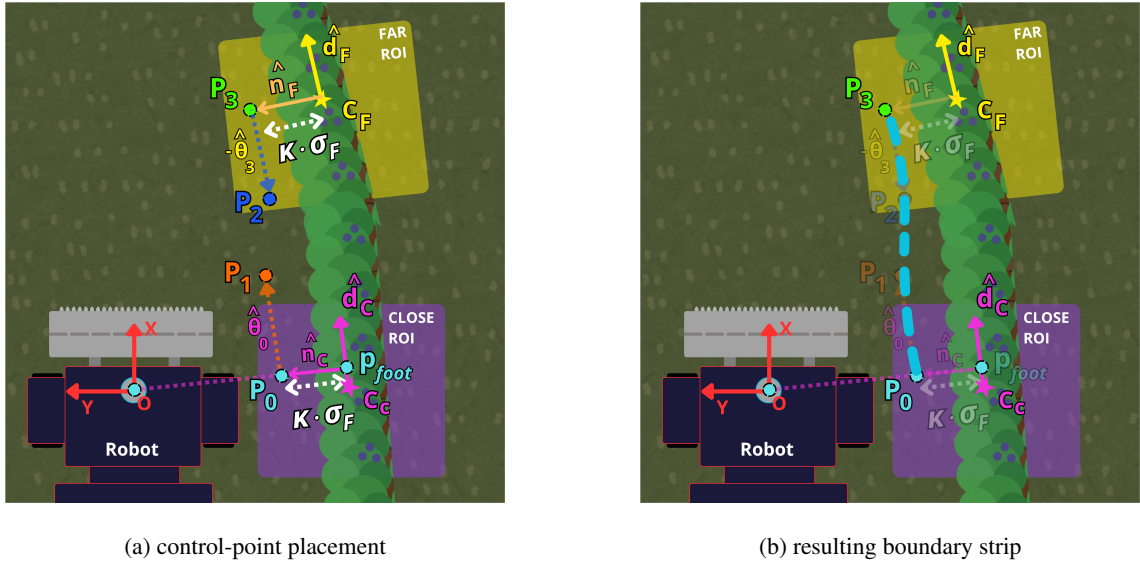


Figure 5.13: Construction of a single per-side boundary strip: (a) placement of the four control points, the start point P_0 anchored on the CLOSE region and the end point P_3 on the FAR region, with P_1 and P_2 placed along their respective headings; (b) the resulting cubic Bézier strip, shown in blue.

Inter-row width. The fallback below requires a measure of the corridor width. While both sides are fully confident ($S = S_{\text{max}}$ on each), the inter-row width is estimated as the orthogonal distance

between the two CLOSE lines.

Let $\mathbf{u} = \mathbf{c}_C^L - \mathbf{c}_C^R$ be the vector between the left and right CLOSE centroids and $\hat{\mathbf{d}}_C^R$ the right CLOSE direction. The term $(\mathbf{u} \cdot \hat{\mathbf{d}}_C^R) \hat{\mathbf{d}}_C^R$ is the orthogonal projection of $\mathbf{u}$ onto the right line; subtracting it from $\mathbf{u}$ removes the along-row component and leaves the component perpendicular to the rows, whose magnitude is the width,

$$W = \|\mathbf{u} - (\mathbf{u} \cdot \hat{\mathbf{d}}_C^R) \hat{\mathbf{d}}_C^R\|, \quad \mathbf{u} = \mathbf{c}_C^L - \mathbf{c}_C^R. \quad (5.35)$$

Equivalently, W is the distance from the left CLOSE centroid to its foot point on the right CLOSE line. Restricting the update to fully confident sweeps prevents W from being corrupted by a degraded side; the last good value is retained for use when a side is subsequently lost. The geometry of this estimate is illustrated in Figure 5.14.

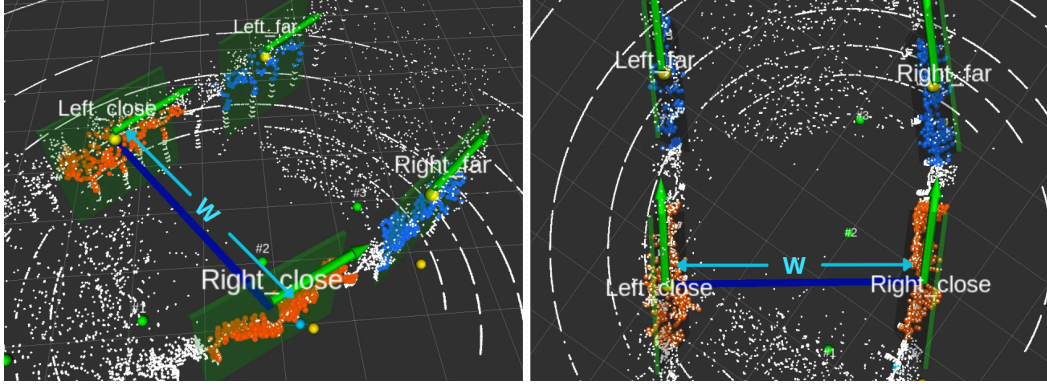


Figure 5.14: Estimation of the inter-row width W as the orthogonal distance between the two CLOSE row lines, shown in (left) top view and (right) side view.

Single-sided fallback. A side is treated as lost when its safety score reaches the floor ($S = S_{\min}$, Section 5.7). Only one row is then observed, and the boundary of the lost side cannot be built from its own, now stale, estimate. It is instead reconstructed from the surviving side using its last valid update: the construction of (5.33) is applied to the good side's lines, but with the inward offset replaced by $W - k_\sigma \sigma$, so that the reconstructed strip is placed a full corridor width across from the good row. This rests on a *local* corridor-symmetry assumption: the two rows are taken to be parallel and a width W apart only over the short look-ahead segment ahead of the robot that the strip spans, not along the whole field. For non-parallel or diverging rows the assumption does not hold exactly, but over this short horizon the reconstructed boundary remains a good approximation of the unobserved row. If both sides are at the floor, no polygon is produced for that sweep.

Polygon assembly and publication. The two strips are joined into a single closed polygon. The left strip is sampled along its arc length, the right strip is sampled and reversed, and the two sequences are concatenated and closed by repeating the first vertex,

$$\mathcal{P}^k = [\mathbf{B}_L^k(u_0), \dots, \mathbf{B}_L^k(u_m), \mathbf{B}_R^k(u_m), \dots, \mathbf{B}_R^k(u_0), \mathbf{B}_L^k(u_0)], \quad (5.36)$$

which traces the corridor outline as a single closed ring that excludes the rows themselves (Figure 5.15). The polygon is published as a point cloud in the map frame.

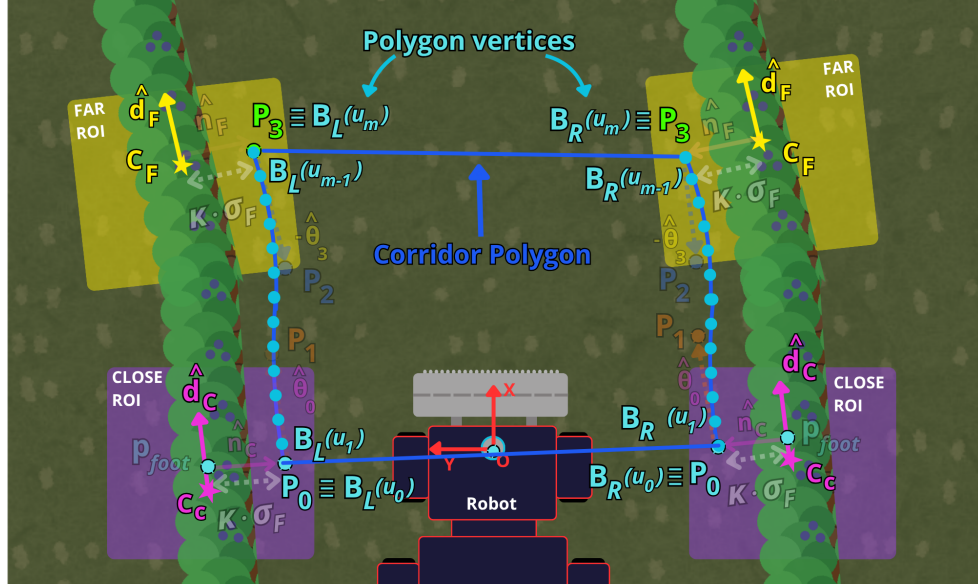


Figure 5.15: Geometric construction of the inter-row polygon $\mathcal{P}$ from the two boundary strips: the left strip sampled along its arc length, the right strip sampled and reversed, and the two sequences concatenated and closed to trace the corridor outline. Figure 5.16 shows the resulting polygon on actual corridor data.

The perception–mapping coupling. Crucially, $\mathcal{P}^k$ is the structure through which perception configures the mapping node: the mapping node clips each LiDAR sweep to $\mathcal{P}^k$ before accumulating it, so that only returns from inside the corridor enter the height map and the cut-line search. Because $\mathcal{P}^k$ is rebuilt every sweep and inset by the current canopy dispersion, this mask is set on-line by perception rather than fixed in advance — the bidirectional perception–mapping coupling identified in Section 3.5, and developed in full in Chapter 6. Figures 5.16 and 5.17 illustrate the polygon on straight and curved corridors and the single-sided fallback.

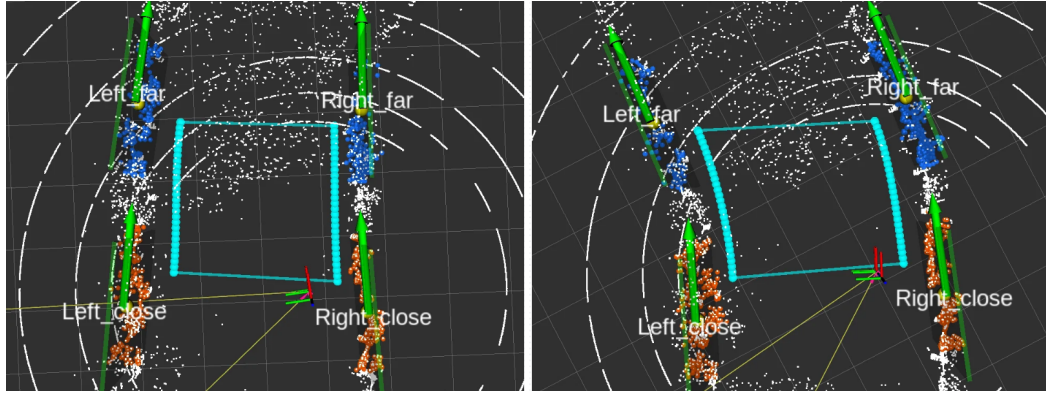


Figure 5.16: Inter-row polygon $\mathcal{P}$ assembled from the two canopy-dispersion-adaptive boundary strips, on (left) a straight corridor and (right) a curved crop corridor. Returns from the rows fall outside the polygon and are excluded from mapping.

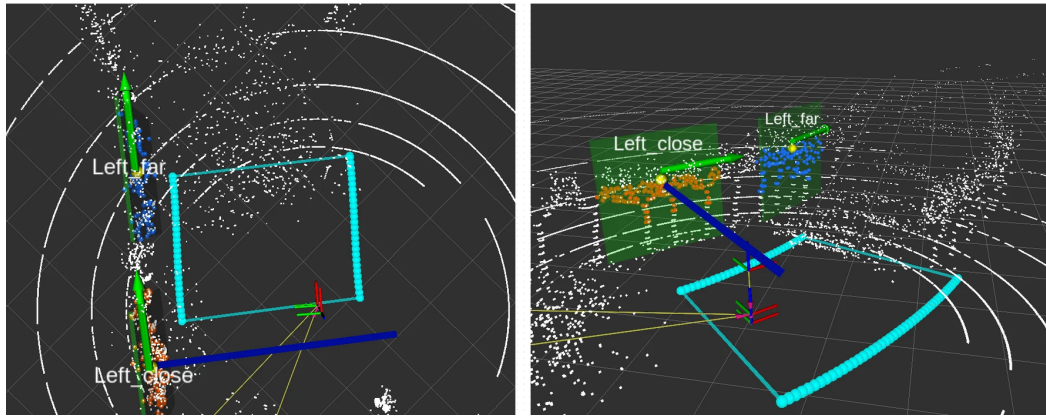


Figure 5.17: Single-sided fallback, in (left) top view and (right) side view. When one row is lost, its boundary is reconstructed from the surviving row offset by the inter-row width W under the corridor-symmetry assumption.

The corridor polygon this defines is the spatial mask consumed by the mapping node's intake stage, evaluated in Section 8.3.1.

5.10 Perception Output Message (RowDetectionResult)

The perception cycle concludes by publishing a single `RowDetectionResult` message that carries the full geometric state of both sides of the corridor to the planner. One message is emitted per sweep, time-stamped and expressed in the map frame, and it is the sole interface from perception to the planner: everything the planner needs to build the reference trajectory and to supervise its behaviour is contained in it. The corridor polygon of Section 5.9 is published separately, as a point cloud, because it is consumed by the mapping node rather than by the planner.

The message is organised per side. For each side it carries the accumulated safety score of Section 5.7; and for each of the side's two regions of interest it carries a validity flag, the line

Table 5.4: Principal fields of the `RowDetectionResult` message. The listed fields are present for both sides; * stands for the `close/far` region prefix.

Field	Type	Meaning
header	—	Time stamp and reference frame (map)
safety_score	int32	Accumulated per-side safety score (5.30)
*_valid	bool	Region holds a valid tracked estimate
*_centroid_{x,y,z}	float32	Line centroid $\mathbf{c}$ (map frame)
*_dir_{x,y}	float32	Line unit direction $\hat{\mathbf{d}}$
*_canopy_dispersion	float32	Canopy dispersion σ (5.31)
*_inlier_ratio	float32	RANSAC inlier ratio ρ ; -1 if no valid plane

centroid and unit direction, the canopy dispersion, and the RANSAC inlier ratio of the fit, the last serving as a per-region confidence that is set to -1 when no valid plane was found on the sweep. The fields are listed in Table 5.4; the same set is repeated for the left and right sides, and within each side for the `CLOSE` and `FAR` regions.

With the perception node now described in full, from the sweep-driven cycle to the message it publishes, the next chapter takes the corridor polygon as its input and develops the vegetation mapping and cut-line detection that build the height map within it; the local planner that turns the `RowDetectionResult` into a continuous reference trajectory is then the subject of Chapter 7.

Chapter 6

Mapping: Corridor Height Mapping and Cut-Line Detection

This chapter develops the mapping node, which turns the corridor that perception delimits into the two products that support the mowing mission: a vegetation-height map of the inter-row floor and the cut line that separates the already-mown swath from the standing vegetation. It begins with the synchronised intake that pairs each LiDAR sweep with the corridor polygon and reduces it to the points lying on the corridor floor; the remaining sections then build the height map from that cloud and detect the cut line that guides the return pass.

6.1 Overview and Processing Cycle

Each LiDAR sweep paired with its corridor polygon triggers one processing cycle of the mapping node, from the synchronised intake through to the publication of the mission and local map products and, when the evidence supports it, the cut line. Figure 6.1 shows this cycle at a glance: intake and corridor extraction (Section 6.2) feed both the mission-level height map (Section 6.3) and the robot-centric grid map (Section 6.4), from which the cut-line detector (Section 6.5) draws its input before publication (Section 6.5.8).

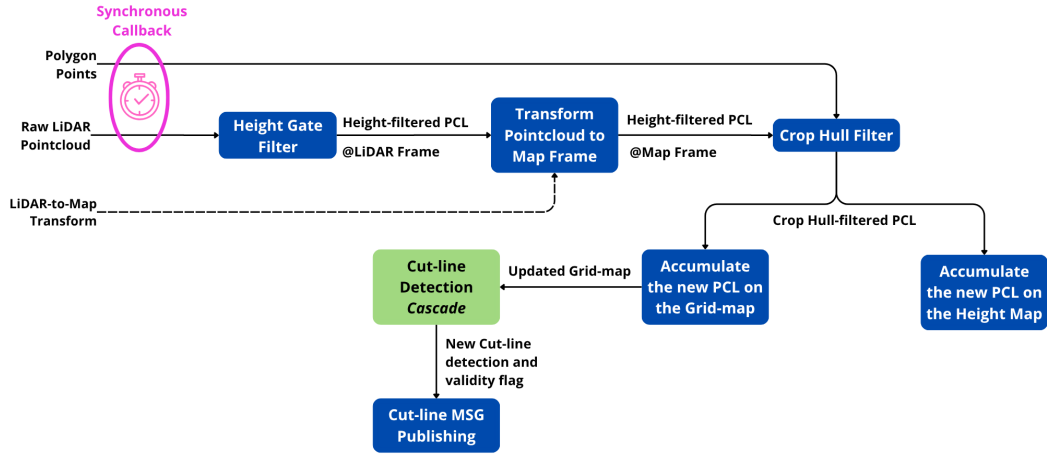


Figure 6.1: Per-sweep processing pipeline of the mapping node, from the synchronised intake to the publication of the height-map products and the cut line.

Cut-line detection, in particular, is not a single step but a cascade of acceptance gates that a candidate must clear before it is accepted and published, from per-section profiling through to cross-section consensus and the final fit. Figure 6.2 previews this cascade; each stage is developed in full in Section 6.5.

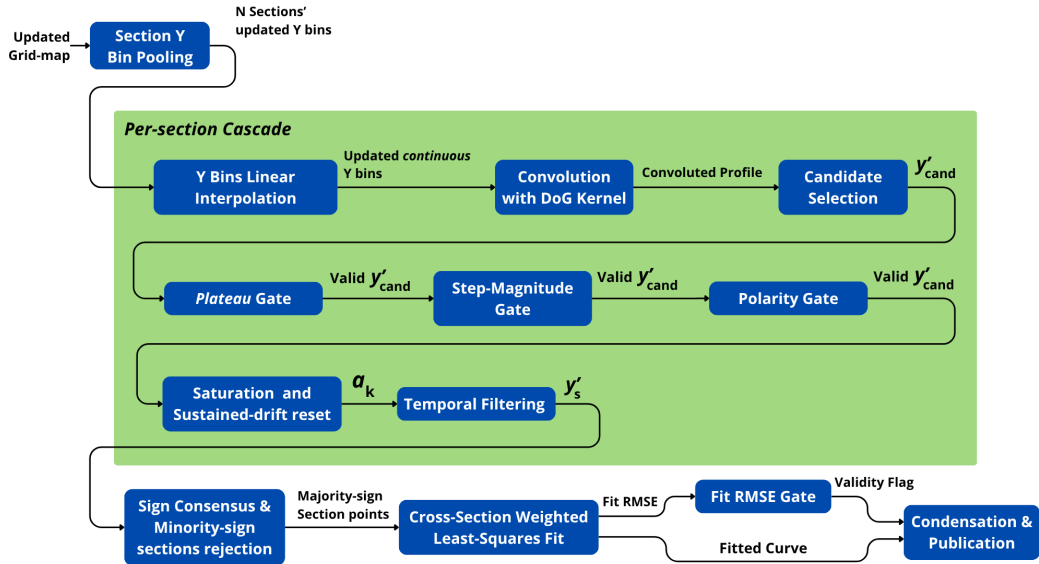


Figure 6.2: The cut-line detection cascade: per-section profiling, the acceptance gates applied to each candidate, cross-section consensus, and the publication of the resulting cut line.

6.2 Synchronised Intake and Corridor Extraction (CropHull)

The mapping node operates on the same LiDAR sweeps as perception, but it does not act on the whole sweep: it acts only on the vegetation inside the corridor that perception has delimited. To do so it must pair each sweep with the boundary polygon computed for it. The node therefore subscribes to two streams, the raw LiDAR cloud on `/raw_lidar` and the corridor polygon on `/mowIT/polygon_points` (Section 5.9), and couples them with an approximate-time synchroniser, so that every cloud is processed together with the polygon produced from the same robot pose. This synchronised intake is the consuming end of the perception–mapping coupling: the boundary that perception builds on-line becomes, here, the spatial filter that selects what is mapped.

Height gating. Before the corridor filter is applied, the sweep is gated in height to retain only the returns near the ground in order to remove outliers and points referring to lateral vine canopy regions that might still overhang the corridor even after the canopy-dispersion margin (Section 5.9). The gate is a pass-through on the vertical coordinate, and it is applied first in the pipeline directly on the raw cloud, in the LiDAR frame, before any coordinate transform and before the corridor hull filter. Because the band is defined in the LiDAR frame, its limits follow from the base height alone, so no transform of the cloud is needed to evaluate them. The pass-through evaluates a single coordinate comparison per point and thus has a low per-point computational cost, yet it discards a large fraction of the sweep. The operations that follow carry a substantially higher per-point cost: the transform into the map frame and, above all, the point-in-hull test of `CropHull`, which evaluates a point-in-polygon predicate for every point. Applying the height gate first reduces the size of the cloud on which these higher-cost operations act, and therefore reduces the total number of point-wise operations performed per sweep.

Let z_b be the height of the robot base expressed in the LiDAR frame, read from the TF tree; assuming the LiDAR sensor is mounted z_L m above the base of the robot, $z_b = -z_L$, the z coordinate of the base (corresponding to the expected ground level) on the LiDAR frame is negative. A point is retained when its vertical coordinate lies within a band of half-width T about the base plane,

$$z \in [z_b - T, z_b + T] \quad (6.1)$$

so that the band spans from T below the ground, which absorbs small terrain dips, to T above it, which sets the ceiling on the vegetation admitted to the map.

The decisive limit is the upper one, $z_b + T$, and the value of T trades two failure modes against each other. If T is set too high, the band reaches up into the lateral vine canopy, and the denser foliage of the side rows that overhangs the corridor is admitted and pollutes the height map. If T is set too low, genuinely tall inter-row vegetation is clipped at the ceiling, so that its height saturates and is under-reported. The value of T is therefore tuned to sit above the tallest grass the system must measure yet below the vine canopy; the value adopted, $T = 0.60$ m, was selected

empirically. Figure 6.3 shows the gated cloud under several height limits and the resulting contamination/saturation trade-off.

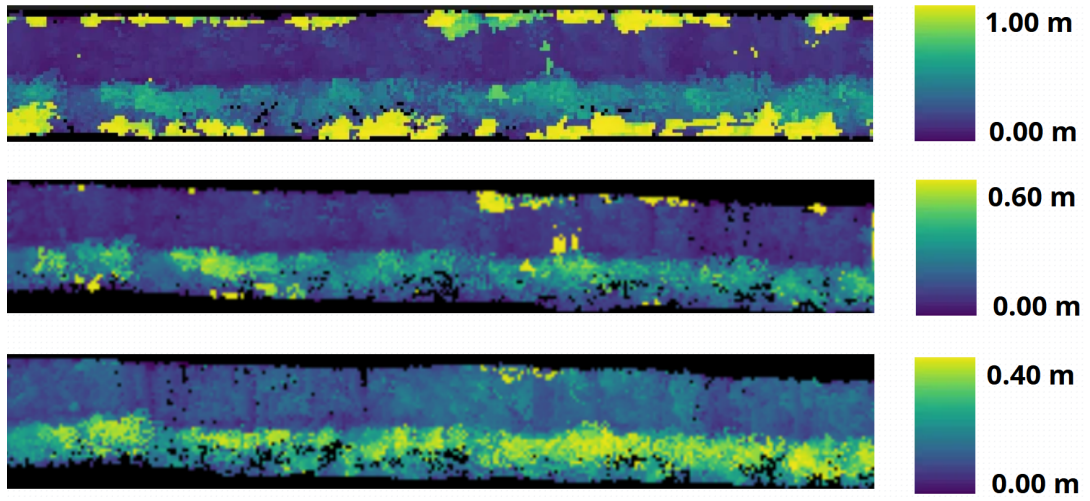


Figure 6.3: Effect of the height-gate half-width T on the retained cloud, and the contamination/saturation trade-off that fixes the adopted value.

Chassis exclusion and corridor extraction. The height-gated cloud is transformed into the map frame, where a negative crop box of the robot footprint (1.0×1.0 m, the guard also used in perception) removes the platform’s own returns. The corridor polygon is then applied as a hull filter. The polygon points received from perception, once transformed into the map frame, define a closed two-dimensional hull; CropHull retains only the points whose horizontal position lies inside this hull and discards the rest, effectively delimiting the navigable area of the inter-row corridor (Figure 6.4).

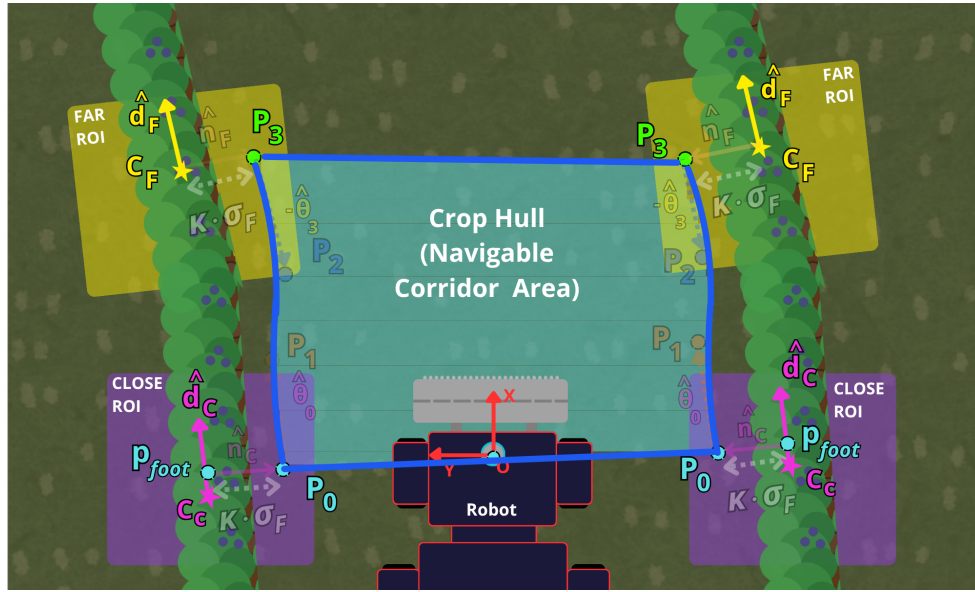


Figure 6.4: Construction of the navigable corridor area from the corridor polygon received from perception: the geometric hull used to gate the mapping node's intake.

Because the polygon is inset from the rows by the canopy-dispersion margin (Section 5.9), most of the lateral vine structures fall outside the hull and are removed, leaving a cloud composed mostly of the low vegetation on the corridor floor. The remaining points of the crop rows, associated with denser foliage regions of the side rows, are removed by the previously mentioned height filter. The pass-through, the crop box and the hull filter are all primitives of the C++ Point Cloud Library (PCL). The resulting corridor-only, height-gated cloud, illustrated in Figure 6.5, is the input to the height map and the cut-line detector developed in the following sections.

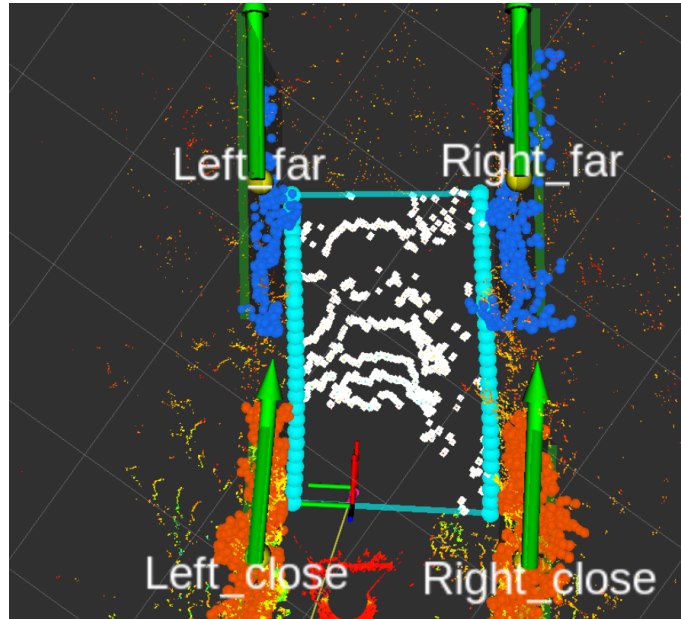


Figure 6.5: Corridor extraction. The height gate and the perception-derived `CropHull` reduce the raw sweep to the vegetation on the corridor floor.

The ordering rationale keeps this intake pipeline cheap: the low-cost height gate already discards the bulk of the sweep, so that the frame transform and the point-in-hull test of `CropHull` operate on only a fraction of the original returns. The point count surviving each stage and the execution time of each operation are reported quantitatively in Section 8.3.1.

6.3 Mission-Level Sparse 2.5D Height Map

The corridor-only cloud produced in Section 6.2 feeds the first of the two mapping products: a mission-level height map of the inter-row vegetation. Its role is not navigation inside the immediate control loop but documentation. It is intended as an accurate, persistent record of the vegetation along the corridor that can be handed to the grower as a survey of the plot or passed to a downstream decision-support or crop-management system. The product the operator ultimately sees is a single image of vegetation height along the traversed corridor.

Because the map records *height*, the value of the product depends entirely on what is allowed to enter it: any tall structure that is not inter-row vegetation would be registered as spurious high vegetation. The vine trunks and the denser foliage of the side rows are exactly such structures, and this is the reason the cloud is restricted to the corridor by the height gate and the `CropHull` of Section 6.2 before it reaches the map. The argument becomes sharper if the height map is later reused for navigation: a traversal cost map keyed on vegetation height would, without that restriction, penalise the centre of the corridor for foliage that in fact belongs to the rows on either side, and would place phantom obstacles at the trunks. Excluding the rows at the source therefore protects the map both as a survey product and as a possible cost layer.

Representation. The map is a two-and-a-half-dimensional field: the ground plane is tessellated into square cells of side r (the resolution, $r = 0.05$ m), and each cell stores statistics of the heights observed within it rather than a full third spatial dimension. A dense array over the mission area would demand that its extent be known in advance and would reserve storage for every cell of the bounding rectangle, the overwhelming majority of which the robot never observes. The map instead retains only the cells that actually receive points, holding them in a hash map keyed by the integer cell index (i, j) ; storage then grows with the observed area rather than with the bounding box, and no prior knowledge of the mission extent is required. The structure is implemented independently of ROS, as a self-contained module operating on point clouds already expressed in the map frame.

A point (x, y) in the map frame is assigned to the cell

$$i = \left\lfloor \frac{x - x_0}{r} \right\rfloor, \quad j = \left\lfloor \frac{y - y_0}{r} \right\rfloor, \quad (6.2)$$

where (x_0, y_0) is the map origin. The origin is anchored once to the robot's starting pose, and the grid axes are aligned with the starting heading, so that the index i measures forward distance from the start of the mission. The use of $\lfloor \cdot \rfloor$ makes the assignment valid for negative coordinates as well as positive ones.

Incremental per-cell statistics. A cell does not retain the points that fall in it. It keeps five running accumulators from which all height statistics are recovered in closed form: the count n , the extremes $z_{\min}$ and $z_{\max}$, and the first two moment sums

$$S_1 = \sum_{p=1}^n z_p, \quad S_2 = \sum_{p=1}^n z_p^2. \quad (6.3)$$

Each incoming point updates the extremes by comparison and the two sums by addition, so the per-point update is a fixed, small set of arithmetic operations. The mean and the population variance are obtained from the accumulators as

$$\bar{z} = \frac{S_1}{n}, \quad \sigma_z^2 = \max\left(0, \frac{S_2}{n} - \bar{z}^2\right), \quad (6.4)$$

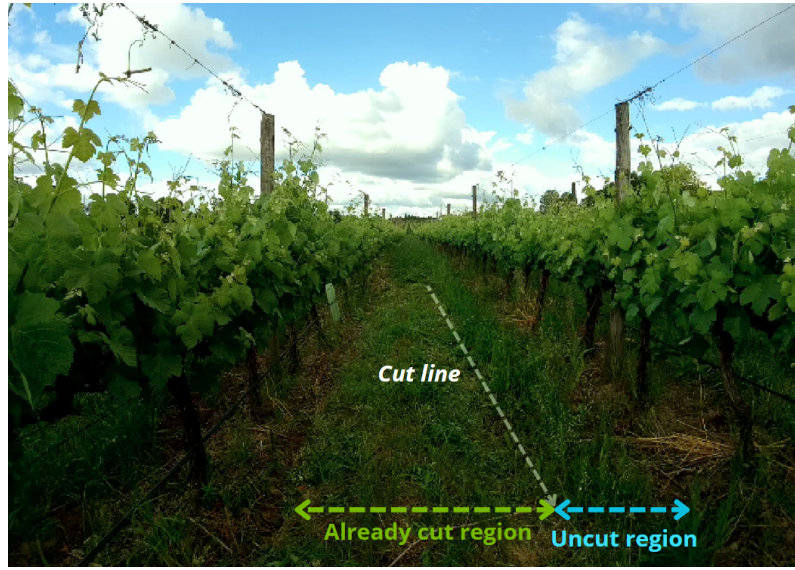
with standard deviation $\sigma_z = \sqrt{\sigma_z^2}$. The clamp at zero guards against the small negative values that finite-precision cancellation can produce in the difference $S_2/n - \bar{z}^2$ when the heights in a cell are nearly equal.

