

Data-Driven Agent-Based Modeling of Aircraft Surface Movement at a Large Airport

A case study for Amsterdam Schiphol Airport

Javier Crespo Núñez



Data-Driven Agent-Based Modeling of Aircraft Surface Movement at a Large Airport

A case study for Amsterdam Schiphol Airport

Thesis report

by

Javier Crespo Núñez

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on 23rd of July of 2026 at 9:30

Thesis committee:

Chair:	Paul Roling
Supervisors:	Alexei Sharpans'kykh Evert Westerveld
External examiner:	Barys Shyrokau
Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	November 2025 – July 2026
Student number:	6296696

An electronic version of this thesis will be made available through the appropriate TU Delft repository or course platform.

Acknowledgments

This thesis has been one of the most interesting and rewarding experiences of my academic journey. It gave me the opportunity to work on a topic that has fascinated me for a long time: the complexity, precision, and constant adaptation of airport operations. I have always been motivated by the way large airports function as living systems, where infrastructure, people, procedures, and uncertainty come together every day. Being able to study that world through data, models, and operational discussions has been a privilege.

I would first like to thank Jan for giving me the first opportunity to start exploring the wonderful world of ground operations at LVNL. That opportunity opened the door to a topic that quickly became much more than a first internship subject, and it shaped the direction of this work from the beginning.

I would also like to thank Marco for always being willing to show me the operation, discuss what was happening on the surface, and help me connect the data and models back to real airport practice. Those conversations were essential for keeping the work grounded in the operation itself, and they made the topic even more engaging.

I am deeply grateful to my family for their support throughout these two years. They were always there when I needed help, patience, time, or simply the right environment to continue. I especially want to thank Isabelle, whose support, encouragement, and example have inspired me to pursue research and opportunities that can create impact and contribute to a more sustainable future.

I also want to thank my fellow KDC and iLabs interns. We shared the pain, the confusion, the late evenings, the endless coffees, and the occasional dinners at the office. Those moments made the difficult parts lighter and turned the thesis process into something far more enjoyable.

I am also grateful to the other students who became interested in this research along the way, and to those who will use, continue, or contribute to it in the future. Knowing that this work can support new ideas, new questions, and future research directions is one of the most rewarding outcomes of the project.

Finally, I would like to thank my supervisors, Alexei, Malte, and Evert. Thank you for your guidance, patience, and support throughout the project. I am especially grateful for the many discussions, technical conversations, and operational reflections that shaped the work. Thank you also for bearing with my many terminology debates and for helping me turn a complex and evolving project into a coherent thesis.

I finish this thesis with gratitude for what I have learned and excitement for what comes next. The work has strengthened my interest in research, aviation, and the use of data-driven methods to better understand and improve complex operations. I hope that this thesis contributes, even in a small way, to the next steps toward more efficient, explainable, and sustainable airport surface operations.

Contents

Acknowledgments	ii
List of Figures	viii
List of Tables	ix
I General Introduction	1
1 Introduction	2
II Scientific Article	3
2 Data-Driven Agent-Based Modeling of Aircraft Surface Movement at a Large Airport	4
2.1 I. Introduction	4
2.2 II. Background	5
2.3 III. Methodology.	7
2.4 IV. Experimental Setup	18
2.5 V. Results	21
2.6 VI. Discussion	24
2.7 VII. Conclusion	27
2.8 References	28
III Literature Review & Research Definition	30
3 Literature Review	31
3.1 Airport Surface Movement Operations and Observed-behavior Modeling.	31
3.2 Surface-surveillance Data and Taxi-trajectory Extraction.	33
3.3 Airport Layout Graphs and Graph-constrained Reconstruction	36
3.4 Path Candidates, Canonical Path Families, and Operational Context	38
3.5 Data-driven Behavioral Model Identification.	40
3.6 Path-choice Modeling	42
3.7 Endpoint-selection Modeling.	44
3.8 Speed-profile Behavior Modeling	45
3.9 Local Conflict-resolution Modeling.	46
3.10 Deterministic Execution Rules and Integrated Model-validation Diagnostics	48
3.11 Evaluation Design, Metrics, and Diagnostic Reporting	49
4 Research Definition	52
4.1 Research Gap	52
4.2 Research Objective	53
4.3 Research Questions	53
4.4 Research Scope and Boundaries	54
4.5 Planning.	55
References	65

IV	Appendices	66
A	Model Calibration and Tuning	67
A.1	Purpose and Relation to the Scientific Paper	67
A.2	Training, Validation, and Test Partitioning	67
A.3	Context Feature Definitions	69
A.4	Context Feature Coverage and Dataset Profile	71
A.5	Bayesian Hyperparameter Optimization.	72
A.6	Training-Data Sensitivity Analysis	73
A.7	Calibration and Tuning Takeaways	75
B	Verification and Validation	76
B.1	Purpose and Relation to the Scientific Paper	76
B.2	Data Reconstruction and Map-Matching Validation.	76
B.3	Path Catalog and Canonical Path-Family Diagnostics	77
B.4	Validation Metrics and Diagnostic Definitions	79
B.5	Path-Choice Component Validation	80
B.6	Endpoint-Selection Component Validation	81
B.7	Speed-Profile Component Validation	83
B.8	Local Conflict-Resolution Component Validation	85
B.9	Integrated Model-Validation Scenario Diagnostics	85
B.10	Verification and Validation Takeaways.	88
C	Operational Interpretation of Selected Model Configurations	89
C.1	Purpose and Relation to the Scientific Paper	89
C.2	Selected Context Features by Component	89
C.3	Operational Interpretation by Learned Behavioral Component.	90
C.4	Prediction-source Support by Component	93
C.5	Context-conditioned Validation Insights	94
C.6	Missing Context and Operational Interpretation Limits	97
C.7	Operational Interpretation Takeaways	98
D	Case Study A: Construction-state Effects on Path Choice	99
D.1	Purpose and Relation to the Scientific Paper	99
D.2	Construction Scenario and Diagnostic Design	99
D.3	Observed Construction-period Path Change	99
D.4	August 2025 Path-choice Diagnostic	100
D.5	Held-out Week-35 Counterfactual	101
D.6	Construction-state Takeaways	102
E	Case Study B: Operational Constraints and Availability Rules	104
E.1	Purpose and Operational Question	104
E.2	Center-runway Active-state Path Procedures.	104
E.3	Canonical Path-family Availability under Procedure Constraints.	106
E.4	Runway 24 S7 Endpoint Restriction	107
E.5	Historical Schedule Versus Observed Runway Use	109
E.6	Operational Interpretation and Limitations	110
E.7	Operational-constraints Takeaways	110
F	Case Study C: Methodology Transfer to Apron-area Taxi Movements	111
F.1	Purpose and Transfer Question	111
F.2	Data Subset and Apron Scope.	111
F.3	Taxiing-phase Extraction	112
F.4	State-machine Map Matching	113
F.5	Path Catalog and Component-input Preparation	115
F.6	Speed-profile Preparation Limitations	116
F.7	Method-transfer Interpretation	117
F.8	Apron-area Transfer Takeaways.	117

Nomenclature

List of Abbreviations

A-CDM	Airport Collaborative Decision Making
A-SMGCS	Advanced Surface Movement Guidance and Control System
ADS-B	Automatic Dependent Surveillance–Broadcast
ASM Ops	Airport surface movement operations
ATC	Air traffic control
BO	Bayesian optimization
BZO	Schiphol operational phase variable used in the context data
CART	Classification and Regression Tree
ECE	Expected calibration error
EHAM	ICAO airport code for Amsterdam Airport Schiphol
ENAC	École Nationale de l’Aviation Civile
ETA	Estimated time of arrival
F1	F1 score; harmonic mean of precision and recall
ISO	International Organization for Standardization
KLM	KLM Royal Dutch Airlines
kt	Knot; unit of speed
LVNL	Luchtverkeersleiding Nederland; Air Traffic Control the Netherlands
MAE	Mean absolute error
MAS	Multi-agent system
OD	Origin–Destination
P10/P90	10th and 90th percentiles
RECAT-EU	European wake-turbulence recategorization

RMSE Root-mean-square error

RWY Runway

SID Standard Instrument Departure

SVG Scalable Vector Graphics

VEMMIS Internal LVNL air traffic management and surveillance database used for Schiphol operational records

List of Symbols

$\bar{\varepsilon}_i$	Mean point-to-path projection error for taxi movement i
$\bar{v}_e(\xi_m)$	Edge-level baseline speed profile for directed edge e
$\chi_{i,j,r}$	Conflict-event indicator for aircraft pair (i, j) at protected taxi resource r
$\Delta_{j,i}^{\text{dep}}$	Departure separation interval between predecessor j and aircraft i
$\Delta_{j,i}^{\text{SID}}$	Same-SID departure-separation add-on between predecessor j and aircraft i
$\Delta_{j,i}^{\text{wake}}$	RECAT-EU wake-turbulence departure minimum between predecessor j and aircraft i
$\hat{\mathbf{p}}_{i,k}$	Projected position of trajectory point $q_{i,k}$ on the airport layout graph
$\hat{\pi}_c^{\text{CART}}(y \mathbf{x})$	CART-stage class-probability estimator for component c
$\hat{\pi}_c^{\text{emp}}(y \mathbf{x})$	Empirical-stage class-probability estimator for component c
$\hat{\pi}_c(y \mathbf{x})$	Final class-probability distribution returned by the three-stage inference procedure
$\hat{r}_{i,e}(\xi_m)$	Predicted speed-profile residual for taxi movement i on directed edge e
$\hat{v}_i(s)$	Path-level target speed profile for taxi movement i
$\hat{v}_{i,e}(\xi_m)$	Predicted edge-relative speed profile for taxi movement i on directed edge e

$\mathbf{p}_{i,k}$	Observed surface position of trajectory point $q_{i,k}$	$\xi_{i,k}$	Edge-relative progress of trajectory point $q_{i,k}$
$\mathbf{x}_{c,i}$	Feature vector for learned behavioral component c and observation i	A^*	A-star directed graph-search algorithm
$\mathcal{C}_o$	Path-choice set for OD pair o	$A_c^{\text{CART}}(\mathbf{x})$	CART-stage acceptance indicator for component c
$\mathcal{D}_{c,S_c}^{\text{emp}}(\mathbf{x})$	Empirical context cell for component c , feature set S_c , and query $\mathbf{x}$	$A_c^{\text{emp}}(\mathbf{x})$	Empirical-stage acceptance indicator for component c
$\mathcal{D}_c^{\text{train}}$	Training-observation set for learned behavioral component c	c	Learned behavioral-component index
$\mathcal{E}_\tau^{\text{ep}}$	Valid endpoint set for endpoint-selection target τ	$c_i(e)$	Edge-relative coverage for directed edge e
$\mathcal{L}_c(\mathbf{x})$	CART leaf set reached by query $\mathbf{x}$ for component c	$\mathcal{C}_{o,k}$	Canonical path family k for OD pair o
$\mathcal{R}^{\text{prot}}$	Set of protected taxi resources	$D(\mathcal{I})$	Number of operating days represented in index set $\mathcal{I}$
$\mathcal{W}$	Set of ISO weeks	$d_{ij}(t)$	Center-to-center distance between aircraft i and j
$\mathcal{W}^{\text{test}}$	Test-week partition	E	Directed-edge set of the airport layout graph
$\mathcal{W}^{\text{train}}$	Training-week partition	e	Directed-edge index
$\mathcal{W}^{\text{val}}$	Validation-week partition	$F1_{\text{edge}}$	Edge-weighted F1 score
$\phi_c^{\text{comp}}(\mathcal{I})$	Component-specific compactness acceptance predicate	$G = (V, E)$	Airport layout graph with node set V and directed-edge set E
$\phi_c^{\text{stab}}(\mathcal{I})$	Component-specific stability acceptance predicate	$h_i^{\text{merge}}(t)$	Merge-entry hold indicator for aircraft i
$\pi_c^{\text{fb}}(y \mid \mathbf{x})$	Fallback-stage class-probability distribution for component c	i	Taxi-movement index
$\rho^{\text{day}}(\mathcal{I})$	Largest single-day share in index set $\mathcal{I}$	$I_{i,r}^{\text{free}}$	Free-motion occupancy interval of aircraft i at protected taxi resource r
ρ_i	Protected radius of aircraft i	$J_c^{\text{val}}(\theta)$	Validation objective score for component c and configuration θ
θ_c^*	Selected configuration for learned behavioral component c	k	Trajectory-point index
$\tilde{\varepsilon}_i(e)$	Median point-to-path projection error for directed edge e	$L(P)$	Length of path P
$\tilde{T}_i$	Graph-aligned taxi trajectory for taxi movement i	l_e	Length of directed edge e
ε_i^{95}	95th-percentile point-to-path projection error for taxi movement i	M	Number of edge-progress grid points
$\varepsilon_{i,k}$	Point-to-path projection error of trajectory point $q_{i,k}$	$N(\mathcal{I})$	Number of observations in index set $\mathcal{I}$
Ξ	Normalized edge-progress grid	$n_i(e)$	Edge-support count for directed edge e
		o	OD-pair index
		$o_i = (d_i, r_i, a_i)$	OD pair for taxi movement i , defined by movement direction, runway stopbar, and apron boundary
		P_i^{fb}	A* fallback path for taxi movement i
		P_i^{obs}	Exact observed path for taxi movement i

P_i^{sel}	Selected path for taxi movement i during model execution	$v_i^{\text{safe}}(t)$	Deterministic safe-speed limit for aircraft i
P_{edge}	Edge-weighted precision	$v_i^{\text{target}}(t)$	Target speed from the speed-profile component for aircraft i
$Q_i(e)$	Set of trajectory points assigned to directed edge e for taxi movement i	$v_{i,k}$	Observed or derived ground speed at trajectory point $q_{i,k}$
$q_{i,k}$	Trajectory point k of taxi trajectory T_i	$X_i^{\text{free}}(t)$	Free-motion trajectory of aircraft i before local conflict resolution
R_{edge}	Edge-weighted recall	$y_{i,j,r}^{\text{act}}$	Movement-derived resolution-action label for aircraft pair (i, j) at protected taxi resource r
$r_{i,e}(\xi_m)$	Speed-profile residual for taxi movement i on directed edge e	$y_{i,j,r}^{\text{delay}}$	Delay target for aircraft pair (i, j) at protected taxi resource r
$RMSE_v$	Speed-profile root-mean-square error	Y_{τ}^{ep}	Endpoint-selection target variable for endpoint target τ
$s_{i,k}$	Path progress of trajectory point $q_{i,k}$	$y_{i,j,r}^{\text{prio}}$	Priority label for aircraft pair (i, j) at protected taxi resource r
T_i	Taxi trajectory for taxi movement i	Y_i^{path}	Path-choice target variable for taxi movement i
t_i^{arr}	Arrival time of aircraft i at the runway-entry stopbar	$y_{c,i}$	Target variable or class label for learned behavioral component c and observation i
t_i^{rel}	Runway-departure release time of aircraft i	$z_{c,i}$	Component observation for learned behavioral component c and taxi movement or event i
$t_{i,k}$	Timestamp of trajectory point $q_{i,k}$		
V	Node set of the airport layout graph		
$v_{i,e}^{\text{obs}}(\xi_m)$	Observed edge-relative speed profile for taxi movement i on directed edge e		
$v_i^{\text{exec}}(t)$	Executed speed of aircraft i at time t		

List of Figures

A.1	Weekly operational-profile clusters	68
A.2	Context-feature coverage profile	71
A.3	Bayesian optimization validation traces	73
A.4	Training-data sensitivity diagnostics	74
B.1	Map-matching validation diagnostics	77
B.2	Path-catalog compression diagnostics	78
B.3	Representative canonical path families	79
B.4	Path-choice validation diagnostics	80
B.5	Non-trivial path-choice diagnostics	81
B.6	Endpoint-selection validation diagnostics	81
B.7	Non-trivial endpoint-selection diagnostics	81
B.8	Endpoint-selection confusion matrices	82
B.9	Spatial endpoint-selection diagnostics	83
B.10	Speed-profile validation diagnostics	84
B.11	Spatial speed-profile error diagnostics	84
B.12	Local conflict-resolution validation diagnostics	85
B.13	Local conflict-resolution confusion matrices	85
B.14	Integrated motion diagnostics	86
B.15	Integrated conflict-emergence diagnostics	86
B.16	Runway-departure queueing diagnostics	87
C.1	Prediction-source shares by component	93
C.2	OD-conditioned error concentration	94
C.3	Selected context-feature validation patterns	95
C.4	Weekly operational-profile validation patterns	96
D.1	Inbound 36R construction-period path usage	100
D.2	Outbound 36L construction-period path usage	100
D.3	August 2025 construction-corridor path-choice diagnostic	101
D.4	Held-out week-35 construction-corridor diagnostic	102
E.1	Center-runway active-state trajectory regimes	104
E.2	Center-runway path-choice counterfactual	105
E.3	Center-runway counterfactual summary	106
E.4	Runway-24/S7 surface-surveillance points	108
F.1	Apron-area study scope	112
F.2	Apron taxiing-phase extraction retention	113
F.3	Apron map-matching acceptance	114
F.4	Apron D map-matching examples	114
F.5	Apron ED map-matching examples	115
F.6	Apron path-catalog preparation	116

List of Tables

4.1	Work-package overview	55
4.2	Detailed work-package tasks and deliverables	56
4.3	Thesis milestones	57
4.4	Indicative thesis timeline	57
A.1	Interpretation of weekly operational-profile clusters	68
B.1	Integrated scenario diagnostics	87
B.2	Runway-departure release diagnostics	88
C.1	Selected empirical and CART context features	90
C.2	Future context directions	97
D.1	August 2025 construction-corridor summary	101
D.2	Held-out week-35 construction-corridor summary	102
E.1	Center-runway standard-corridor counterfactual	106
E.2	OD groups without compliant canonical path families	107
E.3	Runway-24/S7 population denominators	109
E.4	Runway-24/S7 endpoint-availability counterfactual	109
F.1	Apron preprocessing bridge	113
F.2	Apron map-matching summary	115
F.3	Apron path-catalog summary	116

Part I

General Introduction

Introduction

Airport surface movement operations are a critical part of airside performance at large hub airports. Aircraft taxi operations affect runway throughput, delay, predictability, controller workload, and operational resilience. At Amsterdam Airport Schiphol, these operations take place in a dense taxiway network with changing runway configurations, shared surface resources, and many interactions between aircraft. Understanding how aircraft actually move through the current operation is therefore necessary before evaluating future infrastructure changes, optimization methods, decision-support tools, or operational concepts.

This thesis develops a data-driven multi-agent system model for observed aircraft taxi operations at Schiphol. The work focuses on runway-stopbar–apron-boundary taxi movements and uses VEMMIS operational records and surface-surveillance trajectories as the empirical basis. These records are transformed into taxi trajectories, graph-aligned taxi trajectories, exact observed paths, canonical path families, context features, and component-specific input datasets. The resulting model estimates selected taxi-operation behaviors: endpoint selection, path choice, speed-profile behavior, and local conflict resolution. The model is not designed to optimize airport operations. Its purpose is to construct a support-aware observed-behavior baseline that distinguishes historically supported behavior from fallback rules and deterministic execution rules.