Computational cost. Each point affects only its own cell, and locating that cell is an amortised constant-time hash operation, so accumulating a cloud of P points costs $O(P)$ time. Because the raw points are discarded and only the five accumulators per cell are kept, the memory footprint is $O(C)$ in the number of occupied cells C , independent of how many points fall in each cell. This

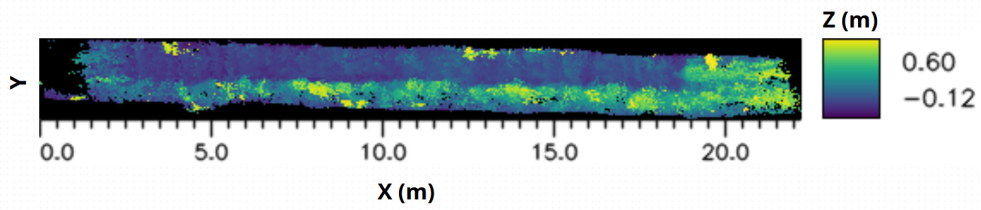
bound is what makes the representation suitable for a mission-level product: the footprint follows the area the robot has actually surveyed and stays bounded over arbitrarily long missions.

This cost profile mirrors a deliberate split between run time and mission end. Accumulation is performed on-line, every corridor cloud being inserted into the map as it is produced, at a per-sweep cost negligible beside the perception and planning work of the same cycle. The rendered product, by contrast, is not required while the robot is navigating; it is therefore generated only once, when the mapping node shuts down. The map is thus populated on-line but rasterised off-line, which keeps the run-time path free of the image generation while still yielding a complete map at the end of the mission.

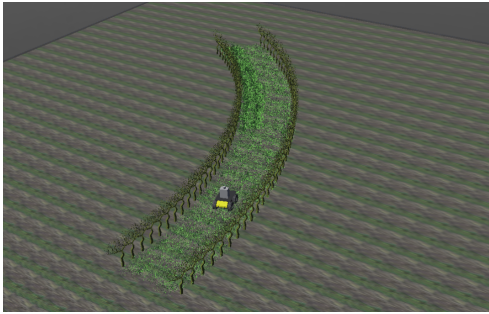
Rendering. At shutdown the occupied cells are rasterised into an image whose pixel grid follows the cell grid. The exported layer is, by default, $z_{\max}$, the per-cell maximum height, which best represents the top of the vegetation canopy; the minimum, the mean, the standard deviation and the point count are available as alternative layers. The occupied values are linearly normalised to the range $[1, 255]$ and rendered through the perceptually uniform viridis colour map, while unobserved cells are held at zero and drawn black, so that the absence of data is never mistaken for low vegetation. A height layer may be shifted by a constant offset before rendering; the node sets this offset to the negative of the base height in the map frame, so that the map stores absolute z yet the exported scale reads as height above the ground. The image is annotated with a colour bar in metres and with a horizontal axis giving the forward distance travelled from the start, as illustrated in Figure 6.6.



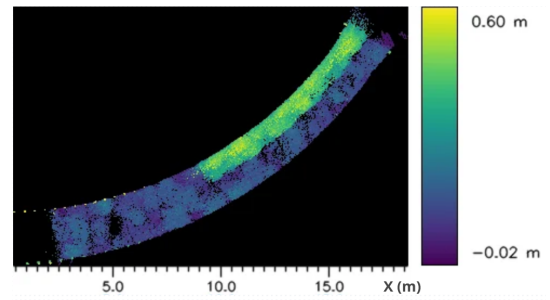
(a.1) Inter-row corridor (real vineyard)



(a.2) Resulting mission-level height map



(b.1) Simulation environment (Webots)



(b.2) Resulting mission-level height map

Figure 6.6: Mission-level sparse height maps exported at the end of a run, shown alongside their respective source environments. (a) Straight inter-row corridor from a real-vineyard rosbag: the field photograph shows the vegetation state at the time of collection, and the height map shows the per-cell maximum vegetation height accumulated during the run. (b) Curved corridor from a Webots simulation: the viewport screenshot shows the simulated environment, and the height map shows the corresponding accumulated result. In both maps, rows are absent because the cloud was restricted to the inter-row corridor before accumulation.

These mission-level maps are assessed qualitatively across the simulated worlds in Section 8.5.

6.4 Robot-Centric Grid Map

The mission-level map of Section 6.3 is a survey product, accumulated over the whole traverse and rendered once at the end. The cut-line detection that guides the return pass has the opposite requirements: it must run continuously, it concerns only the robot’s immediate surroundings, and its result is consumed in the robot’s own frame, where “ahead” and “to the left” have a fixed meaning. The mapping node therefore maintains a second, distinct representation alongside the sparse map: a local, robot-centred height map that follows the platform as it moves. This section describes that representation; the cut-line cascade built upon it is the subject of Section 6.5.

Choice of representation. The local map is built on the `grid_map` library [16], which was adopted for three properties that match this use directly. First, it is computationally efficient to keep centred on a moving robot: the map is held in a circular buffer, so re-centring it on the platform’s new position is performed by shifting buffer indices rather than by copying the stored data, and the maintenance cost per cycle is constant regardless of the distance travelled. Second, it is explicitly robot-centric and multi-layered, designed to hold several co-registered statistical layers over a window that tracks the robot. Third, this combination makes it natural to reason about the structures around the robot, particularly the cut line, in the robot’s own reference frame, rather than in a global frame whose orientation is irrelevant to the lateral control problem. These are the same properties highlighted when the library was introduced in Section 2.3.2.

Geometry and layers. The map is a square window of side $L = 6.0$ m at a resolution of 0.05 m, centred on the robot. It carries three layers, maintained per cell: a point count n_c , a height sum Σ_c , and a mean height $\bar{z}_c$, all initialised to zero. At the start of every cycle the window is re-centred on the robot’s current position (r_x, r_y) in the map frame, which translates the window so that it follows the platform. This re-centring uses the position only and is yaw-agnostic: the window slides with the robot but its axes remain aligned with the map frame; cells that fall outside the new window are cleared, and cells newly brought into view start empty.

Accumulation. The corridor cloud delivered by the intake stage (Section 6.2), already expressed in the map frame, is accumulated into the window. Each point is located in the grid and updates its cell’s layers incrementally,

$$n_c \leftarrow n_c + 1, \quad \Sigma_c \leftarrow \Sigma_c + z, \quad \bar{z}_c = \frac{\Sigma_c}{n_c}, \quad (6.5)$$

so that each occupied cell holds the mean height of the vegetation observed within it, together with the support n_c that will later weight the least-squares fit. As in the sparse map, the points themselves are not retained. Figure 6.7 shows the populated map at a point along a corridor (gray-scale), the cut line appearing as a lateral height step between the standing and the mown vegetation on the left panel.

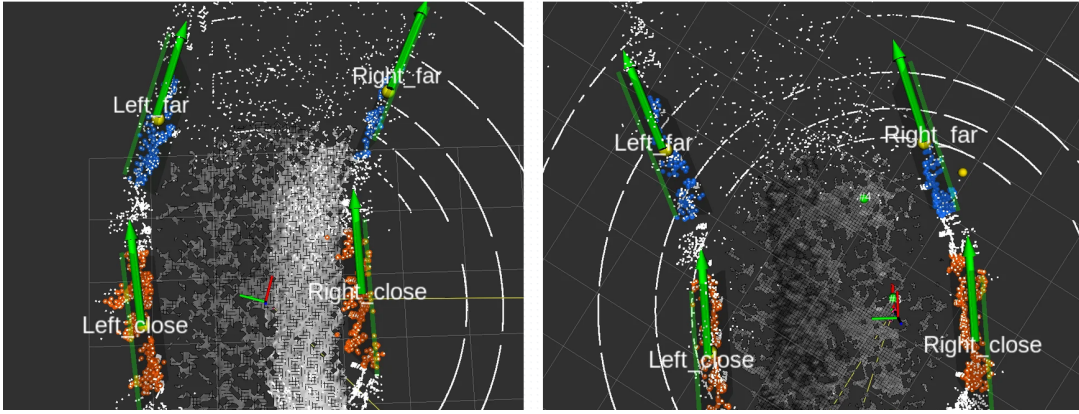


Figure 6.7: Robot-centric grid map at two points along a vineyard corridor. On the left the cut line is clearly visible as a lateral height step between the higher standing vegetation and the lower, already-mown swath; on the right no such step is present, so no cut line can be read from the map.

Projection into the robot frame. Although the window is stored axis-aligned with the map frame, the cut-line detector reasons in the robot frame, with x' pointing forward and y' pointing to the left. Each occupied cell, at map-frame position (x, y) , is projected into that frame by the rigid transform that undoes the robot's pose (r_x, r_y, ψ) ,

$$\begin{aligned} x' &= \cos \psi (x - r_x) + \sin \psi (y - r_y), \\ y' &= -\sin \psi (x - r_x) + \cos \psi (y - r_y), \end{aligned} \tag{6.6}$$

where ψ is the robot heading. Only the cells in the forward half-window, $x' \in [0, L/2]$, are retained, since the cut line is sought ahead of the robot; the projection lets the boundary be expressed as a lateral offset y' at a sequence of forward distances x' , independently of the robot's absolute heading.

The retained forward cells are not processed individually. Those within a forward band ahead of the robot are grouped into a small set of longitudinal *sections*, each of which is reduced to a single point on the cut line, so that the detection pipeline of the next section runs once per section — on a compact one-dimensional summary — rather than over every lateral position in the window. The construction of these sections and the rationale for the decomposition are developed in Section 6.5.1, which opens the cut-line detection cascade.

The grid map is the substrate from which the cascade below extracts the cut line; its results are reported in Section 8.3.

6.5 Cut-Line Detection — the Acceptance-Gate Cascade

The return pass must follow the boundary between the strip already mown on the forward pass and the standing vegetation beside it — the *cut line*. Detecting that boundary on-line, from the robot-centric height map of Section 6.4, is the central contribution of this chapter. The difficulty is that a height step in the lateral profile is necessary but not sufficient evidence of a cut: comparable

steps arise at the corridor edges, at isolated tufts of tall grass, and at gaps in the cover. The detector is therefore built as a *cascade of acceptance gates*, each of which a candidate must pass before it is admitted, answering in turn three questions: does a step exist in a section, is that step physically consistent with a mown/unmown boundary, and do the sections agree on a single coherent line? This section develops the cascade gate by gate, beginning with the sectioned one-dimensional profiles on which it operates and ending with the fitted cut line published to the planner.

Before the cascade is developed, Figure 6.8 illustrates what the detector must read from the data and why the boundary is not always legible. Both panels are viewed from the front, as though the observer stood on the corridor floor and looked longitudinally along it. Because the view looks back along the corridor, the point clouds are laterally mirrored: the returns of the right-hand crop rows appear on the left of the image and those of the left-hand rows on the right. This inversion is immaterial to the argument; what matters are the white points, the returns retained by the CropHull filter of Section 6.2, which characterise the vegetation on the navigable inter-row floor. In panel (a) these points are dispersed and no cut line can be read from them, whereas in panel (b) a cut line is clearly perceptible as the boundary between the raised vegetation on the left and the lower vegetation on the right. Detecting that boundary on-line, when it is present, is the task the remainder of this section addresses.

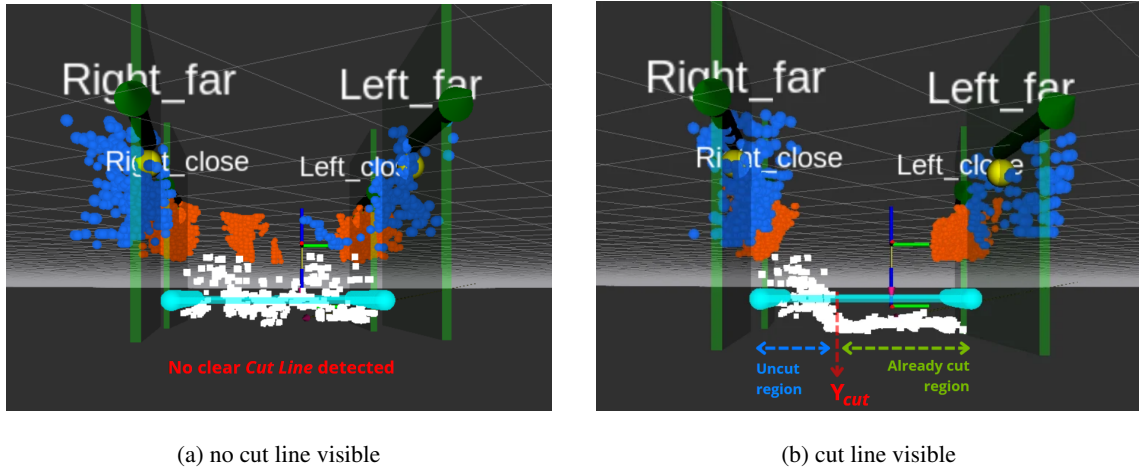


Figure 6.8: The white CropHull-filtered points on the navigable inter-row floor, viewed longitudinally along the corridor from the front. In (a) the (white) points are dispersed and no cut line is discernible; in (b) a cut line is clearly visible as the boundary between the raised vegetation on the left and the lower vegetation on the right. Note that, as the view looks back along the corridor, the point clouds are laterally mirrored, so that the right-hand crop rows appear on the left and the left-hand rows on the right.

6.5.1 Sectioning and 1D Pooled Profiles

The cut line is not estimated from the grid map as a whole. Within the forward half-window $x' \in [0, L/2]$ defined in Section 6.4, the sections are restricted to a longitudinal band $x' \in [x_{\min}, x_{\max}]$ that omits the near and far extremes where the grid holds no usable height data. The near limit $x_{\min} = 1.0$ m skips the blind region directly ahead of the sensor, in which the LiDAR returns

almost no points; the far limit $x_{\max}$ is set to the longitudinal look-ahead distance, beyond which the corridor is not modelled. Confining the sections to this populated band keeps every one of the N sections over data, rather than spreading the same number of sections across the empty near and far cells. The band is partitioned along x' into N longitudinal *sections* of equal depth $w = (x_{\max} - x_{\min})/N$; for a look-ahead of $x_{\max} = 2.5$ m and $N = 10$, each section is $w = 0.15$ m deep. A grid cell is assigned to the section that contains its forward coordinate, $s = \lfloor (x' - x_{\min})/w \rfloor$, and a cell whose forward coordinate falls outside the band takes part in no section. Each section is reduced to a single point on the cut line, whose forward coordinate is fixed at the section's centre,

$$x'_s = x_{\min} + \left(s + \frac{1}{2}\right) w, \quad s = 0, \dots, N-1, \quad (6.7)$$

so the section carries no information about x' ; the unknown is the lateral coordinate y'_s of the mown/unmown boundary within it, which the remainder of the cascade is devoted to estimating. The set of accepted section points $\{(x'_s, y'_s)\}$ is fitted to a smooth curve — the cut line — by the weighted least-squares stage of Section 6.5.7.

Figure 6.9 previews the whole construction, namely the sections and the y bins, and Figure 6.10 displays the per-section points, and the curve fitted through them, before the individual stages are developed in the subsections that follow.

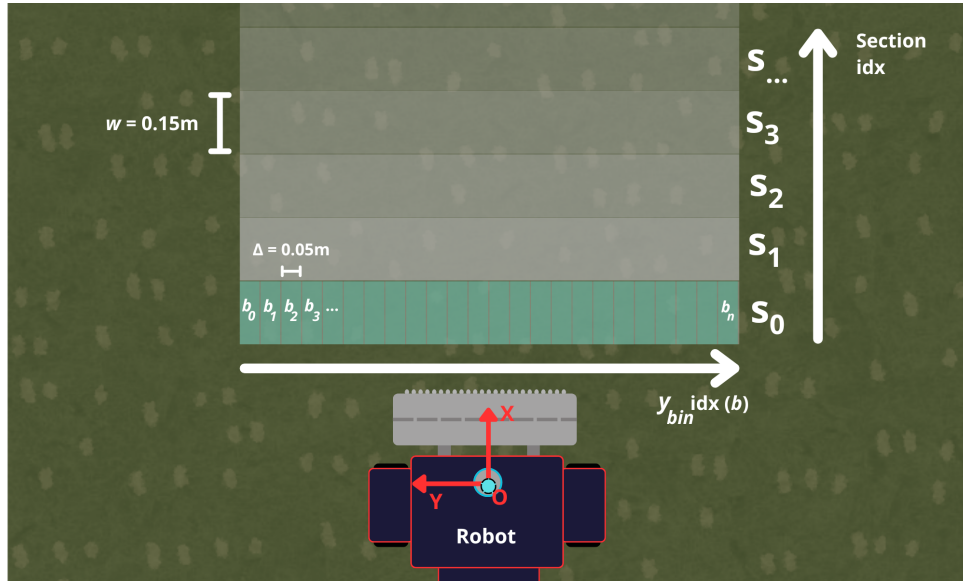


Figure 6.9: Configuration of the sections and the lateral bins within each section. The forward band $[x_{\min}, x_{\max}]$ is divided into N longitudinal sections, each of which pools the retained cells into lateral bins one resolution cell wide. The pooled mean of each bin is the profile value on which the next gate operates.

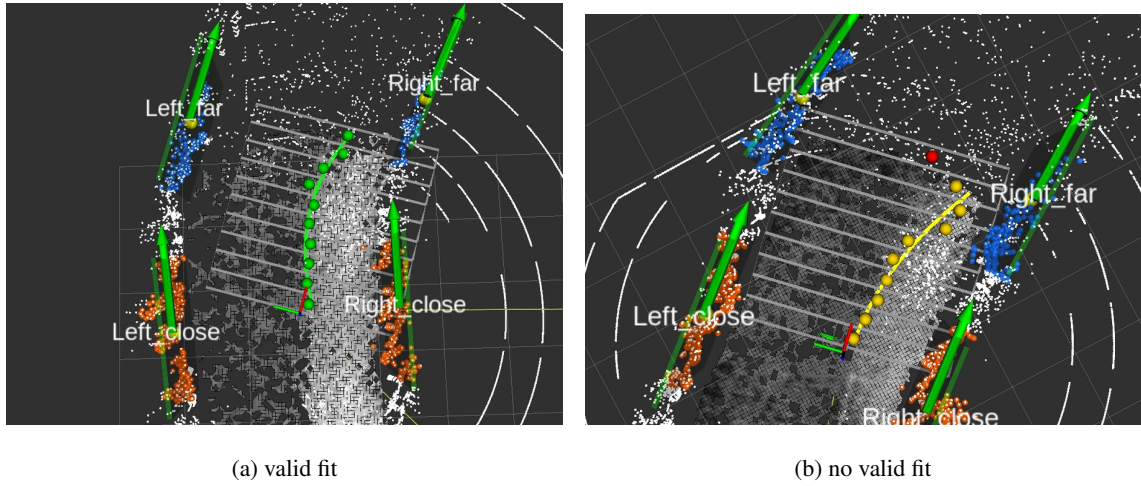


Figure 6.10: Sectioned cut-line detection at a glance: the N sections, the per-section points (x'_s, y'_s) , and the curve fitted through them. (a) a valid case, where the per-section points are mutually consistent and a cut line is fitted; (b) a case where the section points are inconsistent and no valid curve is obtained.

Why a sectioned estimate. Reducing the problem to one point per section, rather than fitting a single line to all corridor cells at once, is a deliberate choice with three motivations. First, the cut line is not in general straight: along a curved corridor it follows the curvature of the rows, so that a single global straight-line fit could not represent it, whereas a sequence of local section points can be fitted by a curve that bends with the corridor. Second, the decomposition makes the estimate robust to partial and contaminated evidence. Each section decides on its own whether it contains a valid boundary, so a section with no detectable step, for instance where the mown stretch has already ended, or one corrupted by spurious returns is rejected individually, without dragging the rest of the estimate with it; a single fit over all cells would instead be pulled by every outlier and could not localise a boundary that occupies only part of the corridor. Third, sectioning turns one two-dimensional detection problem into N independent one-dimensional ones, each operating on a pooled lateral profile that is simpler to reason about, of lower computational cost to evaluate, and able to be filtered over time on its own; the per-section results are recombined only at the final curve fit, where the global shape of the cut line is imposed.

Pooled one-dimensional profile. Within a section, the retained cells are pooled along y' into lateral *bins* one resolution cell wide ($\Delta = 0.05$ m); a cell at lateral coordinate y' falls in bin $b = \lfloor y'/\Delta \rfloor$. Each bin accumulates the sum of the height sums Σ_c and of the point counts n_c of the cells assigned to it, and its profile value is their pooled mean,

$$p(b) = \frac{\sum_{c \in b} \Sigma_c}{\sum_{c \in b} n_c}, \quad (6.8)$$

that is, the mean height of all points that fell anywhere in the bin, weighted naturally by how many points each cell contributed. Pooling several cells per bin raises the support behind each profile

value and so suppresses the per-cell noise that an individual height would carry. The section's total support $C_s = \sum_b \sum_{c \in b} n_c$ is retained as the weight base for the cross-section fit of Section 6.5.7.

A bin that receives no cell is left undefined. To present the matched filter of Section 6.5.2 with a continuous signal, the undefined bins are filled by linear interpolation between their nearest defined neighbours, with constant extrapolation beyond the outermost defined bins; a section with no defined bin at all yields no profile. Finally, an *occupancy gate* requires a section to contain at least a minimum number of defined bins (set to ten) before it is allowed to produce a candidate: sections too sparsely populated to support a reliable profile are discarded at this stage, before any step detection is attempted. The continuous profile $p(y')$ at the fixed forward distance x'_s is the signal on which the next gate operates.

6.5.2 Derivative-of-Gaussian Matched Filtering

The cut line reveals itself, within a section, as a *step* in the lateral profile $p(y')$ built in Section 6.5.1: the standing vegetation on one side of the boundary is taller than the mown swath on the other, so the profile drops sharply as y' crosses the cut. Locating the boundary therefore amounts to locating the steepest transition in p . A bare finite difference would do this only on a clean signal; on the pooled profile, which still carries per-bin noise, differencing amplifies that noise and produces spurious extrema. The remedy is to smooth the profile before differentiating and, because differentiation and convolution commute,

$$\frac{d}{dy'}(p * G_\sigma) = p * G'_\sigma, \quad (6.9)$$

both operations are carried out in a single convolution with the derivative of a Gaussian (DoG). The use of fixed convolution kernels to localise linear structure in this way is well established in the line- and edge-detection literature [26]. Written in full, the normalised Gaussian of scale σ and its first derivative are

$$G_\sigma(u) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{u^2}{2\sigma^2}\right), \quad G'_\sigma(u) = -\frac{u}{\sqrt{2\pi}\sigma^3} \exp\left(-\frac{u^2}{2\sigma^2}\right). \quad (6.10)$$

The DoG is the matched filter for a step edge at scale σ : its response is largest in magnitude where the smoothed profile has its steepest slope, and σ fixes the spatial scale of the steps the filter responds to while suppressing variation finer than it.

Kernel. The continuous derivative is sampled at integer bin offsets to give a discrete kernel of half-width $H = \lfloor 3\sigma \rfloor$ and length $2H + 1$,

$$g[i] = -\frac{i}{\sigma^2} \exp\left(-\frac{i^2}{2\sigma^2}\right), \quad i = -H, \dots, H. \quad (6.11)$$

This is Equation (6.10) with the constant normalising prefactor $1/(\sqrt{2\pi}\sigma)$ dropped; equivalently, it is the derivative of the *unnormalised* Gaussian $\exp(-u^2/2\sigma^2)$. The omission is deliberate and

harmless: that prefactor is a single positive constant multiplying every tap, so it scales every response value uniformly and leaves the argmax — the only quantity the detector uses — unchanged, while sparing one multiplication per tap. The $\pm 3\sigma$ truncation retains essentially all of the derivative's energy. The scale is set in bins, $\sigma = 2.5$ (that is, 0.125 m at the $\Delta = 0.05$ m bin width), giving $H = 7$ and a kernel of 15 taps spanning about 0.7 m.

Response and candidate. The kernel is cross-correlated with the profile over the bins at which it fits entirely inside the support,

$$r(b) = \sum_{i=-H}^H g[i] p(b+i), \quad b = H, \dots, M-1-H, \quad (6.12)$$

where M is the number of profile bins; the H bins at each end are excluded rather than padded, so no artificial edge response is introduced. The value $r(b)$ is the slope of the Gaussian-smoothed profile at bin b : a profile that falls as y' increases yields a response of one sign, a profile that rises the other, and the magnitude grows with the steepness of the transition. The section's candidate boundary is taken at the bin of greatest response magnitude,

$$b^* = \arg \max_{H \leq b \leq M-1-H} |r(b)|, \quad y'_{\text{cand}} = (b_{\min} + b^*) \Delta, \quad (6.13)$$

with $b_{\min}$ the index of the first profile bin and $\Delta = 0.05$ m the bin width; the signed value $r(b^*)$ is retained, as its sign encodes the direction of the step and is examined by the polarity gate of Section 6.5.4. Figure 6.11 shows a section profile, its derivative-of-Gaussian response, and the peak selected as the candidate.

Why further gating is required. The matched filter is a *locator*, not a detector. For any profile that is not perfectly flat, $|r|$ attains a maximum somewhere, so Equation (6.13) always returns a candidate position — even in a section that contains no cut at all, such as one filled with uniform vegetation, or one lying beyond the end of the mown stretch, where the extreme response merely marks the largest residual fluctuation. Identifying the most step-like point in a section is therefore necessary but not sufficient evidence of a boundary. The candidate must still survive two further gates: a magnitude gate, which asks whether the step is large enough to be a genuine height drop (Section 6.5.3), and a geometric-signature gate, which asks whether the step falls in the direction a mown/unmown boundary must (Section 6.5.4). Those gates are the subject of the next two subsections.

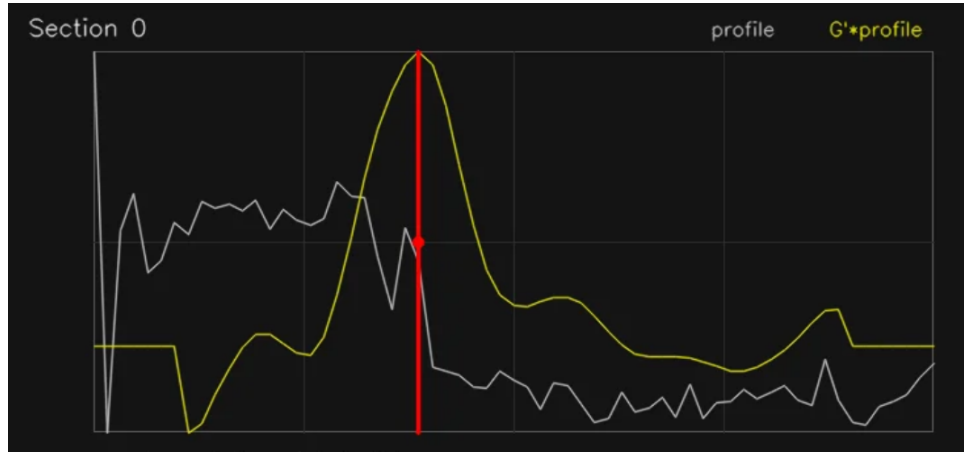


Figure 6.11: Derivative-of-Gaussian matched filtering of a section profile. The response peaks in magnitude at the steepest transition, which the argmax selects as the candidate boundary.

6.5.3 Step-Magnitude Gate

The matched filter of Section 6.5.2 returns a candidate position b^* together with a response magnitude $|r(b^*)|$, but that magnitude is a property of the filtered signal at the chosen scale, not a physical height: it grows with the steepness of the smoothed profile and with the kernel scaling, and cannot be compared against a metric threshold. The first gate of the cascade therefore re-measures the candidate directly on the profile, as the height difference in metres between the vegetation on the two sides of the step, and rejects it when that difference is too small to be a genuine mown/unmown boundary.

Robust plateau heights. A single bin on either side of b^* would be a noisy estimate of the local height, so the step is measured between two *plateaus*, the flatter regions of the profile away from the transition. A guard band of $\text{pad} = \lceil \sigma \rceil$ bins (here $\lceil 2.5 \rceil = 3$ bins, 0.15 m) is skipped on each side of the candidate, so that the transition ramp itself, which spans roughly the kernel scale σ , is excluded. The left plateau is formed from the valid bins at indices $b \leq b^* - \text{pad}$ and the right plateau from those at $b \geq b^* + \text{pad}$, and each plateau is summarised by its median, which is insensitive to the occasional outlier bin,

$$m_L = \text{median}\{p(b) : b \leq b^* - \text{pad}\}, \quad m_R = \text{median}\{p(b) : b \geq b^* + \text{pad}\}. \quad (6.14)$$

The signed step height across the candidate is the difference of the two plateau medians,

$$\Delta h = m_R - m_L. \quad (6.15)$$

Reliability and the magnitude test. The medians are meaningful only if each plateau is supported by enough bins. Each side is therefore required to contribute at least $N_{\min}/3$ valid bins, a third of the section-occupancy floor $N_{\min}$ of Section 6.5.1, here three bins. A candidate whose

plateau falls short, as happens when the transition sits against the edge of the window, is marked unreliable and discarded. A reliable candidate is then admitted only if its step is large enough in magnitude,

$$|\Delta h| \geq h_{\min}, \quad h_{\min} = 0.10 \text{ m}, \quad (6.16)$$

the threshold $h_{\min}$ being set just above the height fluctuation of an undisturbed grass cover, so that flat or gently undulating profiles, in which the argmax of Section 6.5.2 merely marked the largest residual ripple, are rejected here. This gate tests only the *magnitude* of the step; its *sign*, which separates a true mown/unmown boundary from an upward transition into taller vegetation, is left to the geometric-signature gate of Section 6.5.4, for which the bin counts n_L and n_R of the two plateaus are retained.

6.5.4 Geometric-Signature (Boundary-Polarity) Gate

On the forward pass the robot follows one crop row at a fixed lateral standoff and mows the swath beside it, leaving the vegetation in a characteristic arrangement for the return pass: the still-standing, taller vegetation lies closest to the crop rows, and the already-mown, lower vegetation lies toward the interior of the corridor. The cut line is the boundary between the two. A genuine cut is therefore distinguished not by the size of its step, which the previous gate has already tested, but by this arrangement of the two heights, with the taller vegetation on the row side and the lower vegetation toward the centre. A boundary showing the opposite arrangement, the taller vegetation toward the interior, does not correspond to a cut and is rejected whatever its magnitude.

This gate selects it from the geometry of the two-pass mowing strategy, and it is the test that distinguishes a genuine mown/unmown boundary from any other height step. Figure 6.12 contrasts the two arrangements.

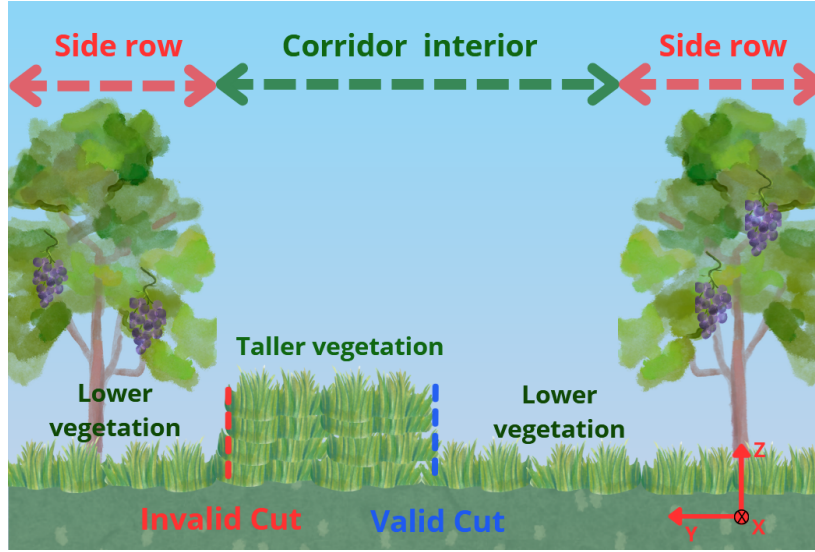


Figure 6.12: Side view of the inter-row corridor, seen as though looking longitudinally along it. The vine rows bound the corridor on the left and right, with the open inter-row floor between them, where a band of taller, still-standing vegetation adjoins the lower, already-mown vegetation. The two polarities a lateral height step may take are contrasted. At the *valid cut* (blue), the taller vegetation lies on the row side and the lower vegetation toward the interior of the corridor, which is the signature of a genuine mown/unmown boundary. At the *invalid cut* (red), the arrangement is reversed, with the taller vegetation toward the interior and the lower vegetation on the row side; this polarity does not correspond to a cut and is rejected whatever its magnitude.

Side of the boundary. Encoding this test requires knowing on which side of the section the boundary lies, which the gate infers from the relative support of the two plateaus of Section 6.5.3. A boundary sits close to one edge of the section, so the plateau on that side is short while the plateau on the opposite side is long. The candidate is accordingly assigned to the side with the smaller plateau: in terms of the bin counts n_L and n_R , the boundary is taken to lie on the left ($+y'$) side when $n_R > n_L$, and on the right ($-y'$) side otherwise.

Polarity condition. With the side fixed, the gate requires the signed step $\Delta h = m_R - m_L$ of Equation (6.15) to have the sign that places the lower vegetation toward the corridor interior. The candidate is accepted only if

$$\begin{cases} \Delta h < 0, & \text{if } n_R > n_L \quad (\text{boundary on the } +y' \text{ side}), \\ \Delta h > 0, & \text{otherwise} \quad (\text{boundary on the } -y' \text{ side}). \end{cases} \quad (6.17)$$

The test is local to the section and independent of which corridor side the robot is following, because the side is read from the data through n_L and n_R rather than assumed; the same rule therefore detects a cut on either side. A candidate that passes both this gate and the magnitude gate yields the section's accepted point (x'_s, y'_s) , with y'_s the lateral coordinate of the candidate bin b^* . Sections whose candidate fails either gate contribute no point on the current sweep, and the

per-section temporal filtering of Section 6.5.5 then decides how that absence is carried forward. The per-section outcome across one sweep is illustrated in Figure 6.13.

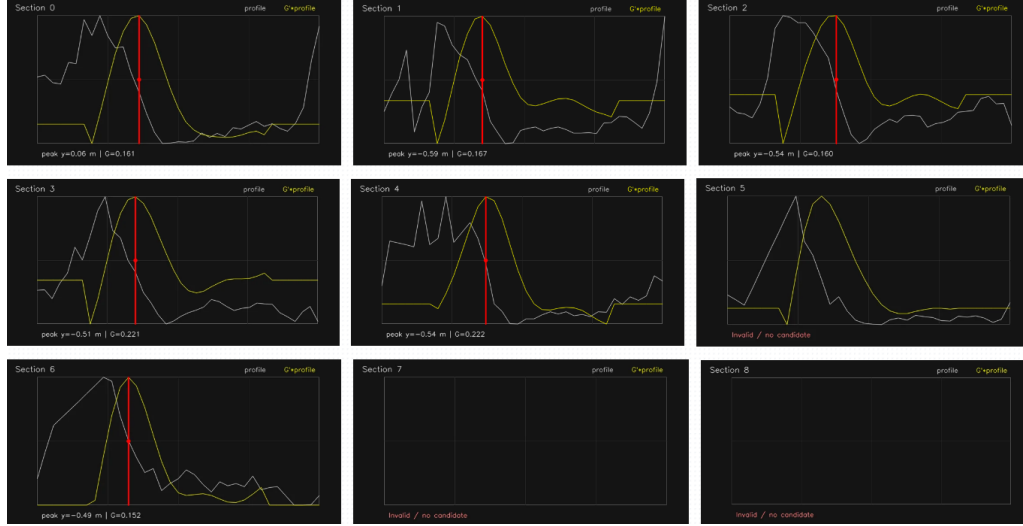


Figure 6.13: Per-section detection outcome for a single sweep. Each section yields at most one accepted point y'_s ; sections without a valid, correctly oriented step contribute none. Nine of the ten sections are shown here, arranged three by three; the tenth and farthest section is omitted, as it lies beyond the populated range and yielded no candidate.

6.5.5 Per-Section Temporal Filtering with Directional Saturation

The gates of Sections 6.5.2 to 6.5.4 produce, on each sweep that passes them, a single candidate lateral offset for the section; write it y'_{cand} . Because the section's forward coordinate x'_s is fixed, this candidate is a scalar, and the sequence of candidates from successive sweeps is a one-dimensional signal that is filtered in time into the section's reported value y'_s .

Two distinct effects make the candidate change from one sweep to the next, and the filter must separate them. The first is measurement noise, which produces isolated, uncorrelated deviations that should be smoothed away. The second is the robot's own motion: the cut line is fixed in the map frame, but y' is measured in the robot frame, so when the robot moves laterally within the corridor the line translates in the robot frame in the direction opposite to that motion, even though it is the same physical line. When the robot is steering toward the cut line, for example, the line approaches in y' on every sweep, and the candidate drifts steadily in one direction. This drift is genuine information about the boundary and must be followed, not suppressed. A filter that simply discarded candidates whose change exceeded a threshold would discard precisely these legitimate motions and lose the line during the very manoeuvre that tracks it. The scheme used here therefore never discards a candidate; it bounds the per-sweep change and then uses the persistence and direction of the bounded changes to separate genuine drift from noise.

Saturation of the per-sweep change. Let $\hat{y}_{k-1}$ be the value accepted on the previous sweep. The per-sweep change is limited to a fixed magnitude Δ_{max} , so the value entering the average is

the candidate when it lies within the limit and the saturated value otherwise,

$$a_k = \begin{cases} y'_{\text{cand}}, & |y'_{\text{cand}} - \hat{y}_{k-1}| \leq \Delta_{\text{max}}, \\ \hat{y}_{k-1} + \text{sign}(y'_{\text{cand}} - \hat{y}_{k-1}) \Delta_{\text{max}}, & \text{otherwise.} \end{cases} \quad (6.18)$$

The limit is set in cells, $\Delta_{\text{max}} = 5\Delta = 0.25$ m at the $\Delta = 0.05$ m bin width. Saturating rather than discarding bounds the influence of any single large deviation: an isolated outlier moves the estimate by at most Δ_{max} instead of being either trusted in full or dropped, and a boundary that is genuinely moving still advances every sweep, at a bounded rate. The decision of whether a sustained change is real is deferred to the direction of successive saturations rather than taken from any single one.

Directional run counting and sustained-drift reset. The two effects are separated by the *direction* of the saturations. Measurement noise saturates the limit only occasionally, and in no consistent direction, so it is absorbed by the cap and by the average that follows. A real lateral motion of the robot relative to the line, by contrast, saturates the limit on consecutive sweeps and always with the same sign, because the line keeps translating the same way in the robot frame for as long as the motion continues. The filter exploits this: each saturation records the sign of its change, consecutive saturations in the same direction are counted, and the count is cleared by any sweep that falls within the limit or by a saturation of the opposite sign. Only once the run reaches $N_{\text{sat}} = 5$ consecutive same-direction saturations is the change accepted as a genuine, sustained drift, that is, as evidence that the robot is moving in the map frame towards or away from the line rather than that the measurement is noisy. The temporal state is then reset and the estimate snaps to the current candidate y'_{cand} , discarding the lagging history so that the section re-acquires the moved boundary at once instead of converging to it slowly.

Moving average and its initialisation. On the first valid detection, and immediately after a reset, the history is pre-filled with $M - 1$ copies of the accepted value, so that the moving average is well defined and unbiased from the first sweep instead of ramping up as samples accumulate. The reported value is the mean of the last M accepted values,

$$y'_s = \frac{1}{M} \sum_{j=0}^{M-1} a_{k-j}, \quad M = 5, \quad (6.19)$$

which is the per-section output carried forward to the cross-section consensus and fit. A representative time history of one section, with the raw candidate, the saturated value, the smoothed output and a sustained-drift reset, is shown in Figure 6.14.

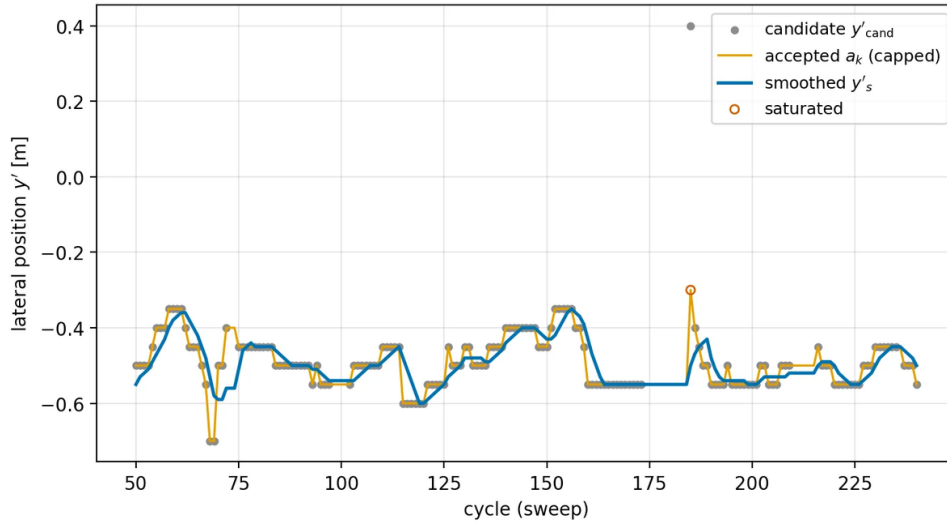


Figure 6.14: Temporal filtering for section 4. Isolated jumps are capped at $\Delta_{\max}$ and absorbed by the moving average, whereas a sustained one-directional run of saturations triggers a reset and the estimate snaps to the drifting boundary.

6.5.6 Cross-Section Sign Consensus

The gates of Sections 6.5.3 and 6.5.4 act on each section independently, and the temporal filter of Section 6.5.5 smooths each section in isolation. A section can therefore pass all of them on the strength of a local feature that is not part of the cut line. The cut line, however, is a single coherent boundary that runs the length of the forward window, so every section that genuinely lies on it must exhibit the same polarity: the taller vegetation on the same side along its whole extent. Cross-section consensus imposes this global constraint, which no per-section test can.

Besides its point (x'_s, y'_s) , each accepted section reports the sign of its height step,

$$q_s = \text{sign}(\Delta h_s) \in \{-1, +1\}, \quad (6.20)$$

taken from the step of Equation (6.15), which encodes the side on which the taller vegetation lies. Let N_v be the number of valid sections on the sweep, and n_+ and n_- the numbers of them with $q_s = +1$ and $q_s = -1$, so that $n_+ + n_- = N_v$. The majority sign and the agreement it commands are

$$q^* = \begin{cases} +1, & n_+ > n_-, \\ -1, & \text{otherwise,} \end{cases} \quad r = \frac{\max(n_+, n_-)}{N_v}. \quad (6.21)$$

Detection gate. The cut line is accepted as present on the sweep only if the agreement reaches a fixed fraction,

$$r \geq \rho_c, \quad \rho_c = 0.75, \quad (6.22)$$

and only if enough sections were valid to support a fit in the first place, $N_v \geq \lceil N/2 \rceil$, which is five for $N = 10$. When the sections disagree, so that $r < \rho_c$, no single boundary is consistently present

across the window, and the cut line is reported absent for the sweep; the same holds when too few sections are valid.

Rejection of minority sections. When consensus holds, the sections whose sign differs from q^* are discarded, and only the majority-sign points are passed to the fit of Section 6.5.7. This removes the occasional section that located an oppositely oriented step, for instance at a gap in the cover or at an isolated clump of tall grass, so that the fit is computed only from points that agree on the geometry of the boundary. Each retained point keeps the weight $w_s = \sqrt{C_s}$ introduced in Section 6.5.1, with C_s the section's total point support, so that better-populated sections carry more influence in the fit.

Recognition of the end of the mown stretch. The consensus gate also provides the detector's stopping condition. While the robot travels along a mown stretch the cut line is present and the sections agree; once the stretch ends and no boundary remains, the surviving section candidates arise only from residual structure and no longer share a sign, so the agreement r falls below ρ_c . Equivalently, the boundary may simply cease to be detected by enough sections, so that the count of valid sections N_v drops below the minimum $\lceil N/2 \rceil$ required for the fit. The end of the mown region is therefore detected either as a loss of consensus or as a loss of sufficient valid sections, in both cases without any dedicated terminal test. Figure 6.13 shows a sweep's sections with their individual signs and the majority they form.

6.5.7 Cross-Section Weighted Least-Squares Fit and Metric Validity Gate

The sections that survive the consensus of Section 6.5.6 provide a set of points $\{(x'_s, y'_s)\}$ along the cut line, each at a fixed forward distance x'_s and each carrying the weight $w_s = \sqrt{C_s}$ of Section 6.5.1. A single curve is fitted through them to form the continuous cut line the planner follows. Because the cut line runs alongside the rows and may bend with a curved corridor, it is modelled, in the robot frame, by a low-order polynomial in the forward coordinate,

$$y'(x') = ax'^2 + bx' + c, \quad (6.23)$$

a quadratic that captures the gentle curvature over the short forward window while remaining low enough in order not to overfit the few section points. When only two points remain the quadratic is underdetermined and the fit falls back to the straight line $y'(x') = ax' + b$.

Weighted least-squares fit. The coefficients are obtained by minimising the weighted sum of squared lateral residuals,

$$\min_{a,b,c} \sum_s w_s (y'_s - y'(x'_s))^2, \quad w_s = \sqrt{C_s}, \quad (6.24)$$

so that better-populated sections influence the fit more, while the square root tempers that influence so that a single very dense section does not dominate the rest. The weighted system is formed by scaling each row of the design matrix by $\sqrt{w_s}$ and is solved by a column-pivoted Householder QR decomposition, which is numerically stable for the small and possibly ill-conditioned polynomial system. The fit is declined when fewer than two points remain, when the section points span too small a range in x' for a curve in x' to be defined, or when the solved coefficients are not finite.

Metric validity gate. Passing the consensus gate establishes that the sections agree on a side; it does not establish that they lie on one smooth curve. The quality of the fit is therefore quantified by the weighted root-mean-square residual,

$$\varepsilon = \sqrt{\frac{\sum_s w_s (y'_s - y'(x'_s))^2}{\sum_s w_s}}, \quad (6.25)$$

expressed in metres and read directly as the typical lateral deviation of the section points from the fitted curve. The cut line is accepted as valid only if this deviation is small,

$$\varepsilon \leq \varepsilon_{\max}, \quad \varepsilon_{\max} = 0.10 \text{ m}. \quad (6.26)$$

A larger residual indicates that the points, though consistent in sign, do not fall on a single smooth boundary, and the line is then withheld from the planner even though a curve was computed.

Sampling and output. The accepted curve is sampled at 30 points evenly spaced over the observed range $[x'_{\min}, x'_{\max}]$, and each sample is transformed from the robot frame into the map frame. These samples are used only for visualisation: they are published as a display of the detected boundary and are convenient for monitoring its evolution over a run, but they are not the representation the planner consumes. The cut line that the planner actually receives, condensed to a single point and a direction, is published as described in Section 6.5.8. Figure 6.10 shows the fitted curve through the per-section points.

6.5.8 Condensation and Publication

The fitted curve and the surviving sections are condensed into a compact representation of the cut line, which is published on every sweep. The `CutLine` message carries five quantities: an anchor point, a direction, the observable span of the line, a validity flag, and the side of the corridor on which the standing vegetation lies.

Curve fit and the published line. The boundary is fitted as a curve in Section 6.5.7, rather than as a straight line, because the corridor, and the cut line along it, may be curved. On a curved corridor a straight-line fit leaves a large residual, so a single value of the validity threshold $\varepsilon_{\max}$ could not serve both straight and curved corridors and would have to be retuned for each. Fitting a quadratic removes that dependence: the model represents the curvature directly, the same threshold

applies in both cases, and the added cost is negligible because the fit is over at most N points, ten here. What is published, however, is not the curve but a single point and a direction, because the planner needs only these to generate its Bézier reference trajectory and does not use the curvature of the line itself (Chapter 7). These, together with the span that reports how far the line is supported by valid sections, are defined next.

Condensed line. The anchor $\mathbf{a}$ is the smoothed point of the nearest section, the one closest to the robot and therefore the least noisy, expressed in the map frame. The direction is the unit chord of the fitted curve, from its nearest sample $\mathbf{q}_N$ to its farthest sample $\mathbf{q}_F$ in the map frame,

$$\hat{\mathbf{u}} = \frac{\mathbf{q}_F - \mathbf{q}_N}{\|\mathbf{q}_F - \mathbf{q}_N\|}, \quad (6.27)$$

which gives the heading of the boundary independently of the robot's orientation. The observable span is the distance between the nearest and farthest *valid* section points, so that when the farther sections are invalid the span contracts and the line is reported as supported only over the nearer part of the window. The condensed line $(\mathbf{a}, \hat{\mathbf{u}})$ is refreshed only on sweeps whose fit passes the metric gate of Section 6.5.7, the weighted-RMSE check; on any other sweep the previous condensed line is retained.

Validity flag. The flag records whether the line is strong enough for the robot to follow. It is set true only when the cut line was detected on the sweep and the fit passed the metric gate of Section 6.5.7: that is, when the sections reached consensus and the weighted residual satisfied $\varepsilon \leq \varepsilon_{\max}$. Any other outcome leaves the flag false.

Side of the standing vegetation. The message also reports the side of the corridor on which the standing, unmown vegetation lies, left or right of the direction of travel. This is the side the return pass must mow toward. It is read directly from the boundary polarity established per section in Section 6.5.4 and agreed across the surviving sections by the sign consensus of Section 6.5.6, so no additional computation is required to obtain it. Reporting the side explicitly spares the planner from inferring it from the relative position of the two crop rows. That inference is ill-conditioned when the cut line runs near the corridor centre, where the rows are almost equidistant: a small error then selects the wrong row and inverts the lateral-offset direction, mirroring the planned endpoint about the line. Supplying the side from the detector, which reads it from the height arrangement rather than from the corridor geometry, removes this failure mode, and the planner consumes the side directly to orient its cutter offset (Chapter 7).

Unconditional publication. A `CutLine` is published on every sweep, whether or not a valid line was found. When the line is valid, the freshly condensed line is sent; when it is not detected or not valid, the last known line is republished with the flag set false. Publishing unconditionally keeps the planner supplied with fresh state on every cycle. The decision of when to switch between

row following and cut-line following is taken by the planner's own temporal hysteresis (Chapter 7), which must receive an up-to-date message each cycle rather than be starved of input whenever detection lapses. The mapping node therefore reports its evidence on every sweep and leaves the switching decision to the planner.

Visualisation outputs. For monitoring, the node also publishes a colour-coded marker of the fitted curve, green when the line is detected and valid, yellow when only one of the two holds, and red when neither, in which case the last known line is shown. It additionally publishes a per-section debug image of the profiles and their derivative-of-Gaussian responses, of the kind shown in Figure 6.13. Neither output is consumed by the planner; both support inspection of the detector during a run.

The detection/abstention rate and the geometric accuracy of this cascade, taken as a whole, are reported in Sections 8.3.2 and 8.3.3.

6.6 Parameter Summary and Calibration

The behaviour of the mapping node is governed by a small set of parameters introduced across Sections 6.2 to 6.5.8. They are collected in Table 6.1 with the values used, so that the pipeline can be reproduced from a single place; the empirical evaluation of the behaviour they produce is the subject of Chapter 8.

Table 6.1: Parameters of the mapping pipeline and the cut-line detection cascade, with the values used. The basis for each is given in the text and in the section where it is introduced.

Symbol	Meaning	Value
T	Height-gate half-width about the base plane	0.60 m
Δ	Map cell size (sparse and grid maps)	0.05 m
L	Robot-centric grid-map window side	6.0 m
N	Number of longitudinal sections	10
$x_{\min}$	Near limit of the section band (sensor blind zone)	1.0 m
$x_{\max}$	Far limit of the section band (longitudinal look-ahead)	2.5 m
$N_{\min}$	Minimum populated bins per section	10
σ	Derivative-of-Gaussian scale	2.5 bins (0.125 m)
$h_{\min}$	Minimum step magnitude	0.08 m
$\Delta_{\max}$	Per-sweep saturation cap on y'	0.25 m (5 cells)
N_{sat}	Same-direction saturations before reset	5
M	FIR window length	5
ρ_c	Cross-section consensus ratio	0.75
$\epsilon_{\max}$	Maximum weighted RMSE of the accepted fit	0.10 m

The parameters divide into two groups. The structural and geometric values, Δ , L , N , $x_{\min}$, $x_{\max}$, σ and M , together with the quadratic order of the fit, follow from the sensor resolution, the look-ahead window and the smoothing scales, and are fixed by design rather than tuned. The decision thresholds T , $N_{\min}$, $h_{\min}$, $\Delta_{\max}$, N_{sat} , ρ_c and $\epsilon_{\max}$ were calibrated on recorded vineyard sequences, each against the failure mode it guards: T sits above the tallest grass to be mapped yet below the vine canopy; $h_{\min}$ above the height fluctuation of an undisturbed cover; $\Delta_{\max}$ at the largest plausible per-sweep motion of the boundary in the robot frame; N_{sat} and ρ_c separate sustained, spatially coherent evidence from noise; and $\epsilon_{\max}$ is set at the lateral deviation beyond which the section points no longer describe a single smooth line.

This chapter has developed the mapping node in full: the synchronised intake and corridor extraction that reduce each sweep to the inter-row vegetation (Section 6.2); the mission-level sparse height map (Section 6.3) and the robot-centric grid map (Section 6.4) built from the corridor cloud; and the acceptance-gate cascade that condenses the per-section detections into a single cut line and publishes it to the planner (Section 6.5). With perception (Chapter 5) and mapping now defined, the local planner that consumes their outputs to generate the robot's motion reference is the subject of Chapter 7.

Chapter 7

Local Planning: FSM, Safety Hysteresis and Continuous Trajectories

This chapter develops the local-planning node, the third node of the system and the one that converts the products of perception and mapping into a reference trajectory for the platform. It consumes the row models that perception publishes on every sweep and the cut line that mapping detects on the return pass, and from them constructs a single continuous reference trajectory, which is handed to the external tracking controller. Around that trajectory it places a supervisor that keeps the mission advancing when detection on one side is momentarily lost and that engages cut-line following once the return pass begins. The chapter opens with the node's state and its detection intake (Section 7.1), then presents the supervisory finite-state machine that orchestrates the node (Section 7.2) before developing, one per section, the mechanisms it invokes: the hysteretic safety scores and the on-line inter-row width (Section 7.3), the side-dependent trajectory endpoint (Section 7.4), the cubic Bézier curves and the rule that chains them into one continuous reference (Section 7.5), the switching of the reference side (Section 7.6), and cut-line following (Section 7.7); it closes with the interface to the tracking controller (Section 7.8).

7.1 Node State and Detection Intake

The local-planning node is the decision-making node of the system. Unlike perception and mapping, it does not subscribe to the LiDAR sensor and holds no point-cloud objects; it owns instead the planning state, namely the finite-state machine, the reference-side decision, and the construction and publication of the reference trajectory. Sensing is thus confined to the other two nodes, and the planner consumes only their published results. This keeps the geometric reconstruction in a single place and lets the planner run at its own control rate, independent of the sweep rate; the node roles and the topic connections are those of the architecture in Chapter 4.

The node subscribes to the per-sweep `RowDetectionResult` message published by perception, which carries, for both sides, the reconstructed `CLOSE` and `FAR` line geometry together with the per-side safety score. From each message the node rebuilds a compact, node-side description

of the two sides: for each side and each region of interest it stores the line centroid, the unit direction, the centroid height, the canopy dispersion, and a validity flag. This reconstruction is a direct field-by-field copy of the message, with no recomputation and no access to the perception data structures, so the planner keeps only a lightweight mirror of the geometry it needs in order to build a trajectory. A live estimate of the inter-row width is refreshed from the same messages at intake; its definition and use are deferred to Section 7.3.

The per-side safety score is consumed exactly as perception reports it. The score is an absolute, bounded quantity that perception owns and accumulates over successive sweeps, saturating at its limits and resetting from its floor (Chapter 5). The planner does not accumulate, filter, or otherwise transform it: it reads the value carried by each message and uses it directly as the input to the supervisory logic. Holding the accumulation entirely on the perception side ensures that the score has a single definition and that the detector and the planner never maintain inconsistent histories of it. How the supervisor interprets the score, through a pair of hysteretic predicates, is developed in Section 7.3.

Detection arrives asynchronously, one message per sweep, whereas the planning logic runs on a fixed-period control cycle, so the two proceed at independent rates. To decouple them, the subscription callback writes each incoming message into a shared buffer under a mutex and marks a detection as available; when the control cycle fires, it acquires the same mutex only long enough to copy that buffer into a private local variable, releases it, and then performs the state evaluation and the trajectory construction on the local copy. The lock is therefore held only for the duration of a copy, so the planning computation never blocks the arrival of new detections, and every control cycle acts on the most recent complete message rather than on one being partially overwritten. Until the first detection has been received the node remains in its initial idle state and publishes no trajectory. The same buffered, mutex-protected pattern is used for the cut-line message, whose role is taken up in Section 7.7.

7.2 The Supervisory FSM

The node's decisions, when to compute a new curve, when to switch the reference side, when to follow the cut line, and when to stop, are arbitrated by a supervisory finite-state machine that runs once per control cycle. The machine is presented here as the top-level controller of the node; each state and the component it invokes is then developed in the sections that follow: reading the safety state (Section 7.3), computing a curve (Section 7.5), switching the reference side (Section 7.6), and following the cut line (Section 7.7). This section gives the five states, the guarded transitions between them, and the behaviour at the edge of the operating envelope; the states and transitions appear in the state diagram of Figure 7.1 and are listed in Table 7.1.

States. The machine has five states. *Init* is the idle entry state, held until detection has stabilised. *ComputeNewCurve* builds and appends one curve (Section 7.5) and publishes the updated trajectory. *Following* tracks the published trajectory and monitors how much of the current curve

remains. *ChangeReferenceSide* performs the reference switch of Section 7.6. *STOP* is a safe idle that clears the trajectory so the platform halts. In normal operation the machine alternates between *Following* and *ComputeNewCurve*, appending a fresh curve each time the current one is nearly consumed.

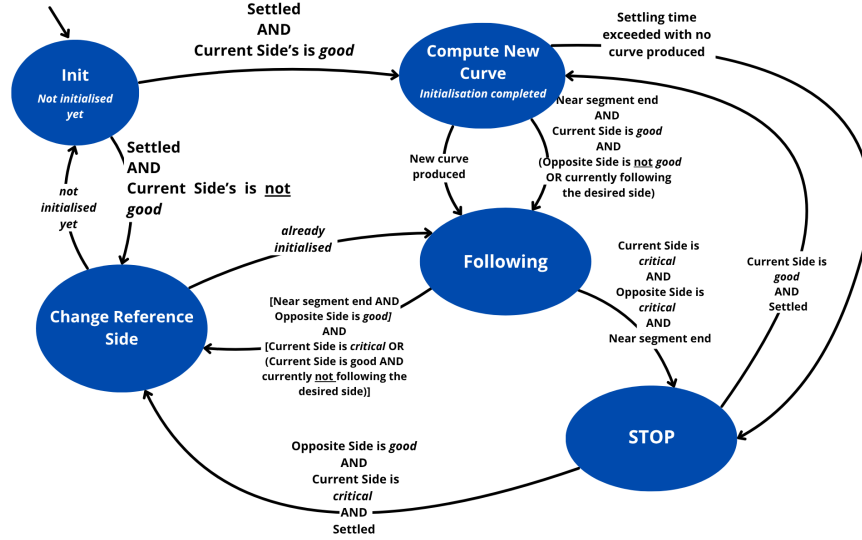


Figure 7.1: State diagram of the supervisory finite-state machine. Edges are the guarded transitions of Table 7.1; in normal operation the machine cycles between *Following* and *ComputeNewCurve*, switching reference side or stopping only at curve boundaries.

Transitions and timing. Every transition is guarded (Table 7.1), and two kinds of guard recur. The first is the pair of hysteretic predicates of Section 7.3, evaluated on the current and on the opposite side, which decide whether to keep following, to switch side, or to stop. The second is a segment-boundary condition: the transitions that leave *Following* fire only as the platform nears the end of the current curve, so a new curve, a side switch, or a stop is acted on at a curve boundary and never in the middle of a segment. This is what allows the chaining rule of Section 7.5 to keep the trajectory G^1 -continuous across these events. The transitions out of *Init* and *STOP* additionally require the machine to be *settled*, meaning that it has remained in the state for longer than a fixed dwell of three seconds. This dwell rejects transient detection noise that would otherwise bounce the machine out of a state and straight back into it: such a state is left only once it has been held for at least three seconds, so a momentary fluctuation that clears within that window triggers no transition. The reference switch is taken in both directions: away from the desired side when it degrades while the opposite side is reliable, and back to the desired side once it recovers, the automatic return of Section 7.6.

STOP as a safe idle. The supervisor relies on at least one of the two rows being reliably detected: the corridor is bilateral, and a trajectory is built from whichever row is trustworthy. When both safety scores are critical at the same time this assumption no longer holds, and neither row offers a reference the planner can trust. Rather than extrapolate from an unreliable detection, the

Table 7.1: Transition table of the supervisory finite-state machine. *Good* and *critical* are the hysteretic predicates of Section 7.3, read on the current or the opposite side; *near segment end* marks the approach to the end of the current trajectory curve; *settled* denotes that the machine has remained in the state for longer than the three-second dwell.

From	Guard	To
Init	settled; current side good	ComputeNewCurve
Init	settled; current side not good	ChangeReferenceSide
ChangeReferenceSide	before initialisation	Init
ChangeReferenceSide	after initialisation	Following
Following	near segment end; current good; on the desired side or opposite not good	ComputeNewCurve
Following	near segment end; opposite good; current critical, or on the non-desired side	ChangeReferenceSide
Following	near segment end; both sides critical	STOP
ComputeNewCurve	new curve produced	Following
ComputeNewCurve	no curve produced within three seconds	STOP
STOP	settled; current side good	ComputeNewCurve
STOP	settled; opposite good; current critical	ChangeReferenceSide

machine enters *STOP* and publishes an empty trajectory, so the platform comes to rest. *STOP* is not terminal: as soon as one side recovers to a reliable state the machine leaves it, resuming curve computation if the current side is good or switching to the recovered side otherwise. The stop is therefore a safe idle that holds the platform still while detection is lost on both sides and releases it automatically when detection returns.

The resulting state occupancy and STOP behaviour are reported in Section 8.4.1.

7.3 Hysteretic Safety Scores and On-line Inter-Row Width

The supervisor reads each side through two quantities that the planner does not itself detect but interprets: the per-side safety score and the inter-row width. This section defines how the absolute safety score is turned into a pair of hysteretic predicates, and how the inter-row width is estimated on-line and anchored to a nominal spacing.

Hysteretic safety predicates. The per-side safety score S_s is the accumulated, saturated confidence that perception maintains for side s (Chapter 5). It is bounded to $S_s \in [S_{\min}, S_{\max}]$, with $S_{\max} = 0$ the safe ceiling and $S_{\min} = -20$ the floor, and it rises by one step on a consistent detection and falls by one step on a failure. The supervisor does not test the raw score against a single level. It derives instead two predicates, a side being *good* only when its score has returned to the ceiling and *critical* only when it has fallen below a lower threshold T_c ,

$$\text{good}(s) \iff S_s = S_{\max}, \quad \text{critical}(s) \iff S_s < T_c, \quad (7.1)$$

with $T_c = -5$. The two thresholds do not coincide, so between them lies a band, $T_c \leq S_s < S_{\max}$, in which a side is neither good nor critical: its detection has degraded but is still tolerated.

This separation is a hysteresis. A side that is currently good becomes critical only after its score has fallen more than five steps below the ceiling, which at one step per sweep requires a sustained loss of detection rather than a single poor sweep; and once critical, it is treated as good again only when the score has recovered all the way back to the ceiling. The band therefore imposes a dwell of several sweeps between the moment a side stops being trusted and the moment it is regarded as recovered, so that the supervisor does not switch reference side or stop on a transient, single-sweep dropout. The supervisory machine of Section 7.2 reads these two predicates, on the current and on the opposite side, to decide when to keep following, to switch side, or to stop.

On-line inter-row width. The supervisor also maintains a live estimate of the inter-row width W , the lateral separation of the two rows (the same quantity illustrated in Figure 5.14). It is computed from the two CLOSE lines as the perpendicular distance between them, by projecting the left CLOSE centroid onto the right CLOSE line and taking the length of the residual,

$$W = \|(\mathbf{c}_L - \mathbf{c}_R) - [(\mathbf{c}_L - \mathbf{c}_R) \cdot \hat{\mathbf{d}}_R] \hat{\mathbf{d}}_R\|, \quad (7.2)$$

where $\mathbf{c}_L$ and $\mathbf{c}_R$ are the left and right CLOSE centroids and $\hat{\mathbf{d}}_R$ is the unit direction of the right CLOSE line. Subtracting the component along $\hat{\mathbf{d}}_R$ removes any longitudinal offset between the two centroids, so W is the true lateral spacing and is independent of how far apart the centroids happen to lie along the rows.

A new value of W is adopted only when both sides are good and both CLOSE lines are valid, that is, only when the geometry on both sides is fully reliable; otherwise the previous estimate is kept. Before any reliable measurement exists, W is initialised from a nominal inter-row spacing, a known planting property of the vineyard supplied as a parameter. That nominal value therefore acts as an absolute lateral reference and as the fallback whenever the measurement conditions are not met. The width gives the line geometry an absolute lateral scale. During the forward pass the planner holds the platform at a fixed desired distance d_{row} from the followed row, a configured offset of the order of one metre. The width is also what lets the planner keep producing trajectories when one side's safety score becomes critical: the lateral reference can then be taken from the surviving row instead of the lost one, without moving the platform. How this reference switch is performed, and why it leaves the platform's position unchanged, is developed in Section 7.6.

The robustness this hysteresis provides under a one-sided detection loss is evaluated in Section 8.4.2.

7.4 Side-Dependent Normals and the Trajectory Endpoint

Each control cycle the trajectory is built between two anchors: a start at the platform and an endpoint placed ahead, on the side being followed. This section constructs that endpoint. It is anchored on the FAR line of the target side, offset into the corridor by the desired distance, and given a heading that blends the direction of the row ahead with the platform's current heading; on

the very first trajectory the construction is softened so that the platform eases onto the reference instead of correcting abruptly.