The report is organized around a scientific article. The article presents the core research contribution: the research gap, methodology, Schiphol case study, component-level validation, integrated model-validation diagnostics, discussion, and conclusion. This article is placed at the beginning of the report so that the main argument can be read as a self-contained scientific study. The remaining chapters provide the thesis-level foundation and supporting evidence needed to understand, verify, and interpret the article in more depth.

After the article, the literature review positions the work within airport surface movement operations, graph-based reconstruction, data-driven behavior identification, multi-agent modeling, and validation methodology. The research definition then states the research gap, objective, research questions, scope, and project plan. The appendices provide the detailed evidence behind the article: Appendix A documents model calibration and tuning; Appendix B provides verification and validation diagnostics; Appendix C interprets the selected model configurations operationally; Appendix D analyzes construction-state effects on path choice; Appendix E studies operational constraints and availability rules; and Appendix F examines methodology transfer to apron-area taxi movements.

Together, the article and supporting chapters show how confidential surface-surveillance data can be converted into a reusable observed-behavior baseline for aircraft taxi operations. The thesis contributes a graph-based and support-aware methodology, demonstrates it on a large Schiphol case study, and identifies the main limitations that remain: static layout representation, missing operational context, movement-derived conflict labels, and simplified deterministic execution rules.

Part II

Scientific Article

Data-Driven Agent-Based Modeling of Aircraft Surface Movement at a Large Airport

J. Crespo Núñez*

Delft University of Technology, Delft, The Netherlands

Airport surface movement operations (ASM Ops) at large hub airports require reusable baselines of observed taxi behavior, but surface-surveillance data are often proprietary, noisy, and difficult to convert into support-aware behavioral models. This paper develops a data-driven multi-agent system (MAS) model for aircraft taxi operations at Amsterdam Airport Schiphol, focused on taxiing movements. Operational records are transformed into taxi trajectories, graph-aligned taxi trajectories, exact observed paths, canonical path families, context features, and component-specific input datasets. Four learned behavioral components estimate endpoint selection, path choice, speed-profile behavior, and local conflict resolution through a shared three-stage inference procedure combining empirical distribution estimators, Classification and Regression Tree (CART) estimators, and fallback rules. The retained dataset contains 883,771 graph-aligned taxi trajectories. Held-out validation shows strong support for selected behaviors, with a mean edge-weighted path F1 of 0.926, a mean speed-profile root-mean-square error (RMSE) of 2.28 kt, and generally low fallback use. The results show that trajectory-derived observations can support an observed-behavior baseline for selected aircraft taxi-operation behaviors, while also identifying limits from static graph representation, missing operational context, movement-derived conflict labels, and simplified deterministic execution rules.

I. Introduction

ASM Ops are a critical part of airside performance at large hub airports. Aircraft taxi operations affect runway throughput, delay, predictability, controller workload, and operational resilience. At Amsterdam Airport Schiphol, these movements take place in a dense taxiway network with changing runway configurations, shared surface resources, and interactions between aircraft. Understanding observed taxi behavior is therefore necessary before evaluating future operational concepts, decision-support tools, or infrastructure changes.

Many airport-surface studies require assumptions about how aircraft move through the current operation. These assumptions often include path choice, endpoint selection, taxi speeds, queueing, and local interaction behavior. When each study defines this baseline separately, results become difficult to compare and conclusions become tied to case-specific modeling choices. A reusable observed-behavior baseline should instead be derived from operational data, represented on a common airport layout graph, and evaluated with explicit support diagnostics.

This paper addresses the lack of a reproducible, graph-based, and support-aware observed-behavior baseline for aircraft taxi operations. The objective is to develop and evaluate a MAS model that reproduces selected taxi-operation behaviors from historical surface-surveillance data. The model is not designed to optimize airport operations. It estimates behavior from observed data and separates learned behavioral components from deterministic execution rules that preserve safe and executable model execution.

The framework transforms operational records from VEMMIS, an internal Luchtverkeersleiding Nederland (LVNL) air traffic management and surveillance database, into graph-aligned taxi trajectories, exact observed paths, canonical path families, context features, and component-specific input datasets. These data products are used to identify four learned behavioral components: endpoint selection, path choice, speed-profile behavior, and local conflict resolution. Each learned component uses a shared empirical, CART, and fallback inference structure. The empirical stage preserves observed behavior when historical support is sufficient, the CART stage provides interpretable generalization when exact support is sparse, and the fallback stage preserves executable outputs when learned support is unavailable.

The Schiphol case study covers aircraft taxi operations from January 2024 to December 2025 within the taxiing scope. The experimental design evaluates the full pipeline from data processing to component-level validation and integrated model validation. This structure allows the paper to assess both isolated learned-component performance and the behavior that emerges when selected component outputs are combined with deterministic execution rules.

*MSc Student, Control and Simulation, Faculty of Aerospace Engineering, Delft University of Technology

The contribution of the paper is an observed-behavior baseline that can support future airport-surface studies. The framework provides structured path, endpoint, speed, and interaction behavior derived from trajectory observations rather than ad-hoc assumptions. It also reports prediction-source diagnostics, so that future users can distinguish historically supported behavior from fallback-based executability. Additional calibration, validation, operational-context, and case-study evidence is provided in Appendix A (Model Calibration and Tuning), Appendix B (Verification and Validation), Appendix C (Operational Interpretation of Selected Model Configurations), Appendix C (Operational Interpretation of Selected Model Configurations), Appendix E (Operational Constraints and Availability Rules), and Appendix F (Methodology Transfer to Apron-area Taxi Movements).

The paper is organized as follows. Section II reviews the literature and methodological context for ASM Ops modeling, graph-based reconstruction, trajectory behavior, data-driven model identification, and model validation. Section III presents the methodology. Section IV describes the Schiphol case study, data partitioning, training workflow, evaluation metrics, and integrated model-validation setup. Section V reports the results. Section VI discusses the main takeaways and limitations. Section VII concludes the paper.

II. Background

A. Airport Surface Movement Operations and MAS Modeling

ASM Ops involve aircraft, controllers, airport infrastructure, runway configurations, shared taxi resources, and local safety constraints. Aircraft taxi operations are therefore not only a path-choice or taxi-time problem. They are part of a coupled movement, control, and infrastructure system in which each movement depends on both the physical layout and the behavior of other traffic. Holistic airport-surface modeling treats these processes jointly [1].

This coupled structure also appears in operational support concepts. Advanced Surface Movement Guidance and Control System (A-SMGCS) specifications separate surveillance, routing, guidance, and airport safety-support services [2]. These functions are distinct, but their outputs interact during surface movement operations: surveillance provides the observed state, routing and guidance shape movement options, and safety-support functions constrain how aircraft use shared resources. This functional decomposition motivates a modeling approach that can represent both individual behavior and system-level interactions.

Agent-based and multi-agent approaches provide such a structure. The agent-based modeling paradigm is commonly used for systems whose aggregate behavior emerges from interactions among autonomous agents and their environment [3]. In ASM Ops, this is directly relevant because surface behavior emerges from interactions between aircraft execution, controller decisions, airport layout constraints, and shared infrastructure resources. Schiphol-related MAS research has also shown the value of representing airport surface movements through interacting operational roles [4].

ASM Ops models can serve different purposes. Optimization models are useful when the goal is to compute efficient surface paths or schedules [5], and fast-time airport-surface models have been used to study surface-operation processes before operational deployment [6]. This paper has a different objective. It uses the MAS representation to organize learned behavioral components and deterministic execution rules for reproducing observed aircraft taxi operations. The resulting integrated model-validation diagnostics assess model consistency and emergent behavior, not the validity of an external operational tool.

B. Surface Surveillance Data and Graph-Based Reconstruction

Surface-surveillance data provide direct observations of aircraft ground motion, but they are not directly usable as model inputs. Radar or Automatic Dependent Surveillance-Broadcast (ADS-B) trajectories may contain noise, missing values, uneven sampling, and spatial or temporal misalignment [7]. Before they can support behavior modeling, these observations must be filtered, scoped to aircraft taxi operations, and reconstructed on a spatial representation of the airport surface [8].

Map matching provides the basic mechanism for this reconstruction. It projects continuous trajectory points onto a graph so that observed motion can be represented as an ordered path through the airport layout. General map-matching literature provides methods for projecting noisy observations onto a graph [9]. Airport-specific map matching adapts this idea to dense taxiway networks, operational restrictions, and surface-surveillance data [7]. Related airport ground-movement studies confirm the relevance of graph-constrained reconstruction for airport surface trajectories [10]. Hidden Markov methods provide a probabilistic alternative for trajectory-to-network matching [11], while spatio-temporal methods use timing and speed constraints to recover paths from sparse data [12].

The airport-surface setting requires more than generic road-network matching. Taxi movements must respect airport layout graph topology, runway stopbars, apron boundaries, and taxiing-scope exclusions. This motivates the graph-constrained reconstruction used in this paper: taxi trajectories are not treated only as coordinate traces, but as graph-aligned taxi trajectories on a shared airport layout graph.

C. Trajectory Behavior, Path Families, and Operational Context

Graph reconstruction makes taxi behavior observable as graph-aligned taxi trajectories. These trajectories contain both a spatial path through the airport layout graph and the speed behavior along that path. The next modeling step is therefore to convert reconstructed trajectory behavior into component-specific alternatives: path candidates for the path-choice component and edge-relative speed profiles for the speed-profile component. In transportation research, path-choice modeling treats the definition of feasible alternatives as a central modeling problem [13]. Airport taxi-route concepts also support route-management analysis and surface-routing applications [14].

Graph search can provide executable paths when observed support is missing [15]. However, observed Ground Control Agent path choice is not assumed to follow shortest paths. For this reason, the methodology uses observed path families as the main path candidates and reserves graph-search paths for fallback cases.

Exact observed paths preserve the spatial sequence of each graph-aligned taxi trajectory, but they are too detailed for direct behavior modeling. Small map-matching differences, local tactical deviations, and minor path variations can create many unique exact observed paths even when the operational trajectory is effectively the same. Trajectory-similarity literature supports grouping similar movement sequences when the goal is to reduce sparsity while preserving movement structure [16]. This motivates the distinction between exact observed paths, which retain spatial reconstruction detail, and path families, which define the path candidates used by the path-choice component.

Trajectory behavior also depends on operational context. Movement direction, runway configuration, aircraft class, time of day, demand state, endpoints, and local traffic interactions all affect the observed path and speed profile of a taxi movement. The methodology therefore attaches context features to graph-aligned taxi trajectories and uses them to construct component-specific input datasets for path choice, endpoint selection, speed-profile behavior, and local conflict resolution.

D. Data-Driven Model Identification

Data-driven behavior identification estimates model outputs from historical observations rather than prescribing them only through shortest-path, optimization-based, or fixed rule-based assumptions. This is appropriate when the objective is to reproduce an observed baseline operation. The model should reflect how aircraft taxi operations were actually executed, while still remaining reliable when the available historical support is uneven.

Empirical distributions provide the most direct way to preserve observed behavior. In well-supported contexts, they estimate output frequencies directly from observations and require few modeling assumptions. Nonparametric empirical-distribution methods estimate quantities without specifying a parametric model [17]. Their main limitation is sparsity: as more operational context is included, exact context cells may contain too few observations for reliable estimation.

Tree-based supervised learning provides a transparent way to generalize beyond exact empirical support. CART partitions the feature space into interpretable leaves and can therefore relate similar operational contexts while preserving a readable decision structure [18]. However, probability estimates from tree leaves require care because leaf support and calibration affect the reliability of class-probability distributions [19]. This makes support and acceptance diagnostics important when tree-based outputs are used as behavioral estimates.

More flexible tree ensembles are common predictive alternatives to single decision trees. Random forests provide one such ensemble approach [20], and gradient boosting is a strong alternative for regression and classification tasks [21]. These methods can improve predictive accuracy, but they are less transparent than a single CART estimator. For this study, interpretability, auditability, and explicit support diagnostics are more important than maximizing black-box predictive performance. This motivates the use of empirical estimates where historical support is sufficient, CART-based generalization where exact support is sparse, and fallback rules where learned support is unavailable.

E. Evaluation, Model Validation, and Deterministic Execution Rules

Evaluation of data-driven airport-surface models must account for the temporal structure of operations. Flight-level random partitioning can place temporally adjacent movements with similar runway configurations, demand patterns,

weather conditions, or traffic states in different data partitions. This can overstate generalization because the training and test data may share the same operating conditions. Time-blocked partitioning reduces this risk by separating complete operating periods. Similar-day grouping has been used in air traffic flow management to compare periods with related demand and operational characteristics [22]. K-medoids clustering is also useful in this setting because each cluster is represented by an observed medoid rather than by a synthetic centroid [23].

Validation must also distinguish predictive performance from learned support. A component can appear accurate if it relies heavily on fallback rules, but fallback outputs do not provide the same evidence of learned behavior as outputs derived from empirical support or CART-based generalization. For support-aware behavior identification, performance metrics should therefore be interpreted together with prediction-source diagnostics. This distinction is important when the objective is not only to predict held-out labels, but also to understand which parts of the model are supported by historical observations.

An observed-behavior MAS model must finally be checked after its components are combined. Learned behavioral components can estimate path choice, endpoint selection, speed behavior, and local conflict-resolution labels, but they do not by themselves guarantee safe and executable movement during integrated model execution. Deterministic execution rules provide this layer. Same-path following interactions, merge-entry holds, and separation speed constraints can be grounded in classical following and braking principles such as Gipps-style car-following [24]. Runway-departure release behavior can also be represented through queueing abstractions [25], with Schiphol-specific departure-separation factors informed by European wake-turbulence recategorization (RECAT-EU) wake category and Standard Instrument Departure (SID) effects [26]. These rules should be interpreted as execution constraints, not as learned behavioral components.

Model verification and validation must be distinguished from model construction and historical-scenario model execution [27]. Integrated model-validation diagnostics are therefore best interpreted as checks of component interoperability, integrated model consistency, and emergent behavior.

III. Methodology

A. Framework Overview

The methodology develops a data-driven MAS model for aircraft taxi operations at Amsterdam Airport Schiphol. The model does not optimize the operation. Instead, it reconstructs observed taxi behavior from VEMMIS operational records and organizes this behavior into learned behavioral components and deterministic execution rules. Multi-agent modeling has previously been applied to airport surface movement operations at Schiphol [4]; here, the MAS structure is used to identify and validate an observed-behavior baseline.

The framework uses VEMMIS air traffic management data. Because raw VEMMIS records cannot be shared, reproducibility relies on clear descriptions of the processing procedure, data fields, derived data products, model inputs, and evaluation design. The same methodology could be applied to other surface-surveillance data sources when comparable trajectory, timing, and operational-context information is available. Figure 1 summarizes the workflow.

The workflow has five stages. First, VEMMIS operational records and surface-surveillance trajectories are converted into scoped taxi trajectories. Second, the modeled taxiing scope is reconstructed on an airport layout graph through graph-constrained map matching. Third, graph-aligned taxi trajectories are transformed into exact observed paths, canonical path families, path-choice sets, context features, and component-specific input datasets. Fourth, learned behavioral components estimate endpoint selection, path choice, speed-profile behavior, and local conflict resolution. Fifth, deterministic execution rules preserve safe and executable model execution during integrated validation.

Throughout the methodology, index i denotes a taxi movement, k a trajectory point, e a directed edge, o an Origin-Destination (OD) pair, and c a learned behavioral component. Sets are denoted by calligraphic letters, vectors by bold symbols, and hats indicate estimated quantities. The processing chain transforms VEMMIS operational records into taxi trajectories T_i , graph-aligned taxi trajectories $\tilde{T}_i$, exact observed paths P_i^{obs} , path-choice sets C_{o_i} , component observations $z_{c,i}$, and learned behavior estimates $\hat{\pi}_c$ or $\hat{v}_c$.

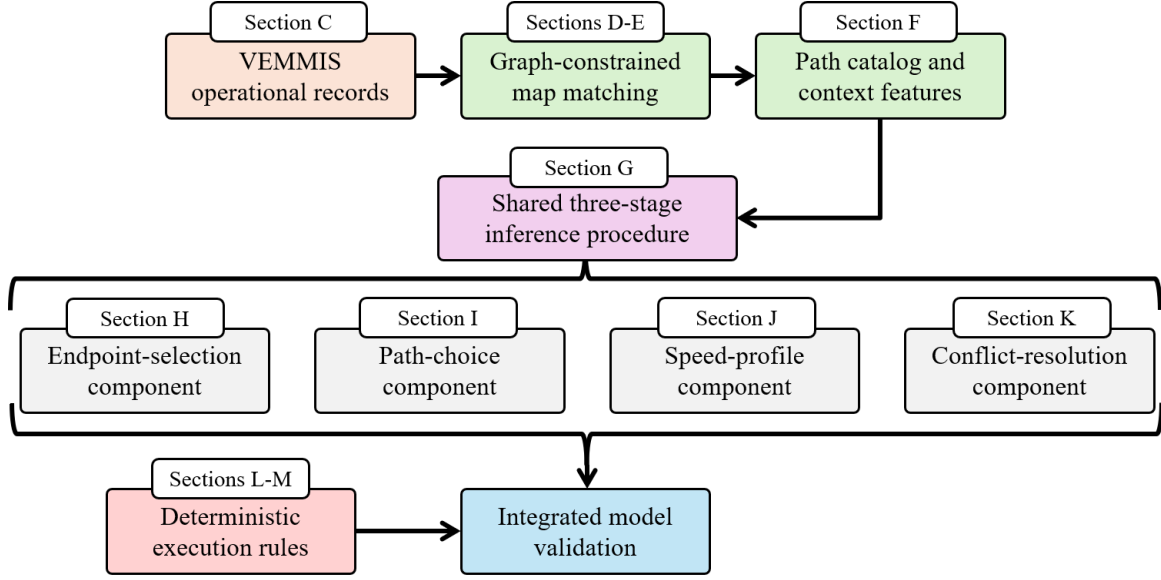


Fig. 1 Overview of the data-processing, model-identification, and deterministic execution-rule pipeline for the MAS model.

The rest of this section follows Fig. 1. Section III.B defines the MAS model, agent categories, component allocation, and the boundary between learned behavioral components and deterministic execution rules. Sections III.C–III.E describe data ingestion, taxiing-scope extraction, airport layout graph construction, and graph-constrained map matching. Section III.F introduces the path catalog, context features, and component-specific input datasets. Section III.G explains the shared three-stage inference procedure. Sections III.H–III.K detail the learned behavioral components, and Sections III.L and III.M describe the deterministic execution rules.

B. MAS Model and Agent Behavior Identification

The MAS model represents aircraft taxi operations as interactions between functional agent categories and an Environment. During model execution, the agents observe the relevant Environment state and produce actions. The model contains three agent categories: the Ground Control Agent, the Aircraft Operator Agent, and the Runway Control Agent. The Environment contains the operational objects and state variables with which the agents interact, such as the airport layout graph, runway stopbars, apron boundaries, protected taxi resources, active traffic state, resource occupancy, and operational context.