Side-dependent inward normal. The offset direction is the inward normal of the target side, the unit vector perpendicular to a line and pointing toward the centre of the corridor. For a unit line direction $\hat{\mathbf{d}} = (d_x, d_y)$ it is

$$\hat{\mathbf{n}}_s = \begin{cases} (d_y, -d_x), & s = \text{Left}, \\ (-d_y, d_x), & s = \text{Right}, \end{cases} \quad (7.3)$$

which in either case points from the row toward the inter-row centre. This is the same side-dependent convention used to place the regions of interest in perception (Chapter 5); it is restated here because the endpoint offset is built on it.

Endpoint on the FAR line. Let $\mathbf{c}_F$ and $\hat{\mathbf{d}}_F$ be the centroid and direction of the target side's FAR line, and $\hat{\mathbf{n}}_F$ its inward normal from Equation (7.3). The endpoint is anchored at the FAR centroid and pushed into the corridor by the desired distance,

$$\mathbf{P}_3 = \mathbf{c}_F + d_{\text{row}} \hat{\mathbf{n}}_F, \quad (7.4)$$

so that $\mathbf{P}_3$ lies at the lateral distance d_{row} from the row ahead, on the side the platform travels. Anchoring the endpoint on the FAR line rather than on the nearer CLOSE line places it at the look-ahead set by the FAR region, a fixed separation ahead of the platform (Chapter 5), which is where the trajectory must aim. Figure 7.2 shows the construction.

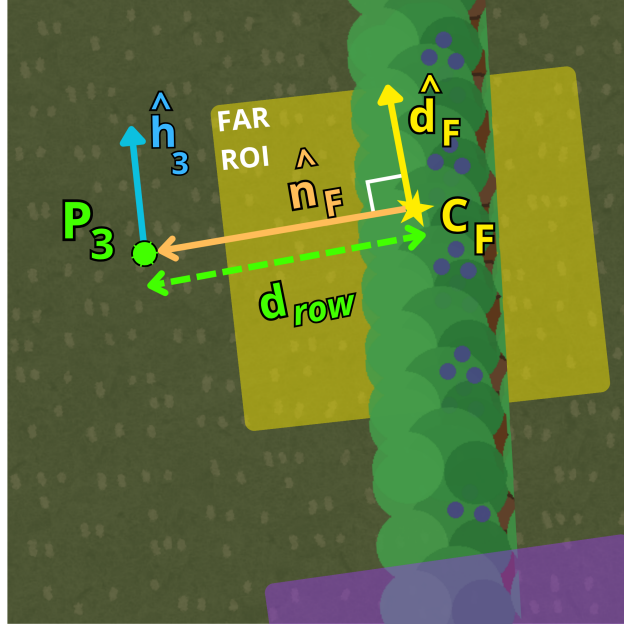


Figure 7.2: Construction of the trajectory endpoint. The endpoint $\mathbf{P}_3$ is anchored on the FAR line, offset into the corridor by the desired distance d_{row} along the inward normal $\hat{\mathbf{n}}_F$, and assigned a heading that averages the FAR direction and the platform’s current heading.

Endpoint heading. The endpoint also carries a heading, which sets the tangent the curve must meet there. Rather than adopt the FAR direction outright, the heading is the normalised average of the FAR line direction and the platform’s current heading $\mathbf{h} = (\cos \psi, \sin \psi)$,

$$\hat{\mathbf{h}}_3 = \frac{\frac{1}{2}\hat{\mathbf{d}}_F + \frac{1}{2}\mathbf{h}}{\left\| \frac{1}{2}\hat{\mathbf{d}}_F + \frac{1}{2}\mathbf{h} \right\|}, \quad \psi_3 = \text{atan2}(\hat{h}_{3,y}, \hat{h}_{3,x}), \quad (7.5)$$

with equal weight on the two terms. Averaging them, rather than taking the row direction alone, keeps the commanded end heading from jumping when the FAR estimate is momentarily noisy, while still turning the platform toward the row ahead.

First-trajectory softening. When the planner produces its first trajectory the platform may start from an arbitrary lateral position, far from the desired offset, and applying the full correction at once would demand a sharp initial turn. Let δ be the platform’s current lateral distance to the followed row, measured as the length of the residual of $\mathbf{P}_0$ projected orthogonally onto the CLOSE line. On the first trajectory only, the lateral target is set halfway between this current distance and the desired offset,

$$d_{\text{row}}^{(1)} = \frac{1}{2}(d_{\text{row}} + \delta), \quad (7.6)$$

so that the first curve takes a shorter lateral step, correcting only half of the standing error, while the platform eases onto the reference over the following cycles rather than in a single one. From the second trajectory onward the full offset d_{row} is used. The start anchor, the endpoint $\mathbf{P}_3$ and its heading ψ_3 are the inputs to the cubic Bézier curve constructed in Section 7.5.

The resulting row-following trajectory error is reported in Section 8.4.3.

7.5 Cubic Bézier Curves and One Continuous Trajectory

Between the start anchor and the endpoint the planner fits a single cubic Bézier curve, and chains successive curves into one continuous reference path. A cubic Bézier is used because it interpolates the two anchors while letting their headings be imposed exactly as the curve's end tangents, which is what lets consecutive segments join smoothly; over the short look-ahead of one segment a single cubic is expressive enough to represent the required bend, and it reuses the curve structure the planner already relies on.

Control-point placement. A cubic Bézier is defined by four points: the endpoints $\mathbf{P}_0$ and $\mathbf{P}_3$, fixed in the previous section, and two interior control points $\mathbf{P}_1$ and $\mathbf{P}_2$ that set the end tangents. The interior points are placed along the start and end headings, ψ_0 and ψ_3 , at a fixed fraction ρ of the chord length,

$$\begin{aligned}\mathbf{P}_1 &= \mathbf{P}_0 + \rho \|\mathbf{P}_3 - \mathbf{P}_0\| (\cos \psi_0, \sin \psi_0), \\ \mathbf{P}_2 &= \mathbf{P}_3 - \rho \|\mathbf{P}_3 - \mathbf{P}_0\| (\cos \psi_3, \sin \psi_3),\end{aligned}\tag{7.7}$$

with $\rho = 0.3$. Placing $\mathbf{P}_1$ ahead of $\mathbf{P}_0$ along ψ_0 and $\mathbf{P}_2$ behind $\mathbf{P}_3$ along ψ_3 makes the curve leave $\mathbf{P}_0$ in the direction ψ_0 and arrive at $\mathbf{P}_3$ in the direction ψ_3 , the defining tangent property of a cubic Bézier; ρ sets how far the curve reaches along those headings before bending toward the other anchor. Figure 7.3 shows the construction.

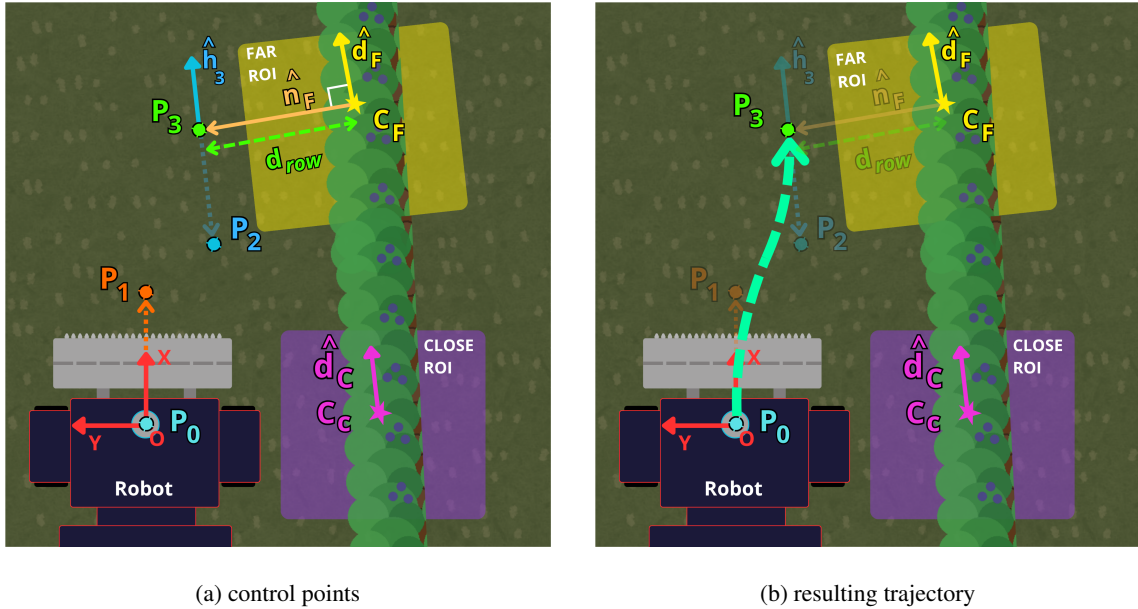


Figure 7.3: Construction of one cubic Bézier segment. (a) The interior control points $\mathbf{P}_1, \mathbf{P}_2$ are offset from the endpoints along the start and end headings by $\rho \|\mathbf{P}_3 - \mathbf{P}_0\|$; (b) the resulting curve leaves $\mathbf{P}_0$ along ψ_0 and reaches $\mathbf{P}_3$ along ψ_3 .

Power-basis evaluation. Rather than evaluate the curve through repeated linear interpolation, it is converted once to its power (monomial) basis,

$$\mathbf{B}(u) = \mathbf{c}_0 + \mathbf{c}_1 u + \mathbf{c}_2 u^2 + \mathbf{c}_3 u^3, \quad u \in [0, 1], \quad (7.8)$$

whose coefficients follow from the control points,

$$\begin{aligned} \mathbf{c}_0 &= \mathbf{P}_0, & \mathbf{c}_1 &= 3(\mathbf{P}_1 - \mathbf{P}_0), \\ \mathbf{c}_2 &= 3(\mathbf{P}_0 - 2\mathbf{P}_1 + \mathbf{P}_2), & \mathbf{c}_3 &= (\mathbf{P}_3 - \mathbf{P}_0) + 3(\mathbf{P}_1 - \mathbf{P}_2). \end{aligned} \quad (7.9)$$

In this form a point follows from a single cubic polynomial in each coordinate, and the derivative is available in closed form as $\mathbf{B}'(u) = \mathbf{c}_1 + 2\mathbf{c}_2 u + 3\mathbf{c}_3 u^2$, so every sample costs a fixed, small number of arithmetic operations.

Arc-length discretisation. The controller consumes the curve as a sequence of points, and these are placed at an approximately constant spacing δ_s in arc length rather than at a constant step in the parameter u , whose rate of advance varies along the curve. At a given u the parameter is advanced by

$$\Delta u = \frac{\delta_s}{\|\mathbf{B}'(u)\|}, \quad (7.10)$$

because $\|\mathbf{B}'(u)\| \Delta u$ is the first-order arc length swept by that step, so each step advances about δ_s along the curve. With $\delta_s = 0.1$ m the curve is sampled into points roughly 0.1 m apart, independent of where the parameter happens to move quickly or slowly, which gives the controller a spatially uniform reference; the summed segment lengths give the curve's total length.

One continuous trajectory. A trajectory is a sequence of such curves held in a bounded history, the oldest discarded as new ones are appended. The first curve starts at the robot pose; every subsequent curve starts at the previous curve's endpoint and inherits its heading. Writing $\mathbf{P}_0^k, \psi_0^k$ for the start of curve k and $\mathbf{P}_3^k, \psi_3^k$ for its endpoint, the chaining invariant is

$$\mathbf{P}_0^k = \mathbf{P}_3^{k-1}, \quad \psi_0^k = \psi_3^{k-1}, \quad (7.11)$$

so each curve begins exactly where the previous one ends, with the same heading. Because the start point and start heading of curve k thus coincide with the endpoint and end heading of curve $k - 1$, and because the control-point placement makes each curve tangent to its start heading, consecutive curves meet with the same position and the same tangent direction: the concatenation is continuous in position and tangent, that is G^1 -continuous. The trajectory is a single object, not indexed by side or by mode. When the reference side is switched (Section 7.6) or the planner moves between row-following and cut-line following (Section 7.7), only the rule that computes the next endpoint changes; the new curve still begins at the previous endpoint with its heading, so the G^1 continuity holds across the transition. Continuity is therefore a structural consequence

of the chaining rule rather than something re-established at each special case, and the platform is handed one continuous reference path throughout (Section 7.8).

The position and tangent continuity achieved at curve joins is quantified in Section 8.4.1.

7.6 Reference-Side Switching with Complement Offset

The platform always travels down one corridor, but the planner is free to take its lateral reference from either row. This section describes how it chooses which row to reference, how it changes that choice when detection on the followed row degrades, and how it returns to the original choice once that row recovers. The change of reference moves neither the platform nor the trajectory: it is only a change of which measurement the offset is taken from.

Desired and current reference side. Two notions of side are kept apart. The *desired* side is the row the mission is configured to follow; it is fixed for the whole run and is never switched. The *current* side is the row the planner references at the moment, and a flag records whether the two coincide. At start-up the current side is set to the desired side. Only the current side is ever changed, and only by the supervisor.

Switching the reference, not the path. When detection on the current side degrades while the other side remains reliable, the supervisor switches the current side to the other row. The switch itself is a single reassignment of the current side; no part of the trajectory is cleared or recomputed. By the chaining invariant of Section 7.5 the next curve simply begins at the previous endpoint, so the reference path stays G^1 -continuous across the switch. The platform is not displaced either, because the lateral offset is referred to the new row through a complementary distance. While the desired side is followed the offset is the desired distance d_{row} ; while the other side is followed it is the complement

$$d_{\text{opp}} = W - d_{\text{row}}, \quad (7.12)$$

clamped to a small positive minimum, with W the inter-row width of Section 7.3. Referring the offset to the surviving row at this complementary distance places the target at the same physical position in the corridor, as Figure 7.4 shows. What changes at a switch is therefore only the measurement geometry, the row and offset from which the endpoint is built, and not the platform's path through the corridor. The per-side safety scores are left untouched by a switch, as they belong to the physical rows and not to the reference choice.

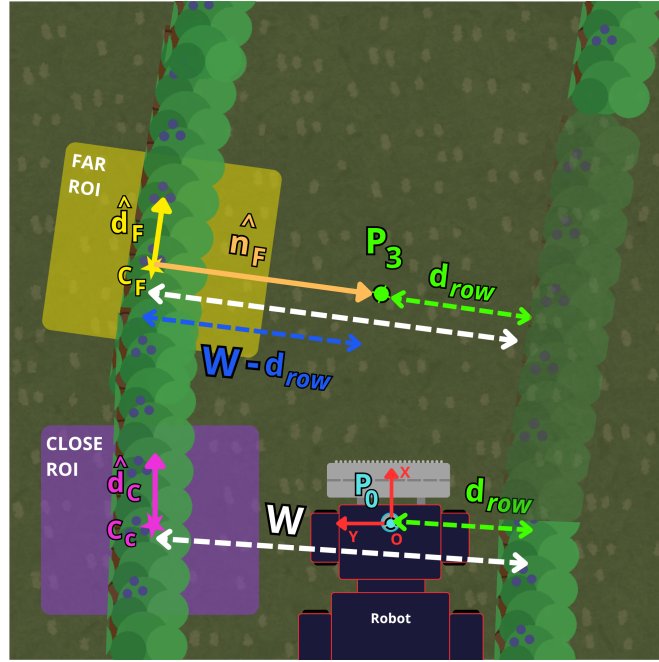


Figure 7.4: Holding the lateral position through a one-sided loss of detection. While both rows are valid the platform is kept at the desired distance d_{row} from the followed (right) row; when that row is lost, the reference moves to the surviving (left) row at the complementary distance $W - d_{row}$, where W is the last trusted inter-row width, so the measurement geometry switches without moving the platform.

Automatic return. The switch is not permanent. While the planner is referencing the non-desired side and the desired side recovers to a reliable state, the supervisor switches the current side back to the desired one, by the same single reassignment and with the offset returning to d_{row} . Switches in either direction are taken near the end of the current trajectory segment, where a new curve is appended in any case, so they fall on a curve boundary and never interrupt a segment. The desired side is thus kept as the home reference, and the other side serves only as a temporary stand-in while detection on the home row is degraded. These switches, with the conditions that trigger them, are part of the supervisory transitions presented in Section 7.2.

This switching mechanism is the one exercised in the one-sided-loss robustness results of Section 8.4.2.

7.7 Cut-Line Following

On the return pass the platform mows the strip left beside the swath cut on the forward pass, and it references the cut line, the boundary between the mown and the standing vegetation, rather than a row. This section covers when cut-line following is engaged, how the endpoint is built from the cut line, and why the transition into and out of this mode keeps the trajectory continuous.

Confidence hysteresis. The mapping node publishes a cut line on every sweep with a validity flag (Chapter 6). The planner accumulates a confidence score over these messages, raising it by one step on a valid cut line and lowering it by one on an invalid one, bounded to $[-20, 0]$. Cut-line following is engaged only when the score reaches the ceiling 0 and disengaged only when it falls below a threshold of -5 ; between the two the current mode is held. This is the same hysteresis applied to the safety scores in Section 7.3: it requires a sustained run of valid detections to enter cut-line following and a sustained run of failures to leave it, so the planner does not flicker between row-following and cut-line following on an intermittent detection. A further guard ignores a cut line whose detected span is below a minimum, even at full confidence, since too short a segment cannot fix a reliable direction.

Endpoint from the cut line. When cut-line following is engaged the endpoint is built from the cut line in place of the `FAR` row. Let $\mathbf{a}$ and $\hat{\mathbf{d}}_\ell$ be the cut line's anchor and unit direction, the latter oriented to point along the direction of travel. The platform $\mathbf{P}_0$ is projected orthogonally onto the line, giving the foot $\mathbf{f}$; the endpoint is taken a look-ahead ℓ further along the line and offset laterally by half the cutter width toward the followed vine,

$$\mathbf{f} = \mathbf{a} + [(\mathbf{P}_0 - \mathbf{a}) \cdot \hat{\mathbf{d}}_\ell] \hat{\mathbf{d}}_\ell, \quad \mathbf{P}_3 = \mathbf{f} + \ell \hat{\mathbf{d}}_\ell + \frac{1}{2} w_c \hat{\mathbf{n}}_v, \quad (7.13)$$

where w_c is the cutter width, $\hat{\mathbf{n}}_v = -\hat{\mathbf{n}}_s$ the unit normal pointing toward the followed vine, and ℓ the detected cut-line span clamped to $[0.5, 2.0]$ m. The half-cutter offset places the path so that the edge of the cutter reaches up to the line, so the pass mows the strip between the line and the followed row. Figure 7.5 shows the construction.

Side of the offset. The normal $\hat{\mathbf{n}}_v$ points toward the standing vegetation, so the sense of the half-cutter offset depends on which corridor side that vegetation occupies. The planner does not infer this side from which crop row lies closer to the line; it reads it from the `CutLine` message, in which the mapping node reports it from the boundary polarity (Chapter 6). The geometric inference is unreliable when the cut line runs near the corridor centre, where the two rows are almost equidistant: a small error there selects the wrong side, inverts $\hat{\mathbf{n}}_v$, and places the endpoint symmetrically on the wrong side of the line. Taking the side from the detector, which reads it from the height arrangement, removes this ambiguity; the geometric nearest-row test is retained only as a fallback for the rare message that carries no side.

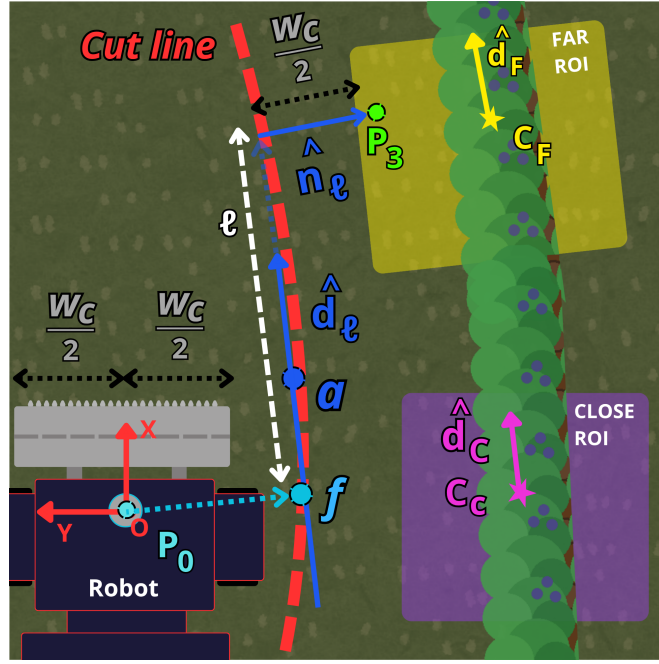


Figure 7.5: Endpoint construction for cut-line following. The platform is projected onto the cut line, advanced by the span-based look-ahead ℓ , and offset by half the cutter width toward the followed vine so the cutter reaches the line; a collision guard keeps the endpoint at least the safety distance d_s from the nearest vine row.

Collision guard. The offset endpoint must not drive the cutter into the vine. A guard keeps the endpoint at least a safety distance d_s from the vine row on the standing-vegetation side reported by the mapping node, which need not be the currently referenced side and whose FAR line and canopy dispersion are used. The safety distance grows with that row's canopy dispersion,

$$d_s = \min\left(\frac{1}{2}w_c + \kappa \sigma_F, d_{\text{row}}\right), \quad \kappa = 1.5, \quad (7.14)$$

with σ_F the FAR canopy dispersion of that row and the desired distance d_{row} as an upper bound. Writing ρ_v for the signed lateral distance from the endpoint to that vine's FAR line, measured toward the corridor centre, the endpoint is pushed inward whenever it lies closer than d_s ,

$$\mathbf{P}_3 \leftarrow \mathbf{P}_3 + \max(0, d_s - \rho_v) \hat{\mathbf{n}}_F, \quad (7.15)$$

where $\hat{\mathbf{n}}_F$ is the inward normal of that row. Figure 7.6 contrasts the two cases. The endpoint heading is set by the same equal-weight blend of the cut-line direction and the current heading used for the row endpoint (Section 7.4).

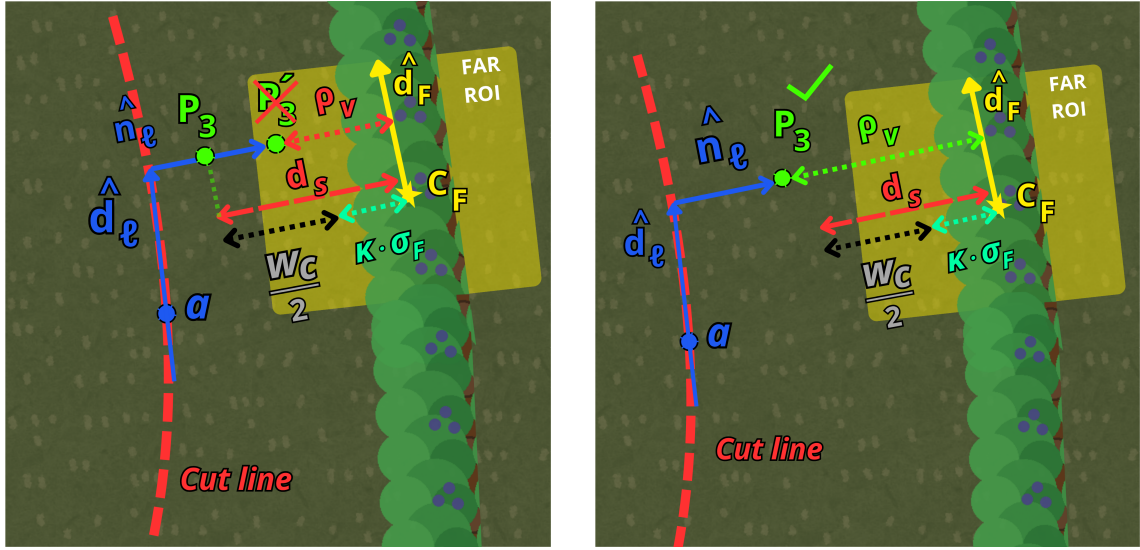
(a) $\rho_v < d_s$: endpoint pushed in.(b) $\rho_v \geq d_s$: endpoint unchanged.

Figure 7.6: The cut-line collision guard. The endpoint is kept at least the safety distance d_s from the FAR line of the nearest vine row: (a) when it is closer than d_s it is displaced toward the corridor interior by the deficit $d_s - \rho_v$; (b) when it already lies beyond d_s it is left in place.

First trajectory onto the cut line. Engaging cut-line following can require the platform to cross the corridor: when the standing vegetation reported by the detector lies on the side opposite the row the platform was following, the first cut-line endpoint sits across the corridor from the current path, and a single curve to it would turn sharply. As for the first row-following trajectory (Section 7.4), the planner softens only this first transition. While it is generated, the look-ahead ℓ is extended and the projection foot is advanced along the line, so the endpoint is placed farther ahead, and the half-cutter offset is scaled down so the path does not yet reach fully to the line. The resulting curve is gentler; from the second cut-line curve onward the nominal look-ahead and the full offset are restored, so the cutter eases onto the line over the following sweeps rather than in a single step. When the cut line already lies on the followed side, no softening is applied and the first curve uses the nominal parameters.

Seamless transition. Cut-line following changes how the endpoint is computed, not the trajectory it feeds. The cut-line endpoint enters the same curve construction and the same chaining rule of Section 7.5, so a curve built in cut-line mode starts at the previous endpoint exactly as a row curve does, and the reference path stays G^1 -continuous when the planner enters or leaves the mode. Because the cut-line direction is oriented to the direction of travel, the same construction serves both the forward pass and the return pass, so the turn from the forward mowing pass to the return pass is taken without breaking the reference, as one continuous trajectory.

The resulting cut-line-following trajectory error is reported in Section 8.4.4.

Table 7.2: Fields of the published `tribe_nav_msgs/Trajectory` message that the planner populates each cycle. Nested rows are the fields of each `BezierCurve` entry.

Field	Type	Content
<code>bezier_curves</code>	<code>BezierCurve[]</code>	ordered list of cubic curves forming the trajectory
<code>x, y</code>	<code>float[4]</code>	coordinates of the four control points $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$
<code>frame_id</code>	<code>string</code>	reference frame of the control points (<code>map_sn1</code>)
<code>velocity</code>	<code>float</code>	target speed along the curve
<code>resolution</code>	<code>float</code>	sampling step requested of the controller

7.8 Interface to the Trajectory Controller

The planner’s product is the reference trajectory, and its last task each cycle is to hand that trajectory to the platform’s tracking controller. This is done through a custom ROS 2 message, `tribe_nav_msgs/Trajectory`, published on a dedicated topic. The message represents the trajectory as a list of cubic Bézier curves; each curve is carried by its four control points ($\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$) together with the reference frame, a target velocity, and a sampling resolution (Table 7.2). Each control cycle the planner fills this message from the curve it has just appended and publishes it; a stop is signalled by publishing an empty trajectory, which clears the controller’s reference and brings the platform to rest.

The tracking controller that consumes this message is an external component, proprietary to the platform’s developers, and is treated here as a black box. The planner’s responsibility ends at producing and publishing the geometric reference; how the controller samples the curves and follows them is outside the scope of this work. The behaviour of the complete perception, mapping and planning system that produces this reference is evaluated in Chapter 8.

Chapter 8

Results and Discussion

This chapter evaluates the perception, mapping and local-planning stack of Chapters 5, 6 and 7 on controlled Webots scenarios with analytic ground truth. The evaluation is organised by module: Section 8.2 assesses the crop-row perception of Chapter 5, Section 8.3 assesses the corridor mapping and cut-line detection of Chapter 6, and Section 8.4 assesses the reference generation of Chapter 7. Because the worlds carry exact row and cut-line geometry, the fitted models, cut-line estimates and generated trajectories are compared directly with their analytic references in the horizontal plane; quantities without an external geometric reference, such as the RANSAC inlier ratio, finite-state-machine occupancy and execution time, are reported as internal-consistency indicators. Alongside this simulated campaign, a rosbag recorded in a real vineyard corridor (Section 8.1.1) exercises the perception and mapping subsystems under genuine sensor noise and field conditions; lacking analytic ground truth and a closed-loop planning log, it is scored only through the same kind of internal-consistency indicators, reported alongside their simulated counterparts wherever both are available. The tracking controller is outside this scope, so the planning results quantify the generated reference trajectory rather than closed-loop platform error.

8.1 Experimental Setup

The campaign uses twenty Webots worlds: five corridor shapes evaluated under four task families. The five shapes are a straight corridor with a 15° heading, three circular corridors of decreasing radius (20 m, 10 m and 5 m) and one S-shaped corridor of changing-sign curvature built from two 8 m arcs. Across the shapes the straight world isolates the baseline problem while the curved worlds place a growing demand on predictive region placement, cut-line fitting and reference generation; the 5 m arc is retained as an extreme stress test rather than a nominal operating point. The four families isolate the operating modes of the two-pass mowing strategy and are summarised in Table 8.1.

Table 8.1: The four task families and, for each, what it verifies and the expected outcome.

Family		Corridor content, what it verifies, and expected outcome
Forward	pass	20 m corridor with uniform low grass and no cut line. Verifies row following in isolation: the row reference should track the configured offset, and the cut-line detector should not return any valid detection.
(FP)		
Backward	pass	
(BP)		
Transition		20 m corridor with a cut line along the whole length. Verifies cut-line detection and following: the detector should report a valid line on the large majority of frames and the reference should settle on the half-cutter offset from the boundary; a few missed frames at the start or under strong curvature are expected.
(TP)		
Gap test		20 m corridor with low grass over the first half and a cut line over the second. Verifies the switch from row following to cut-line following: the detector should abstain in the first half and detect in the second, and a short trajectory transient at the switch is expected.
(GAP)		
		20 m corridor with no cut line in which the right crop row is absent over $s \in [7, 13]$ m. Verifies one-sided-loss robustness: the right-row detection rate is expected to collapse over the gap, which is the induced condition and not a fault, while the planner should keep a valid reference on the left row without stopping.

Table 8.1 fixes the meaning of each family used throughout the chapter. In every cut-bearing world the standing vegetation lies on the left of travel and the mown swath on the right, so the cut line is a lateral height step parallel to the rows. The five shapes are common to the families; an overview of the corridor geometry is shown in Figure 8.1.

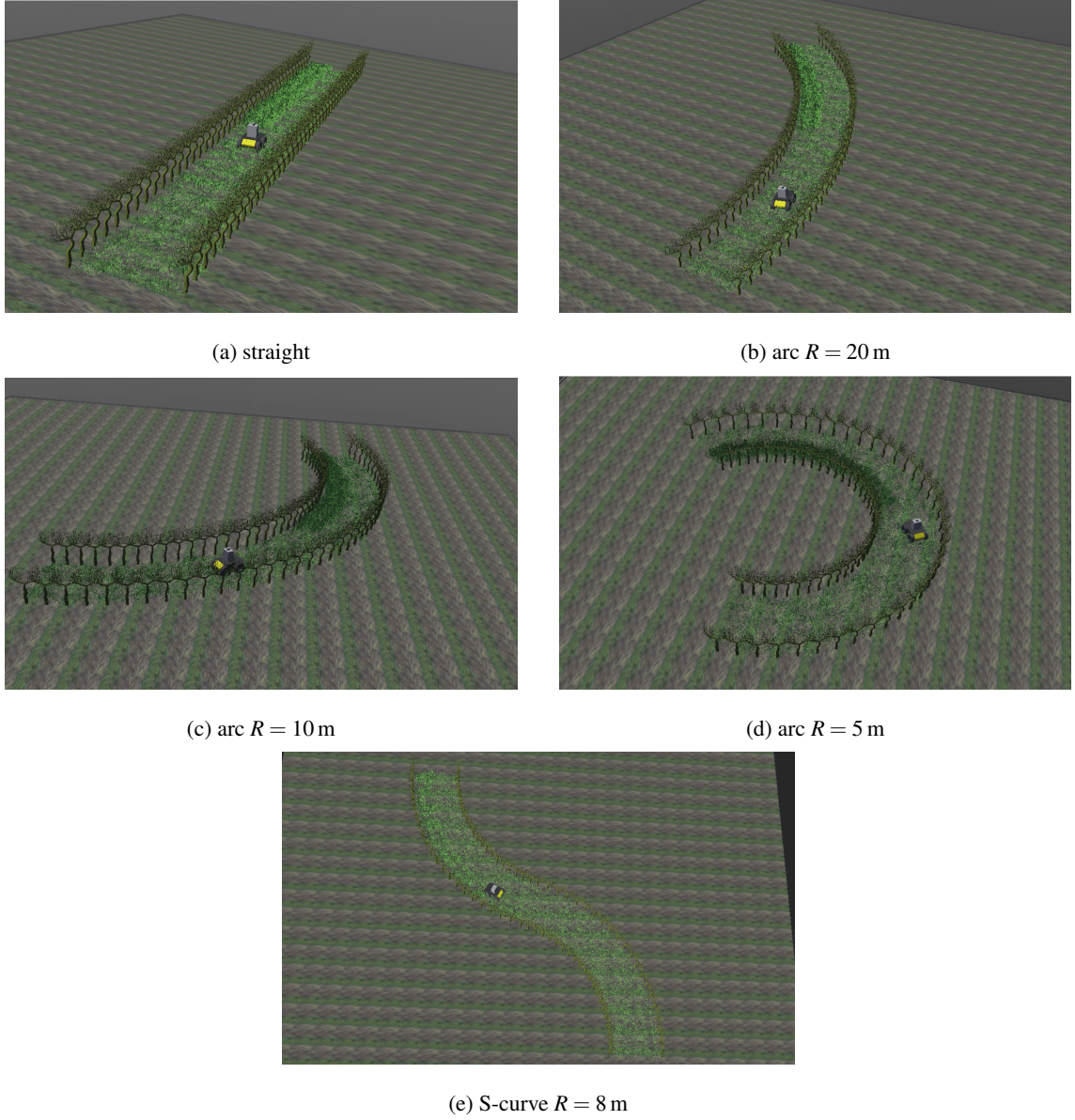


Figure 8.1: The five corridor shapes used by every family: (a) straight, (b)–(d) three arcs of decreasing radius, and (e) the S-curve.