The historical data-processing pipeline is separate from the MAS model itself. VEMMIS operational records, taxi trajectories, graph-aligned taxi trajectories, context features, path catalogs, and component-specific input datasets are used to identify the model properties from observed operations. Once these properties are estimated, the MAS model uses them as behavior rules, action spaces, and execution constraints during model validation. This distinction separates the model that is executed from the data and procedures used to infer its behavior.

The learned behavioral components approximate agent decisions from observed data. The internal reasoning of controllers and flight crews is not observed directly, so each learned component treats the corresponding agent behavior as a black box: observed Environment conditions are associated with a trajectory-derived action, class label, or speed profile. In this paper, the Ground Control Agent is associated with endpoint selection, path choice, and local conflict resolution; the Aircraft Operator Agent is associated with speed-profile behavior and aircraft-side execution; and the Runway Control Agent is represented through a runway-departure release rule. These are functional modeling abstractions and do not imply that actual ATC or pilot commands were observed in the VEMMIS records.

Deterministic execution rules are treated separately from learned behavioral components. They map model-execution state to speed, hold, or release outputs in order to preserve safe and executable integrated model behavior. They are therefore part of the executed MAS model, but they are not evidence that the corresponding behavior was learned from data.

Table 1 Agent-category allocation of learned behavioral components and deterministic execution rules.

Model element	Role in the MAS model
Ground Control Agent	learned decision rules for endpoint selection, path choice, and local conflict resolution
Aircraft Operator Agent	learned speed-profile behavior; deterministic following-and-merge guard for aircraft-side execution
Runway Control Agent	deterministic runway-departure release rule at runway-entry stopbars
Environment	airport layout graph; runway stopbars; apron boundaries; protected taxi resources; active traffic and resource state; operational context available during model execution

The allocation in Table 1 sets the modeling boundary for the rest of the methodology. The learned components reproduce or generalize observed agent behavior from trajectory-derived observations. The deterministic execution rules preserve safety and executability during integrated model validation.

C. Data Ingestion and Taxiing-Scope Extraction

The data-ingestion and preprocessing layer converts VEMMIS operational records into scoped taxi trajectories for graph reconstruction and behavior identification. VEMMIS contains flight records, schedule records, and surface-surveillance trajectories. These records provide the historical observations from which agent behavior is inferred, but the raw surveillance coverage extends beyond the modeled taxiing scope and may include runway, apron, approach, and other airside detections.

A taxi movement denotes one inbound or outbound aircraft movement within the modeled taxiing scope. Inbound movements run from runway stopbar to apron boundary, while outbound movements run from apron boundary to runway stopbar.

Definition 1 (Taxi trajectory). *For taxi movement i , the taxi trajectory T_i is the ordered sequence of surface-surveillance trajectory points retained between the movement’s runway stopbar and apron boundary.*

$$T_i = \langle q_{i,1}, q_{i,2}, \dots, q_{i,n_i} \rangle, \quad q_{i,k} = (\mathbf{p}_{i,k}, t_{i,k}, v_{i,k}), \quad (1)$$

where $\mathbf{p}_{i,k} \in \mathbb{R}^2$ is the observed surface position, $t_{i,k}$ is the timestamp, and $v_{i,k}$ is the observed or derived ground speed.

The preprocessing procedure isolates this taxiing-scope interval and prepares it for graph reconstruction. It normalizes identifiers and metadata, filters position and speed signals with a rolling median filter and derivative filter [8], chains compatible trajectory fragments, and infers runway stopbar and apron-boundary milestones. A movement is retained only when movement direction, runway stopbar, apron boundary, trajectory points, and taxiing-scope start and end times are available, mutually consistent, and inside the modeled taxiing scope.

The scope excludes apron-internal movements, gate-area movements, pushback, de-icing, private-aviation movements, eastern maintenance-area movements, and runway 22/04. These exclusions keep the modeled focused on commercial aircraft taxi operations.

D. Airport Layout Graph and Environment Representation

The airport layout graph provides the spatial representation used throughout the methodology. During model identification, it supports graph-constrained map matching, endpoint inference, path-catalog construction, speed-profile alignment, and local conflict-event extraction. During MAS model execution, it is one of the environmental objects through which agents interact with endpoints, directed edges, protected taxi resources, path candidates, and resource-occupancy states.

Definition 2 (Airport layout graph). *The airport layout graph is a directed graph that represents the traversable surface infrastructure within the modeled taxiing scope. Nodes represent operational layout points, and directed edges represent traversable centerline elements with allowed travel direction and precise geometric shape.*

$$G = (V, E), \quad E \subseteq V \times V. \quad (2)$$

Each node $v \in V$ represents an operational location, such as a taxiway junction, runway stopbar, apron boundary, holding location, or other relevant layout point. Each directed edge $e = (v_a, v_b) \in E$ represents a traversable centerline element between two nodes. The graph is derived from an École Nationale de l’Aviation Civile (ENAC) Scalable Vector Graphics (SVG) layout of Amsterdam Airport Schiphol, in which edge shapes are stored as precise Bezier-derived centerlines.

The graph elements also contain layout-derived attributes used by the methodology. Node attributes identify operational position categories and their role in the modeled taxiing scope. Edge attributes include name, element type, length, available dimension fields, directionality, restrictions, operational labels, and other metadata. A bidirectional physical taxiway segment may therefore be represented by two opposite directed edges with shared geometry but different travel direction or operational attributes.

Figure 2 shows the airport layout graph and modeled spatial scope. The left panel presents the graph used as the spatial reference. The right panel places the modeled scope in geographic context and highlights excluded regions, including later infrastructure additions (in yellow and purple) and the eastern maintenance/private-aviation area (in red).



Fig. 2 Airport layout graph representation and modeled spatial scope.

E. Map Matching

Graph-constrained map matching projects taxi trajectories onto the airport layout graph to produce graph-aligned taxi trajectories [9]. The procedure adapts the airport-surface trajectory approach of Tran et al. [7] to the Schiphol layout and the modeled taxiing scope. Its purpose is to convert noisy surface-surveillance observations into topologically valid graph paths with trajectory-point projections, path progress, and reconstruction-quality diagnostics.

Definition 3 (Graph-aligned taxi trajectory). *A graph-aligned taxi trajectory augments a taxi trajectory with its reconstructed directed-edge sequence, projected positions, path progress, edge-relative progress, matched directed edges, and point-to-path projection errors.*

$$\tilde{T}_i = \left(T_i, P_i^{\text{obs}}, \{\hat{\mathbf{p}}_{i,k}, s_{i,k}, \xi_{i,k}, e_{i,k}, \varepsilon_{i,k}\}_{k=1}^{n_i} \right), \quad (3)$$

where P_i^{obs} is the reconstructed directed-edge sequence, $\hat{\mathbf{p}}_{i,k}$ the projected position of trajectory point $q_{i,k}$, $s_{i,k}$ path progress, $\xi_{i,k}$ edge-relative progress, $e_{i,k}$ the matched directed edge, and $\varepsilon_{i,k}$ the point-to-path projection error.

If trajectory point $q_{i,k}$ is projected on edge $e_{i,j}$ of P_i^{obs} , its path progress is

$$s_{i,k} = \sum_{\ell < j} l_{e_{i,\ell}} + \xi_{i,k} l_{e_{i,j}}, \quad (4)$$

and its point-to-path projection error is

$$\varepsilon_{i,k} = \|\mathbf{p}_{i,k} - \hat{\mathbf{p}}_{i,k}\|_2. \quad (5)$$

The map-matching procedure first prepares the taxi trajectory and generates candidate edges near each trajectory point. Candidate edges are scored using spatial proximity, motion direction, and graph plausibility. Short gaps in evidence are repaired through graph connectivity, after which an endpoint-constrained path is reconstructed between the inferred runway stopbar and apron boundary. The output is retained only if the reconstructed path is topologically valid and satisfies movement-level and edge-level reconstruction-quality checks.

Reconstruction quality is evaluated through both edge-level and movement-level diagnostics. Edge-level diagnostics describe whether each directed edge in the reconstructed path is supported by nearby trajectory points. For movement i and directed edge e , the supporting trajectory-point set is

$$Q_i(e) = \{k : e_{i,k} = e\}. \quad (6)$$

For non-empty $Q_i(e)$, the edge-level evidence is summarized by the number of assigned trajectory points, the covered fraction of the edge, and the median point-to-path projection error:

$$\begin{aligned} n_i(e) &= |Q_i(e)|, \\ c_i(e) &= \max_{k \in Q_i(e)} \xi_{i,k} - \min_{k \in Q_i(e)} \xi_{i,k}, \\ \tilde{\varepsilon}_i(e) &= \text{median}_{k \in Q_i(e)} \varepsilon_{i,k}. \end{aligned} \quad (7)$$

Here, $n_i(e)$ is the edge-support count, $c_i(e)$ is the edge-relative coverage, and $\tilde{\varepsilon}_i(e)$ is the representative projection error for the edge. If $Q_i(e) = \emptyset$, the edge automatically fails the support criteria; in that case $n_i(e) = 0$, while $c_i(e)$ and $\tilde{\varepsilon}_i(e)$ are not evaluated.

Movement-level diagnostics summarize the projection quality of the full graph-aligned taxi trajectory:

$$\begin{aligned} \bar{\varepsilon}_i &= \frac{1}{n_i} \sum_{k=1}^{n_i} \varepsilon_{i,k}, \\ \varepsilon_i^{95} &= \text{quantile}_{0.95} \left(\{\varepsilon_{i,k}\}_{k=1}^{n_i} \right). \end{aligned} \quad (8)$$

The mean projection error $\bar{\varepsilon}_i$ measures the average spatial fit, while the 95th-percentile projection error ε_i^{95} captures large local deviations.

A graph-aligned taxi trajectory is retained when the reconstructed path satisfies the edge-support, projection-error, endpoint-consistency, and topology criteria, where $E(P_i^{\text{obs}})$ denotes the directed-edge set of the reconstructed path.:

$$\begin{aligned} n_i(e) &\geq n_{\min} \quad \forall e \in E(P_i^{\text{obs}}), \\ c_i(e) &\geq c_{\min} \quad \forall e \in E(P_i^{\text{obs}}), \\ \tilde{\varepsilon}_i(e) &\leq \varepsilon_{\max} \quad \forall e \in E(P_i^{\text{obs}}), \\ \bar{\varepsilon}_i &\leq \bar{\varepsilon}_{\max}, \\ \varepsilon_i^{95} &\leq \varepsilon_{\max}^{95}, \\ \eta_i^{\text{end}} &= 1, \\ \eta_i^{\text{top}} &= 1. \end{aligned} \quad (9)$$

Here, η_i^{end} indicates that the reconstructed path is consistent with the inferred runway stopbar and apron boundary, and η_i^{top} indicates that the reconstructed directed-edge sequence is topologically valid on the airport layout graph. In the result analysis, high-error exclusions are reported using the 95th-percentile criterion $\varepsilon_i^{95} > 80$ m.

Figure 3 summarizes the graph-constrained map-matching logic. Trajectory points are linked to candidate edges, graph connectivity is restored where needed, endpoints constrain the reconstructed path, and projection-error diagnostics determine whether a case is repaired or excluded.

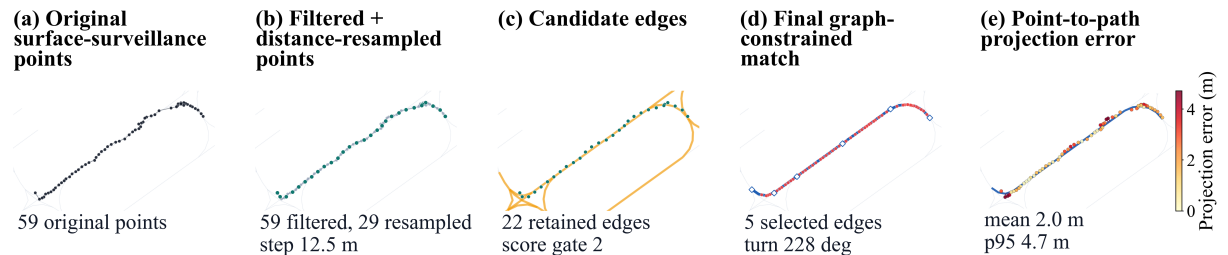


Fig. 3 Conceptual overview of the graph-constrained map-matching procedure and reconstruction-quality checks.

F. Path Catalog and Operational Context

The path catalog converts graph-aligned taxi trajectories into the discrete path candidates used by the path-choice component. The same graph-aligned trajectories are also linked to operational context features, which are later used to construct component-specific input datasets for the learned behavioral components.

Definition 4 (Exact observed path, canonical path family, and path-choice set). *For each OD pair, graph-aligned taxi trajectories yield exact observed paths. Similar exact observed paths are grouped into canonical path families, and the supported canonical path families form the path-choice set used by the path-choice component.*

For taxi movement i , the exact observed path is the reconstructed directed-edge sequence

$$P_i^{\text{obs}} = \langle e_{i,1}, e_{i,2}, \dots, e_{i,n_i^P} \rangle, \quad e_{i,j} \in E, \quad (10)$$

with path length

$$L(P_i^{\text{obs}}) = \sum_{j=1}^{n_i^P} l_{e_{i,j}}. \quad (11)$$

The OD pair of movement i is

$$o_i = (d_i, r_i, a_i), \quad (12)$$

where d_i is movement direction, r_i is the runway stopbar, and a_i is the apron boundary. An OD group is a coarser runway-apron grouping used only for aggregated diagnostics.

For OD pair o , let $\mathcal{P}_o^{\text{obs}}$ be the set of exact observed paths assigned to that OD pair. A canonical path family is a subset of these exact observed paths:

$$C_{o,k} \subseteq \mathcal{P}_o^{\text{obs}}. \quad (13)$$

The OD-specific path-choice set is the set of supported canonical path families:

$$C_o = \{C_{o,1}, C_{o,2}, \dots, C_{o,K_o}\}. \quad (14)$$

Canonical path families are formed by comparing exact observed paths within the same OD pair. The comparison uses mutual directed-edge-count coverage and reference-normalized length difference:

$$\text{cov}_{\text{cnt}}(P, Q) = \frac{|E(P) \cap E(Q)|_{\text{multi}}}{|E(P)|_{\text{multi}}}, \quad (15)$$

$$\delta_L(P, Q) = 100 \frac{|L(P) - L(Q)|}{L(Q)}. \quad (16)$$

Here, $E(P)$ and $E(Q)$ denote the directed-edge multisets of paths P and Q . Two exact observed paths pass the canonical grouping threshold when

$$\text{cov}_{\text{cnt}}(P, Q) \geq 0.70, \quad \text{cov}_{\text{cnt}}(Q, P) \geq 0.70, \quad \delta_L(P, Q) \leq 15. \quad (17)$$

An exact observed path opens a new canonical path family only if it reaches the representative-support rule for its OD pair. In this study, the minimum representative support is 1% of the OD-pair traffic.

Operational context is then attached to the graph-aligned taxi trajectories and their path-choice sets using normalized flight identifiers and time-aligned operational intervals. These context features provide the conditions under which endpoint selection, path choice, speed-profile behavior, and local conflict resolution are later estimated.

G. Shared Three-Stage Inference Procedure

The learned behavioral components use a shared inference structure. The purpose is to preserve observed behavior when historical support is sufficient, generalize across related operational contexts when exact support is sparse, and preserve executable outputs when learned support is unavailable.

Definition 5 (Component observation and three-stage inference procedure). *A component observation pairs the feature vector used by a learned behavioral component with its target variable. For discrete learned components, the three-stage inference procedure estimates a class-probability distribution using an empirical stage, a CART stage, or a fallback stage, and records the prediction source as a diagnostic.*

$$\mathbf{z}_{c,i} = (\mathbf{x}_{c,i}, y_{c,i}), \quad (18)$$

where $\mathbf{x}_{c,i} \in \mathcal{X}_c$ is the feature vector and $y_{c,i} \in \mathcal{Y}_c$ is the target variable. For discrete components, $y_{c,i}$ is a class label; for the speed-profile component, it is a residual profile.

The first stage is the empirical stage. For a selected empirical feature set S_c , the empirical context cell for query $\mathbf{x}$ is

$$\mathcal{D}_{c,S_c}^{\text{emp}}(\mathbf{x}) = \{i \in \mathcal{D}_c^{\text{train}} : \mathbf{x}_{c,i,S_c} = \mathbf{x}_{S_c}\}. \quad (19)$$

The exact-match condition in Eq. (19) is applied to categorical or discretized context features. For discrete targets, the empirical distribution estimator is

$$\hat{\pi}_c^{\text{emp}}(y | \mathbf{x}) = \frac{\sum_{i \in \mathcal{D}_{c,S_c}^{\text{emp}}(\mathbf{x})} \mathbf{1}(y_{c,i} = y)}{|\mathcal{D}_{c,S_c}^{\text{emp}}(\mathbf{x})|}. \quad (20)$$

The estimator is evaluated only for non-empty empirical context cells; empty cells automatically fail the empirical acceptance criteria. The empirical estimate is used only when the empirical context cell satisfies component-specific acceptance criteria. For any index set $\mathcal{I}$, define the support, temporal-coverage, and single-day dominance diagnostics as

$$\begin{aligned} N(\mathcal{I}) &= |\mathcal{I}|, \\ D(\mathcal{I}) &= |\{\text{day}(i) : i \in \mathcal{I}\}|, \\ \rho^{\text{day}}(\mathcal{I}) &= \max_d \frac{|\{i \in \mathcal{I} : \text{day}(i) = d\}|}{|\mathcal{I}|}. \end{aligned} \quad (21)$$

Here, $N(\mathcal{I})$ is the number of observations, $D(\mathcal{I})$ is the number of operating days represented in the set, and $\rho^{\text{day}}(\mathcal{I})$ is the largest single-day share.

The remaining stability and compactness checks are component-specific predicates. Let $\phi_c^{\text{stab}}(\mathcal{I}) \in \{0, 1\}$ and $\phi_c^{\text{comp}}(\mathcal{I}) \in \{0, 1\}$ indicate whether the historical observations are stable and compact enough for component c . The empirical acceptance indicator is then

$$A_c^{\text{emp}}(\mathbf{x}) = \mathbf{1} \left\{ \begin{aligned} &N(\mathcal{D}_{c,S_c}^{\text{emp}}(\mathbf{x})) \geq N_{c,\min}^{\text{emp}}, \\ &D(\mathcal{D}_{c,S_c}^{\text{emp}}(\mathbf{x})) \geq D_{c,\min}^{\text{emp}}, \\ &\rho^{\text{day}}(\mathcal{D}_{c,S_c}^{\text{emp}}(\mathbf{x})) \leq \rho_{c,\max}^{\text{emp}}, \\ &\phi_c^{\text{stab}}(\mathcal{D}_{c,S_c}^{\text{emp}}(\mathbf{x})) = 1, \\ &\phi_c^{\text{comp}}(\mathcal{D}_{c,S_c}^{\text{emp}}(\mathbf{x})) = 1. \end{aligned} \right\}. \quad (22)$$

Thus, the empirical stage is accepted only when the matching historical observations are numerous enough, spread over enough operating days, not dominated by a single day, stable, and compact for the component being estimated. The thresholds in Eq. (22) are component-specific hyperparameters selected during model training and tuning.