Figure 8.1 situates the shapes that recur in every later table and figure, where each row is labelled by its family and shape.

Every world carries an exact analytic geometry, which is what allows it to serve as ground truth for both the row-perception and the trajectory metrics reported in this chapter. Each corridor centreline is generated by integrating a signed curvature profile $\kappa(s)$ along the arc length s from a fixed start pose; an approximate single-vine mesh is then concatenated along that centreline at a 0.5 m pitch, and the two vine rows are placed as parallel offsets at ± 1.25 m, giving the 2.50 m inter-row spacing. Each run is 20 m long. The shapes differ only in $\kappa(s)$: the straight corridor takes $\kappa(s) = 0$ with a 15° start heading, so its centreline is $y = x \tan 15^\circ$; the three circular corridors take $\kappa(s) = 1/R$ with $R \in \{20, 10, 5\}$ m, tracing arcs of decreasing radius; and the S-curve takes

$\kappa(s) = +1/R$ over the first half of the run and $\kappa(s) = -1/R$ over the second, with $R = 8m$, joining two opposing arcs G^1 -continuously at mid-length and admitting no single closed-form Cartesian description. Because the row meshes follow directly from these expressions, they provide the analytic reference against which perception and planning are scored.

8.1.1 Real-Vineyard Rosbag

Alongside the Webots campaign, a single continuous rosbag recorded with the onboard 3D LiDAR in a real vineyard corridor (Figure 8.2) is replayed offline through the same perception and mapping nodes used in simulation, without modification. In this rosbag, the robot was remote-controlled by a person to follow along the inter-row corridor of the vineyard, without any autonomous navigation capabilities. The run carries no analytic row or cut-line geometry ground truth, and since it was tested *offline*, no closed-loop planning output was generated, so it cannot be compared against an external reference the way the Webots worlds are. It is instead used to check whether the internal-consistency indicators obtained in simulation, namely RANSAC support, inter-row width stability, the point-cloud reduction achieved by the intake stage, the cut-line fit residual, and the per-cycle cost of each stage, hold under genuine sensor noise, lighting and canopy irregularity.

Cut-line detection scored as a classifier against ground truth is also *not* evaluated on this run, for lack of a surveyed boundary, and remains a simulation-only result. The rosbag results are reported alongside their simulated counterparts at the relevant points of Sections 8.2 and 8.3.



(a)



(b)

Figure 8.2: The real-vineyard inter-row corridor where the rosbag used in this chapter was recorded.

8.2 Perception Results

The row detector of Chapter 5 is assessed on the forward-pass (*FP*) worlds, which isolate row following with no cut line; the gap test is brought in only to expose detection loss. Row-line accuracy, support, availability, inter-row width and per-sweep cost are reported in turn.

8.2.1 Row-Line Accuracy and Support

The fitted CLOSE and FAR lines (Chapter 5) are compared with the analytic rows: lateral error is the perpendicular distance to the ground-truth row and heading error the angle to the row tangent. To keep the tables compact, Tables 8.2 and 8.3 report, per shape, the RMSE averaged over the four ROIs (left and right $\times$ CLOSE and FAR); the full per-ROI distributions are kept in the figures.

Table 8.2: Row-line perception lateral RMSE on the forward-pass (FP) worlds, averaged over the four ROIs (left/right $\times$ CLOSE/FAR). N valid sweeps.

Scenario	N	mean RMSE [m]
Straight	380	0.089
Arc R=20 m	354	0.078
Arc R=10 m	390	0.092
Arc R=5 m	328	0.155
S-curve (R=8 m)	392	0.128

Table 8.3: Row-line perception heading RMSE on the forward-pass (FP) worlds, averaged over the four ROIs (left/right $\times$ CLOSE/FAR). N valid sweeps.

Scenario	N	mean RMSE [deg]
Straight	380	4.99
Arc R=20 m	354	4.13
Arc R=10 m	390	4.88
Arc R=5 m	328	6.45
S-curve (R=8 m)	392	5.69

Tables 8.2 and 8.3 show the row-line perception error staying small as curvature increases, the heavier tails falling on the FAR regions. The per-ROI distributions behind the averages are shown in Figures 8.3 and 8.4.

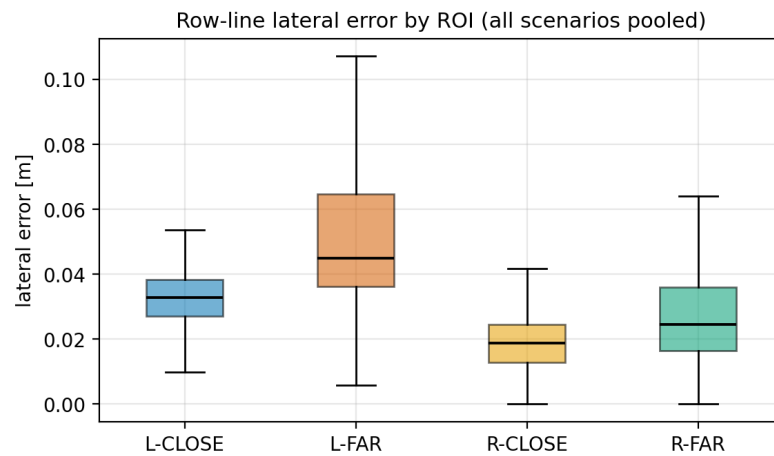


Figure 8.3: Row-line perception lateral error distribution by side and region, *FP* worlds.

Figure 8.3 confirms the lateral error concentrating near zero, the FAR regions carrying the spread that the per-shape average in Table 8.2 smooths out.

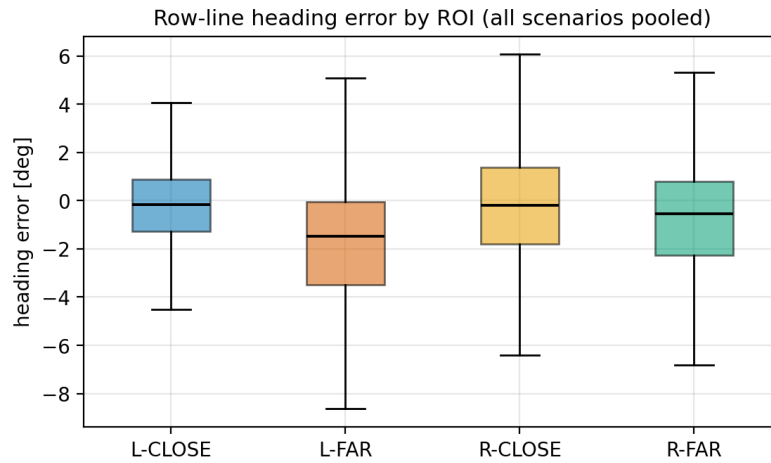


Figure 8.4: Row-line perception heading error distribution by side and region, *FP* worlds.

Figure 8.4 shows the heading error remaining small and unbiased. The fits must also be well supported; Table 8.4 reports the RANSAC inlier ratio, averaged over the ROIs, and Figure 8.5 its distribution.

Table 8.4: RANSAC inlier ratio on the forward-pass (FP) worlds, averaged over the four ROIs. N valid sweeps.

Scenario	N	mean inlier	$\% \geq 0.50$
Straight	380	0.848	97.3
Arc R=20 m	354	0.844	97.2
Arc R=10 m	390	0.817	95.9
Arc R=5 m	328	0.835	97.2
S-curve (R=8 m)	392	0.800	95.3

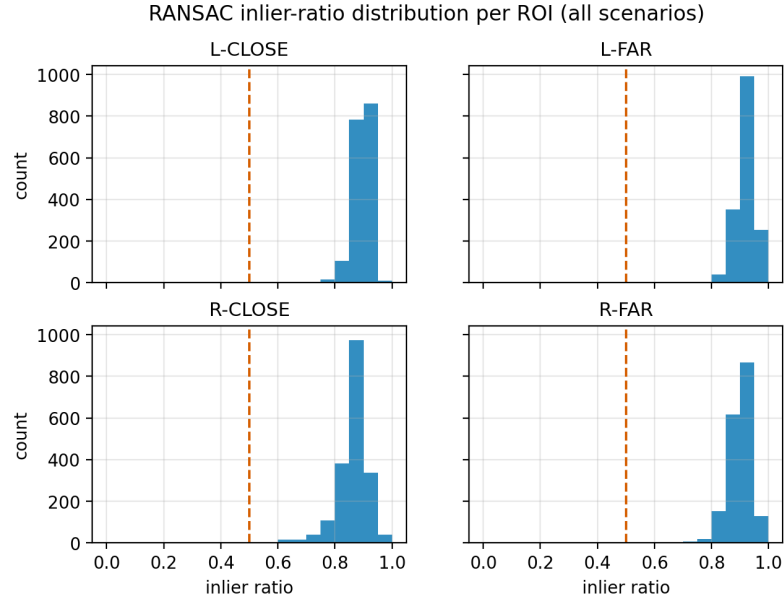


Figure 8.5: RANSAC inlier-ratio distribution by ROI, *FP* worlds; the dashed line marks the acceptance threshold.

Table 8.4 and Figure 8.5 together show that the centimetre-level accuracy is obtained from well-supported fits, so the geometric estimates are trustworthy wherever they are produced.

The same support metric is checked on the real-vineyard rosbag (Section 8.1.1), where no analytic row is available to score lateral or heading error directly but the RANSAC inlier ratio carries over unchanged. Table 8.5 reports the inlier ratio per ROI over the recorded run, and Figure 8.6 its distribution.

Table 8.5: RANSAC inlier ratio per ROI for the real-vineyard rosbag: mean, standard deviation, and the fraction of valid fits with inlier ratio at or above 0.50. N = number of valid sweeps.

ROI	N valid	mean ratio	STD.dev	$\% \geq 0.50$
L-CLOSE	237	0.813	0.060	100.0
L-FAR	237	0.857	0.090	100.0
R-CLOSE	237	0.786	0.055	100.0
R-FAR	237	0.827	0.052	100.0

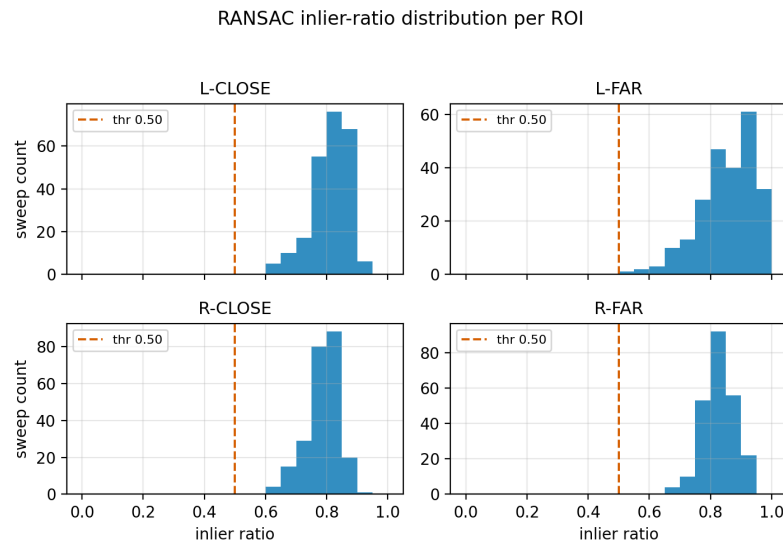


Figure 8.6: RANSAC inlier-ratio distribution by ROI, real-vineyard rosbag; the dashed line marks the acceptance threshold of 0.50.

The mean ratio lies between 0.79 and 0.86 across the four ROIs (Table 8.5), in the same range as the simulated forward-pass figures of Table 8.4, and every valid sweep on the rosbag clears the 0.50 acceptance gate, against 95.3 to 97.3 % in simulation. The fits behind the rosbag estimates therefore remain well supported under real sensor noise and canopy irregularity, not only under the analytic conditions of the Webots worlds.

8.2.2 Availability, Width and Cycle Time

Availability is the quantity the mapping and planning nodes consume when a side becomes unavailable, so it is the natural place to bring in the gap test. Table 8.6 reports the valid-fit rate per side, each side the mean of its CLOSE and FAR regions, on the forward-pass worlds as a baseline and on the gap worlds.

Table 8.6: Valid-fit rate per side on the forward-pass and gap worlds; each side is the mean of its CLOSE and FAR regions. The right side collapses over the induced gap.

Scenario	sweeps	left valid %	right valid %
FP Straight	395	96.2	96.2
FP Arc R=20 m	366	96.2	97.3
FP Arc R=10 m	391	99.5	99.7
FP Arc R=5 m	395	66.8	99.5
FP S-curve (R=8 m)	436	89.9	90.1
GAP Straight	403	94.5	69.7
GAP Arc R=20 m	432	91.0	62.7
GAP Arc R=10 m	521	99.4	70.1
GAP Arc R=5 m	437	87.4	53.3
GAP S-curve (R=8 m)	400	98.0	61.8

Table 8.6 is the perception signature of the induced loss: the right-side rate falls to roughly 55 to 70 % on the *GAP* worlds, where the right row is physically absent over part of the corridor, while the left side and the *FP* baseline stay near 90 to 100 %. The detector reports the loss truthfully rather than fabricating a line, which is the input the planner needs to fall back to the surviving side (Section 8.4.2).

The perceived inter-row width and the per-sweep cycle time are reported on the forward-pass worlds in Tables 8.7 and 8.8; Table 8.8 reports the time taken to process one sweep, not a computational cost in any other sense.

Table 8.7: Inter-row width on the forward-pass worlds against the nominal 2.5 m spacing. N sweeps.

Scenario	N	mean W [m]	STD.dev [m]	mean err vs 2.50 [m]
Straight	395	2.444	0.007	-0.056
Arc R=20 m	366	2.445	0.009	-0.055
Arc R=10 m	391	2.208	0.725	-0.292
Arc R=5 m	395	1.418	1.221	-1.082
S-curve (R=8 m)	436	2.118	0.839	-0.382

Table 8.8: Per-sweep perception cycle time on the forward-pass worlds. N sweeps.

Scenario	N	cycle mean [ms]	cycle STD.dev [ms]	cycle max [ms]	raw pts mean
Straight	395	13.58	2.77	21.67	30000
Arc R=20 m	366	12.58	2.83	19.89	30000
Arc R=10 m	391	12.72	2.63	21.83	30000
Arc R=5 m	395	11.67	2.32	18.76	30000
S-curve (R=8 m)	436	12.81	2.98	21.15	30000

Table 8.7 shows the width estimate staying close to the nominal 2.5 m spacing on the near-straight worlds and degrading under strong curvature, where the two rows are no longer parallel. Table 8.8 shows the per-sweep cycle time remaining well within a sweep period, supporting the detector's use as an on-line component.

The rosbag offers no surveyed inter-row spacing to compare against, but it gives a genuine internal-consistency check over a continuous real run. Table 8.9 reports the per-sweep width statistics and Figure 8.7 its evolution over the run.

Table 8.9: Inter-row width W measured per sweep: mean, standard deviation, and observed range. No fixed ground-truth spacing is assumed; these figures assess the internal consistency of the estimator across the run. N sweeps.

N	mean W [m]	STD.dev [m]	min [m]	max [m]	range [m]
237	2.032	0.042	1.929	2.133	0.204

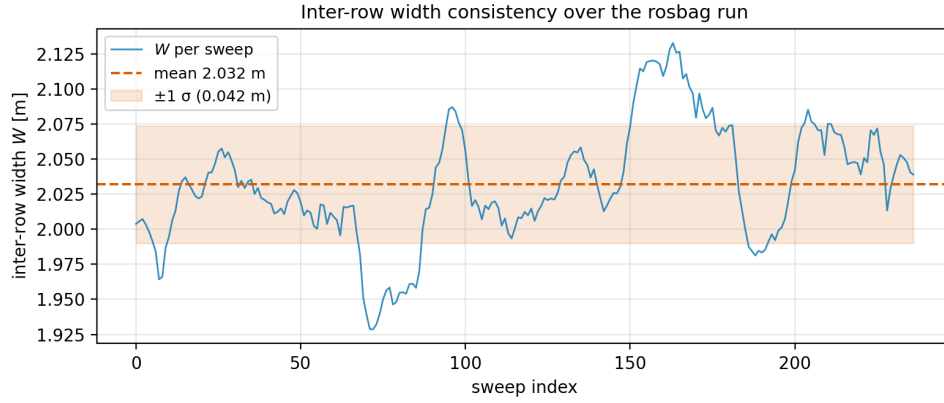


Figure 8.7: Inter-row width per sweep over the real-vineyard rosbag run, with the run mean and $\pm 1\sigma$ band.

The estimate stays close to its own 2.032 m run mean, with a standard deviation of 0.042 m and a full observed range of 0.204 m (Table 8.9); this is the same order of stability seen on the near-straight simulated worlds (Table 8.7), now obtained from a continuously varying real corridor rather than an analytically fixed one.

Per-sweep and per-cycle cost are likewise checked on the rosbag. Table 8.10 reports the cycle time of the full perception sweep and of each mapping stage over the recorded run; the mapping-stage rows are discussed together with their simulated counterparts in Section 8.3.3.

Table 8.10: Cycle time per subsystem: mean, standard deviation, maximum, and 95th-percentile latency. Times in milliseconds. N frames.

Subsystem	N	mean [ms]	STD.dev [ms]	max [ms]	P95 [ms]
Perception (full cycle)	237	17.17	3.43	28.23	22.73
Mapping — height gate	223	0.49	0.11	0.85	0.68
Mapping — CropHull	223	1.10	0.25	2.22	1.58
Mapping — grid update	223	0.23	0.13	0.84	0.56
Mapping — cut-line det.	223	3.72	4.38	27.93	11.58
Mapping — total	223	6.41	4.57	31.34	15.04

The perception sweep averages 17.17 ms (P95 22.73 ms), somewhat above the 11.67 to 13.58 ms range of Table 8.8; this is consistent with the larger raw point count returned by the real sensor (Table 8.13), discussed in Section 8.3.1, rather than with any change in the underlying algorithm. Both the per-sweep and the per-cycle figures stay well inside the sweep period, so the on-line cost margin established in simulation carries over to the field run.

8.3 Mapping and Cut-Line Detection Results

The mapping node of Chapter 6 is evaluated through point-cloud intake, cut-line detection counts, cut-line geometric error and mapping-cycle cost. The four families exercise the detector under,

respectively, no cut line (*FP* and *GAP*), a full-length cut line (*BP*) and a half-length cut line (*TP*).

8.3.1 Point-Cloud Intake

The mapping node applies a height gate and then the perception-derived corridor polygon as a spatial mask (Chapter 6). Table 8.11 reports how many points survive each stage and Table 8.12 their cost.

Table 8.11: Mapping intake point counts on the cut-bearing (*BP*, *TP*) worlds. N frames.

Scenario	N	raw pts	after height	after CropHull
BP Straight	373	30000	10431	590
BP Arc R=20 m	328	30000	10412	535
BP Arc R=10 m	322	30000	10441	512
BP Arc R=5 m	103	30000	10098	567
BP S-curve (R=8 m)	374	30000	10389	553
TP Straight	368	30000	10530	451
TP Arc R=20 m	387	30000	10481	438
TP Arc R=10 m	386	30000	10433	437
TP Arc R=5 m	249	30000	10214	490
TP S-curve (R=8 m)	351	30000	10272	443

Table 8.12: Mapping intake processing time on the cut-bearing (*BP*, *TP*) worlds.

Scenario	N	height [ms]	CropHull [ms]	grid [ms]	cutline [ms]	total [ms]
BP Straight	373	0.30	1.98	0.23	5.39	8.86
BP Arc R=20 m	328	0.28	1.81	0.18	4.94	8.16
BP Arc R=10 m	322	0.28	1.75	0.19	5.04	8.15
BP Arc R=5 m	103	0.27	1.72	0.20	5.27	8.34
BP S-curve (R=8 m)	374	0.29	1.89	0.19	5.16	8.40
TP Straight	368	0.27	1.79	0.15	4.87	7.99
TP Arc R=20 m	387	0.27	1.81	0.15	4.99	8.09
TP Arc R=10 m	386	0.27	1.81	0.15	5.01	8.12
TP Arc R=5 m	249	0.29	1.87	0.19	5.28	8.60
TP S-curve (R=8 m)	351	0.28	1.82	0.16	5.40	8.54

Tables 8.11 and 8.12 show the raw cloud being reduced to the corridor-floor region at a cost compatible with on-line operation; they are reported on the cut-bearing (*BP*, *TP*) worlds, where mapping and cut-line detection are exercised.

The same reduction is checked on the real-vineyard rosbag (Section 8.1.1), where the real LiDAR sweep returns substantially more raw points per frame than the simulated sensor. Table 8.13 reports the mean point count surviving each stage.

Table 8.13: Mapping intake point counts: mean raw points and mean surviving points after the height-gate and CropHull filtering stages, with the mean fraction removed at each step. N frames.

N	raw pts (mean)	after height gate (mean)	after CropHull (mean)	% removed (height)	% removed (CropHull)
223	57600	9935	1089	82.8	89.0

Table 8.13 shows a raw sweep of roughly 57 600 points being reduced to 1 089 surviving points, just under 2 % of the raw cloud. The two filtering stages redistribute that reduction differently from simulation: the height gate removes a larger share here (82.8 %, against roughly 65 % on the *BP* worlds of Table 8.11) and the CropHull mask a smaller further share of what remains (89.0 %, against over 94 %), consistent with a denser, noisier real sweep being thinned earlier by the height gate; the overall fraction reaching the corridor-floor region is nevertheless close to its simulated counterpart, where roughly 2 % of the raw cloud survives both stages.

8.3.2 Cut-Line Detection and Abstention

The detector is evaluated as a classifier against the cut-line layout of each family. Frames whose look-ahead window lies in a cut region are positive and should yield a valid line; frames in a no-cut region are negative and the detector should abstain. Table 8.14 reports the resulting confusion counts.

Table 8.14: Cut-line detection rates on the cut-bearing worlds. TP and FN are percentages of the positive frames; TN and FP of the negative frames. The *BP* worlds have no negative region, so their abstention columns are not defined.

Scenario	pos. frames	neg. frames	TP %	FN %	TN %	FP %
BP Straight	373	0	89.5	10.5	–	–
BP Arc R=20 m	328	0	88.7	11.3	–	–
BP Arc R=10 m	322	0	86.0	14.0	–	–
BP Arc R=5 m	103	0	83.5	16.5	–	–
BP S-curve (R=8 m)	374	0	94.4	5.6	–	–
TP Straight	197	146	88.8	11.2	100.0	0.0
TP Arc R=20 m	201	162	91.0	9.0	100.0	0.0
TP Arc R=10 m	215	140	83.3	16.7	100.0	0.0
TP Arc R=5 m	114	101	93.9	6.1	100.0	0.0
TP S-curve (R=8 m)	192	133	94.8	5.2	100.0	0.0

Table 8.14 captures the intended behaviour of the acceptance-gate cascade of Chapter 6. In the *FP* and *GAP* families every frame is negative and the detector records no false positives, that is, it abstains throughout a corridor that contains no cut line. In the *BP* family the whole run is positive and the detector recovers the line on the large majority of frames, and in the *TP* family it separates the no-cut first half from the cut second half. The cascade thus knows when not to detect, the property that makes the cut line usable as a safety-critical reference.

8.3.3 Cut-Line Geometric Error

Where a valid line is produced, its external accuracy is the lateral error to the analytic mown–unmown boundary, shown for the cut-bearing worlds in Figure 8.8 and detailed in Table 8.15; the internal weighted-least-squares fit residual is reported in Table 8.16.

Table 8.15: Cut-line external geometric accuracy on the cut-bearing worlds (positive valid frames).

Scenario	<i>N</i>	lat mean [m]	lat STD.dev [m]	lat max [m]	orient mean [deg]	orient STD.dev [deg]
BP Straight	334	0.110	0.066	0.455	2.04	5.09
BP Arc R=20 m	291	0.111	0.069	0.380	3.04	4.80
BP Arc R=10 m	277	0.084	0.049	0.294	4.75	3.47
BP Arc R=5 m	86	0.050	0.039	0.143	7.50	5.71
BP S-curve (R=8 m)	353	0.104	0.054	0.457	1.46	5.36

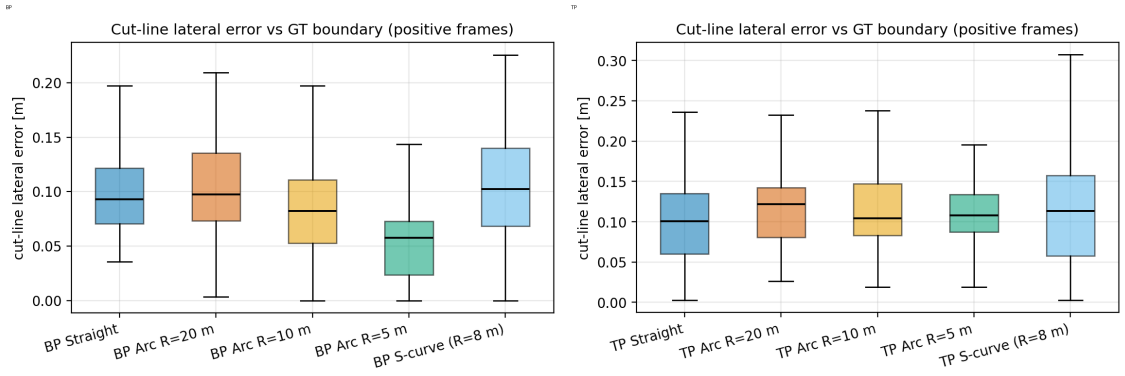


Figure 8.8: Cut-line lateral error against the ground-truth boundary: *Backward Pass* on the left; and *Transition Pass* on the right.

Figure 8.8 is the principal evidence that the node detects the cut line as a geometric boundary rather than reacting to any local height change; the error is reported only on the cut-bearing *BP* and *TP* worlds, the only ones that contain a cut line.

Table 8.16: Cut-line weighted-least-squares fit RMSE on the cut-bearing (*BP*, *TP*) worlds.

Scenario	<i>N</i>	RMSE mean [m]
BP Straight	334	0.019
BP Arc R=20 m	291	0.019
BP Arc R=10 m	277	0.018
BP Arc R=5 m	86	0.028
BP S-curve (R=8 m)	353	0.019
TP Straight	175	0.021
TP Arc R=20 m	183	0.022
TP Arc R=10 m	179	0.019
TP Arc R=5 m	107	0.026
TP S-curve (R=8 m)	182	0.031

Table 8.16 shows the fit residual sitting at the millimetre-to-centimetre level, confirming that accepted lines are well modelled.

The same internal fit-quality check is available on the real-vineyard rosbag (Section 8.1.1): no analytic mown–unmown boundary exists there to compute the external lateral error of Figure 8.8, but the WLS residual itself is computed identically. Table 8.17 reports the residual over every valid detection of the run, and Figure 8.9 its distribution.

Table 8.17: Cut-line WLS fit RMSE on valid detections: mean, standard deviation, maximum, and fraction within the 0.10 m acceptance gate. N is the number of frames in which a cut line was successfully detected.

N valid det.	mean RMSE [m]	STD.dev [m]	max RMSE [m]	% ≤ 0.10 m
158	0.033	0.019	0.099	100.0

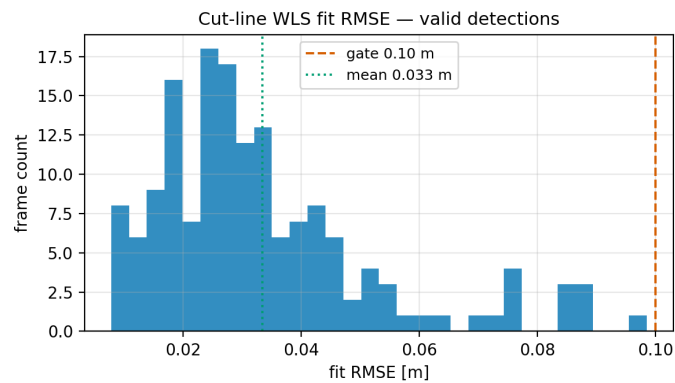


Figure 8.9: Cut-line weighted-least-squares fit RMSE on valid detections, real-vineyard rosbag; the dashed line marks the 0.10 m acceptance gate.

Of the 223 mapping frames in the run, 158 produce a valid cut-line detection, and every one of them fits within the 0.10 m acceptance gate, with a mean residual of 0.033 m (Table 8.17). This sits slightly above the millimetre-to-centimetre residuals of the simulated cut-bearing worlds (Table 8.16) but comfortably inside the same gate, so accepted lines remain well modelled under real sensor noise. Table 8.18 reports the mapping-cycle time.

Table 8.18: Cut-line and mapping-cycle time on the cut-bearing worlds. N frames.

Scenario	N	t_{cutline} mean [ms]	STD.dev [ms]	max [ms]	t_{total} mean [ms]
BP Straight	373	5.39	5.72	33.48	8.86
BP Arc R=20 m	328	4.94	5.12	24.54	8.16
BP Arc R=10 m	322	5.04	5.14	33.36	8.15
BP Arc R=5 m	103	5.27	5.55	21.29	8.34
BP S-curve (R=8 m)	374	5.16	5.43	32.18	8.40
TP Straight	368	4.87	5.11	31.17	7.99
TP Arc R=20 m	387	4.99	5.17	23.88	8.09
TP Arc R=10 m	386	5.01	5.23	29.85	8.12
TP Arc R=5 m	249	5.28	5.51	30.13	8.60
TP S-curve (R=8 m)	351	5.40	5.67	39.50	8.54

Table 8.18 supports the use of the mapping node as an on-line component of the sensing-to-reference pipeline. The same margin holds on the rosbag: Table 8.10 (Section 8.2.2) shows the cut-line detection step remaining the dominant and most variable mapping cost, consistent with the breakdown of Table 8.18, with the four mapping stages together a small fraction of the frame period.

8.4 Local-Planning Results

The planner of Chapter 7 is evaluated through curve continuity, finite-state-machine behaviour, one-sided-loss robustness, planning cost and the geometric accuracy of the generated row-following and cut-line-following references.

8.4.1 Continuity, State Occupancy and Cost

The single continuous trajectory of Chapter 7 requires consecutive Bézier segments to join without a position gap or tangent jump. Across all worlds these join discontinuities are negligible: the chained reference stays continuous to within sub-millimetre position gaps and sub-degree tangent jumps, so it remains G^1 -continuous through every curve change. Table 8.19 reports the state occupancy on the transition and gap worlds, where the planner changes mode (TP) or reference side (GAP); each contiguous STOP block is compressed to one effective cycle to avoid inflating STOP time by the delay between the end of a mission and the operator stopping the run.

Table 8.19: Finite-state-machine occupancy (% of effective cycles, STOP blocks compressed) on the transition and gap worlds, where the planner changes mode or reference side.

Scenario	Init	ChangeReferenceSide	Following	ComputeNewCurve	STOP
TP Straight	3.9	0.0	94.5	1.4	0.1
TP Arc R=20 m	3.9	0.0	94.3	1.6	0.1
TP Arc R=10 m	11.7	0.3	86.4	1.6	0.1
TP Arc R=5 m	4.5	0.0	93.9	1.5	0.0
TP S-curve (R=8 m)	3.7	0.0	94.3	1.8	0.1
GAP Straight	3.9	0.3	94.4	1.3	0.1
GAP Arc R=20 m	3.8	0.3	94.6	1.2	0.1
GAP Arc R=10 m	11.3	0.5	86.9	1.2	0.1
GAP Arc R=5 m	11.0	0.4	87.6	1.0	0.0
GAP S-curve (R=8 m)	2.5	0.3	95.8	1.3	0.1

Table 8.19 shows the planner spending the large majority of each mission in the Following state, with brief excursions to recompute a curve and almost no STOP time before the end of the run. Table 8.20 reports the per-cycle planning time.

Table 8.20: Per-cycle planning time (forward-pass worlds, representative). N cycles.

Scenario	N	mean [ms]	STD.dev [ms]	max [ms]
Straight	727	0.136	0.059	0.565
Arc R=20 m	746	0.135	0.083	1.382
Arc R=10 m	765	0.130	0.061	0.530
Arc R=5 m	739	0.124	0.059	0.812
S-curve (R=8 m)	806	0.131	0.061	0.777

Table 8.20 confirms that the planner operates as an on-line reference generator rather than an offline post-processing step.

8.4.2 One-Sided-Loss Robustness

The gap test induces a genuine one-sided loss: over $s \in [7, 13]$ m the right row is physically absent and its detection rate collapses (Table 8.6). Table 8.21 reports, over the cycles spent inside that interval, the fraction in which the planner kept a usable reference instead of stopping.