If the empirical stage is not accepted, the second stage uses a CART estimator. CART partitions the feature space into interpretable leaves using context features [18]. Let $\ell_c(\mathbf{x})$ denote the CART leaf reached by query $\mathbf{x}$. The corresponding leaf set is

$$\mathcal{L}_c(\mathbf{x}) = \{i \in \mathcal{D}_c^{\text{train}} : \ell_c(\mathbf{x}_{c,i}) = \ell_c(\mathbf{x})\}. \quad (23)$$

For discrete targets, the CART-stage class-probability estimator is

$$\hat{\pi}_c^{\text{CART}}(y | \mathbf{x}) = \frac{\sum_{i \in \mathcal{L}_c(\mathbf{x})} \mathbf{1}(y_{c,i} = y)}{|\mathcal{L}_c(\mathbf{x})|}. \quad (24)$$

The CART estimator is evaluated on leaves produced by a fitted tree, which contain training observations by construction. Leaves that do not satisfy the acceptance criteria are not used for model execution. The CART-stage estimate is used only when the reached leaf satisfies component-specific leaf-acceptance criteria. Define the dominant class probability in the leaf as

$$p_c^{\text{dom}}(\mathbf{x}) = \max_{y \in \mathcal{Y}_c} \hat{\pi}_c^{\text{CART}}(y | \mathbf{x}). \quad (25)$$

The CART-stage acceptance indicator is

$$A_c^{\text{CART}}(\mathbf{x}) = \mathbf{1} \left\{ \begin{array}{l} N(\mathcal{L}_c(\mathbf{x})) \geq N_{c,\min}^{\text{CART}}, \\ D(\mathcal{L}_c(\mathbf{x})) \geq D_{c,\min}^{\text{CART}}, \\ \rho^{\text{day}}(\mathcal{L}_c(\mathbf{x})) \leq \rho_{c,\max}^{\text{CART}}, \\ \phi_c^{\text{stab}}(\mathcal{L}_c(\mathbf{x})) = 1, \\ \phi_c^{\text{comp}}(\mathcal{L}_c(\mathbf{x})) = 1, \\ p_c^{\text{dom}}(\mathbf{x}) \geq p_{c,\min}^{\text{CART}}. \end{array} \right\}. \quad (26)$$

The functions ϕ_c^{stab} and ϕ_c^{comp} are component-specific acceptance predicates selected during model training and tuning. They are not universal statistical tests; they define when a component-specific empirical cell or CART leaf is considered reliable enough for model execution.

Thus, the CART stage is accepted only when the reached leaf contains enough historical support, has sufficient temporal coverage, is not dominated by one operating day, is stable and compact for the component, and has a sufficiently dominant class. These thresholds are also component-specific hyperparameters selected during model training and tuning.

The final discrete prediction follows the empirical, CART, and fallback order:

$$\hat{\pi}_c(y | \mathbf{x}) = \begin{cases} \hat{\pi}_c^{\text{emp}}(y | \mathbf{x}), & A_c^{\text{emp}}(\mathbf{x}) = 1, \\ \hat{\pi}_c^{\text{CART}}(y | \mathbf{x}), & A_c^{\text{emp}}(\mathbf{x}) = 0, A_c^{\text{CART}}(\mathbf{x}) = 1, \\ \hat{\pi}_c^{\text{fb}}(y | \mathbf{x}), & A_c^{\text{emp}}(\mathbf{x}) = 0, A_c^{\text{CART}}(\mathbf{x}) = 0. \end{cases} \quad (27)$$

The fallback stage provides a component-specific executable output when neither learned stage is accepted. It is therefore part of the inference procedure, but it is not interpreted as evidence of learned behavior.

The prediction-source diagnostic records whether an output came from the empirical stage, CART stage, fallback stage, baseline rule, deterministic execution rule, or unsupported state. This diagnostic is reported together with predictive performance because it distinguishes learned support from fallback-based executability.

Figure 4 summarizes the inference sequence. The discrete learned components use Eq. (27); the speed-profile component uses the same support-and-fallback principle through edge-level baseline and residual estimation.

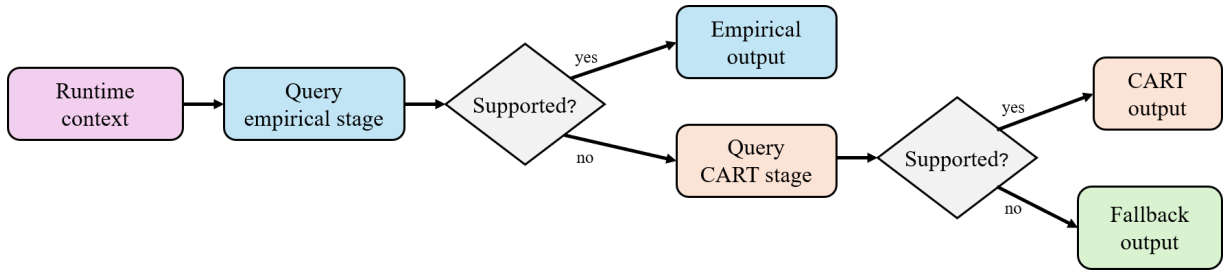


Fig. 4 Empirical-CART-fallback inference stages used by the learned behavioral components.

H. Ground Control Agent: Endpoint-Selection Component

The endpoint-selection component represents the Ground Control Agent's selection of the operational endpoints that bound the modeled taxiing scope. An endpoint is either a runway stopbar or an apron boundary. The component is learned from observed taxi movements: it associates the available operational context with the endpoint used in the historical trajectory, without assuming that the underlying controller decision was directly observed.

The four endpoint-selection targets are the inbound runway stopbar, inbound apron boundary, outbound apron boundary, and outbound runway stopbar. For target τ , the endpoint-choice distribution is

$$\hat{\pi}_\tau^{\text{ep}}(e | \mathbf{x}) = \widehat{P}(Y_\tau^{\text{ep}} = e | X = \mathbf{x}), \quad e \in \mathcal{E}_\tau^{\text{ep}}, \quad (28)$$

where $\mathcal{E}_\tau^{\text{ep}}$ is the valid endpoint set for target τ .

Endpoint selection is sequential when one endpoint constrains the feasible or likely value of the next endpoint. If endpoint target τ_1 is selected before target τ_2 , the selected upstream endpoint is added to the feature vector for the second target:

$$\mathbf{x}_{\tau_2}^{\text{ep}} = [\mathbf{x}_{\tau_2,0}^{\text{ep}}, \hat{e}_{\tau_1}]. \quad (29)$$

During supervised training, the observed upstream endpoint is available in the component-specific input dataset. During model execution, the selected upstream endpoint is passed forward. Gate and stand information may be used as context features, but assigning endpoints at stand level is outside the modeled taxiing scope.

I. Ground Control Agent: Path-Choice Component

The path-choice component represents the Ground Control Agent's selection of a graph path within the modeled taxiing scope. For taxi movement i , the available path candidates are the canonical path families in the OD-specific path-choice set C_{o_i} . The component is learned from observed graph-aligned taxi trajectories: it associates operational context with the canonical path family used in the historical movement, without assuming that an actual controller route instruction was observed.

For path candidate $C_{o_i,k}$, the path-choice distribution is

$$\hat{\pi}^{\text{path}}(C_{o_i,k} \mid \mathbf{x}_i) = \hat{P}(Y_i^{\text{path}} = C_{o_i,k} \mid X_i = \mathbf{x}_i), \quad C_{o_i,k} \in C_{o_i}. \quad (30)$$

Here, $\mathbf{x}_i$ contains the context features available for taxi movement i , and Y_i^{path} is the observed canonical path family.

When the learned path-choice stages do not provide an accepted output, directed graph search provides executable fallback geometry:

$$P_i^{\text{fb}} = A^*(G, o_i). \quad (31)$$

Here, A^* denotes directed graph search on the airport layout graph between the endpoints of OD pair o_i , respecting movement direction [15]. Fallback paths preserve executability but are not interpreted as learned Ground Control Agent choices.

J. Aircraft Operator Agent: Speed-Profile Component

The speed-profile component represents aircraft-side movement execution along a selected graph path. Endpoint selection and path choice determine where the aircraft travels; the speed-profile component determines how speed varies along that path. Its output is therefore a path-progress-dependent target speed profile rather than a discrete class label.

Definition 6 (Edge-speed observation and speed-profile residual). *The speed-profile component represents observed edge-relative speed behavior through accepted edge-speed observations and models deviations from an edge-level baseline using speed-profile residuals.*

For taxi movement i and directed edge e , an accepted edge-speed observation is represented on a normalized edge-progress grid:

$$v_{i,e}^{\text{obs}}(\xi_m), \quad \xi_m \in \Xi, \quad \Xi = \{\xi_1, \dots, \xi_M\} \subset [0, 1]. \quad (32)$$

The implemented default uses $M = 21$ grid points. An edge-speed observation is accepted only when the traversal has sufficient trajectory-point evidence and satisfies the selected plausibility checks. The point-density criterion is

$$n_{i,e} \geq \max \left(n_{\min}, \left\lceil \rho_{\min} \frac{l_e}{100} \right\rceil \right), \quad (33)$$

where $n_{i,e}$ is the number of trajectory points available for the edge traversal, l_e is the edge length, $n_{\min}$ is the absolute minimum point count, and $\rho_{\min}$ is the minimum point density per 100 m. Additional accepted-observation checks use traversal coverage, speed-sample ratio, and backward-step thresholds from the selected extraction parameter set.

The edge-level baseline speed profile is the median observed speed curve over the accepted edge-speed observations:

$$\bar{v}_e(\xi_m) = \text{median}_{i \in \mathcal{I}_e} v_{i,e}^{\text{obs}}(\xi_m), \quad (34)$$

where $\mathcal{I}_e$ is the set of accepted edge-speed observations for edge e . The speed-profile residual is the deviation from this edge-level baseline:

$$r_{i,e}(\xi_m) = v_{i,e}^{\text{obs}}(\xi_m) - \bar{v}_e(\xi_m). \quad (35)$$

The learned component estimates residual profiles rather than absolute speed profiles. The predicted edge-relative speed profile is

$$\hat{v}_{i,e}(\xi_m) = \max \{0, \bar{v}_e(\xi_m) + \hat{r}_{i,e}(\xi_m)\}. \quad (36)$$

When no accepted empirical or CART residual is available, the component falls back to the edge-level baseline:

$$\hat{r}_{i,e}(\xi_m) = 0. \quad (37)$$

Let $P_i^{\text{sel}} = \langle e_1, \dots, e_{n_i^P} \rangle$ denote the selected path for taxi movement i . The path-level target speed profile is assembled by placing each predicted edge-relative profile along this path:

$$\hat{v}_i(s) = \hat{v}_{i,e_j}(\xi), \quad s = \sum_{\ell < j} l_{e_\ell} + \xi l_{e_j}, \quad e_j \in P_i^{\text{sel}}. \quad (38)$$

Supported empirical context cells and CART leaves store median residual curves and 10th/90th percentile residual curves. The maximum operator in Eq. (36) keeps the target speed physically valid.

K. Ground Control Agent: Local Conflict-Resolution Component

The local conflict-resolution component represents movement-derived interaction behavior at non-runway protected taxi resources within the modeled taxiing scope. It focuses on pairwise encounters where two aircraft would compete for the same local resource before any conflict-resolution behavior is applied. The component excludes runway crossings, runway resources, rerouting, pushback, apron-internal movement, and same-path following interactions, which are handled outside this learned component.

Definition 7 (Conflict event and movement-derived labels). *A conflict event is a pairwise event in which two free-motion trajectories compete for the same protected taxi resource within overlapping or insufficiently separated time intervals. The associated priority, resolution-action, and delay labels are inferred from trajectory behavior and free-motion counterfactuals; they are not observed ATC commands.*

The protected-resource set contains graph nodes, apron-boundary resources, and reviewed geometric crossing points:

$$\mathcal{R}^{\text{prot}} = \mathcal{R}^{\text{node}} \cup \mathcal{R}^{\text{apron}} \cup \mathcal{R}^{\text{merge}} \cup \mathcal{R}^{\text{cross}}, \quad (39)$$

where $\mathcal{R}^{\text{node}}$ contains protected graph nodes, $\mathcal{R}^{\text{apron}}$ contains apron-boundary resources, $\mathcal{R}^{\text{merge}}$ contains merge resources, and $\mathcal{R}^{\text{cross}}$ contains reviewed geometric crossings where paths intersect without sharing a graph node.

For aircraft i , the free-motion trajectory is the trajectory that would result from the selected path and speed-profile output before local conflict resolution is applied:

$$X_i^{\text{free}}(t) = \left(\mathbf{p}_i^{\text{free}}(t), v_i^{\text{free}}(t), s_i^{\text{free}}(t) \right). \quad (40)$$

Here, $\mathbf{p}_i^{\text{free}}(t)$, $v_i^{\text{free}}(t)$, and $s_i^{\text{free}}(t)$ are the free-motion position, speed, and path progress at time t .

For protected taxi resource r , the free-motion occupancy interval of aircraft i is

$$I_{i,r}^{\text{free}} = [t_{i,r}^{\text{in}}, t_{i,r}^{\text{out}}]. \quad (41)$$

A pairwise conflict event between aircraft i and j at resource r is represented as

$$\chi_{i,j,r} = \mathbf{1} \left\{ I_{i,r}^{\text{free}} \cap I_{j,r}^{\text{free}} \neq \emptyset \vee |t_{i,r}^{\text{pass}} - t_{j,r}^{\text{pass}}| < \Delta_r \right\}. \quad (42)$$

This expression captures the two ways in which a local conflict can arise: overlapping free-motion occupancy intervals or passage times that are closer than the resource-specific separation threshold Δ_r . The conflict-generation procedure applies this logic through resource-specific checks for protected nodes, apron boundaries, merge resources, and reviewed geometric crossings.

The priority label records which aircraft passed the protected taxi resource first in the observed trajectory. Cases with equal or unresolved passing times are excluded from the priority-label training set.

$$y_{i,j,r}^{\text{prio}} = \begin{cases} i, & t_{i,r}^{\text{obs}} < t_{j,r}^{\text{obs}}, \\ j, & t_{j,r}^{\text{obs}} < t_{i,r}^{\text{obs}}. \end{cases} \quad (43)$$

The movement-derived resolution-action label for the own aircraft is

$$y_{i,j,r}^{\text{act}} \in \{\text{PROCEED}, \text{SLOW_DOWN}, \text{HOLD_SHORT}\}. \quad (44)$$

When available, the delay target is the difference between observed and free-motion resource-passing time:

$$y_{i,j,r}^{\text{delay}} = t_{i,r}^{\text{obs}} - t_{i,r}^{\text{free}}. \quad (45)$$

These labels approximate how the interaction was resolved in the observed movement data. They are inferred from trajectory timing and free-motion counterfactuals, not from ATC instruction logs or directly observed controller commands.

L. Aircraft Operator Agent: Following-and-Merge Guard

The following-and-merge guard is a deterministic execution rule for aircraft-side execution. It is applied after endpoint selection, path choice, and speed-profile construction. Its purpose is to keep the model-executed trajectory safe and executable when aircraft follow the same path, approach an occupied downstream edge, or enter a merge resource. The rule is not a learned behavioral component.

At each time step, the executed speed is bounded by the target speed profile and the safe-speed limit:

$$v_i^{\text{exec}}(t + \Delta t) = \min \{v_i^{\text{target}}(t + \Delta t), v_i^{\text{safe}}(t + \Delta t)\}. \quad (46)$$

Here, v_i^{target} is the speed requested by the speed-profile component, while v_i^{safe} is the deterministic limit imposed by downstream occupancy, following, and merge conditions.

For a leader j and follower i , the protected center-gap requirement is

$$d_{ij}(t) \geq d_{\min} + \rho_i + \rho_j, \quad (47)$$

where $d_{ij}(t)$ is the center-to-center distance along the relevant path, $d_{\min} = 30$ m is the required clearance between protected radii, and ρ_i, ρ_j are aircraft protected radii. The implemented rule also uses category-dependent radii, an acceleration bound $a_{\max} = 0.4$ m/s², a deceleration bound $b_{\max} = 0.5$ m/s², an 8 s following time gap, and soft-gap logic to reduce stop-go chatter.

The acceleration envelope constrains the executed speed between consecutive time steps:

$$v_i(t) - b_{\max} \Delta t \leq v_i^{\text{exec}}(t + \Delta t) \leq v_i(t) + a_{\max} \Delta t. \quad (48)$$

A merge-entry hold is triggered when a downstream merge resource is occupied and the predicted entry would violate the protected gap or safe-speed condition:

$$h_i^{\text{merge}}(t) = \mathbf{1} \{ \Omega_i^{\text{merge}}(t) = 1 \}. \quad (49)$$

Here, $\Omega_i^{\text{merge}}(t)$ denotes the merge-entry violation predicate. The detailed safe-speed function also accounts for edge-entry buffers, release margins, braking-envelope distance, and downstream occupancy. These terms are implementation-level execution constraints, not learned behavior.

M. Runway Control Agent: Runway-Departure Release Rule

The runway-departure release rule is a deterministic execution rule for outbound aircraft at runway-entry stopbars. It represents the runway-boundary release constraint used during integrated model validation, not a learned runway-control clearance model. Queueing abstractions provide a basis for representing runway departure processes [25].

For runway stopbar group r , let $Q_r(t)$ be the first-come, first-served outbound queue. For the first aircraft in a runway-stopbar queue, $t_i^{\text{rel}} = t_i^{\text{arr}}$. If aircraft i follows aircraft j , the release time is

$$t_i^{\text{rel}} = \max \{ t_i^{\text{arr}}, t_j^{\text{rel}} + \Delta_{j,i}^{\text{dep}} \}. \quad (50)$$

Here, t_i^{arr} is the time at which aircraft i reaches the runway-entry stopbar, and $\Delta_{j,i}^{\text{dep}}$ is the required departure separation interval between predecessor j and aircraft i .

The implemented departure separation interval is

$$\Delta_{j,i}^{\text{dep}} = \max \left\{ \Delta_{j,i}^{\text{base}}, \Delta_{j,i}^{\text{wake}} \right\} + \Delta_{j,i}^{\text{SID}}. \quad (51)$$

The current integrated model-validation setup uses $\Delta^{\text{base}} = 80$ s, a RECAT-EU wake-turbulence departure minimum $\Delta_{j,i}^{\text{wake}}$ where available, and a 20 s same-SID add-on $\Delta_{j,i}^{\text{SID}}$. Intermediate runway-entry stopbars are diagnostic only and do not add a release penalty. The rule therefore provides a deterministic runway-boundary release constraint for model execution.

IV. Experimental Setup

A. Schiphol Case Study and Input Scope

The main case study uses aircraft taxi operations at Amsterdam Airport Schiphol from January 2024 to December 2025. The input consists of monthly taxi-trajectory records derived from VEMMIS operational records and processed according to the pipeline in Section III.C. The modeled taxiing scope, endpoint definitions, and exclusion rules are defined in the Methodology and are kept fixed throughout the main experiments.

The main experimental setup focuses on the observed-behavior baseline for aircraft taxi trajectories. Three appendix case studies complement this baseline. Appendix D (Construction Data) examines the effect of temporary infrastructure changes, Appendix E (Operational Constraints) examines known operational rules and procedure constraints, and Appendix F (Apron Area) tests how the methodology transfers to a more complex apron-area setting.

B. Data Split

The learned behavioral components are trained, tuned, and evaluated using a pooled 2024-2025 week-blocked, cluster-stratified training, validation, and test split. The split unit is the ISO week, using the week-numbering standard of the International Organization for Standardization (ISO): all observations from the same week are assigned to the same partition. This avoids flight-level random partitioning, which could place temporally adjacent operations with similar runway configurations, demand patterns, weather conditions, or traffic states in different partitions.

Weeks from both years are pooled before splitting so that no partition represents only one calendar year or operating regime. Training, validation, and test partitions therefore each contain operational conditions from the full two-year period.

Stratification is based on weekly operational profiles. Let $\mathcal{W}$ be the set of ISO weeks and let $\mathbf{u}_w$ be the profile vector for week w :

$$\mathbf{u}_w = [u_{w,1}, u_{w,2}, \dots, u_{w,p}]. \quad (52)$$

The profile summarizes operational conditions such as traffic volume, inbound/outbound mix, runway configuration, apron and OD-pair mix, aircraft and wake category, airline/operator composition, temporal and seasonal factors, demand state, and weather or operating state. Similar-day grouping has been used in air traffic flow management to compare operating periods with related demand and operational characteristics [22].

Weekly operational profiles are grouped with k-medoids clustering on a custom operational-distance matrix. The distance matrix combines normalized distances across feature groups so that no single feature group dominates the clustering. K-medoids is suitable here because each cluster is represented by an observed week, or medoid, rather than by a synthetic centroid [23]. The final split uses six operational-profile clusters for stratified assignment.

Complete weeks are assigned to the training, validation, and test partitions:

$$\begin{aligned} \mathcal{W} &= \mathcal{W}^{\text{train}} \cup \mathcal{W}^{\text{val}} \cup \mathcal{W}^{\text{test}}, \\ \mathcal{W}^a \cap \mathcal{W}^b &= \emptyset, \quad a, b \in \{\text{train, val, test}\}, \quad a \neq b. \end{aligned} \quad (53)$$

The target proportion is approximately 70/15/15 while maintaining balanced coverage of operational-profile clusters and calendar periods.

C. Model Training, Tuning, and Evaluation Workflow

All learned behavioral components use the same training, tuning, and evaluation workflow. Each component has its own component-specific input dataset and target variable, but the selection protocol is shared. Training observations

build the empirical and CART stages, validation observations guide feature-set selection and hyperparameter tuning, and test observations remain held out until the selected component configuration is fixed.

(1) Feature-importance ranking: For each component, an exploratory CART model is first fitted using the available context features. This model is not used as the final component. Its purpose is to rank context features before candidate feature-set screening. Classification trees rank features by impurity reduction for categorical targets, while regression trees rank features by squared-error reduction for speed-profile residual targets.

(2) Candidate feature-set screening: The ranked features are used to construct ordered candidate feature sets for the empirical and CART stages. Early candidates contain fewer features and therefore provide broader support; later candidates contain richer context but are more sparse. Candidate feature sets are evaluated on the validation partition using component-specific performance metrics and prediction-source diagnostics. The three strongest empirical feature sets and the three strongest CART feature sets are crossed to form the feature-set combinations passed to Bayesian hyperparameter optimization.

(3) Bayesian hyperparameter optimization: Bayesian hyperparameter optimization tunes the candidate component configurations from the screening step. Bayesian optimization is commonly used for expensive validation-based machine-learning objectives [28]. In this study, it tunes component-specific support thresholds, acceptance criteria, CART parameters, and fallback settings. The validation objective score combines predictive quality with prediction-source diagnostics. The test partition is not used during this step.

(4) Locked test evaluation: The configuration with the best validation objective score is selected as the final component configuration. It is then evaluated once on the held-out test partition using the metrics in Section IV.D. The resulting component outputs are exported for component-level validation and integrated model-validation diagnostics.

The selected component configuration is

$$\theta_c^* = \arg \max_{\theta \in \Theta_c} J_c^{\text{val}}(\theta), \quad (54)$$

where Θ_c is the candidate configuration space and J_c^{val} is the component-specific validation objective score. Objective weights are tuning settings, not universal evaluation metrics. Full feature-set screening results, Bayesian optimization traces, and training-data sensitivity diagnostics are reported in Appendix A (Model Calibration and Tuning).

D. Evaluation Metrics and Diagnostics

The same metric families are used during candidate feature-set screening, Bayesian optimization, and final held-out evaluation, but their roles differ. Validation metrics guide feature-set screening, hyperparameter tuning, and model selection. Test metrics are reported only after the selected component configuration is fixed. Full metric definitions, denominators, and component-specific diagnostic variants are provided in Appendix B (Verification and Validation).

Metrics are matched to the output type of each component. Discrete learned components are evaluated with classification performance, probabilistic predictive performance, calibration performance, and prediction-source diagnostics. The speed-profile component is evaluated with regression performance and predictive-uncertainty diagnostics. Integrated model-validation diagnostics are reported separately because they evaluate whether selected component outputs and deterministic execution rules remain coherent when executed together.

Path choice: Path choice is evaluated both as graph-path recovery and as path-candidate classification. Edge-weighted F1 compares the predicted path with the observed exact path using directed-edge overlap. Weighted F1 over path-candidate labels evaluates whether the predicted canonical path family matches the observed family, with class weights based on test support. Probabilistic predictive performance is reported with normalized Brier score and normalized predictive entropy, and calibration performance is reported with expected calibration error (ECE). Results are reported for all OD pairs and, where relevant, for the non-trivial subset that excludes OD pairs with only one available path candidate.

For edge-weighted F1, let O be the weighted overlap of the observed and predicted directed-edge multisets, L_{obs} the observed path length, and L_{pred} the predicted path length:

$$P_{\text{edge}} = \frac{O}{L_{\text{pred}}}, \quad R_{\text{edge}} = \frac{O}{L_{\text{obs}}}. \quad (55)$$

The edge-weighted F1 score is

$$F1_{\text{edge}} = \begin{cases} 2P_{\text{edge}}R_{\text{edge}}/(P_{\text{edge}} + R_{\text{edge}}), & P_{\text{edge}} + R_{\text{edge}} > 0, \\ 0, & P_{\text{edge}} + R_{\text{edge}} = 0. \end{cases} \quad (56)$$

When edge lengths are available, O , L_{obs} , and L_{pred} are length-weighted; otherwise, directed-edge counts are used.

Endpoint selection: Endpoint selection is evaluated separately for the inbound runway stopbar, inbound apron boundary, outbound apron boundary, and outbound runway stopbar targets. Weighted F1 measures support-weighted endpoint-class recovery, while macro F1 is used as a diagnostic for rare endpoint classes. Probabilistic predictive performance is reported with normalized Brier score and normalized predictive entropy, and calibration performance is reported with ECE. Confusion matrices are used as diagnostics to identify whether errors occur between nearby or operationally similar endpoints.

Speed-profile component: The speed-profile component is evaluated as a continuous residual profile around an edge-level baseline. The main error metric is root-mean-square error in knots:

$$RMSE_v = \sqrt{\frac{1}{|I|M} \sum_{(i,e) \in I} \sum_{m=1}^M \left(\hat{v}_{i,e}(\xi_m) - v_{i,e}^{\text{obs}}(\xi_m) \right)^2}. \quad (57)$$

Here, I is the set of evaluated edge-speed observations and M is the number of edge-progress grid points. Distributional performance is summarized with median, interquartile-range, and high-percentile RMSE values. Predictive uncertainty is evaluated with central 80% prediction-interval diagnostics, including interval score, interval width, empirical coverage, and coverage gap. Edge-level RMSE maps are used to localize where residual speed prediction is weaker.

Local conflict resolution: Local conflict resolution is evaluated using movement-derived priority labels and resolution-action labels. Priority-label and resolution-action performance are reported with target-class weighted F1. Probabilistic predictive performance is summarized with normalized Brier score and normalized predictive entropy, and calibration performance is reported with ECE. Delay behavior is evaluated with mean absolute error (MAE) when delay targets are available. Confusion matrices are used to identify priority swaps and resolution-action substitutions. These metrics compare against labels inferred from trajectory behavior and free-motion counterfactuals, not directly observed ATC instruction logs.

For model selection, each component uses a weighted validation objective score. Objective terms are normalized so that larger values indicate better validation performance. Lower-is-better terms, such as error, Brier score, calibration error, fallback share, and unsupported share, are inverted or penalized before aggregation. Objective weights are component-specific tuning settings and are therefore not interpreted as universal evaluation metrics.

All learned components report prediction-source diagnostics. These diagnostics show how often outputs come from empirical support, CART generalization, baseline behavior, deterministic execution rules, fallback rules, or unsupported states. They are interpreted together with predictive performance: high performance with high fallback use is weaker evidence of learned behavior than high performance from empirical-stage or CART-stage outputs.

Integrated model-validation diagnostics are separate from learned-component metrics. They check whether selected model-execution outputs can be consumed together in a graph-based execution sequence. Reported diagnostics include model-execution completion rate, protected-resource events, following-and-merge guard activations, runway-departure release events, timing errors, conflict-event agreement, queueing diagnostics, and trajectory-consistency diagnostics.

E. Training-Data Sensitivity Design

The training-data sensitivity analysis tests whether component performance depends mainly on the amount of training data, the time span covered by the training data, or both. The validation and test partitions remain fixed while only the training partition is varied. This isolates training-data effects from changes in the evaluation population.

Two training-subset families are used. Fraction-based subsets include 5%, 10%, 25%, 50%, 75%, and 100% of the stratified training weeks. Chronological-window subsets use expanding historical windows up to the full training period. The resulting diagnostics show whether additional observations, broader temporal coverage, or their combination improves performance, learned support, and fallback share. Full sensitivity curves and component-level interpretations are provided in Appendix A (Model Calibration and Tuning).

F. Integrated Model-Validation Setup

The integrated model-validation setup combines the selected learned behavioral components from Sections III.H–III.K with the deterministic execution rules from Sections III.L and III.M. The purpose is to check whether the component outputs, graph-based movement representation, protected-resource logic, and deterministic execution rules remain coherent when executed together.

Three two-hour historical validation scenarios are selected from the held-out test partition: a preferential runway-configuration scenario, a conflict-rich scenario, and a high-runway-variability scenario. Each scenario stresses a different aspect of the integrated MAS model. The preferential runway-configuration scenario represents a high-volume primary operating setup, the conflict-rich scenario emphasizes protected-resource interactions, and the high-runway-variability scenario tests the model under frequent runway-configuration changes.

Each scenario uses the same historical-scenario model-execution setup. Historical records provide activation times, movement direction, runway and apron context, and operational state variables. During model execution, the endpoint-selection and path-choice components select the scoped movement endpoints and path candidate, and the speed-profile component constructs the target speed profile along the selected path. The protected-resource monitor detects local encounters, while deterministic execution rules handle same-path following, merge-entry holds, separation speed constraints, braking, and runway-departure release at runway-entry stopbars.

Table 2 summarizes the scenarios and traffic composition. Detailed scenario-level motion, conflict-event, and runway-departure queueing diagnostics are provided in Appendix B (Verification and Validation).

Table 2 Two-hour historical validation scenarios used for integrated model validation.

Scenario	Time window	Total movements	Inbound	Outbound	Runway configurations
Preferential runway configuration	2025-06-09 05:48–07:48	192	116	76	3
Conflict-rich	2024-05-20 07:30–09:30	142	65	77	2
High runway variability	2025-01-02 11:00–13:00	165	82	83	7

V. Results

A. Data Processing, Reconstruction, and Dataset Characterization

The processing pipeline produces a large graph-aligned taxi dataset from the VEMMIS operational records. The main losses occur during taxiing-scope extraction and graph reconstruction, while the subsequent context-assignment step removes only a small number of additional trajectories. Table 3 summarizes the retention sequence by movement direction.

Table 3 Data-retention summary from raw schedule records to graph-aligned and context-assigned taxi trajectories. Counts are rounded to the nearest thousand where shown with k .

Processing stage	Inbound count	Inbound share	Outbound count	Outbound share	Interpretation
Raw candidate records	~ 462k	100.00%	~ 464k	100.00%	Candidate records before taxiing-scope extraction and graph reconstruction
Graph-aligned data	~ 453k	98.04%	~ 431k	92.71%	Main retention loss occurs before path-catalog construction
Context-assigned data	~ 453k	98.00%	~ 431k	92.69%	Context joining removes only 266 additional trajectories

Most excluded cases are caused by irreparable surveillance-data corruption in the source records. Minor inconsistencies can often be corrected during preprocessing, but unrealistic trajectory logs remain the dominant exclusion source. The directional difference in retention indicates that outbound taxi trajectories contain more reconstruction failures than inbound taxi trajectories.

The retained taxi movements are aligned to the airport layout graph with high reconstruction quality. The median movement-level 95th-percentile point-to-path projection error is 8.71 m. Evidence-aware fallback is used for less than 1% of trajectories, and reduced-graph shortest-path recovery is needed in only four cases. High-error exclusions, defined by a 95th-percentile point-to-path projection error above 80 m, represent 0.46% of the retained set and are spatially concentrated around runway 27, runway 09, and apron S. Detailed map-matching repair logic, projection-error diagnostics, and reconstruction-failure breakdowns are provided in Appendix B (Verification and Validation).

The graph-aligned taxi trajectories are then organized into exact observed paths and grouped into canonical path families. The resulting catalog contains 883,771 graph-aligned taxi trajectories, 12,735 exact observed paths, and 3,002 canonical path families. This reduces the average number of OD-specific path candidates from 8.81 exact observed paths to 2.09 canonical path families. The reduction lowers sparsity while retaining the dominant operational path candidates used by the path-choice component. Expanded path-catalog statistics and canonical path-family examples are provided in Appendix B (Verification and Validation).

The final dataset is also enriched with operational context features. The most common runway configuration accounts for 19.22% of movements, midday is the predominant time group, most movements occur under very-high-density conditions, and aircraft class D and operator KLM are the most frequent. These distributions show that the dataset is broad enough to support context-conditioned behavior identification, while still reflecting the dominant operating patterns at Schiphol. Expanded operational-context and error-concentration analyses are provided in Appendix C (Operational Interpretation of Selected Model Configurations).

Overall, the data-processing results show that the pipeline produces a large, graph-aligned, and context-rich taxi dataset. The retained data provide sufficient reconstruction quality, path-catalog coverage, and operational-context coverage for the learned behavioral components.

B. Model Training and Tuning

The model-training results summarize how the final learned behavioral components are selected before held-out testing. Each component uses its own component-specific input dataset and target variable, but all components follow the same validation-stage workflow: candidate context features are ranked, empirical and CART feature sets are screened, and selected candidate configurations are tuned with Bayesian hyperparameter optimization. The test partition remains unused during this process.

Table 4 lists the selected empirical and CART feature sets and the validation objective score for each learned behavioral component. The scores identify the selected configuration within each component, but they are not directly comparable across components because each validation objective uses component-specific performance and prediction-source terms. Here, BZO denotes the Schiphol operational phase variable and ETA denotes estimated time of arrival.

Table 4 Selected feature sets and validation objective scores per learned behavioral component.

Component	Empirical selected context	CART selected context	Validation objective score
Path choice	Runway configuration	Runway configuration; scheduled gate area; 30-min direction-demand count; 30-min local-density count; taxi start minute of day; minutes since runway-configuration change; minutes until runway-configuration change	0.898
Endpoint: inbound runway stopbar	Exit speed +1000 m; aircraft class; local density; exit speed +500 m	Exit speed +1000 m; aircraft class; local density; exit speed +500 m; exit speed at threshold; time of day; exit speed +250 m; airline	0.705
Endpoint: apron boundary	Aircraft class; gate area; local density	Aircraft class; gate area; local density; runway role; time of day; weekday; BZO phase; same-apron demand; airline	0.799
Endpoint: outbound runway stopbar	Aircraft class; SID; BZO phase; time of day; local density; weekday; runway configuration; airline	Aircraft class; SID; BZO phase; time of day; local density; weekday; runway configuration	0.743
Speed-profile component	Previous-edge speed; aircraft class; apron	Previous-edge speed; airline; aircraft class	0.825
Local conflict resolution	Relative ETA bin; own direction; resource group; other direction	Relative ETA; own distance to resource; other distance to resource; own direction	0.867

The selected feature sets are consistent with the operational meaning of each component. Path choice uses runway configuration as a compact empirical context, while the CART stage adds gate area, demand, density, timing, and runway-configuration transition features. Endpoint selection uses target-specific feature sets; apron-boundary targets share the same feature stack, while runway-stopbar targets require more runway-side and speed-related context. The speed-profile component uses residual features, and local conflict resolution emphasizes timing, direction, and resource geometry.

The training-data sensitivity analysis supports the same interpretation. Larger training sets generally reduce fallback dependence and improve validation performance, but the effect is component-specific. Path choice benefits strongly from additional data, endpoint-selection fallback shares decrease especially for runway-side targets, and the speed-profile and local conflict-resolution components already have broad learned support under smaller subsets. Full candidate feature-set screening, Bayesian optimization traces, and training-data sensitivity diagnostics are provided in Appendix A (Model Calibration and Tuning).

In summary, the final learned behavioral components are selected through a locked validation-stage process. The selected configurations are then evaluated once on the held-out test partition in the next subsection.

C. Component-Level Model Performance

Table 5 summarizes the held-out test performance of the final learned behavioral components. The table reports the primary metric for each component together with probabilistic or uncertainty diagnostics and prediction-source

diagnostics. Prediction-source shares are essential for interpretation because empirical-stage and CART-stage outputs provide learned behavioral evidence, while fallback outputs preserve executability without representing learned behavior.

Table 5 Locked held-out component-validation summary. Values are computed from machine-readable figure-support summaries.