Table 8.21: One-sided-loss robustness (GAP worlds). Within the arc-length window where the right row is physically absent, the number of control cycles and the fraction in which the planner kept a usable reference (did not enter STOP), navigating on the remaining left row.

Scenario	gap cycles	maintained (non-STOP)	% maintained
GAP Straight	204	204	100.0
GAP Arc R=20 m	207	207	100.0
GAP Arc R=10 m	209	209	100.0
GAP Arc R=5 m	226	226	100.0
GAP S-curve (R=8 m)	205	205	100.0

Table 8.21 shows the planner maintaining a valid reference on the surviving left row throughout the gap in every shape, so the loss of one side does not halt the mission. This is the practical demonstration of the hysteretic per-side safety logic of Chapter 7.

8.4.3 Row-Following Trajectory Error

Row-following references are compared with the analytic followed row shifted laterally to the corridor interior by the configured 1.0 m offset, on the forward-pass worlds (the clean row-following case) and on the gap worlds, where following continues on the surviving row. Tables 8.22 and 8.23 report the lateral and heading error of the generated samples.

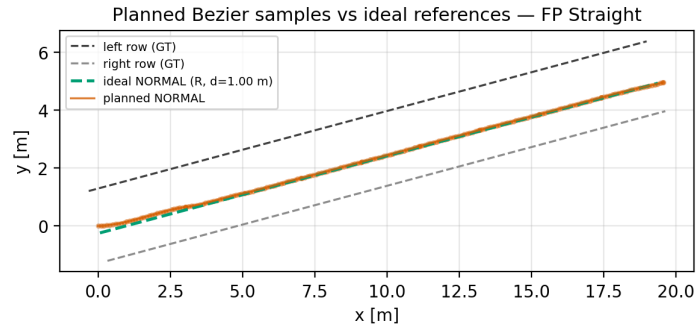
Table 8.22: Row-following trajectory lateral error on the forward-pass and gap worlds. N samples.

Scenario	N	lat mean [m]	lat STD.dev [m]	lat RMSE [m]	lat max [m]
FP Straight	216	0.043	0.046	0.063	0.250
FP Arc R=20 m	222	0.053	0.064	0.084	0.250
FP Arc R=10 m	228	0.068	0.072	0.099	0.255
FP Arc R=5 m	187	0.062	0.083	0.103	0.277
FP S-curve (R=8 m)	224	0.049	0.084	0.097	0.265
GAP Straight	215	0.028	0.061	0.067	0.250
GAP Arc R=20 m	222	0.036	0.077	0.085	0.250
GAP Arc R=10 m	217	0.063	0.074	0.097	0.254
GAP Arc R=5 m	187	0.041	0.146	0.151	0.336
GAP S-curve (R=8 m)	214	0.040	0.090	0.098	0.263

Table 8.23: Row-following trajectory heading error on the forward-pass and gap worlds. N samples.

Scenario	N	head mean [deg]	head STD.dev [deg]	head RMSE [deg]	head max [deg]
FP Straight	216	-0.83	2.16	2.31	13.86
FP Arc R=20 m	222	-0.97	2.51	2.69	6.39
FP Arc R=10 m	228	-0.74	4.90	4.95	10.58
FP Arc R=5 m	187	-2.06	8.50	8.72	24.56
FP S-curve (R=8 m)	224	-0.89	6.01	6.06	16.81
GAP Straight	215	-0.79	2.55	2.66	13.87
GAP Arc R=20 m	222	-1.13	3.23	3.41	8.69
GAP Arc R=10 m	217	-1.19	5.32	5.44	14.42
GAP Arc R=5 m	187	-2.46	13.42	13.61	39.14
GAP S-curve (R=8 m)	214	-1.04	5.85	5.93	15.48

The spatial form of these references is shown per world, one figure each: the *Forward Pass* overlays in Figures 8.10, 8.11, 8.12, 8.13 and 8.14, and the *GAP* overlays in Figures 8.15, 8.16, 8.17, 8.18 and 8.19; each overlays the analytic rows, the ideal offset reference and the generated samples.

Figure 8.10: Planned row-following reference, *Forward Pass* family, straight corridor.

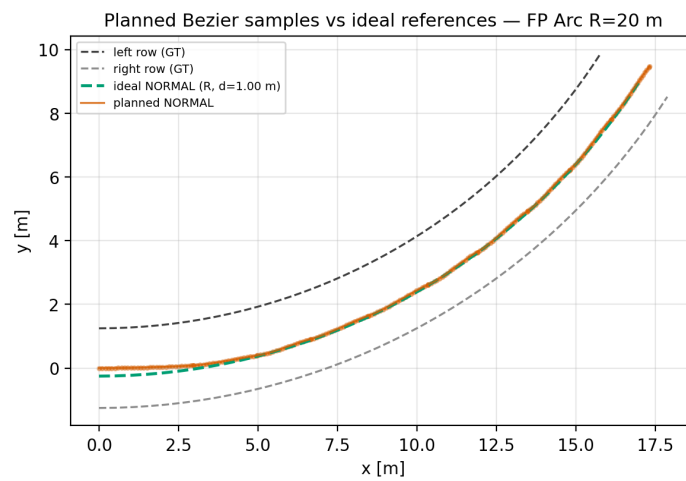


Figure 8.11: Planned row-following reference, *Forward Pass* family, 20 m arc.

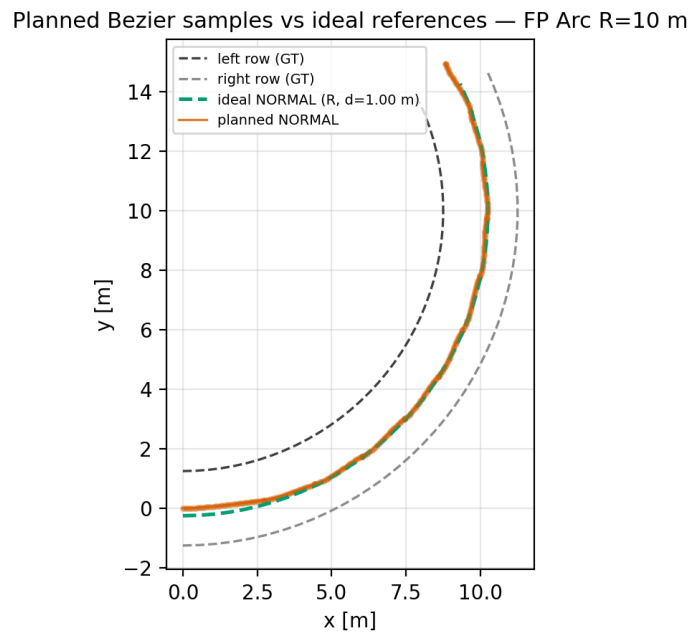
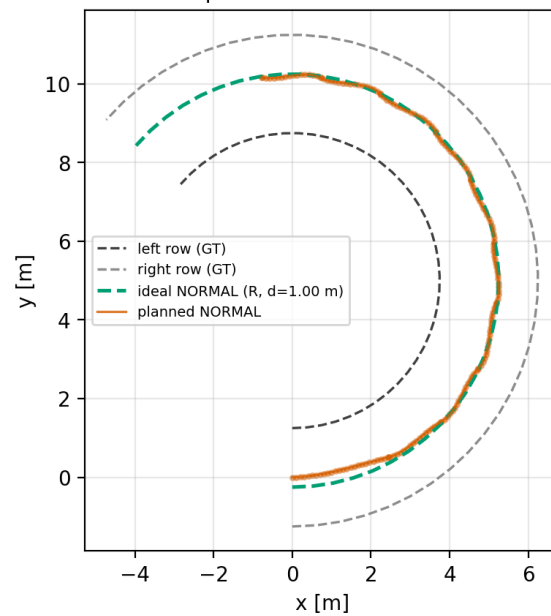
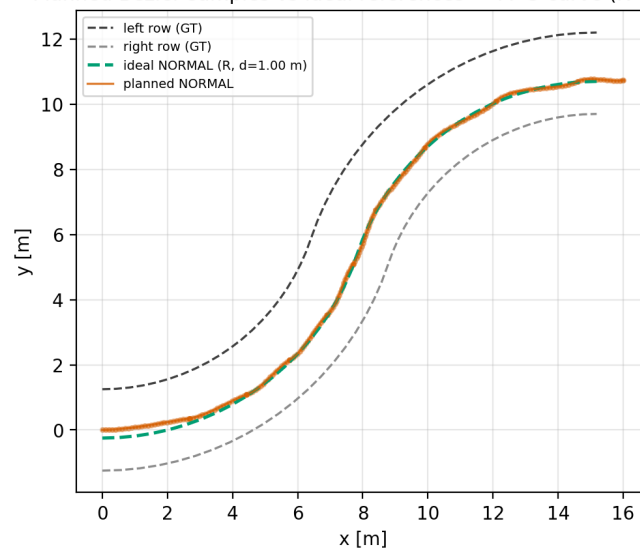


Figure 8.12: Planned row-following reference, *Forward Pass* family, 10 m arc.

Planned Bezier samples vs ideal references — FP Arc R=5 m

Figure 8.13: Planned row-following reference, *Forward Pass* family, 5 m arc.

Planned Bezier samples vs ideal references — FP S-curve (R=8 m)

Figure 8.14: Planned row-following reference, *Forward Pass* family, S-curve.

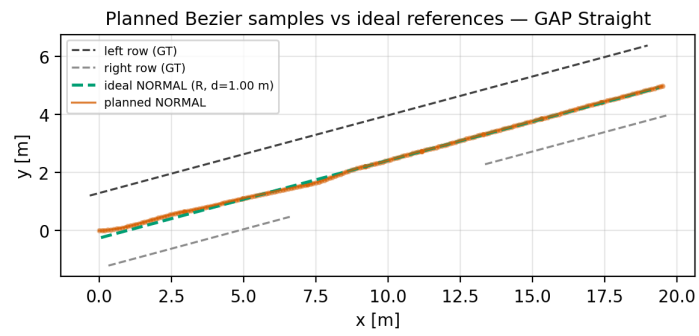


Figure 8.15: Planned row-following reference, *GAP* family, straight corridor; the reference stays on the left row through the missing-row interval.

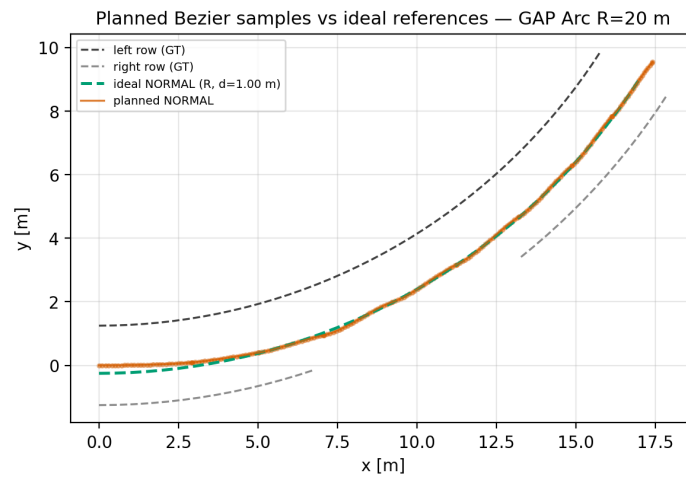
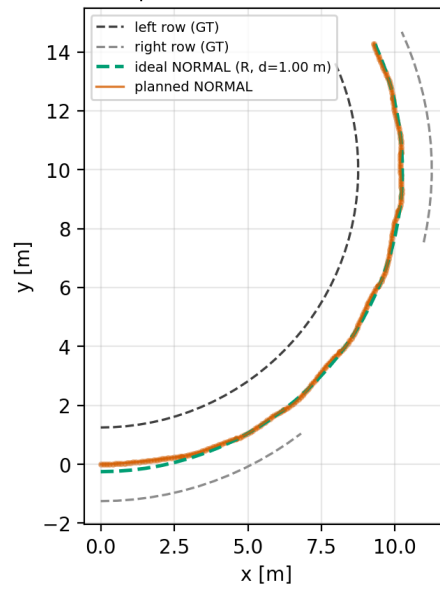
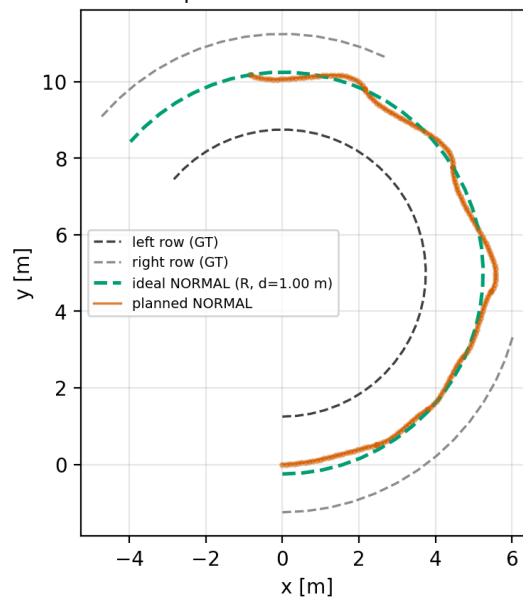


Figure 8.16: Planned row-following reference, *GAP* family, 20 m arc.

Planned Bezier samples vs ideal references — GAP Arc R=10 m

Figure 8.17: Planned row-following reference, *GAP* family, 10 m arc.

Planned Bezier samples vs ideal references — GAP Arc R=5 m

Figure 8.18: Planned row-following reference, *GAP* family, 5 m arc.

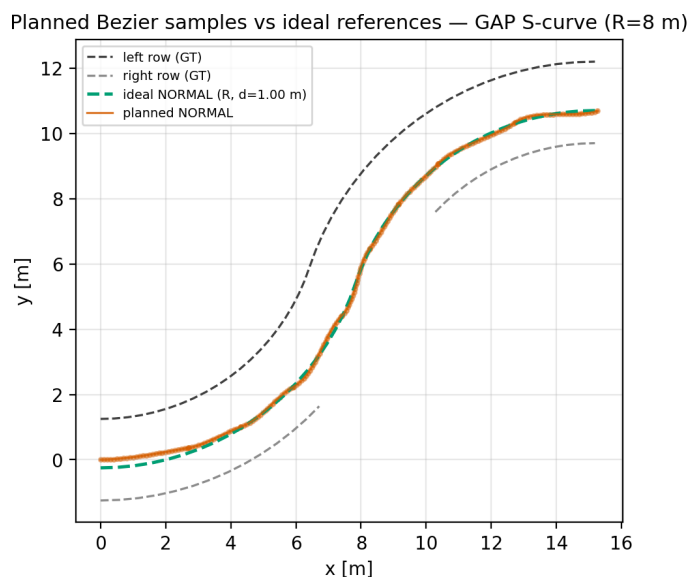


Figure 8.19: Planned row-following reference, *GAP* family, S-curve.

The per-cycle evolution of the error is shown per world, one figure each: the *Forward Pass* traces in Figures 8.20, 8.21, 8.22, 8.23 and 8.24, and the *GAP* traces in Figures 8.25, 8.26, 8.27, 8.28 and 8.29; they exhibit the transient-to-steady-state behaviour described in Chapter 7, the early cycles carrying a larger error while the platform moves from the corridor centre toward the reference side, after which the error settles, and the *GAP* traces stay bounded while only the left row is available.

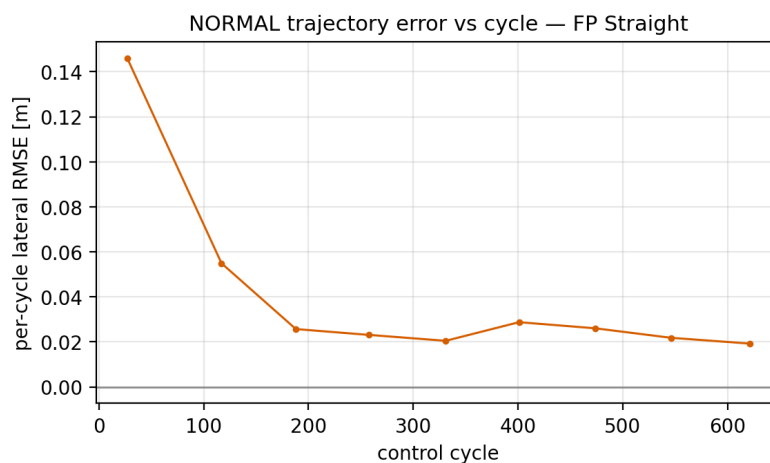


Figure 8.20: Row-following lateral error over the planning cycle, *Forward Pass* family, straight corridor.

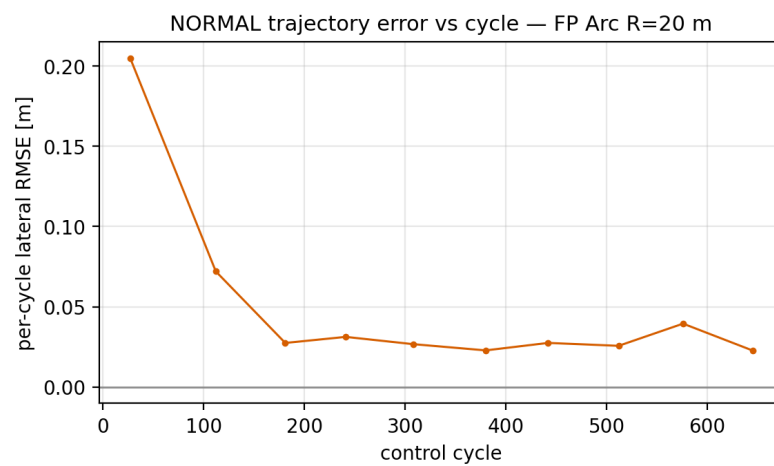


Figure 8.21: Row-following lateral error over the planning cycle, *Forward Pass* family, 20 m arc.

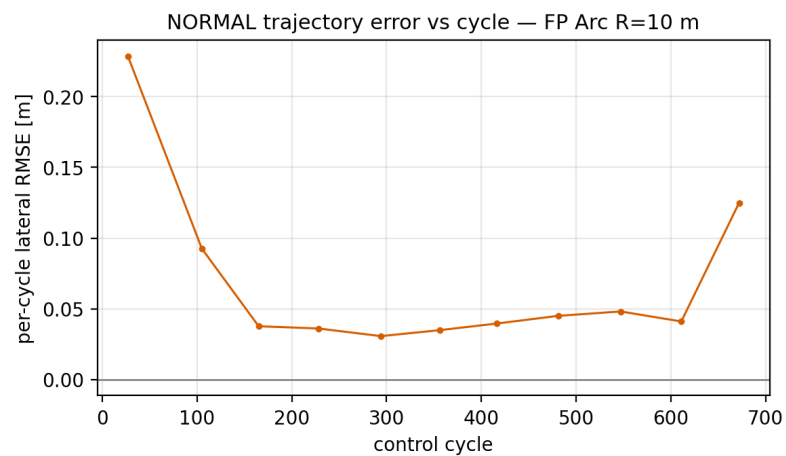


Figure 8.22: Row-following lateral error over the planning cycle, *Forward Pass* family, 10 m arc.

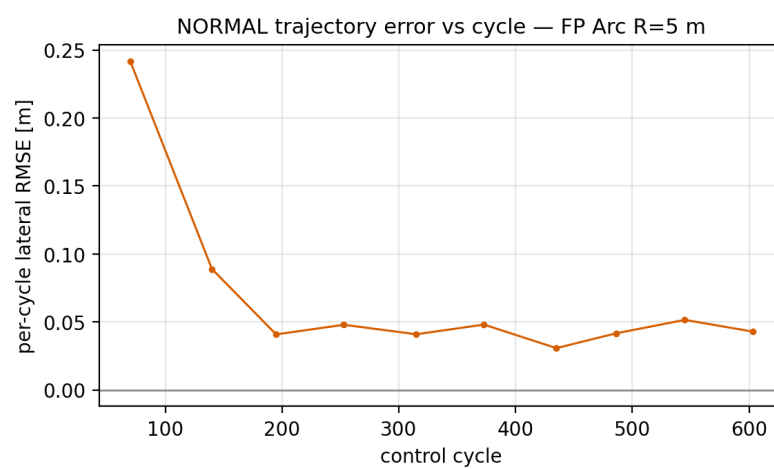


Figure 8.23: Row-following lateral error over the planning cycle, *Forward Pass* family, 5 m arc.

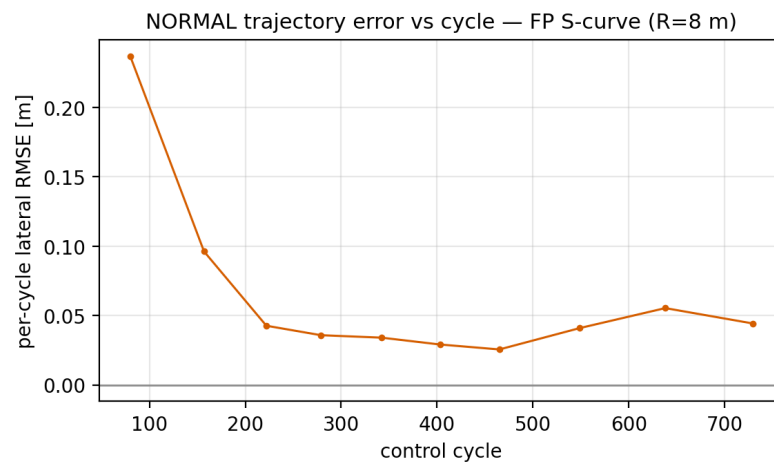


Figure 8.24: Row-following lateral error over the planning cycle, *Forward Pass* family, S-curve.

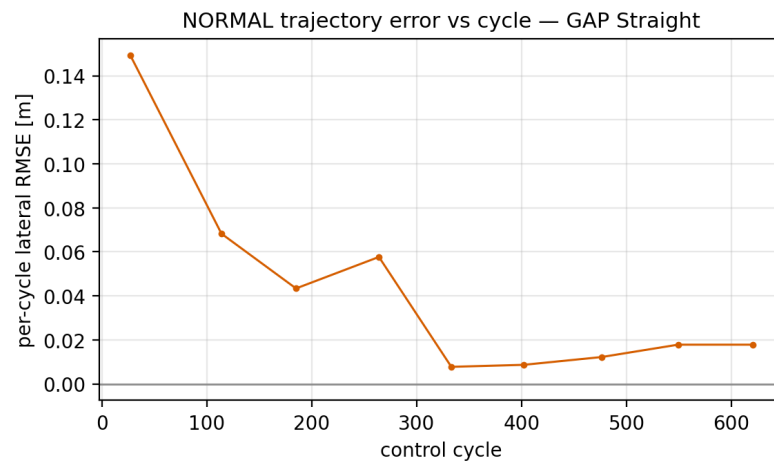


Figure 8.25: Row-following lateral error over the planning cycle, *GAP* family, straight corridor.

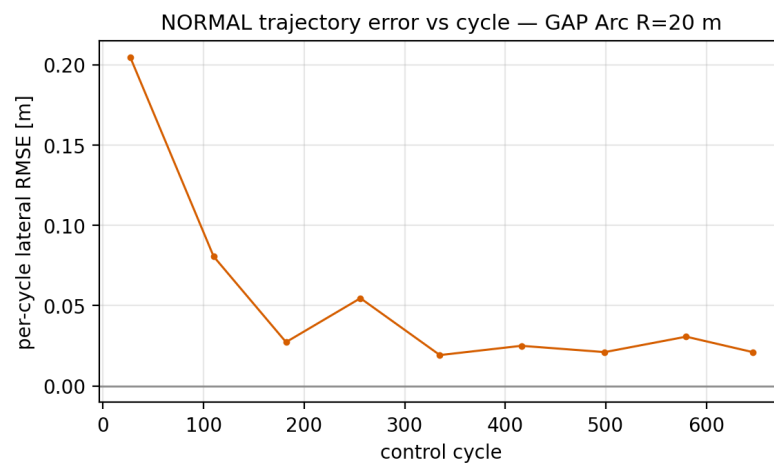


Figure 8.26: Row-following lateral error over the planning cycle, *GAP* family, 20 m arc.

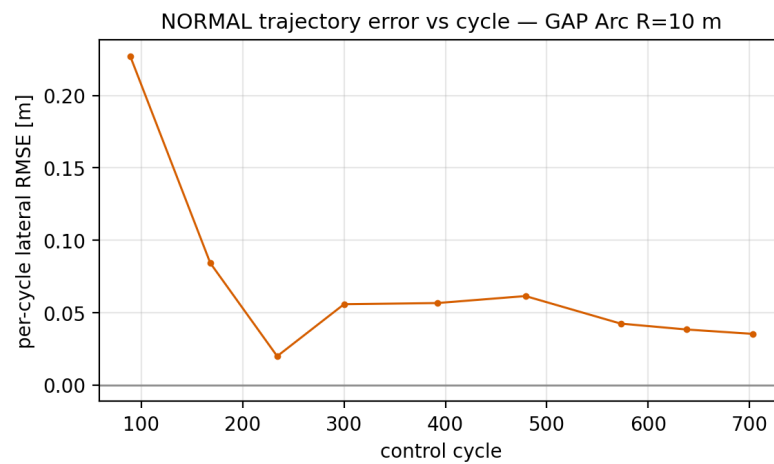


Figure 8.27: Row-following lateral error over the planning cycle, *GAP* family, 10 m arc.

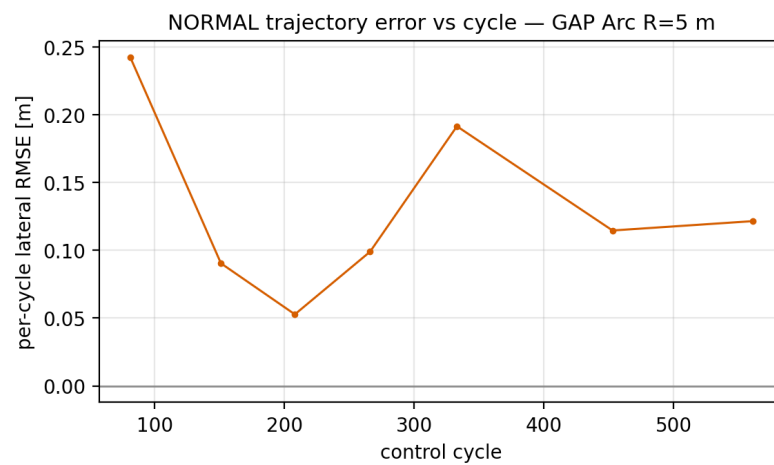


Figure 8.28: Row-following lateral error over the planning cycle, *GAP* family, 5 m arc.

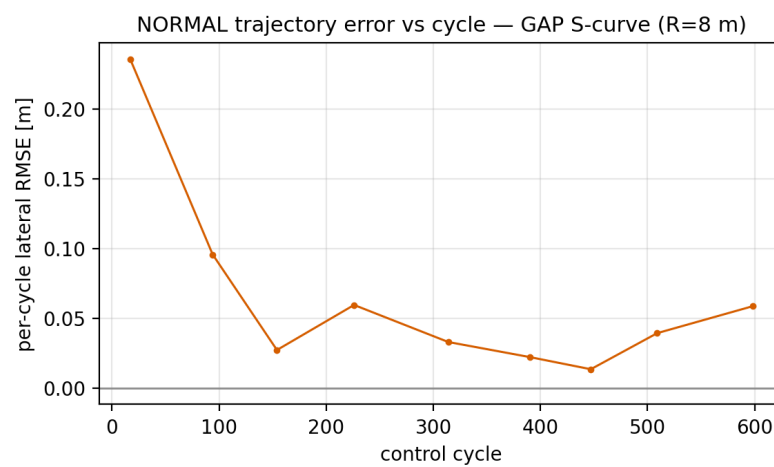


Figure 8.29: Row-following lateral error over the planning cycle, *GAP* family, S-curve.

These per-world traces confirm the bounded steady-state error after the initial transient, including the *GAP* family where the reference is held on a single row. The distribution of the error is summarised in Figure 8.30.

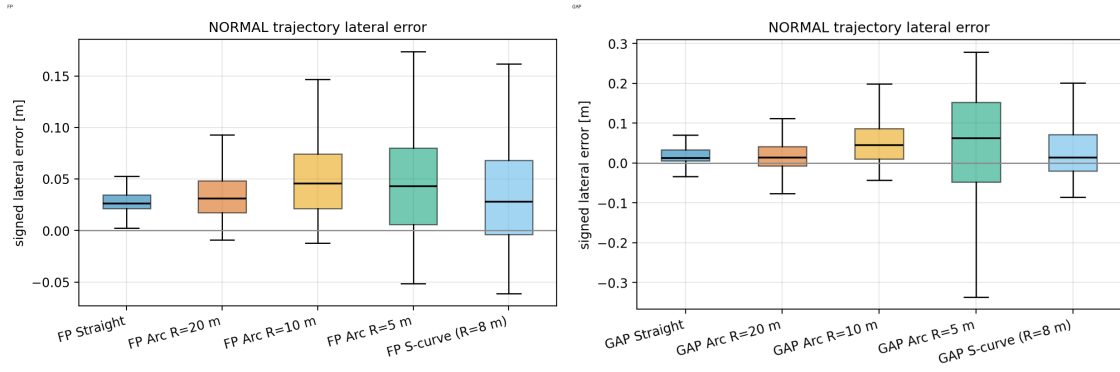


Figure 8.30: Distribution of the row-following lateral error: *FP* on the left; and *GAP* on the right.

Figure 8.30 reads together with Tables 8.22 and 8.23: the tables give the scenario-level statistics and the figure the distributional shape.

8.4.4 Cut-Line-Following Trajectory Error

Cut-line following closes the loop between mapping and planning: the planner shifts the detected boundary by half the cutter width toward the standing vegetation, the side now reported in the cut-line message (Chapter 6), so that the cutter edge reaches the line. The trajectory error against the offset boundary is reported in Tables 8.24 and 8.25, on the cut-bearing *Backward Pass* and *Transition Pass* worlds.

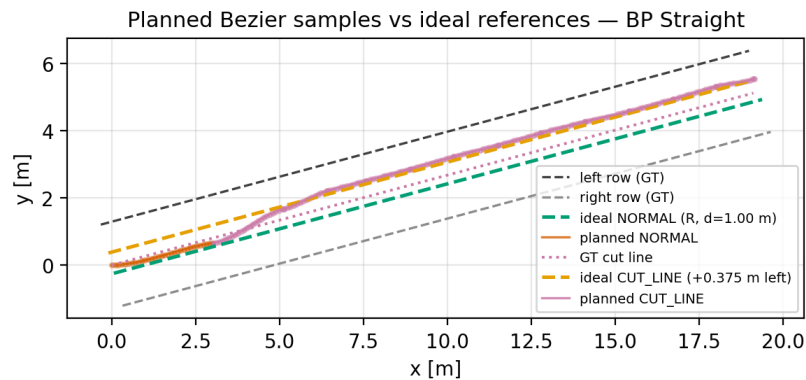
Table 8.24: Cut-line-following trajectory lateral error on the cut-bearing worlds. N samples.

Scenario	N	lat mean [m]	lat STD.dev [m]	lat RMSE [m]	lat max [m]
BP Straight	187	0.028	0.151	0.153	0.527
BP Arc R=20 m	189	0.035	0.155	0.159	0.497
BP Arc R=10 m	168	-0.137	0.154	0.206	0.620
BP Arc R=5 m	26	-0.380	0.184	0.421	0.648
BP S-curve (R=8 m)	176	-0.111	0.163	0.197	0.620
TP Straight	83	-0.039	0.215	0.217	0.610
TP Arc R=20 m	99	-0.086	0.186	0.204	0.599
TP Arc R=10 m	108	-0.198	0.169	0.260	0.613
TP Arc R=5 m	51	-0.247	0.204	0.319	0.648
TP S-curve (R=8 m)	97	-0.120	0.189	0.223	0.593

Table 8.25: Cut-line-following trajectory heading error on the cut-bearing worlds. N samples.

Scenario	N	head mean [deg]	head STD.dev [deg]	head RMSE [deg]	head max [deg]
BP Straight	187	1.79	4.83	5.14	16.92
BP Arc R=20 m	189	2.48	5.91	6.40	17.07
BP Arc R=10 m	168	2.53	7.82	8.20	18.95
BP Arc R=5 m	26	6.58	15.67	16.72	27.35
BP S-curve (R=8 m)	176	2.44	6.76	7.17	19.17
TP Straight	83	4.71	8.10	9.33	21.25
TP Arc R=20 m	99	3.84	6.29	7.34	18.65
TP Arc R=10 m	108	3.08	9.27	9.73	22.19
TP Arc R=5 m	51	5.70	13.45	14.49	24.47
TP S-curve (R=8 m)	97	3.93	8.29	9.13	22.62

The spatial form of the cut-line reference is shown per world, one figure each: the *Backward Pass* overlays in Figures 8.31, 8.32, 8.33, 8.34 and 8.35, and the *Transition Pass* overlays in Figures 8.36, 8.37, 8.38, 8.39 and 8.40; each overlays the rows, the ground-truth cut line and the generated samples.

Figure 8.31: Planned cut-line reference, *Backward Pass* family, straight corridor.