Component	Target/output type	Primary metric	Probabilistic or uncertainty diagnostic	Prediction-source diagnostic
Path choice	Canonical path family and graph path	Exact observed path F1 mean 0.926; canonical weighted F1 mean 0.861	Normalized Brier mean 0.087; ECE 0.010	86.53% empirical; 10.17% CART; 3.30% fallback
Endpoint selection	Inbound runway stopbar; inbound apron boundary; outbound apron boundary; outbound runway stopbar	Weighted F1: 0.696; 0.898; 0.902; 0.766	Normalized Brier mean: 0.196; 0.066; 0.062; 0.140. ECE: 0.012; 0.004; 0.002; 0.016	Fallback share: 1.01%; 0.32%; 0.30%; 6.90%; no unsupported cases
Speed-profile component	Edge-relative speed profiles	Mean RMSE 2.279 kt; mean 90th-percentile absolute error 2.962 kt; mean bias 0.089 kt	Central-80% interval score 9.340 kt; coverage 84.21%; coverage gap 4.21 percentage points	91.10% empirical; 8.87% CART; 0.033% fallback; 0.0002% unsupported
Local conflict resolution	Priority and resolution-action labels	Weighted F1: priority 0.856; resolution action 0.830; delay MAE 29.7 s	Normalized Brier mean: 0.104 and 0.126; ECE: 0.054 and 0.060	45.62% empirical; 54.38% CART; no fallback or unsupported cases

The component-level results show that the learned behavioral components differ in difficulty and support. Path choice captures most observed path geometry, although non-trivial OD pairs with multiple path candidates remain harder than single-candidate OD pairs. Endpoint selection is strongest for apron-boundary targets and weaker for rare runway-stopbar classes. The speed-profile component achieves low aggregate speed error with near-complete learned residual support. Local conflict resolution recovers the main inferred priority and movement-derived resolution-action patterns, but remaining errors are concentrated in priority swaps and proceed or slow-down substitutions.

These results support the central claim that trajectory-derived observations can identify selected taxi-movement behaviors. They also show why prediction-source diagnostics are necessary: predictive performance alone does not distinguish learned behavior from fallback-based executability. Full target-level plots, non-trivial-case views, confusion matrices, prediction-source distributions, edge-level speed diagnostics, and component-specific interpretation notes are provided in Appendix B (Verification and Validation).

D. Integrated Model-Validation Diagnostics

The integrated model-validation results evaluate whether selected component outputs remain coherent when they are combined in historical-scenario model execution. The integrated setup connects path choice, speed-profile behavior, local conflict resolution, protected-resource monitoring, same-path following, merge-entry holds, and runway-departure release. These results are interpreted as component-interoperability and emergent-timing diagnostics, not as operational deployment validation.

The three scenarios use historical endpoints to isolate the effects of path timing, speed-profile behavior, protected-resource conflicts, following-and-merge constraints, and runway-departure release. Table 6 summarizes the integrated model-validation results.

Table 6 Compact integrated model-validation summary. Planning completion is computed from planned and input movement counts in the machine-readable scenario run summaries.

Historical validation scenario	Planning completion	Taxi-time / timing-error summary	Protected-resource / conflict-event diagnostic	Queueing / runway-departure release diagnostic
Preferential runway configuration	100% (192/192)	Realized mean taxi-time error +0.30 min; nominal −0.21 min	0.42 model-executed vs. 0.19 historical conflict events/movement; conflict-pair F1 24.6%	76 compared runway releases; mean absolute release-time error 1.26 min; mean queue delay 0.55 min
Conflict-rich	100% (142/142)	Realized mean taxi-time error −0.87 min; nominal −1.50 min	0.69 model-executed vs. 1.49 historical conflict events/movement; conflict-pair F1 18.0%	77 compared runway releases; mean absolute release-time error 2.45 min; mean queue delay 0.55 min
High runway variability	100% (165/165)	Realized mean taxi-time error +7.67 min; nominal +0.01 min	0.46 model-executed vs. 0.12 historical conflict events/movement; conflict-pair F1 19.2%	76 compared runway releases; mean absolute release-time error 1.61 min; mean queue delay 0.55 min

Planning completion is 100% in all three scenarios, which shows that the selected outputs can be combined into executable graph-based movement plans. The deterministic execution rules have little effect on pointwise speed-profile

error, but they can strongly affect timing. The preferential runway-configuration scenario changes from a nominal taxi-time error of -0.21 min to a realized error of $+0.30$ min. The conflict-rich scenario remains early relative to the processed historical taxi time, changing from -1.50 min to -0.87 min. The high-runway-variability scenario is the main timing outlier: its nominal taxi-time error is $+0.01$ min, while the realized model-execution error increases to $+7.67$ min. This indicates that the timing deviation is driven by downstream component interactions and release constraints, not by the nominal speed-profile component alone.

Conflict-pair agreement remains limited across the scenarios. These values should not be interpreted as component-level local conflict-resolution accuracy. They measure whether integrated model execution produces similar protected-resource encounters after path choice, speed-profile behavior, following-and-merge constraints, and runway-departure release effects are combined. Scenario-level motion, conflict-emergence, and runway-departure queueing diagnostics are provided in Appendix B (Verification and Validation).

Overall, the integrated model-validation diagnostics show that the selected model-execution outputs can be combined into an executable graph-based sequence. They also show that deterministic execution rules create timing effects that are not visible in isolated component metrics.

VI. Discussion

The results show that surface-surveillance trajectories can be transformed into a support-aware observed-behavior baseline for aircraft taxi operations. The framework identifies selected agent behaviors from historical data, evaluates them at component level, and checks their interoperability through integrated model validation. This section summarizes the main implications and limitations of the results.

Takeaway 1: Surface-surveillance data can support learned components for selected taxi-operation behaviors

The results show that VEMMIS operational records can be converted into learned behavioral components without requiring ATC instruction logs. Graph-aligned taxi trajectories, context features, and component-specific input datasets provide enough structure to estimate endpoint selection, path choice, speed-profile behavior, and local conflict resolution.

This transformation depends on more than data volume. It requires taxiing-scope extraction, graph-constrained map matching, canonical path-family grouping, and prediction-source diagnostics. The high retention rate, low projection errors, and broad context coverage show that this is feasible for the taxi scope at Schiphol. Surface-surveillance data can therefore support structured identification of agent behavior when the raw observations are converted into a common graph-based representation.

Takeaway 2: Representation quality determines what the learned components can recover

The airport layout graph and path catalog are not neutral preprocessing outputs. They define the spatial and operational representation on which all learned components depend. When the graph-aligned taxi trajectory is accurate, the downstream path, speed, endpoint, and conflict labels are meaningful. When the representation is incomplete or unstable, the learned components inherit that limitation.

Localized reconstruction issues illustrate this dependency. In the P-Holding area, aircraft may move over paved surface in ways that do not follow the modeled centerlines precisely, which increases point-to-path projection error. Apron S further shows that a static graph cannot represent temporary construction states unless those states are included in the data model. These cases show that graph accuracy, endpoint definitions, and layout-state information are not only reconstruction details; they directly affect behavior identification. Detailed reconstruction and path-catalog diagnostics are provided in Appendix B (Verification and Validation), and the construction case is analyzed in Appendix C (Operational Interpretation of Selected Model Configurations).

Takeaway 3: Context features and prediction-source diagnostics are part of the model result

The selected feature sets reflect the operational structure of aircraft taxi operations. Path choice depends on runway configuration, gate area, demand, density, timing, and runway-configuration transitions. Endpoint selection is target-specific: apron-boundary targets are strongly tied to gate and apron context, while runway-stopbar targets depend more on runway-side, aircraft, speed, SID, and operational-state information. Speed-profile behavior depends on aircraft class, operator, previous-edge speed, and local motion continuity. Local conflict resolution depends mainly on timing, direction, and protected-resource geometry.

Prediction-source diagnostics are needed to interpret these results. A component with high predictive performance but high fallback use provides weaker evidence of learned behavior than a component whose outputs come mainly from the empirical or CART stages. In the final selected configurations, fallback use is generally low, while the balance between empirical and CART outputs differs by component. This confirms that prediction-source diagnostics are not only implementation checks; they describe where the model is supported by observed behavior and where it relies on fallback-based executability.

Takeaway 4: Component performance follows the operational structure of each target

The component-level results are consistent with the operational difficulty of the targets. Path choice performs well because many OD pairs have stable and repeated canonical path families. Performance decreases for non-trivial OD pairs because these cases involve real choices between competing path candidates rather than structurally determined paths.

Endpoint selection is more heterogeneous. Apron-boundary targets perform better because they are closely tied to gate and apron area. Runway-stopbar targets are harder because runway exit behavior depends on aircraft-side execution after landing, while runway entry behavior depends on operational constraints and departure procedures. Some endpoint errors are therefore better interpreted as missing operational constraints than as random classification errors.

The speed-profile component performs well as a baseline-plus-residual model. The edge-level baseline captures typical speed behavior, while residual estimates account for context-specific deviations. Local conflict resolution also captures the main movement-derived priority and resolution-action patterns, but action-label distinction remains harder than priority recovery. In particular, proceed and slow-down actions can be difficult to separate from trajectory data alone, and hold-short cases are relatively rare.

Takeaway 5: Integrated model validation reveals coupling effects that component metrics cannot show

Component-level validation evaluates isolated learned outputs. Integrated model validation tests a different question: whether selected outputs can be combined with deterministic execution rules in a coherent graph-based model execution. This distinction is important because timing, conflicts, and queues emerge from the interaction between components.

The historical-scenario results show that deterministic execution rules can create timing deviations that are not visible in isolated speed-profile metrics. The high-runway-variability scenario is the clearest example: nominal timing remains close to the processed historical taxi time, while realized model-execution timing drifts after following, merge, and runway-departure release constraints are applied. This indicates that the main timing error is caused by downstream interactions, not by the nominal speed-profile component alone.

Conflict-pair agreement is also sensitive to integration effects. A small change in path choice, speed profile, release timing, or following behavior can change whether two aircraft meet at the same protected taxi resource. Low integrated conflict-pair F1 should therefore not be interpreted as direct failure of the local conflict-resolution component. It shows that protected-resource encounters are highly sensitive to the coupled timing of the full model execution. Detailed integrated diagnostics are provided in Appendix B (Verification and Validation).

Limitation 1: Raw operational data cannot be shared

The first limitation is data accessibility. VEMMIS is an internal LVNL database, so the raw operational records cannot be distributed with the paper. This limits direct external reproducibility from raw data. The paper addresses this by defining the processing steps, data units, derived data products, model inputs, evaluation design, and aggregate diagnostics. The resulting methodology can be reproduced on comparable surface-surveillance datasets, but the exact Schiphol records remain proprietary.

Limitation 2: The modeled taxiing scope is deliberately restricted

The main model focuses on runway-stopbar–apron-boundary aircraft taxi operations. This scope excludes apron-internal movements, gate-area movements, pushback, de-icing, private-aviation movements, eastern maintenance-area movements, runway 22/04, and full tower-clearance behavior. These exclusions make the modeling problem tractable and keep the learned components focused on the main runway–apron taxiing process, but they also limit the operational completeness of the model.

The apron-area case study in Appendix F (Methodology Transfer to Apron-area Taxi Movements) shows that parts of the methodology can transfer to more complex surface areas, but not without adaptation. Apron movements require

different phase extraction, boundary logic, and graph-alignment assumptions. The main paper should therefore be interpreted as a model for aircraft taxi operations within the defined taxiing scope, not as a complete model of all airside surface movement.

Limitation 3: Static graph representation misses temporary layout states

The airport layout graph provides the common spatial reference for reconstruction, path catalogs, speed profiles, and conflict-event extraction. However, it is static. It cannot fully represent temporary layout changes, construction closures, or local paved-surface behavior that does not follow the modeled centerlines.

The construction case in Appendix C (Operational Interpretation of Selected Model Configurations) illustrates this limitation. When temporary construction changes the available taxiway corridors, a model trained without construction-state context may assign probability to path candidates that were historically avoided or unavailable during that period. This shows that future versions should include time-dependent layout states or construction-context features when such information is available.

Limitation 4: Missing operational rules can appear as prediction errors

Some prediction errors reflect missing operational constraints rather than weak statistical learning. For example, endpoint-selection and path-choice errors may arise when runway-specific restrictions, temporary procedures, sector practices, or local availability rules are not encoded as context features. The model can only infer these effects indirectly from the available historical observations.

Appendix E (Operational Constraints and Availability Rules) shows that known operational rules can explain some apparently anomalous path or endpoint outputs. This suggests that a stronger model should combine trajectory-derived learning with explicit operational-rule information where such rules are known and active. The learned components should therefore be viewed as an observed-behavior baseline, not as a complete representation of all operational procedures.

Limitation 5: Local conflict labels are movement-derived, not observed commands

The local conflict-resolution component uses labels inferred from trajectory behavior and free-motion counterfactuals. These labels are useful for reproducing movement-derived interaction patterns, but they do not reveal actual controller intent. A slow-down or hold-short label indicates how the observed movement differed from its free-motion counterfactual; it is not an observed ATC instruction.

The protected-resource definition is also incomplete. It includes protected nodes, apron-boundary resources, merge resources, and reviewed geometric crossings, but close non-intersecting edges can still represent operationally relevant proximity. Future work should improve protected-resource definitions and, where possible, compare movement-derived labels with operational logs or expert-reviewed interaction cases.

Limitation 6: Deterministic execution rules simplify integrated model behavior

The following-and-merge guard and runway-departure release rule preserve safe and executable model execution, but they are simplified deterministic rules. They do not represent the full range of controller decisions, pilot behavior, runway sequencing logic, or tactical local adjustments. Their role is to make integrated model execution feasible and interpretable, not to provide learned evidence of those behaviors.

The integrated model-validation results reveal this limitation. Deterministic execution rules can induce timing drift, especially in scenarios with high runway variability or strong queueing interactions. These deviations are useful diagnostics because they show where the current integrated model requires refinement. They also clarify the boundary of the contribution: the paper validates component interoperability and emergent model behavior, but it does not claim operational deployment validity.

VII. Conclusion

This paper developed and evaluated a data-driven MAS model for observed aircraft taxi operations at Amsterdam Airport Schiphol. The objective was to construct a reproducible, graph-based, and support-aware observed-behavior baseline from surface-surveillance data. The framework separates the historical data-processing pipeline from the executed MAS model: VEMMIS operational records are used to infer behavior rules, while the MAS model organizes

the learned behavioral components and deterministic execution rules during integrated model validation.

The results show that historical surveillance records can be transformed into a large graph-aligned taxi dataset for the taxiing scope. The final dataset contains 883,771 graph-aligned taxi trajectories. Map matching provides a shared airport-layout-graph representation, and canonical path-family grouping reduces exact-observed-path sparsity while preserving the main path candidates needed for path choice.

The held-out component results show that trajectory-derived observations can support learned behavioral components for selected taxi-operation behaviors. Path choice captures most observed path geometry, with a mean edge-weighted path F1 of 0.926. Endpoint selection performs best for apron-boundary targets and remains more difficult for runway-stopbar targets. The speed-profile component achieves a mean RMSE of 2.28 kt with near-complete empirical or CART residual support. Local conflict resolution recovers the main movement-derived priority and resolution-action patterns, while remaining limited by labels inferred from trajectory behavior rather than observed ATC instructions.

Prediction-source diagnostics are essential for interpreting these results. High predictive performance is stronger evidence of learned behavior when outputs come from empirical or CART stages than when they depend on fallback rules. The selected components generally show low fallback use, but the balance between empirical and CART support differs by target. This confirms that the framework identifies not only what the model predicts, but also where those predictions are supported by historical observations.

Integrated model validation shows that selected component outputs can be combined into executable graph-based historical-scenario model executions. Planning completion reaches 100% in the three representative scenarios. The integrated diagnostics also show that deterministic execution rules can create timing effects that isolated component metrics do not reveal. In the high-runway-variability scenario, the realized mean taxi-time error increases to +7.67 min despite near-zero nominal timing error, showing that downstream interactions and release constraints are important sources of emergent model behavior.

The main limitations are the proprietary nature of VEMMIS, the restricted runway-stopbar–apron-boundary scope, the use of a static airport layout graph, missing operational-rule and construction-state context, movement-derived conflict labels, and simplified deterministic execution rules. Future work should add time-dependent layout states, encode known operational constraints, refine protected-resource and conflict-label definitions, and improve the deterministic execution rules for following, merging, queueing, and runway-departure release. These extensions would strengthen the framework as a reusable observed-behavior baseline for airport-surface analysis, decision-support evaluation, and future MAS-based studies.

References

- [1] Wilke, S., Majumdar, A., and Ochieng, W. Y., "Airport Surface Operations: A Holistic Framework for Operations Modeling and Risk Management," *Safety Science*, Vol. 63, 2014, pp. 18–33. <https://doi.org/10.1016/j.ssci.2013.10.015>, URL <https://doi.org/10.1016/j.ssci.2013.10.015>.
- [2] EUROCONTROL, "EUROCONTROL Specification for Advanced Surface Movement Guidance and Control System (A-SMGCS) Services," Tech. Rep. EUROCONTROL-SPEC-171, EUROCONTROL, 2020. URL <https://www.eurocontrol.int/sites/default/files/2020-04/eurocontrol-specification-a-smgcs-v-2-0.pdf>, edition 2.0.
- [3] Macal, C. M., and North, M. J., "Tutorial on Agent-Based Modelling and Simulation," *Journal of Simulation*, Vol. 4, No. 3, 2010, pp. 151–162. <https://doi.org/10.1057/jos.2010.3>, URL <https://doi.org/10.1057/jos.2010.3>.
- [4] von der Burg, M., and Sharpanskykh, A., "Modelling and Analysis of Autonomous Airport Surface Movement Operations Based on Multi-Agent Planning: Explorative Case Study at Amsterdam Airport Schiphol," *European Journal of Transport and Infrastructure Research*, Vol. 25, No. 1, 2025. <https://doi.org/10.59490/ejtir.2025.25.1.7459>, URL <https://doi.org/10.59490/ejtir.2025.25.1.7459>.
- [5] Roling, P. C., and Visser, H. G., "Optimal Airport Surface Traffic Planning Using Mixed-Integer Linear Programming," *International Journal of Aerospace Engineering*, Vol. 2008, 2008, p. 732828. <https://doi.org/10.1155/2008/732828>, URL <https://doi.org/10.1155/2008/732828>.
- [6] Zelinski, S., and Windhorst, R., "Modelling and Simulating Airport Surface Operations with Gate Conflicts," *The Aeronautical Journal*, Vol. 123, No. 1259, 2019, pp. 1–19. <https://doi.org/10.1017/aer.2018.111>, URL <https://doi.org/10.1017/aer.2018.111>.
- [7] Tran, T.-N., Pham, D.-T., and Alam, S., "A Map-Matching Algorithm for Ground Movement Trajectory Representation using A-SMGCS Data," *2020 International Conference on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT)*, 2020, pp. 1–8. <https://doi.org/10.1109/AIDA-AT48540.2020.9049181>, URL <https://doi.org/10.1109/AIDA-AT48540.2020.9049181>.
- [8] Olive, X., Waltert, M., Mori, R., and Mouyon, P., "Filtering Aircraft Surface Trajectories Using Information on the Taxiway Structure of Airports," , 2024. <https://doi.org/10.21256/ZHAW-34908>, URL <https://doi.org/10.21256/ZHAW-34908>, iCRAT 2024 contribution.
- [9] Quddus, M. A., Ochieng, W. Y., and Noland, R. B., "Current Map-Matching Algorithms for Transport Applications: State-of-the Art and Future Research Directions," *Transportation Research Part C: Emerging Technologies*, Vol. 15, No. 5, 2007, pp. 312–328. <https://doi.org/10.1016/j.trc.2007.05.002>, URL <https://doi.org/10.1016/j.trc.2007.05.002>.
- [10] Wang, X., Brownlee, A. E. I., Weiszer, M., Woodward, J. R., Mahfouf, M., and Chen, J., "An Interval Type-2 Fuzzy Logic Based Map Matching Algorithm for Airport Ground Movements," *IEEE Transactions on Fuzzy Systems*, Vol. 31, No. 2, 2023, pp. 582–595. <https://doi.org/10.1109/TFUZZ.2022.3221793>, URL <https://doi.org/10.1109/TFUZZ.2022.3221793>.
- [11] Newson, P., and Krumm, J., "Hidden Markov Map Matching Through Noise and Sparseness," *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Association for Computing Machinery, New York, NY, USA, 2009, pp. 336–343. <https://doi.org/10.1145/1653771.1653818>, URL <https://doi.org/10.1145/1653771.1653818>.
- [12] Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., and Huang, Y., "Map-Matching for Low-Sampling-Rate GPS Trajectories," *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Association for Computing Machinery, New York, NY, USA, 2009, pp. 352–361. <https://doi.org/10.1145/1653771.1653820>, URL <https://doi.org/10.1145/1653771.1653820>.
- [13] Prato, C. G., "Route Choice Modeling: Past, Present and Future Research Directions," *Journal of Choice Modelling*, Vol. 2, No. 1, 2009, pp. 65–100. [https://doi.org/10.1016/S1755-5345\(13\)70005-8](https://doi.org/10.1016/S1755-5345(13)70005-8), URL [https://doi.org/10.1016/S1755-5345\(13\)70005-8](https://doi.org/10.1016/S1755-5345(13)70005-8).
- [14] EUROCONTROL Experimental Centre, "Standard Airport Taxi Route Study," Tech. Rep. EEC Technical/Scientific Report No. 2009-002, EUROCONTROL Experimental Centre, 2009. URL https://www.eurocontrol.int/sites/default/files/library/002_Standard_airport_taxi_route_study.pdf.
- [15] Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, 1968, pp. 100–107. <https://doi.org/10.1109/TSSC.1968.300136>, URL <https://doi.org/10.1109/TSSC.1968.300136>.