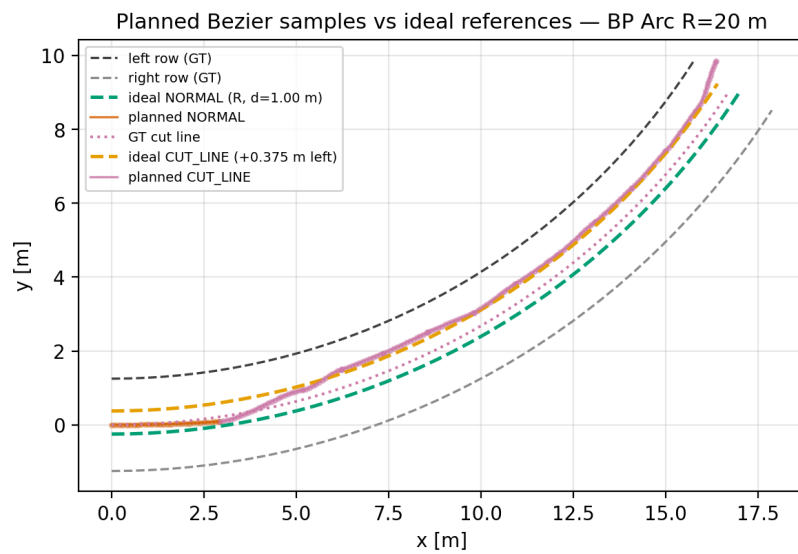


Figure 8.32: Planned cut-line reference, *Backward Pass* family, 20 m arc.

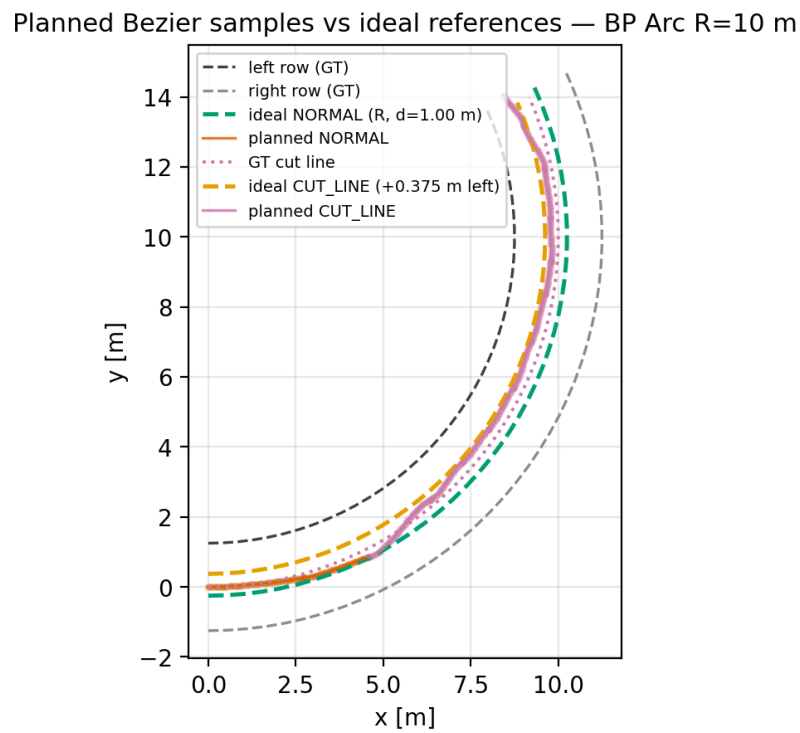
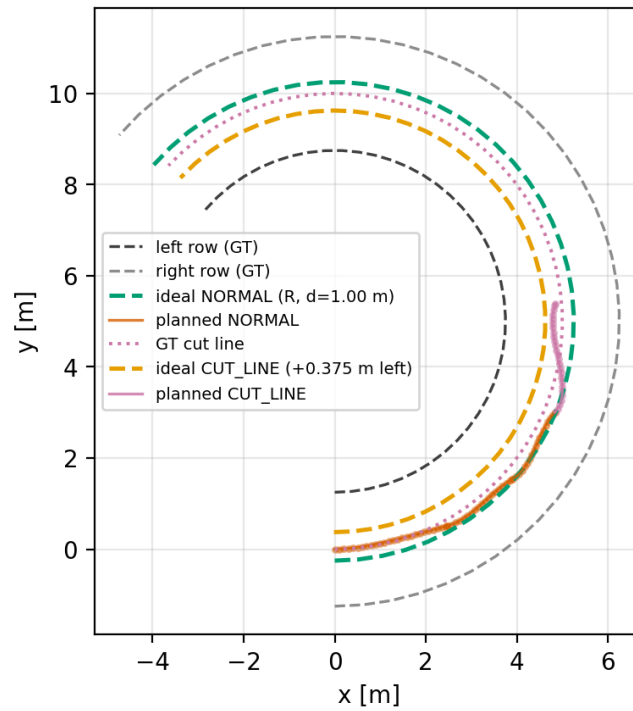
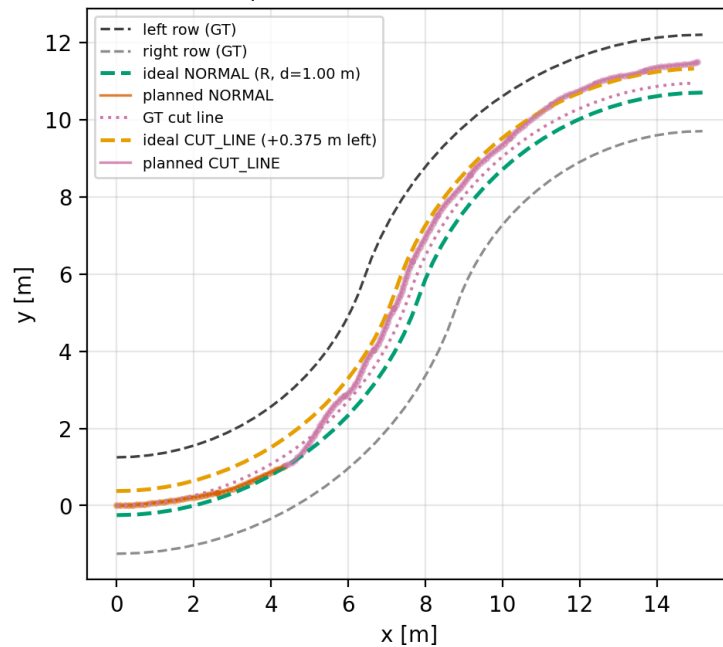


Figure 8.33: Planned cut-line reference, *Backward Pass* family, 10 m arc.

Planned Bezier samples vs ideal references — BP Arc R=5 m

Figure 8.34: Planned cut-line reference, *Backward Pass* family, 5 m arc.

Planned Bezier samples vs ideal references — BP S-curve (R=8 m)

Figure 8.35: Planned cut-line reference, *Backward Pass* family, S-curve.

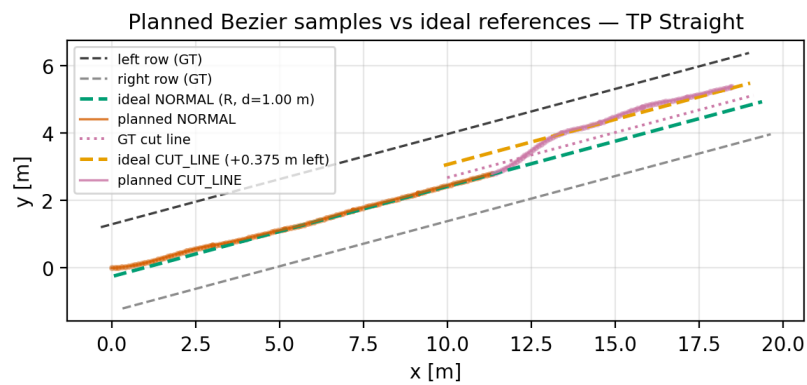


Figure 8.36: Planned cut-line reference, *Transition Pass* family, straight corridor.

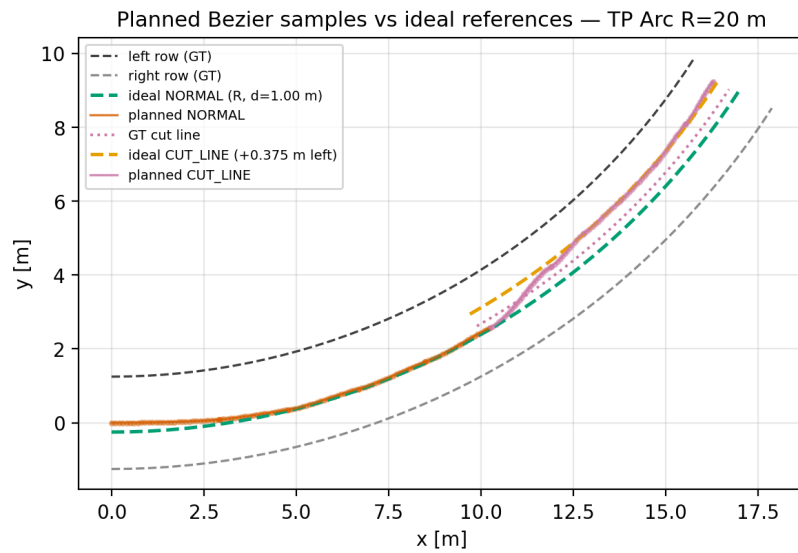
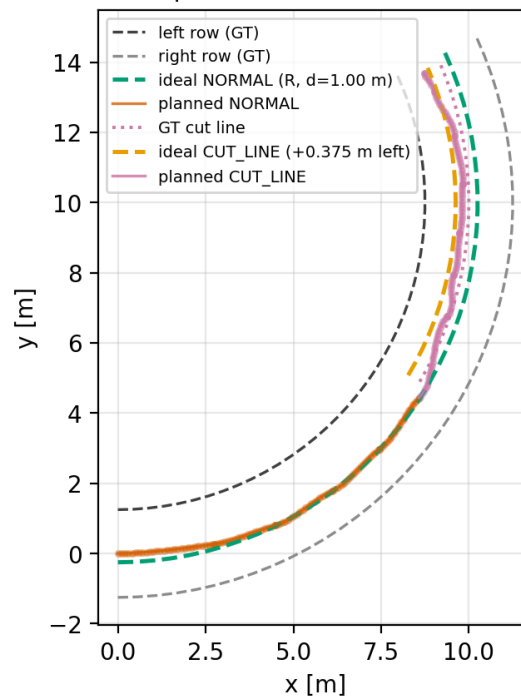
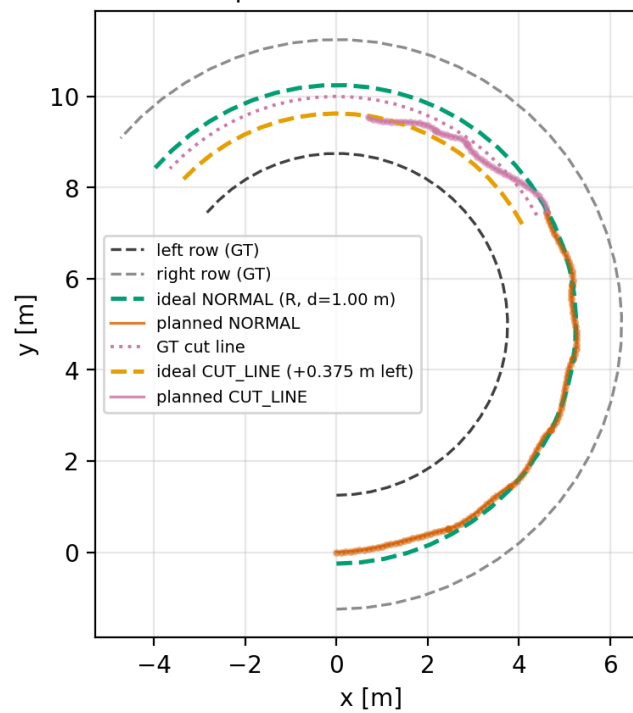


Figure 8.37: Planned cut-line reference, *Transition Pass* family, 20 m arc.

Planned Bezier samples vs ideal references — TP Arc R=10 m

Figure 8.38: Planned cut-line reference, *Transition Pass* family, 10 m arc.

Planned Bezier samples vs ideal references — TP Arc R=5 m

Figure 8.39: Planned cut-line reference, *Transition Pass* family, 5 m arc.

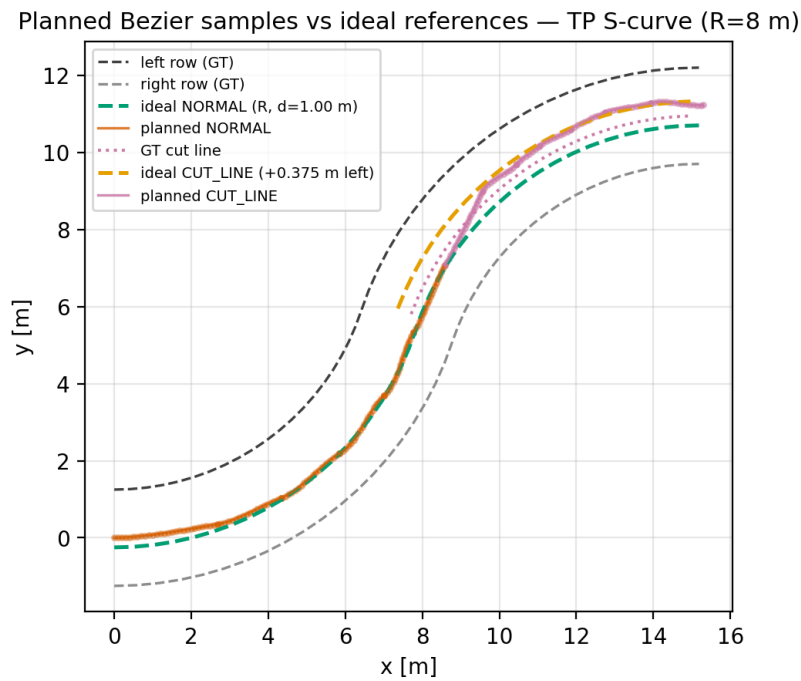


Figure 8.40: Planned cut-line reference, *Transition Pass* family, S-curve.

The per-cycle evolution of the error is shown per world, one figure each: the *Backward Pass* traces in Figures 8.41, 8.42, 8.43, 8.44 and 8.45, and the *Transition Pass* traces in Figures 8.46, 8.47, 8.48, 8.49 and 8.50; in the transition case a short transient appears as the planner switches onto the boundary.

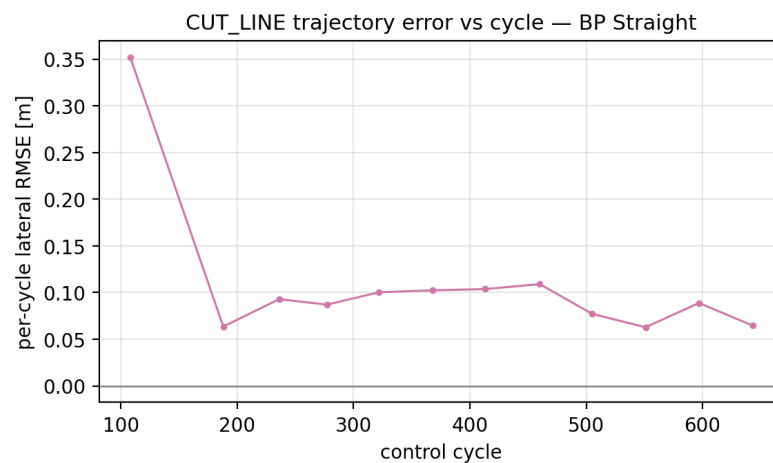


Figure 8.41: Cut-line-following lateral error over the planning cycle, *Backward Pass* family, straight corridor.

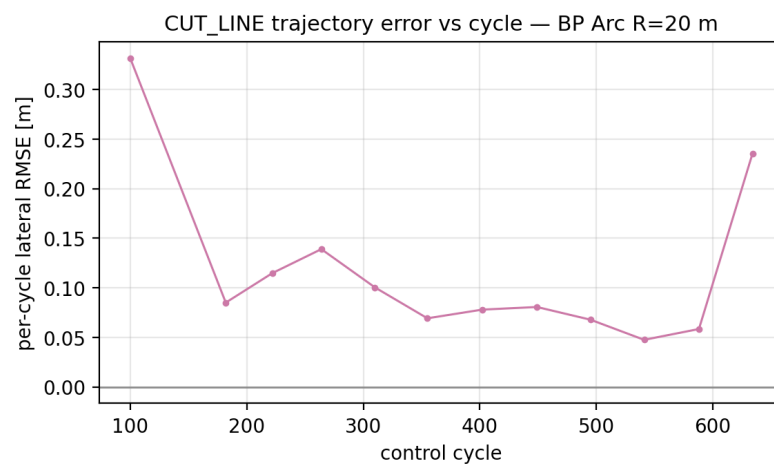


Figure 8.42: Cut-line-following lateral error over the planning cycle, *Backward Pass* family, 20 m arc.

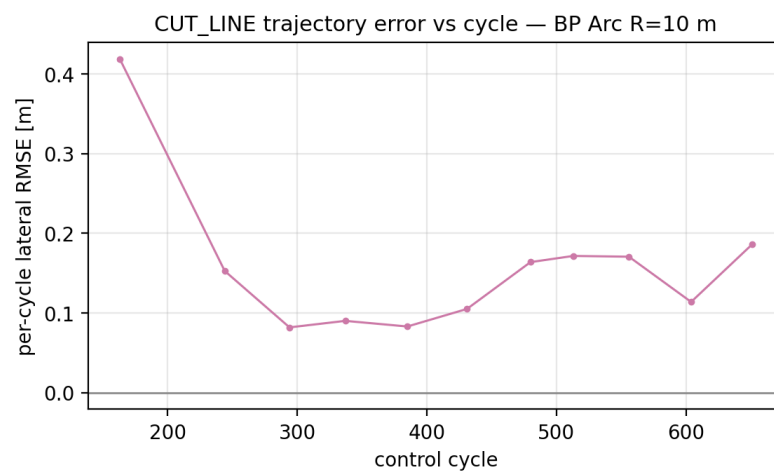


Figure 8.43: Cut-line-following lateral error over the planning cycle, *Backward Pass* family, 10 m arc.

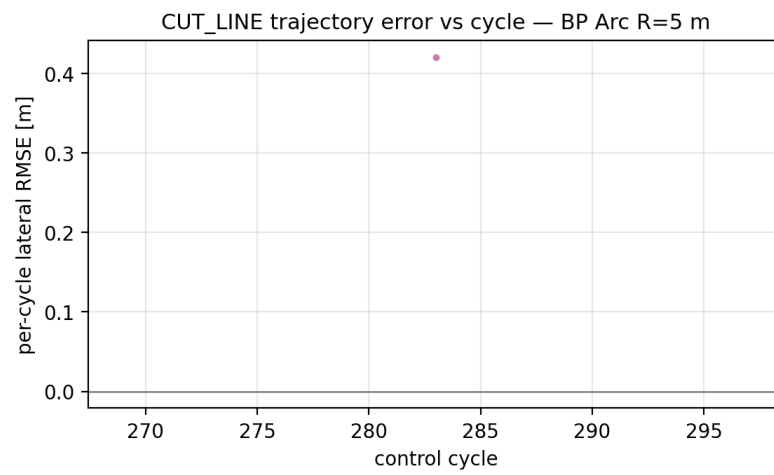


Figure 8.44: Cut-line-following lateral error over the planning cycle, *Backward Pass* family, 5 m arc.

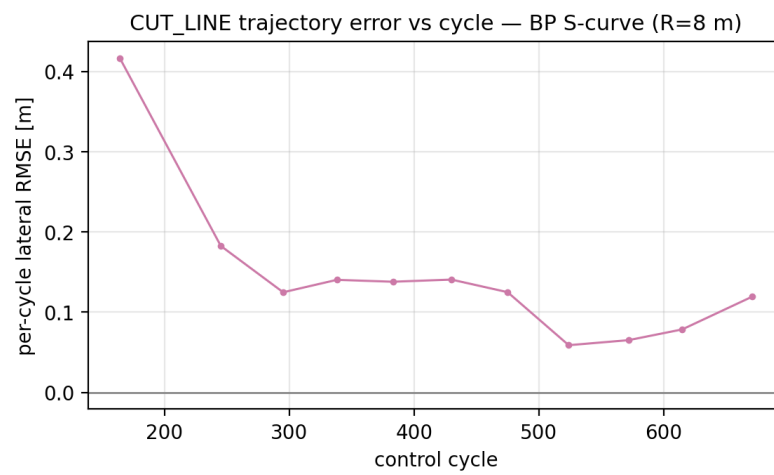


Figure 8.45: Cut-line-following lateral error over the planning cycle, *Backward Pass* family, S-curve.

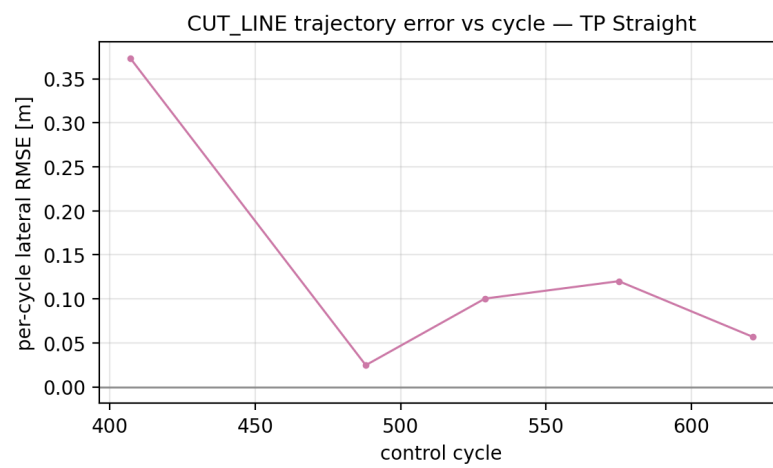


Figure 8.46: Cut-line-following lateral error over the planning cycle, *Transition Pass* family, straight corridor.

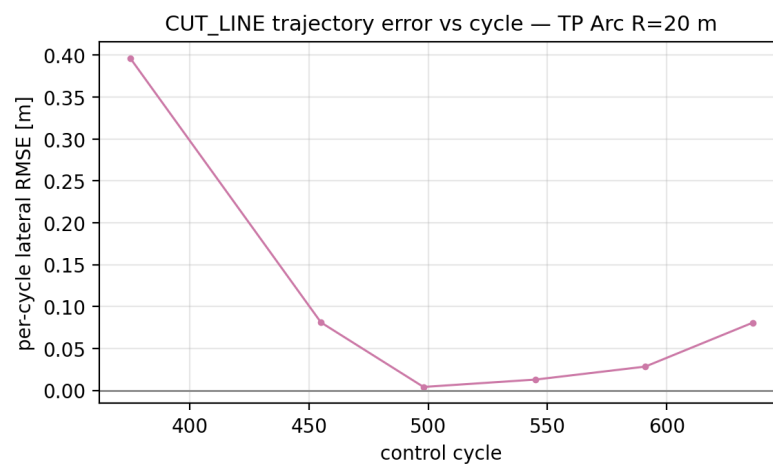


Figure 8.47: Cut-line-following lateral error over the planning cycle, *Transition Pass* family, 20 m arc.

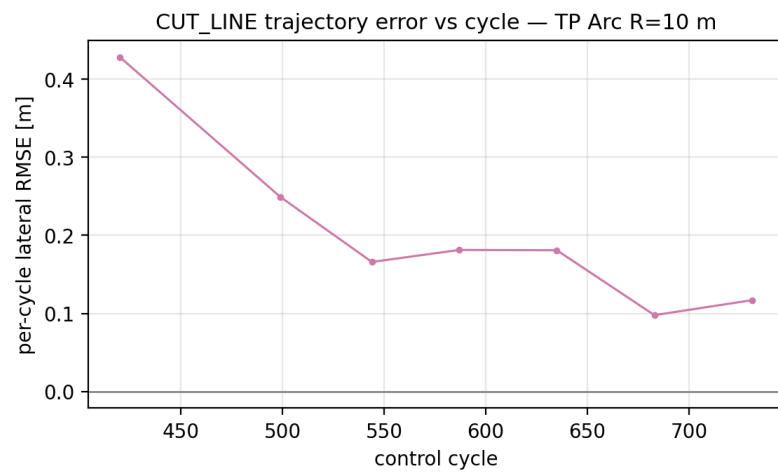


Figure 8.48: Cut-line-following lateral error over the planning cycle, *Transition Pass* family, 10 m arc.

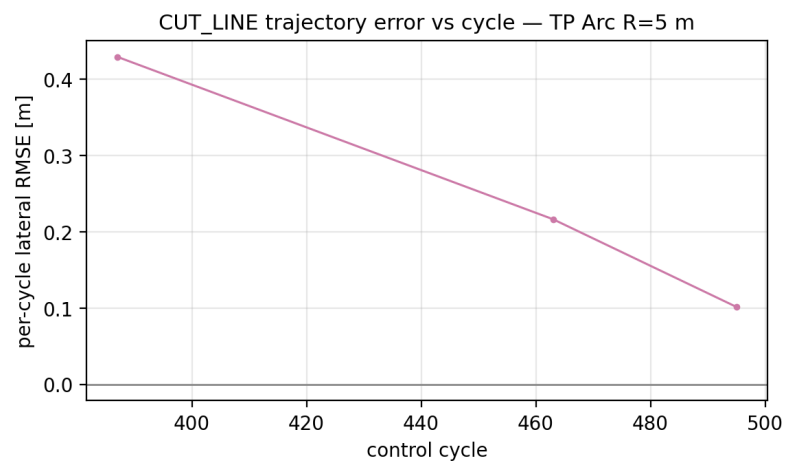


Figure 8.49: Cut-line-following lateral error over the planning cycle, *Transition Pass* family, 5 m arc.

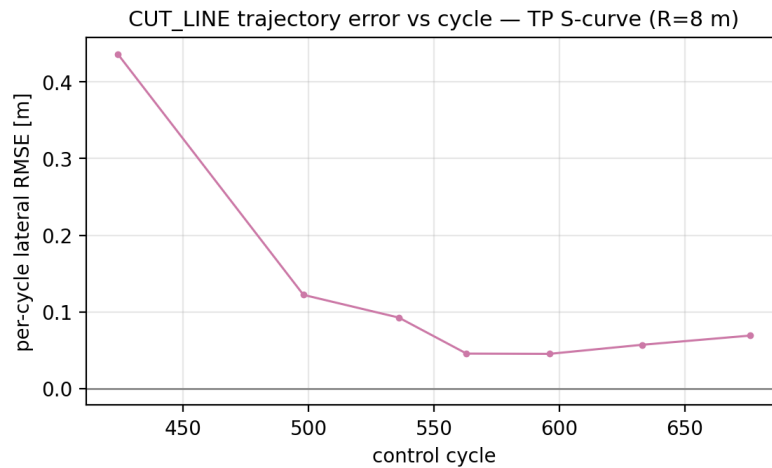


Figure 8.50: Cut-line-following lateral error over the planning cycle, *Transition Pass* family, S-curve.

These per-world traces show the reference settling onto the boundary after detection. Their distribution is summarised in Figure 8.51.

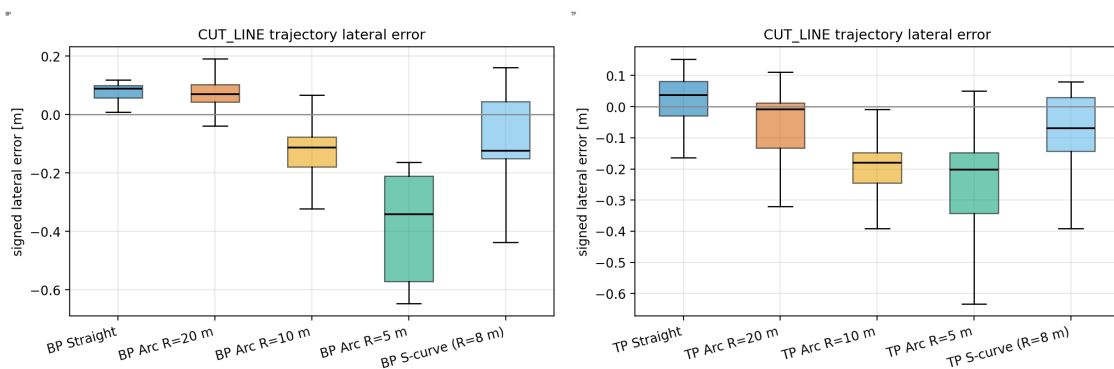


Figure 8.51: Distribution of the cut-line-following lateral error: *Backward Pass* on the left; and *Transition Pass* on the right.

Figure 8.51 validates the final geometric step of the two-pass strategy: turning a detected boundary into an actuator-offset reference.

8.5 Height-Map Products

The mission-level height map is the by-product of normal operation (Chapter 6). It is not treated as dense absolute height validation, since centimetre-level ground truth for every cell would require a dedicated survey; instead it verifies that the system accumulates corridor-floor vegetation in the map frame while excluding the lateral rows through the perception-derived polygon. The maps are shown for the transition (*Transition Pass*) family in Figure 8.52, one per corridor shape; the

Transition Pass worlds are chosen because they are the most dynamic, containing both a no-cut and a cut region within the same mission.

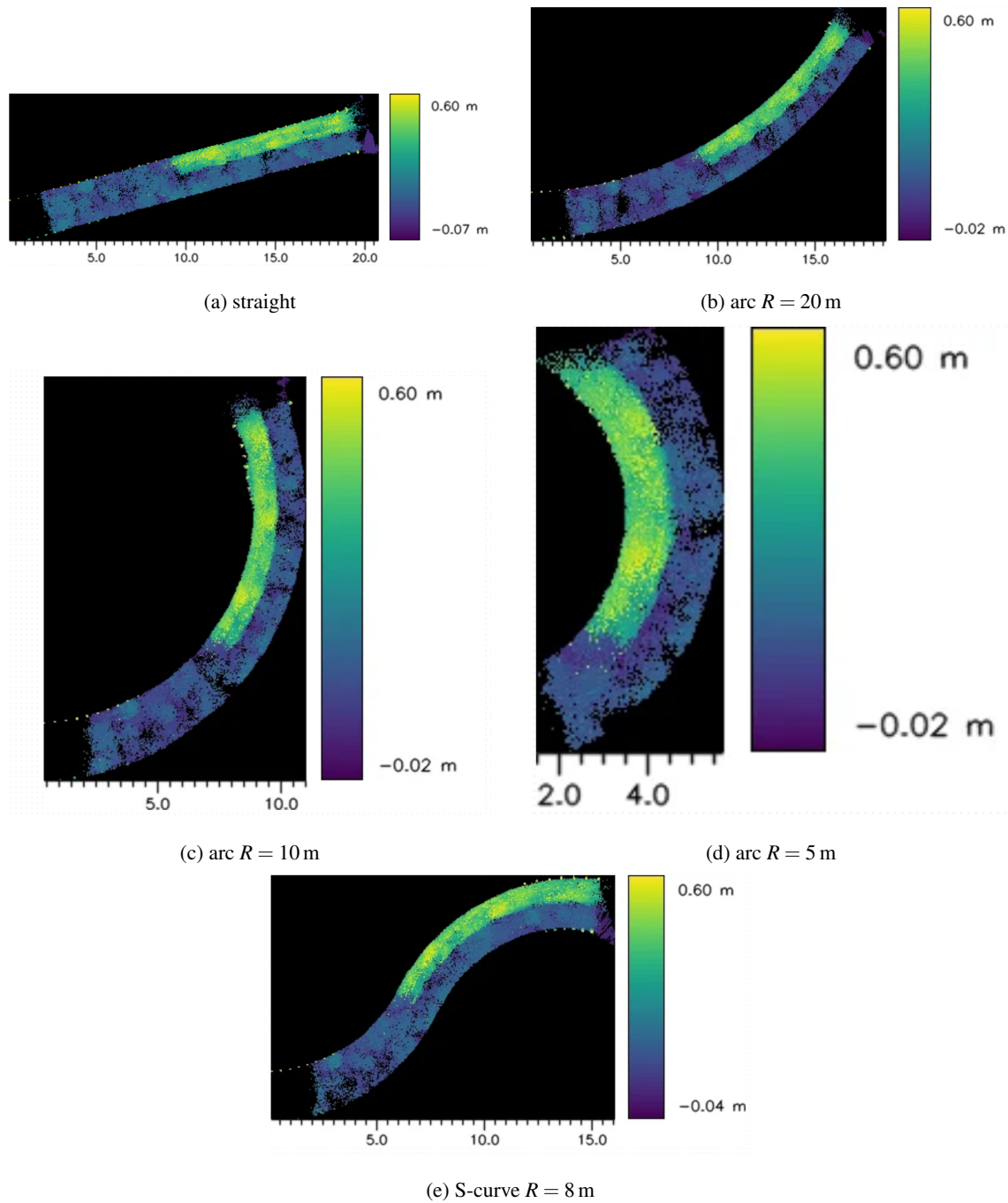


Figure 8.52: Mission-level height maps for the transition (TP) family, one per corridor shape: (a) straight, (b)–(d) the three arcs, and (e) the S-curve.

Figure 8.52 shows the accumulated corridor-floor vegetation remaining spatially coherent across the shapes, with the lateral rows excluded by the mask.