- [16] Hu, D., Chen, L., Fang, H., Fang, Z., Li, T., and Gao, Y., “Spatio-Temporal Trajectory Similarity Measures: A Comprehensive Survey and Quantitative Study,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 36, No. 5, 2024, pp. 2191–2212. <https://doi.org/10.1109/TKDE.2023.3323535>, URL <https://doi.org/10.1109/TKDE.2023.3323535>.
- [17] Wasserman, L., *All of Nonparametric Statistics*, Springer, New York, 2006. <https://doi.org/10.1007/0-387-30623-4>, URL <https://doi.org/10.1007/0-387-30623-4>.
- [18] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J., *Classification and Regression Trees*, Wadsworth International Group, Belmont, CA, 1984.
- [19] Provost, F., and Domingos, P., “Tree Induction for Probability-Based Ranking,” *Machine Learning*, Vol. 52, No. 3, 2003, pp. 199–215. <https://doi.org/10.1023/A:1024099825458>, URL <https://doi.org/10.1023/A:1024099825458>.
- [20] Breiman, L., “Random Forests,” *Machine Learning*, Vol. 45, No. 1, 2001, pp. 5–32. <https://doi.org/10.1023/A:1010933404324>, URL <https://doi.org/10.1023/A:1010933404324>.
- [21] Friedman, J. H., “Greedy Function Approximation: A Gradient Boosting Machine,” *The Annals of Statistics*, Vol. 29, No. 5, 2001, pp. 1189–1232. <https://doi.org/10.1214/aos/1013203451>, URL <https://doi.org/10.1214/aos/1013203451>.
- [22] Kuhn, K. D., “A Methodology for Identifying Similar Days in Air Traffic Flow Management Initiative Planning,” *Transportation Research Part C: Emerging Technologies*, Vol. 69, 2016, pp. 411–429. <https://doi.org/10.1016/j.trc.2016.05.014>, URL <https://doi.org/10.1016/j.trc.2016.05.014>.
- [23] Kaufman, L., and Rousseeuw, P. J., *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, New York, 1990.
- [24] Gipps, P. G., “A Behavioural Car-Following Model for Computer Simulation,” *Transportation Research Part B: Methodological*, Vol. 15, No. 2, 1981, pp. 105–111. [https://doi.org/10.1016/0191-2615\(81\)90037-0](https://doi.org/10.1016/0191-2615(81)90037-0), URL [https://doi.org/10.1016/0191-2615\(81\)90037-0](https://doi.org/10.1016/0191-2615(81)90037-0).
- [25] Simaiakis, I., and Balakrishnan, H., “A Queuing Model of the Airport Departure Process,” *Transportation Science*, Vol. 50, No. 1, 2016, pp. 94–109. <https://doi.org/10.1287/trsc.2015.0603>, URL <https://doi.org/10.1287/trsc.2015.0603>.
- [26] Okina, A., and Dijkstra, F., “RECAT-EU for Departures at Schiphol,” Tech. Rep. FWY-2020-01, Knowledge and Development Centre Mainport Schiphol, 2020. URL <https://kdc-mainport.nl/wp-content/uploads/2021/01/FWY-2020-01-RECAT-EU-for-departures-v1.0.pdf>, version V1.0; executed by FerWay and MovingDot.
- [27] Sargent, R. G., “Verification and Validation of Simulation Models,” *Journal of Simulation*, Vol. 7, No. 1, 2013, pp. 12–24. <https://doi.org/10.1057/jos.2012.20>, URL <https://doi.org/10.1057/jos.2012.20>.
- [28] Snoek, J., Larochelle, H., and Adams, R. P., “Practical Bayesian Optimization of Machine Learning Algorithms,” *Advances in Neural Information Processing Systems*, Vol. 25, 2012. URL https://papers.nips.cc/paper_files/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html.

Part III

Literature Review & Research Definition

Literature Review

3.1. Airport Surface Movement Operations and Observed-behavior Modeling

This section positions aircraft taxi operations within airport surface movement operations (ASM Ops). It explains why observed taxi behavior should be treated as part of a coupled operational system rather than as an isolated movement problem. It also explains why this thesis requires a data-driven observed-behavior baseline instead of only an optimization model. The section ends by identifying the behavioral components that structure the rest of the literature review: path choice, endpoint selection, speed-profile behavior, local conflict resolution, deterministic execution rules, and support-aware validation.

3.1.1. ASM Ops as a Coupled Operational System

ASM Ops combine aircraft movement, airport infrastructure, runway configuration, controller decisions, flight-crew execution, local procedures, surveillance systems, and shared surface resources. This makes aircraft taxi operations a system-level phenomenon. A taxi movement is not only a path between two points on the airport layout. It is also the result of runway use, apron access, traffic interactions, local procedures, and tactical decisions made under safety constraints.

The A-SMGCS concept reflects this system view by organizing surface movement through surveillance, routing, guidance, and airport safety-support functions. The ICAO manual presents A-SMGCS as a way to support safe and efficient movement under varying visibility, traffic, and airport-complexity conditions [1]. EUROCONTROL's specification also separates A-SMGCS into interacting service layers rather than treating surveillance, routing, and safety support as independent functions [2]. This layered view is useful for this thesis because the learned components also separate observed behavior into functional parts while keeping them connected through a common graph and operational context.

Holistic airport-surface modeling treats surface operations as coupled movement, control, and infrastructure processes. Wilke et al. argue that airport surface performance depends on interactions between infrastructure, traffic flows, control decisions, and operational constraints [3]. This perspective supports the thesis framing: observed taxi behavior should not be reduced to isolated taxi-time values or shortest paths. It should be reconstructed as a set of context-dependent behaviors embedded in the airport-surface system.

The coupled nature of ASM Ops is visible in taxi-time and congestion studies. Idris et al. show that taxi-out time is strongly related to runway queueing and departure demand [4]. Simaiakis et al. show that surface congestion affects taxi time and emissions because aircraft spend more time moving slowly, stopping, or waiting with engines running [5]. Clewlow et al. show that arrival operations can affect departure taxi behavior because arrivals consume runway or crossing capacity and change the conditions encountered by departures [6]. Together, these studies support the need to model taxi behavior as context-dependent and interactive.

For this thesis, the important implication is that observed taxi trajectories encode operational decisions and constraints. They show how the surface system repeatedly absorbs demand, uses runway and apron access, resolves local interactions, and adapts to recurring contexts. The trajectories do not reveal the

internal reasoning of controllers or pilots directly. They do, however, provide observable evidence from which selected behaviors can be inferred.

3.1.2. Modeled Taxiing Scope

This thesis focuses on aircraft taxi operations within a runway-stopbar–apron-boundary scope. Inbound taxi movements run from runway stopbar to apron boundary after landing. Outbound taxi movements run from apron boundary to runway stopbar before departure. This scope captures the taxiway-system portion of the operation, where path choice, endpoint selection, speed-profile behavior, and local conflict resolution are strongly expressed.

The scope deliberately excludes apron-internal movement, gate-area maneuvering, pushback, de-icing, private-aviation movements, eastern maintenance-area movements, and the full runway-clearance process. These exclusions are not claims that the omitted processes are unimportant. They define a tractable behavioral layer where surface-surveillance trajectories and operational records can support data-driven behavior identification.

This scope also separates the model from a complete airport-surface simulator. The thesis does not claim to model every operational decision from gate to runway or runway to gate. It constructs an observed-behavior baseline for the main runway-stopbar–apron-boundary taxiing process. This distinction is important because it keeps the learned behavioral components tied to observable trajectory evidence.

3.1.3. Optimization Models and Observed-behavior Baselines

Many airport-surface studies formulate taxi operations as optimization problems. These models specify objectives such as minimizing delay, taxi time, fuel burn, queueing, or conflicts subject to operational constraints. Marin presents airport taxi planning as a management problem in which ground movements must be planned over constrained surface infrastructure [7]. Clare and Richards combine taxiway routing with runway scheduling and show that routing and runway sequencing decisions are interdependent [8]. Bertsimas and Frankovich represent airport operations as a networked traffic-flow optimization problem across interacting airport resources [9]. Ravizza et al. combine empirical taxi-time estimation with a structured ground-movement model, which illustrates the value of linking data-driven time prediction with explicit surface-movement structure [10].

These optimization studies are important because they show that airport surface movement is constrained, networked, and operationally consequential. They also show why a baseline of current behavior matters. A prescriptive model can recommend paths, sequences, or release decisions that differ from observed operations. To interpret such differences, researchers need a clear reference for what the operation usually does under comparable conditions.

An optimization model and an observed-behavior baseline therefore answer different questions. Optimization asks what the operation should do under a stated objective and constraint set. An observed-behavior baseline asks what the operation tends to do in the historical data. The baseline is not a substitute for optimization. It is an empirical reference that later optimization, decision-support, or automation studies can compare against.

Without an empirical baseline, studies often rely on ad hoc assumptions. These assumptions may include shortest-path routing, fixed taxi speeds, simplified runway queues, or generic separation rules. Such assumptions are useful for abstraction, but they can distort comparisons if they do not resemble the airport's actual operating patterns. A data-driven baseline can reduce this problem by representing path choice, endpoint selection, speed-profile behavior, and local conflict resolution from historical observations.

3.1.4. MAS Modeling for Observed Taxi Behavior

The multi-agent system paradigm is useful when system-level behavior emerges from interactions among autonomous or semi-autonomous entities. Macal and North present agent-based modeling as a paradigm for systems in which aggregate behavior arises from interactions between agents and their environment [11]. In ASM Ops, this is directly relevant because taxi behavior emerges from aircraft execution, controller decisions, infrastructure constraints, runway configuration, and shared taxi resources.

The PhD thesis by von der Burg develops MAS models for airport surface movement operations and provides a structured way to organize surface operations through interacting operational roles [12]. That

work focuses on models, methods, and tools for next-generation concepts of operations. This thesis uses the same broad MAS motivation but applies it to a different purpose: identifying an observed-behavior baseline from historical surveillance data.

The MAS framing is valuable here because it separates functional behaviors. The Ground Control Agent can be associated with endpoint selection, path choice, and local conflict resolution. The Aircraft Operator Agent can be associated with speed-profile behavior and aircraft-side execution. The Runway Control Agent can be represented through a deterministic runway-departure release rule. These are modeling abstractions. They do not imply that actual controller or pilot commands are observed in the data.

This separation helps preserve an important evidential boundary. Learned behavioral components estimate relationships between observed environment conditions and trajectory-derived outputs. Deterministic execution rules preserve safe and executable model execution. Treating these as different model elements prevents deterministic safeguards from being misinterpreted as learned behavior.

3.1.5. Reusable Research Infrastructure

Reusable research infrastructure improves comparability across studies. BlueSky illustrates this principle in air traffic management by providing open tools for simulation and reproducible experimentation [13]. Mesa plays a similar role for agent-based modeling by providing a reusable Python framework for building and analyzing agent-based models [14]. AMELIA is closer to the airport-surface domain because it provides a dataset and benchmark for airport surface movement forecasting [15].

These examples do not provide the observed-behavior baseline developed in this thesis. They do, however, support the broader idea that common data products, common definitions, and reusable model structures improve research quality. This thesis follows that principle by transforming proprietary operational records into reusable derived objects: taxi trajectories, graph-aligned taxi trajectories, exact observed paths, canonical path families, context features, component observations, and prediction-source diagnostics.

The reusable object is therefore not the raw VEMMIS database. The reusable object is the method for producing structured behavioral evidence from surveillance and operational records. This matters because the raw records cannot be shared publicly. The thesis must therefore provide reproducibility through definitions, processing logic, aggregate evidence, validation diagnostics, and appendix-level support.

3.1.6. Component-based Behavioral Representation

A monolithic taxi-behavior model would be difficult to interpret. It would also mix different operational functions that depend on different data and context. Path choice, endpoint selection, speed-profile behavior, and local conflict resolution are not the same prediction problem. They use different target variables, have different support structures, and respond to different operational contexts.

A component-based representation allows each behavior to be identified and validated separately. Path choice uses canonical path families as path candidates. Endpoint selection estimates runway-stopbar and apron-boundary targets. Speed-profile behavior estimates residual speed profiles around edge-level baselines. Local conflict resolution estimates movement-derived priority and resolution-action labels for pairwise protected-resource encounters.

This component structure also improves diagnosis. A path-choice error can be related to OD structure, runway configuration, or path-candidate support. An endpoint-selection error can be related to runway-side availability or apron-side geography. A speed-profile error can be localized to a directed edge, local motion state, or aircraft class. A local conflict-resolution error can be related to near-tie timing or protected-resource geometry. The literature review therefore proceeds from data reconstruction to component-specific model identification and finally to validation.

3.2. Surface-surveillance Data and Taxi-trajectory Extraction

Surface-surveillance trajectories are the empirical basis of the observed-behavior baseline. They are also imperfect, proprietary, and not directly usable as learned-component inputs. This section reviews surveillance sources, track-quality challenges, taxiing-scope extraction, filtering, context joining, and re-

producibility constraints. The goal is to show why a substantial data-processing layer is required before model identification.

3.2.1. Operational Surface-surveillance Sources

Airport surface surveillance can be provided by surface movement radar, multilateration, ADS-B, ASDE-X-type systems, and A-SMGCS data fusion. These sources provide time-stamped observations of aircraft positions and sometimes motion attributes. FAA documentation on ASDE-X illustrates how surveillance fusion can support surface awareness at complex airports [16]. Bloem et al. discuss airport surveillance fusion for surface state estimation, which shows why multiple data sources may be combined for operational awareness [17]. Pledgie et al. also emphasize the need to infer surface state from imperfect observations rather than assuming that raw surveillance tracks are complete [18].

Public ADS-B data sources have enabled many air-traffic studies, but their use near the ground requires caution. Strohmeier et al. describe the OpenSky Network as a large open aviation dataset that supports reproducible research at scale [19]. Schafer et al. analyze OpenSky data quality and show that coverage and integrity issues must be considered before using ADS-B trajectories for detailed operational inference [20]. This distinction matters for airport-surface modeling because small spatial or temporal errors can change the inferred graph edge.

VEMMIS provides the Schiphol operational records used in this thesis. It includes flight-level information, schedule records, and surface-surveillance trajectories. Because the raw database is proprietary, the thesis cannot make raw records the reproducibility mechanism. It must instead define the data units, processing steps, derived products, and validation diagnostics used to support the observed-behavior baseline.

3.2.2. Raw Track-quality Challenges

Surface-surveillance trajectories may contain noise, uneven sampling, missing points, false jumps, duplicated fragments, identity interruptions, or inconsistent speed and heading attributes. These problems are not only data-cleaning nuisances. They directly affect graph reconstruction, path catalog construction, speed-profile estimation, and conflict-event extraction.

Tran et al. report that A-SMGCS trajectories require airport-specific map matching because dense taxiway layouts and noisy surveillance data make simple geometric projection insufficient [21]. Olive and Basora discuss trajectory-event extraction and show that operational trajectory analysis depends on identifying meaningful movement phases from raw aircraft data [22]. Olive's ADS-B preprocessing work shows that trajectory quality improves only after plausibility filtering and careful treatment of missing or inconsistent motion attributes [23]. These studies support the thesis choice to treat taxi-trajectory extraction as a methodological step rather than as a trivial data import.

The airport-surface setting makes these issues especially sensitive. Taxiway edges can be close together. Apron and holding areas can be wide. Aircraft may deviate from modeled centerlines without violating operational practice. A small position error can therefore imply a different directed edge, a different exact observed path, or a different protected-resource encounter. The data-processing pipeline must preserve useful movement evidence while excluding trajectories that are physically inconsistent or unsuitable for graph reconstruction.

3.2.3. Taxiing-scope Extraction

Taxiing-scope extraction converts broad surface-surveillance records into taxi trajectories within the runway-stopbar–apron-boundary scope. This step identifies the movement direction, runway stopbar, apron boundary, start time, end time, and retained trajectory points. It is necessary because raw surveillance records may include approach detections, runway occupancy, apron-internal movement, gate-area maneuvering, or fragmented tracks outside the modeled scope.

Mirmohammadsadeghi and colleagues show that taxi-event extraction can be treated as a problem of detecting operationally meaningful movement phases from surveillance data [24]. Schultz et al. emphasize that airport operational milestones can be inferred from data only when the relevant movement phases and operational events are defined clearly [25]. These contributions support the thesis distinction between raw surveillance records, taxi trajectories, graph-aligned taxi trajectories, and component observations.

This scope extraction is also important for validation. If the taxiing-scope interval is too long, the model may learn behavior from gate-area or runway-clearance processes that are outside its stated scope. If the interval is too short, it may miss endpoint behavior or local interactions near boundaries. A clear taxiing-scope definition therefore protects the interpretation of every learned component.