8.6 Discussion

8.6.1 Synthesis

The central engineering question posed in Chapter 1 is whether a purely LiDAR-based perception–mapping–planning stack can reliably localise crop rows, detect the mown–unmown vegetation boundary, and convert both geometric signals into smooth local references suitable for autonomous inter-row mowing, without relying on GPS, pre-built maps, or multi-modal sensor fusion. The evidence presented in Sections 8.2–8.5 supports an affirmative answer within the scope of the validated scenarios, and three properties of the aggregate results deserve interpretive emphasis beyond their individual numerical values.

First, the Dual-ROI row detector maintains a coherent geometric output across five corridor morphologies spanning straight geometry to a tight $R = 5$ m stress arc. Lateral RMSE rises predictably from approximately 8 cm on the straight and $R = 20$ m worlds to 15 cm on the $R = 5$ m case (Table 8.2), and heading RMSE from roughly 4° to 6.5° (Table 8.3). Both increases are geometrically explicable: as the inter-row corridor curves more sharply, the two crop rows cease to be locally parallel within any finite observation window, and the RANSAC line fit progressively estimates a secant of the curved row rather than its tangent. Crucially, RANSAC inlier ratios remain above 0.80 across all shapes (Table 8.4), and the fraction of sweeps exceeding the acceptance threshold stays above 95 %, confirming that accuracy and geometric support degrade together and in a manner that the downstream acceptance logic can interpret correctly. The consistency of this behaviour across morphologically diverse corridors demonstrates that the method’s dependence on vegetation geometry is stable under the curvature range representative of real vineyard inter-row sections.

Second, the cut-line detector records a false-positive rate of 0.0% across every world in which no mown boundary is present, comprising both the forward-pass and gap-test families, five corridor shapes each (Table 8.14). A false positive at this stage would cause the planner to commit prematurely to cut-line-following mode, displacing the generated reference towards a non-existent boundary and directing the cutter into standing vegetation. The demonstrated abstention behaviour is therefore a safety-critical architectural property: it confirms that the acceptance-gate cascade of Chapter 6 actively suppresses responses to any point-cloud discontinuity that does not simultaneously satisfy the combined geometric, height, and spatial constraints imposed by the gate. The internal WLS fit residual remaining at the millimetre-to-centimetre level for all accepted detections (Table 8.16) further confirms that accepted lines are geometrically well-founded, not merely coincidentally passing a loose threshold.

Third, the one-sided-loss robustness result (Table 8.21) closes the loop between the perception and planning layers. When the right row is physically absent over the induced $s \in [7, 13]$ m gap, the right-side valid-fit rate falls truthfully to 53–70 % (Table 8.6), and the planner responds by holding a valid reference on the surviving left row throughout the gap in every tested shape. The hysteretic per-side safety logic of Chapter 7 thus provides the intended fallback mechanism: a partial failure in one perception channel does not halt the mission and does not silently corrupt the reference. The

real-vineyard rosbag of Section 8.1.1 corroborates the internal-consistency indicators assessable without surveyed ground truth, namely the RANSAC inlier ratio (Table 8.5) and the inter-row width (Table 8.9), and both hold within the same range as their simulated counterparts under genuine sensor noise and canopy irregularity, with per-cycle timing budgets similarly respected (Table 8.10).

8.6.2 Comparison with the state of the art

Situating these results within the published literature requires explicit acknowledgement of the methodological divide that separates virtually all prior work from the present evaluation. The thesis metrics are obtained in a controlled simulation environment with analytic ground truth and ideal odometry, whereas the literature predominantly reports figures from closed-loop field trials in which sensor noise, terrain deformation, localisation drift, and dynamic occlusion from wind and variable illumination all contribute jointly to the measured deviation. Table 8.26 summarises the most directly comparable published figures alongside the corresponding values obtained in this thesis; the methodological differences that condition each comparison are discussed below.

Table 8.26: Quantitative comparison between thesis results and selected state-of-the-art references. Thesis figures are open-loop reference errors against analytic ground truth in Webots simulation. Literature figures are from real-field closed-loop deployments unless otherwise stated.

Domain	Reference	Sensor	Metric	Literature	Thesis
Row detection	Shi et al. (2025) [45]	Vision + 2-D LiDAR	Lateral error RMSE	6.6 cm	7.8–15.5 cm
			LiDAR-only baseline RMSE	9.0 cm	–
			Navigation success	95.4 %	–
Row tracking	Liu et al. (2023) [28]	RGB-D camera	Positional deviation, mean	17–22 cm	6.3–10.3 cm [†]
			Heading deviation, mean	1.7–1.9°	2.3–5.0° [†]
Mowing boundary	Fukukawa et al. (2016) [17]	Monocular camera	Positional error, std	4.6 cm [‡]	4.9–6.9 cm
			Angular error, std	1.8° [‡]	3.5–5.1°
			False-positive rate	Not reported	0.0 %

[†] Thesis values are open-loop reference errors against analytic ground truth and are not directly equivalent to the closed-loop platform deviations reported by Liu et al.

[‡] Fukukawa et al. evaluated 12 static images at three fixed lawn locations; dynamic corridor traversal was not performed.

The row-detection lateral RMSE of 7.8–15.5 cm is larger than the 6.6 cm lateral RMSE reported by Shi et al. [45] with a combined vision–LiDAR fusion system in a real pomegranate orchard. Two structural differences modulate this comparison. First, the Shi et al. system augments

2-D LiDAR trunk clusters with RGB bounding boxes from a YOLOv8-based detector, providing complementary spatial cues, namely texture, colour, and silhouette, that a single 3-D LiDAR cannot replicate by design; the present method is explicitly single-sensor. Second, their figure measures the deviation of the robot's physical trajectory from a surveyed centreline, integrating sensor noise, controller lag, and terrain slip in a closed loop, whereas the thesis lateral RMSE measures only the geometric error of the RANSAC-fitted row lines against the analytic ground-truth rows, isolating the perception stage from all subsequent platform dynamics. The single-modality 2-D LiDAR RANSAC comparator reported by Shi et al. achieves a lateral error RMS of 9.0 cm under the same real-field conditions, a figure that brackets the thesis values from above in a comparable single-sensor configuration and suggests that the additional RMSE is attributable in part to operating without visual cues rather than to a limitation of the RANSAC line-fitting strategy itself.

The comparison with Liu et al. [28] is structurally analogous. Their reported positional deviation of 17–22 cm for closed-loop row tracking in a real vineyard exceeds the thesis row-following trajectory RMSE of 6.3–10.3 cm on the three nominal shapes, whilst their heading deviation of 1.7–1.9° is smaller than the thesis heading RMSE of 2.3–5.0°. The former discrepancy reflects additional position error accumulated by a physical platform undergoing wheel slip, terrain gradient, and localisation drift on top of the reference-generation error characterised here. The latter inversion arises because the BEV-based controller of Liu et al. imposes a strong heading-alignment constraint that the platform continuously corrects in a closed loop, whereas the thesis heading RMSE includes the angular component of the Bézier look-ahead geometry during the initial transient before the corridor is geometrically well-characterised. The data suggest that, by isolating the perception-to-reference-generation component from closed-loop platform dynamics, the Dual-ROI pipeline produces geometrically competitive output relative to vision-based approaches that rely on richer, more complex sensor stacks. A further distinction that the numerical comparison does not capture is the morphological scope of the evaluation: the thesis provides a systematic account of performance across five corridor shapes, whereas field evaluations of row tracking are rarely replicated across a comparable range of curvatures within a single publication.

For the mowing boundary specifically, Fukukawa et al. [17]—the closest published comparator in task definition—report a positional error standard deviation of 4.6 cm from a monocular camera applied to static lawn images using Gabor filtering and RANSAC. The thesis cut-line lateral standard deviation on the backward-pass worlds is 4.9–6.9 cm across the three nominal shapes (Table 8.15), a comparable range achieved by a fundamentally different sensing principle—a 3-D point-cloud height step rather than a 2-D texture discontinuity. The scope of the two evaluations is, however, incommensurable: the Fukukawa et al. results derive from twelve static images at three fixed locations with no dynamic traversal, whereas the thesis figures are computed over hundreds of frames covering complete corridor traversals under varying curvature and both full and partial cut-line configurations. The numerical proximity should therefore be understood as an order-of-magnitude consistency check, and the absence of an equivalent false-positive characterisation in the Fukukawa et al. study prevents a fuller comparison of the detection selectivity of the two

approaches.

The thesis validation consequently occupies a methodologically distinct but complementary position in the literature. It provides a complete systematic characterisation of a cascade-gated LiDAR height-step boundary detector across five corridor geometries and four task families, with a formally documented false-positive rate of zero.

8.6.3 Limitations and critical scenarios

The results reveal two categories of performance degradation that reflect structured limitations of the current implementation rather than stochastic measurement noise, and both are anticipated by the geometric assumptions underlying the algorithm.

The first is curvature-induced degradation in the perception layer. At $R = 5$ m—a deliberate stress test retained to establish an upper bound on the degradation, and not representative of the nominal inter-row geometry of common vineyards—the Dual-ROI lateral RMSE rises to 0.155 m and the heading RMSE to 6.5° . The mechanism is the same one that produces the apparent width collapse in Table 8.7: as the corridor curves sharply, RANSAC fits a chord across the curved row within the fixed observation window, and the chord diverges from the row tangent in proportion to the local curvature. This error propagates into the planning layer, raising the row-following trajectory RMSE to 10.3 cm lateral and 8.7° heading at the same shape (Tables 8.22 and 8.23), and the cut-line-following lateral RMSE to 42.1 cm on the backward-pass variant (Table 8.24), where the short valid detection window produces a sparse and geometrically distorted set of cut-line observations on which to anchor the Bézier reference. These stress-test figures establish the curvature boundary at which the straight-line fitting assumption breaks down; an adaptive ROI whose angular extent contracts as estimated corridor curvature increases would mitigate the degradation at the cost of higher parametric complexity.

The second category is transition-phase instability in the mapping layer. On the transition-pass worlds at $R = 10$ m and $R = 20$ m, the lateral standard deviation of the cut-line external error reaches 0.47–0.83 m (Table 8.15), driven by a small number of frames near the no-cut/cut boundary in which the look-ahead window straddles the transition. In those frames the height-step gradient is spatially ambiguous—the window contains only a partial step—and the acceptance gate either rejects the detection (a false negative) or accepts a line whose lateral anchor is displaced from the true boundary. Critically, this is not a failure of the internal fitting: the WLS residual remains at the centimetre level for all accepted frames (Table 8.16). It is a finite-window boundary artefact. The practical implication is that the first valid detections immediately after a no-cut/cut transition carry reduced reliability, and a minimum-dwell or confidence-accumulation mechanism in the planner’s state machine could delay commitment to cut-line-following mode until a sufficient number of consistent detections have been received.

8.6.4 Technological and scientific implications

From a system-architecture standpoint, the validation demonstrates that a complete pipeline from raw 3-D point cloud to smooth local reference trajectory can be executed on commodity hardware within the on-line timing budget at every stage (Tables 8.8, 8.12, 8.18, and 8.20), without GPU acceleration or multi-sensor time-alignment pipelines. The key architectural choice enabling this is the early reduction of the raw point cloud to a height-gated, spatially constrained corridor-floor subset through the perception-derived corridor polygon mask (Table 8.11). This step simultaneously bounds the computational load and eliminates structural clutter—vine posts, trellis wires, and canopy returns—that would otherwise generate spurious height-step responses in the mapping layer. The cascade-gated detection chain then operates on a geometrically filtered, semantically constrained point set, and it is this progressive filtering, rather than any single algorithmic novelty, that is primarily responsible for the zero false-positive rate documented in Section 8.3.2. The pipeline’s reliance on 3-D point-cloud geometry rather than colour, texture, or semantic classification further implies that its core detection logic is structurally agnostic to illumination conditions—an advantage over vision-based boundary detectors under the adverse lighting scenarios reported in the vineyard navigation literature [1, 28, 45].

From a scientific standpoint, the work provides a characterisation of the zero-false-positive property of a cascade-gated LiDAR height-step boundary detector across five corridor geometries and four task families in a vineyard mowing context. The result carries a direct agronomic interpretation: a system that does not hallucinate a mowing boundary under uniform vegetation will not prematurely direct the cutter into standing crop, eliminating one of the principal failure modes of autonomous mowing systems. Demonstrating this property under ideal simulation conditions is a necessary precondition for field deployment, not a sufficient one; it establishes that the detection logic is architecturally sound before the additional variability of real canopy, terrain perturbation, and long-duration sensor drift are introduced. The chapter should therefore be read as a controlled validation of the algorithm interfaces at the reference-generation level, corroborated on one real-vineyard run, and not as a final agronomic field trial.

The height-map accumulated during normal operation (Section 8.5) illustrates a secondary benefit of the architecture: mission-level vegetation-height information is produced without any additional sensing or processing beyond what the detection pipeline already performs. In the context of precision viticulture, such a record could support post-mission assessments of canopy-height uniformity or sward growth rate, informing inter-operation decisions such as differential growth-regulator application. Its practical utility would require validation against reference measurements of the kind reported by Jimenez-Berni et al. [21] and Obanawa et al. [40], and constitutes a natural direction for future work alongside longer-duration field trials.

In summary, the developed stack fulfils its design objective at the reference-generation level: it detects and tracks crop rows, constructs the inter-row corridor, maps the corridor-floor vegetation, detects the mown–unmown boundary, and converts both row and boundary geometry into G^1 -continuous local references, whilst degrading safely when a row is lost. The principal engineering

result is the zero-false-positive rate of the cut-line detector, which ensures that the planner receives a trustworthy binary signal and cannot be driven into unsafe mode transitions by spurious height-step responses. The principal open question is the transfer from analytic simulation to continuous field operation under real canopy variability, terrain perturbation, and long-duration sensor drift—for which the one real-vineyard run of Section 8.1.1 provides encouraging but necessarily limited evidence. Addressing that gap through instrumented field trials constitutes the primary next step towards a field-deployable autonomous inter-row mowing system. The tracking controller and large-scale coverage optimisation remain outside the claims established here.

Chapter 9

Conclusions and Future Work

This dissertation investigated the perception, mapping, and local planning components required for a LiDAR-based inter-row mowing robot to navigate autonomously and perform vegetation-aware maintenance in vineyard corridors. The nine functional requirements stated in Section 2.4 served as the engineering criteria throughout the work. This chapter summarises the degree to which each requirement was met (Section 9.1), and proposes concrete directions for extending the work (Section 9.2).

9.1 Main Conclusions

This work demonstrates that the perception, mapping, and local planning requirements set out in Section 2.4 can be addressed by a coherent LiDAR-only stack operating in real time. The Dual ROI model recovers crop-row geometry at centimetric accuracy across a range of corridor shapes and curvatures, providing the spatial reference for autonomous inter-row navigation (R2, R6). The same LiDAR data feeds a height-mapping module that characterises inter-row vegetation at both mission and local scale, distinguishing the corridor floor from the lateral rows and accumulating a spatial record of canopy height suitable for agronomic use (R1). From the local height map, the cut-line cascade detects the boundary between the already-mown swath and the adjacent standing vegetation without reliance on camera imagery, departing from the dominant approach in the precision-mowing literature and offering inherent robustness to illumination variability (R4, R7). A Bézier-spline reference generator unifies these outputs into continuous, safe trajectories for both row-following and cut-line-following modes, degrades gracefully under single-sided row loss, and composes the two modes into a single vegetation-aware two-pass mission (R5, R6, R8, R9). Dynamic cutting-height regulation (R3) falls outside the scope of this dissertation; the height estimates and coverage maps produced here provide the substrate for such a controller as future work.

9.2 Future Work

The results and limitations identified above point to several concrete directions for extending the system.

Field validation in operational vineyards. The most immediate priority is deployment in a real vineyard carrying out the full two-pass mission. Beyond confirming algorithmic correctness, such a campaign would expose the system to the soil, canopy, lighting, and localisation variability that simulation cannot replicate. An initial per-vineyard tuning phase — adjusting crop box dimensions and the initial inter-row width estimate — is expected to be brief, after which the Dual ROI model's on-line adaptation should sustain reliable operation without further manual intervention.

Curvature-continuous reference generation. The Bézier planner guarantees position and tangent continuity (G^1) at join points between successive planning cycles. Full curvature continuity (G^2) was not pursued because the evidence shows that the strategy of relaxing the look-ahead gain on the initial segment of each cycle is sufficient to keep the reference smooth enough for the current controller. Nonetheless, curvature discontinuities impose a transient demand on the tracking controller at every cycle boundary. Extending the planner to enforce G^2 continuity — for instance, by matching the curvature derivative at join points through a higher order spline — would reduce this transient and further smooth the executed trajectory.

Advanced trajectory tracking controller. The PID-plus-feedforward controller used in field and laboratory runs operates safely in the range of $0.3\text{--}0.5\text{ m s}^{-1}$. Higher traversal speeds would directly improve field productivity but require a controller that can anticipate the look-ahead curvature of the reference and enforce lateral error bounds under the resulting centrifugal loading. A model predictive controller (MPC) that internalises the reference curvature and explicitly constrains lateral deviation is a natural candidate and would also provide a principled framework for incorporating actuator and terrain constraints. This development would additionally satisfy requirement R3 if extended to include real-time cutting-height regulation.

Independent validation of mission height maps. The mission-level height maps are evaluated qualitatively in this work. A quantitative accuracy study using an independent reference sensor — such as a structured-light scanner or a stereo camera mounted on the same platform — would permit height error characterisation in real conditions and establish the maps as a calibrated agronomic product rather than a qualitative planning aid.

Actuator-state-aware coverage mapping. The current height map records vegetation height at the moment of the robot's passage regardless of whether the cutting actuator was engaged. Extending the map with a binary actuator-engagement layer would distinguish areas that were actively mown from areas that were merely traversed. Post-mission, this layer could be fused with the height-before-cut estimate to produce a spatial record of the resulting canopy height, enabling agronomic reporting of covered area, achieved cut height, and residual untreated zones. Such a product would close the loop between the vegetation-aware planning demonstrated here and practical vineyard management requirements.

References

- [1] Diego Aghi, Vittorio Mazzia, and Marcello Chiaberge. “Local Motion Planner for Autonomous Navigation in Vineyards with a RGB-D Camera-Based Algorithm and Deep Learning Synergy”. In: *Machines* 8.2 (2020), p. 27. DOI: [10.3390/machines8020027](https://doi.org/10.3390/machines8020027).
- [2] Dionisio Andújar et al. “Discriminating crop, weeds and soil surface with a terrestrial LiDAR sensor”. In: *Sensors* 13.11 (2013), pp. 14662–14675. DOI: [10.3390/s131114662](https://doi.org/10.3390/s131114662).
- [3] Katherine Aro et al. “A Nonlinear Model Predictive Controller for Trajectory Planning of Skid-Steer Mobile Robots in Agricultural Environments”. In: *Proc. IEEE Conf. AgriFood Electronics (CAFE)*. 2023, pp. 65–69. DOI: [10.1109/CAFE58535.2023.10291643](https://doi.org/10.1109/CAFE58535.2023.10291643).
- [4] Mostafa Rahimi Azghadi et al. “Precision Robotic Spot-Spraying: Reducing Herbicide Use and Enhancing Environmental Outcomes in Sugarcane”. In: *Computers and Electronics in Agriculture* 235 (2025), p. 110365. DOI: [10.1016/j.compag.2025.110365](https://doi.org/10.1016/j.compag.2025.110365).
- [5] Tijmen Bakker et al. “Systematic design of an autonomous platform for robotic weeding”. In: *Journal of Terramechanics* 47.2 (2010), pp. 63–73. DOI: [10.1016/j.jterra.2009.06.002](https://doi.org/10.1016/j.jterra.2009.06.002).
- [6] Eric R. Benson, John F. Reid, and Qin Zhang. “Machine vision-based guidance system for agricultural grain harvesters using cut-edge detection”. In: *Biosystems Engineering* 86.4 (2003), pp. 389–398. DOI: [10.1016/j.biosystemseng.2003.07.002](https://doi.org/10.1016/j.biosystemseng.2003.07.002).
- [7] Riccardo Bertoglio et al. “A Map-Free LiDAR-Based System for Autonomous Navigation in Vineyards”. In: *Proc. European Conf. Mobile Robots (ECMR)*. 2023. DOI: [10.1109/ECMR59166.2023.10256347](https://doi.org/10.1109/ECMR59166.2023.10256347).
- [8] Alessandro Biglia et al. “3D point cloud density-based segmentation for vine rows detection and localisation”. In: *Computers and Electronics in Agriculture* 199 (2022), p. 107166. DOI: [10.1016/j.compag.2022.107166](https://doi.org/10.1016/j.compag.2022.107166).
- [9] David Brás, André Rodrigues Baltazar, and André Silva Pinto de Aguiar. “[TITLE PLACEHOLDER — LiDAR-Based Vegetation Sensing and Autonomous Navigation for Inter-Row Mowing in Vineyards]”. In: *Proc. 13th Iberian Robotics Conference (ROBOT)*. Submitted. 2026.
- [10] Chao Cheng et al. “Recent Advancements in Agriculture Robots: Benefits and Challenges”. In: *Machines* 11.1 (2023), p. 48. DOI: [10.3390/machines11010048](https://doi.org/10.3390/machines11010048).
- [11] Wonjae Cho et al. “Using multiple sensors to detect uncut crop edges for autonomous guidance systems of head-feeding combine harvesters”. In: *Engineering in Agriculture, Environment and Food* 7.3 (2014), pp. 115–121.
- [12] David C. Conner and Justin Willis. “Flexible Navigation: Finite state machine-based integrated navigation and control for ROS enabled robots”. In: *Proc. SoutheastCon 2017*. 2017, pp. 1–8. DOI: [10.1109/SECON.2017.7925266](https://doi.org/10.1109/SECON.2017.7925266).

- [13] Stefani Daryanto et al. “Quantitative synthesis on the ecosystem services of cover crops”. In: *Earth-Science Reviews* 185 (2018), pp. 357–373. DOI: [10.1016/j.earscirev.2018.06.013](https://doi.org/10.1016/j.earscirev.2018.06.013).
- [14] Stepan Dergachev, Kirill Muravyev, and Konstantin Yakovlev. *2.5D Mapping, Pathfinding and Path Following for Navigation of a Differential Drive Robot in Uneven Terrain*. arXiv:2209.07252. 2022. DOI: [10.48550/arXiv.2209.07252](https://doi.org/10.48550/arXiv.2209.07252).
- [15] Mohamed Elbanhawi, Milan Simic, and Reza Jazar. “Continuous Path Smoothing for Car-Like Robots Using B-Spline Curves”. In: *Journal of Intelligent & Robotic Systems* 80.S1 (2015), pp. 23–56. DOI: [10.1007/s10846-014-0172-0](https://doi.org/10.1007/s10846-014-0172-0).
- [16] Péter Fankhauser and Marco Hutter. “A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation”. In: *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Ed. by Anis Koubaa. Cham, Switzerland: Springer, 2016, pp. 99–120. DOI: [10.1007/978-3-319-26054-9_5](https://doi.org/10.1007/978-3-319-26054-9_5).
- [17] Tatsuya Fukukawa et al. “Vision-Based Mowing Boundary Detection Algorithm for an Autonomous Lawn Mower”. In: *Journal of Advanced Computational Intelligence and Intelligent Informatics* 20.1 (2016), pp. 49–56. DOI: [10.20965/jaciii.2016.p0049](https://doi.org/10.20965/jaciii.2016.p0049).
- [18] Vitor A. H. Higuti et al. “Under canopy light detection and ranging-based autonomous navigation”. In: *Journal of Field Robotics* 36.3 (2019), pp. 547–567. DOI: [10.1002/rob.21852](https://doi.org/10.1002/rob.21852).
- [19] Matteo Iovino et al. “Comparison Between Behavior Trees and Finite State Machines”. In: *IEEE Transactions on Automation Science and Engineering* (2024). arXiv:2405.16137.
- [20] Nuwan Jayasuriya et al. “Machine vision based plant height estimation for protected crop facilities”. In: *Computers and Electronics in Agriculture* 218 (2024), p. 108669. DOI: [10.1016/j.compag.2024.108669](https://doi.org/10.1016/j.compag.2024.108669).
- [21] Jose A. Jimenez-Berni et al. “High Throughput Determination of Plant Height, Ground Cover, and Above-Ground Biomass in Wheat with LiDAR”. In: *Frontiers in Plant Science* 9 (2018), p. 237. DOI: [10.3389/fpls.2018.00237](https://doi.org/10.3389/fpls.2018.00237).
- [22] J. Johnson et al. *Robotic Multimodal Data Acquisition for In-Field Deep Learning Estimation of Cover Crop Biomass*. arXiv:2506.22364. 2025. DOI: [10.48550/arXiv.2506.22364](https://doi.org/10.48550/arXiv.2506.22364).
- [23] Dimitrios Kateris et al. “Weed Mapping in Vineyards Using RGB-D Perception”. In: *Engineering Proceedings* 9 (2021), p. 30. DOI: [10.3390/engproc2021009030](https://doi.org/10.3390/engproc2021009030).
- [24] Hannah E. Keen and Karsten Berns. “Generation of Elevation Maps for Planning and Navigation of Vehicles in Rough Natural Terrain”. In: *Advances in Service and Industrial Robotics (RAAD 2019)*. Vol. 980. Advances in Intelligent Systems and Computing. Cham, Switzerland: Springer, 2019, pp. 476–483. DOI: [10.1007/978-3-030-19648-6_56](https://doi.org/10.1007/978-3-030-19648-6_56).
- [25] Wan-Soo Kim et al. “Weakly Supervised Crop Area Segmentation for an Autonomous Combine Harvester”. In: *Sensors* 21.14 (2021), p. 4801. DOI: [10.3390/s21144801](https://doi.org/10.3390/s21144801).
- [26] Johannes Kneip, Patrick Fleischmann, and Karsten Berns. “Crop Edge Detection Based on Stereo Vision”. In: *Intelligent Autonomous Systems 15 (IAS-15)*. 2019, pp. 639–651. DOI: [10.1007/978-3-030-01370-7_50](https://doi.org/10.1007/978-3-030-01370-7_50).
- [27] Lin Li et al. “An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells”. In: *Remote Sensing* 9.5 (2017), p. 433. DOI: [10.3390/rs9050433](https://doi.org/10.3390/rs9050433).

- [28] Ertai Liu et al. “Vision-Based Vineyard Navigation Solution with Automatic Annotation”. In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*. 2023, pp. 4234–4241. DOI: [10.1109/IROS55552.2023.10341261](https://doi.org/10.1109/IROS55552.2023.10341261).
- [29] Ruiji Liu, Francisco Yandun, and George Kantor. “Towards Over-Canopy Autonomous Navigation: Crop-Agnostic LiDAR-Based Crop-Row Detection in Arable Fields”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. arXiv:2403.17774. 2025.
- [30] Xu Liu et al. “High-Fidelity and Curvature-Continuous Path Smoothing With Quadratic Bézier Curve”. In: *IEEE Transactions on Intelligent Vehicles* (2024).
- [31] Yusen Luo et al. “Stereo-vision-based multi-crop harvesting edge detection for precise automatic steering of combine harvester”. In: *Biosystems Engineering* 215 (2022), pp. 115–128.
- [32] Zhen Ma et al. “Advances in Crop Row Detection for Agricultural Robots: Methods, Performance Indicators, and Scene Adaptability”. In: *Agriculture* 15.20 (2025), p. 2151. DOI: [10.3390/agriculture15202151](https://doi.org/10.3390/agriculture15202151).
- [33] Parikshit Maini, Burak M. Gonultas, and Volkan Isler. “Online Coverage Planning for an Autonomous Weed Mowing Robot With Curvature Constraints”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5445–5452. DOI: [10.1109/LRA.2022.3154006](https://doi.org/10.1109/LRA.2022.3154006).
- [34] Flavio B. P. Malavazi et al. “LiDAR-only based navigation algorithm for an autonomous agricultural robot”. In: *Computers and Electronics in Agriculture* 154 (2018), pp. 71–79. DOI: [10.1016/j.compag.2018.08.034](https://doi.org/10.1016/j.compag.2018.08.034).
- [35] Chrysanthos Maraveas. “Image Analysis Artificial Intelligence Technologies for Plant Phenotyping: Current State of the Art”. In: *AgriEngineering* 6.3 (2024), pp. 3375–3407. DOI: [10.3390/agriengineering6030193](https://doi.org/10.3390/agriengineering6030193).
- [36] Olivier Michel. “Webots: Professional Mobile Robot Simulation”. In: *International Journal of Advanced Robotic Systems* 1.1 (2004), pp. 39–42.
- [37] Takahiro Miki et al. “Elevation Mapping for Locomotion and Navigation using GPU”. In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*. 2022, pp. 2273–2280. DOI: [10.1109/IROS47612.2022.9981507](https://doi.org/10.1109/IROS47612.2022.9981507).
- [38] Alessandro Navone, Mauro Martini, and Marcello Chiaberge. “Autonomous robotic pruning in orchards and vineyards: A review”. In: *Smart Agricultural Technology* 12 (2025), p. 101283. DOI: [10.1016/j.atech.2025.101283](https://doi.org/10.1016/j.atech.2025.101283).
- [39] Hassan Nehme et al. “LiDAR-based structure tracking for agricultural robots: Application to autonomous navigation in vineyards”. In: *Journal of Intelligent & Robotic Systems* 103.4 (2021), p. 61. DOI: [10.1007/s10846-021-01519-7](https://doi.org/10.1007/s10846-021-01519-7).
- [40] Hiroyuki Obanawa et al. “Portable LiDAR-Based Method for Improvement of Grass Height Measurement Accuracy: Comparison with SfM Methods”. In: *Sensors* 20.17 (2020), p. 4809. DOI: [10.3390/s20174809](https://doi.org/10.3390/s20174809).
- [41] Srđan Popić et al. “Optimized autonomous navigation for field robots: Extended results and practical deployment”. In: *Precision Agriculture* 27.1 (2026), p. 5. DOI: [10.1007/s11119-025-10303-8](https://doi.org/10.1007/s11119-025-10303-8).
- [42] David Reiser et al. “Development of an Autonomous Electric Robot Implement for Intra-Row Weeding in Vineyards”. In: *Agriculture* 9.1 (2019), p. 18. DOI: [10.3390/agriculture9010018](https://doi.org/10.3390/agriculture9010018).

- [43] K. A. C. Ruzlan et al. “Weed control efficiency of unmanned aerial vehicle spray in replanting oil palm plantation areas”. In: *Weed Science* 73 (2025), e37. DOI: [10.1017/wsc.2024.91](https://doi.org/10.1017/wsc.2024.91).
- [44] Alexander Schepelmann et al. “Visual segmentation of lawn grass for a mobile robotic lawnmower”. In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*. 2010, pp. 734–739. DOI: [10.1109/IROS.2010.5650430](https://doi.org/10.1109/IROS.2010.5650430).
- [45] Zhikang Shi et al. “Vision and 2D LiDAR Fusion-Based Navigation Line Extraction for Autonomous Agricultural Robots in Dense Pomegranate Orchards”. In: *Sensors* 25.17 (2025), p. 5432. DOI: [10.3390/s25175432](https://doi.org/10.3390/s25175432).
- [46] Junya Shiraishi et al. “Efficient construction of 3D-Gaussian clusters and elevation maps for autonomous mobile robot navigation in non-planar environments”. In: *Advanced Robotics* 39.16 (2025), pp. 1026–1040. DOI: [10.1080/01691864.2025.2544167](https://doi.org/10.1080/01691864.2025.2544167).
- [47] Kenneth Renny Simba, Naoki Uchiyama, and Shigenori Sano. “Real-time smooth trajectory generation for nonholonomic mobile robots using Bézier curves”. In: *Robotics and Computer-Integrated Manufacturing* 41 (2016), pp. 31–42.
- [48] Matteo Sperti et al. *Non-linear Model Predictive Control for Multi-task GPS-free Autonomous Navigation in Vineyards*. arXiv:2404.05343. 2024. DOI: [10.48550/arXiv.2404.05343](https://doi.org/10.48550/arXiv.2404.05343).
- [49] Emmanouil Tziolas et al. “Assessing the Economic Performance of Multipurpose Collaborative Robots toward Skillful and Sustainable Viticultural Practices”. In: *Sustainability* 15.4 (2023), p. 3866. DOI: [10.3390/su15043866](https://doi.org/10.3390/su15043866).
- [50] Jian Wu et al. “An Improved Timed Elastic Band (TEB) Algorithm of Autonomous Ground Vehicle (AGV) in Complex Environments”. In: *Sensors* 21.24 (2021), p. 8312. DOI: [10.3390/s21248312](https://doi.org/10.3390/s21248312).
- [51] Yibo Zhou et al. “Rolling 2D Lidar-Based Navigation Line Extraction Method for Modern Orchard Automation”. In: *Agronomy* 15.4 (2025), p. 816. DOI: [10.3390/agronomy15040816](https://doi.org/10.3390/agronomy15040816).
- [52] Hongbo Zhu et al. “YOLOX-based blue laser weeding robot in corn field”. In: *Frontiers in Plant Science* 13 (2022), p. 1017803. DOI: [10.3389/fpls.2022.1017803](https://doi.org/10.3389/fpls.2022.1017803).
- [53] Kazuki Zushida et al. “Application and Analysis of Random Forest Algorithm for Estimating Lawn Grass Lengths in Robotic Lawn Mower”. In: *International Journal of Mechanical Engineering and Applications* 9.1 (2021), pp. 6–14. DOI: [10.11648/j.ijmea.20210901.12](https://doi.org/10.11648/j.ijmea.20210901.12).