3.2.4. Filtering, Smoothing, and Conditioning

Filtering and conditioning prepare raw trajectory points for reconstruction. Typical steps include removing impossible jumps, checking speed plausibility, smoothing noisy position or speed signals, joining compatible fragments, and excluding incomplete records. These steps must be conservative enough to remove corrupt data but not so aggressive that they erase operationally meaningful behavior such as stops, holds, or slow movement.

Bloem and Churchill describe methods for filtering airport surface movement trajectories and show that data conditioning is necessary before airport-surface analysis [26]. Olive's surface-filtering work also supports the need for robust trajectory cleaning before extracting behavior from surface data [27]. These studies justify the thesis emphasis on data-retention diagnostics and reconstruction-quality checks.

Filtering is not independent of the modeling objective. A taxi-time model may tolerate small spatial errors if timing remains accurate. A graph-path model may be more sensitive to spatial errors near inter-sections. A speed-profile component must preserve speed variation, stops, and acceleration patterns. A conflict-resolution component must preserve local timing near protected resources. The preprocessing design must therefore be linked to the downstream behavioral components.

3.2.5. Operational Context Joining

A taxi trajectory alone does not contain all information needed for behavior identification. It must be linked to flight records, schedule records, runway configuration, aircraft class, operator, SID, gate or apron information, traffic density, and operating state. These context variables explain why similar paths may be used differently under different conditions. They also help prevent the model from treating all taxi movements between the same endpoints as interchangeable.

Taxi-time prediction studies repeatedly show that operational context matters. Vargo et al. use machine-learning methods for airport taxi-time prediction and demonstrate that contextual features can improve prediction beyond simple distance or time-of-day assumptions [28]. Wang et al. analyze taxi-time features and show that operational variables contribute to surface-movement prediction quality [29]. Yin et al. review taxi-time prediction and emphasize that runway use, traffic demand, weather, and aircraft characteristics influence airport surface outcomes [30].

For this thesis, context joining is not only a predictive feature-engineering step. It defines the conditions under which observed behavior is interpreted. A path-choice observation consists of a path candidate and the operational context in which that path was used. A speed-profile observation consists of an edge-relative speed profile and the motion and aircraft context around it. A conflict-resolution observation consists of a pairwise encounter and the timing, direction, and protected-resource context that describe it.

3.2.6. Reproducibility with Proprietary Data

Proprietary surveillance data create a reproducibility challenge. Other researchers cannot reproduce the exact processing pipeline from raw VEMMIS records if those records cannot be shared. The thesis must therefore rely on transparent data definitions, processing logic, aggregate diagnostics, and derived evidence rather than raw-data release.

The FAIR data and reproducible research literature supports this distinction. The Research Data Alliance emphasizes that reproducibility depends on clear metadata, provenance, workflow documentation, and access conditions, not only on open raw data [31]. Mitchell et al. discuss the need to track data provenance and processing lineage so that users can understand how derived data products were produced [32]. Spinielli et al. show that reproducibility in air navigation service research can require sharing methods, aggregate data products, and evaluation artifacts when operational data cannot be fully released [33].

This thesis follows that logic. It defines taxi trajectories, graph-aligned taxi trajectories, exact observed paths, canonical path families, component observations, prediction-source diagnostics, and validation

metrics. It also reports reconstruction-quality diagnostics and appendix-level evidence. These elements cannot replace raw-data release completely, but they make the scientific claims inspectable and transferable.

3.3. Airport Layout Graphs and Graph-constrained Reconstruction

Graph-constrained reconstruction is the bridge between raw surveillance trajectories and learned taxi-operation behavior. This section reviews airport layout graphs, map-matching method families, airport-specific reconstruction challenges, endpoint-constrained path reconstruction, and quality diagnostics. The section establishes why the thesis uses graph-aligned taxi trajectories as the core reconstructed data product.

3.3.1. Airport Layout Graphs as Spatial Reference

The airport layout graph provides the common spatial reference for path choice, endpoint selection, speed-profile behavior, and local conflict resolution. Nodes represent operational locations such as taxiway junctions, runway stopbars, apron boundaries, or holding locations. Directed edges represent traversable centerline elements with allowed travel direction and layout-derived attributes.

Graph-based representations are common in airport-surface planning because they provide a compact way to represent feasible movement over taxiway and runway infrastructure. He et al. use graph-search methods for airport taxiway path finding, which illustrates how graph topology supports routing and path construction [34]. Pant et al. use safety-aware taxi planning on graph-like airport infrastructure, which shows the continued relevance of graph representations for airport surface movement [35]. Butler et al. combine airport taxi planning with control barrier functions and motion-planning constraints, which further illustrates how graph and trajectory representations can be linked [36].

For this thesis, the graph is not only a routing object. It is also the coordinate system for behavior identification. Exact observed paths are directed-edge sequences. Path candidates are canonical path families within OD pairs. Edge-relative speed profiles are defined on directed edges. Protected-resource encounters are derived from graph nodes, merge locations, apron boundaries, and geometric crossings. The graph therefore connects the data-processing and model-identification layers.

3.3.2. Map-matching Method Families

Map matching projects noisy trajectory observations onto a graph. Quddus et al. review map-matching algorithms and distinguish geometric, topological, probabilistic, and advanced methods [37]. Quddus et al. also discuss map-matching integrity and show that quality assessment is necessary when map-matched outputs are used for downstream applications [38]. These reviews are important because they show that map matching is not a single technique but a family of methods that combine observation geometry, network topology, and plausibility constraints.

Hidden Markov map matching treats the true path as a latent state sequence and uses probabilistic emissions and transitions to infer the most likely route. Newson and Krumm present this approach for noisy and sparse GPS trajectories [39]. Spatio-temporal map matching adds timing and speed consistency to path reconstruction, which is useful when data are sparse or when geometric proximity alone is ambiguous [40]. These methods show why topology and temporal evidence can improve reconstruction beyond nearest-edge projection.

Airport-specific map matching adapts these principles to dense taxiway networks and surface-surveillance data. Tran et al. develop A-SMGCS map-matching methods for airport ground trajectories and show that airport-specific constraints are needed [21]. Wang et al. also show that graph-constrained reconstruction is relevant for airport surface trajectories and that airport-specific geometry and topology must be considered [41]. Gao et al. use semantic airport-surface trajectory interpretation, which supports the broader idea that graph alignment enables higher-level operational understanding [42].

3.3.3. Airport-surface Reconstruction Challenges

Airport-surface map matching differs from road-network map matching. Taxiway centerlines can be close together. Intersections are dense. Aircraft may move across wide paved areas in ways that do not exactly follow modeled centerlines. Local operating practices can also cause deviations from the nominal graph representation. These features make nearest-edge projection unreliable in isolation.

The graph candidate with the smallest distance to a trajectory point may not be operationally correct. A nearby edge may have the wrong travel direction. A geometrically plausible edge may not connect to the inferred endpoint. A short gap in surveillance data may hide an intersection traversal that must be recovered through topology. These conditions require map matching to combine spatial evidence, direction, connectivity, endpoint consistency, and quality diagnostics.

This challenge links map matching to the rest of the thesis. If a graph-aligned taxi trajectory is wrong, the exact observed path is wrong. If the exact observed path is wrong, canonical path-family grouping and path-choice labels can be wrong. If projected positions are wrong, edge-relative speed profiles and protected-resource encounters can be wrong. Reconstruction quality is therefore not a preprocessing detail. It is a validity requirement for the observed-behavior baseline.

3.3.4. Candidate Generation, Scoring, and Connectivity Recovery

Graph-constrained map matching usually starts by generating candidate directed edges near each trajectory point. Candidate-edge scoring then combines spatial proximity, motion direction, and graph plausibility. In airport-surface data, candidate scoring must also respect airport layout features such as stopbars, runway-side boundaries, apron-side boundaries, and one-way graph elements.

Connectivity recovery is needed when trajectory evidence is incomplete. Surveillance gaps or sparse sampling can create missing edge evidence between otherwise plausible candidate edges. A graph search can recover a short topological connection when the recovered path is consistent with endpoints and projection diagnostics. This recovery should be interpreted as reconstruction support rather than as a behavioral route-choice assumption.

The distinction between reconstruction recovery and behavior modeling is important. Connectivity recovery creates a topologically valid graph-aligned taxi trajectory from incomplete observations. It does not claim that a controller selected a shortest path. The behavioral interpretation begins only after the graph-aligned taxi trajectory is validated and grouped into exact observed paths and canonical path families.

3.3.5. Endpoint-constrained Path Reconstruction

Endpoint constraints reduce ambiguity in dense airport layouts. For inbound movements, the reconstructed path must connect a runway stopbar to an apron boundary. For outbound movements, it must connect an apron boundary to a runway stopbar. These constraints are operationally meaningful because they define the modeled taxiing scope.

Endpoint-constrained reconstruction also links preprocessing to component modeling. The runway stopbar and apron boundary become part of the OD pair. They also become targets for endpoint selection. The reconstructed path becomes the exact observed path. The projected trajectory points become the basis for path progress, edge-relative progress, speed-profile behavior, and conflict-event extraction.

This means that endpoint inference, map matching, and component input construction cannot be treated as isolated tasks. A reconstruction error at the endpoint can change the OD pair, the path-choice set, and the component observation. This is why endpoint consistency is part of the reconstruction-quality diagnostics.

3.3.6. Quality Diagnostics and Exclusion

Map-matching quality must be evaluated before graph-aligned trajectories are used for behavior identification. Relevant diagnostics include point-to-path projection error, edge support, edge-relative coverage, endpoint consistency, topology validity, and movement-level 95th-percentile projection error. These diagnostics identify both aggregate reconstruction quality and local representation problems.

High-error exclusions protect downstream learning. A corrupted trajectory can create false exact observed paths. It can also create incorrect endpoint labels, unrealistic speed profiles, or false protected-resource encounters. Exclusion is therefore part of data validity. It should not be interpreted as a failure of the learned behavioral components.

Localized reconstruction problems are still informative. They may indicate surveillance gaps, static-graph limitations, construction-state mismatches, or wide paved areas where aircraft do not follow the

graph centerline. Appendix-level reconstruction diagnostics are useful because they show whether problems are global or local. This supports the thesis distinction between generally reliable graph alignment and localized representation limits.

3.3.7. Exact Observed Paths

The direct path output of map matching is the exact observed path. It is the ordered directed-edge sequence obtained from one graph-aligned taxi trajectory. Exact observed paths preserve reconstruction detail and support traceability from the learned path-choice labels back to individual taxi movements.

Exact observed paths are necessary for evaluation. They allow edge-weighted overlap metrics to compare predicted and observed graph paths. They also allow path length, path progress, and edge-relative speed profiles to be assembled consistently. However, exact observed paths are too detailed to serve directly as path-choice classes in all cases. Minor local deviations or reconstruction differences can create many rare exact paths. This motivates the path-catalog step reviewed next.

3.4. Path Candidates, Canonical Path Families, and Operational Context

This section reviews the literature and modeling logic behind path candidates. It distinguishes feasible paths, shortest paths, exact observed paths, and canonical path families. It then explains why operational context is required to interpret path behavior.

3.4.1. Feasible Paths, Shortest Paths, and Observed Paths

A directed airport layout graph can contain many feasible paths between a runway stopbar and an apron boundary. A shortest path is the path that minimizes a chosen graph cost, such as distance or nominal travel time. An exact observed path is the reconstructed directed-edge sequence followed by one observed taxi movement. A path candidate is a supported option used by the path-choice component within an OD pair.

These categories must remain separate. Feasible paths describe physical possibility. Shortest paths describe graph-search output under a chosen objective. Exact observed paths describe empirical evidence. Canonical path families describe the alternatives used for behavior modeling. Treating these objects as equivalent would incorrectly replace observed operational behavior with graph optimality.

Graph search still has a role. The A* algorithm is a classical method for graph search and provides a practical way to construct executable fallback paths when observed support is unavailable [43]. In this thesis, A* fallback paths preserve executability. They are not interpreted as learned Ground Control Agent path choices.

3.4.2. Alternative-set Generation in Choice Modeling

Choice modeling requires a set of alternatives. Train explains that discrete choice models estimate probabilities over alternatives available to the decision maker [44]. Prato reviews route-choice set generation and shows that the definition of alternatives strongly affects route-choice modeling [45]. These principles apply directly to airport taxi path choice because the model can only assign probability to path candidates that have been defined.

The candidate set must avoid two errors. If relevant observed paths are omitted, the model cannot recover historical behavior. If too many irrelevant or near-duplicate paths are included, the model becomes sparse and difficult to interpret. The path catalog must therefore balance coverage and parsimony.

Route-choice literature also shows that candidate alternatives are not purely geometric objects. Yao et al. discuss route-choice set generation and show that candidate sets must represent plausible alternatives rather than every possible graph path [46]. Zhao et al. use inverse reinforcement learning for route choice, which illustrates another way to infer route preferences from observed trajectories [47]. These studies reinforce the idea that candidate paths must be grounded in observed behavior and operational plausibility.

3.4.3. Airport Taxi Routes as Operational Constructs

Airport taxi routes are operational constructs. They depend on airport layout, runway configuration, local procedures, traffic demand, and controller practice. EUROCONTROL's taxi-route concept supports route-management analysis and surface-routing applications in A-SMGCS contexts [48]. This source contributes a domain-specific view of taxi routes as operationally managed surface-movement structures.

Taxi-time prediction literature also supports the need for operational context. Yin et al. review taxi-time prediction and identify runway use, aircraft characteristics, traffic conditions, and weather as important factors [30]. This review does not define path candidates, but it reinforces the broader point that surface behavior depends on operational state.

For this thesis, path candidates are defined within OD pairs. The OD pair fixes movement direction, runway stopbar, and apron boundary. Within that OD pair, several canonical path families may remain. The selected path candidate can then depend on runway configuration, gate area, demand, density, time of day, and runway-transition timing.

3.4.4. Exact-path Proliferation and Sparsity

Exact observed paths can proliferate because of small graph-alignment differences, local tactical deviations, missing surveillance points, or graph granularity. Tran et al. show that airport trajectory reconstruction can be sensitive to dense taxiway topology and surveillance noise [21]. Wang et al. also emphasize that airport-surface map matching must handle dense graph structure and operational constraints [41]. Schlosser et al. show that ground trajectories require careful processing before they can support higher-level analysis [49].

This proliferation creates sparsity when exact paths are used directly as labels. Minor edge differences can split one operational path into several rare classes. At the same time, important branches, runway crossings, or apron-side approaches must not be merged away. A path catalog must therefore reduce noise while preserving operationally meaningful path candidates.

3.4.5. Canonical Path Families

Canonical path families group similar exact observed paths within an OD pair. They reduce sparsity by representing repeated path structures rather than every minor reconstruction variant. Bian et al. review trajectory clustering and show that clustering results depend strongly on the chosen similarity representation [50]. Hu et al. discuss trajectory similarity and reinforce that movement-sequence grouping should preserve the structure relevant to the downstream task [51]. These references support the thesis choice to use graph-sequence similarity rather than only geometric proximity.

For airport taxi paths, similarity must preserve directed-edge sequence structure. Two paths that share most edges but diverge at an operationally important branch may represent different path candidates. Two paths that differ only by a minor reconstruction detail may represent the same operational path. The grouping rule must therefore be interpretable, traceable, and conservative around important branches.

Canonical path families provide this compromise. They reduce exact-path sparsity while preserving traceability to graph-aligned taxi trajectories. They also allow the path-choice component to estimate distributions over operationally meaningful path candidates rather than over every exact path variant.

3.4.6. OD Pairs, OD Groups, and Rare Paths

OD pairs define the primary conditioning structure for path choice. A specific OD pair is defined by movement direction, runway stopbar, and apron boundary. This level of detail is necessary because path options depend strongly on both runway-side and apron-side endpoints.

OD groups are coarser runway–apron groupings used for aggregated diagnostics. They are useful when exact OD pairs become sparse or when results need to be interpreted at a higher spatial level. The distinction between OD pairs and OD groups helps balance detailed modeling and readable diagnostics.

Rare paths create a specific challenge. They may represent real but infrequent operational choices. They may also be artifacts of reconstruction noise or incomplete surveillance. Treating all rare exact paths as separate path candidates would increase sparsity. Merging them blindly would hide unusual but meaningful behavior. The thesis therefore keeps rare exact paths traceable while using support criteria to decide whether they become separate path candidates.

3.4.7. Operational Context for Path Candidates

Operational context is required because the same OD pair can exhibit different path behavior under different conditions. Runway configuration changes the active runway-side structure. Gate area changes apron-side access. Demand and density change surface-traffic pressure. Time of day reflects recurring schedule and operating patterns. Runway-configuration transition timing captures periods when path use may change around operational transitions.

This context dependence is central to the observed-behavior baseline. The path-choice component does not only learn which path candidates exist. It learns how their probabilities vary with operational context and support. This creates a bridge between graph reconstruction and data-driven behavioral model identification.

3.5. Data-driven Behavioral Model Identification

This section reviews the model-identification logic used by the learned behavioral components. It discusses empirical distributions, support and sparsity, CART estimators, alternatives to CART, probabilistic reliability, fallback rules, prediction-source diagnostics, and model-selection discipline. The goal is to show why the thesis uses a support-aware empirical–CART–fallback procedure.

3.5.1. From Context-enriched Records to Learned Behavior

A learned behavioral component estimates a relationship between context features and a trajectory-derived output. The output can be a class label, such as endpoint target, path candidate, priority label, or resolution-action label. The output can also be a continuous residual profile, as in the speed-profile component. In both cases, the model uses historical observations to estimate how behavior varies with context.

This framing is different from directly observing agent intention. The VEMMIS records do not contain complete controller instruction logs or pilot decision states. The model therefore treats agent behavior as a black box. Observed environment conditions are associated with trajectory-derived actions, labels, or profiles. This is appropriate for an observed-behavior baseline, but it requires careful wording: the model estimates behavior patterns, not internal mental states.

3.5.2. Empirical Distributions and Support

Empirical distributions are the most direct way to preserve observed behavior. In a supported context cell, the empirical distribution estimates the output probabilities from observed frequencies. Wasserman describes nonparametric estimation as a way to estimate quantities without assuming a parametric functional form [52]. This is useful for airport taxi behavior because many context-specific decisions may be repeated often enough to estimate directly.

The limitation is sparsity. As more context features are added, exact context cells become smaller. A highly detailed empirical cell may contain too few observations to estimate a reliable distribution. It may also be dominated by one day or one unusual operating period. This creates a trade-off between operational specificity and statistical support.

Support checks are therefore essential. Minimum observation counts reduce the risk of estimating behavior from too few cases. Temporal coverage checks reduce the risk of learning from one isolated day. Single-day dominance checks prevent one operating day from controlling a context cell. Stability and compactness checks help ensure that the observed outcomes are coherent enough to use directly.

3.5.3. CART for Interpretable Generalization

CART estimators provide a transparent way to generalize beyond exact empirical support. Breiman et al. introduced Classification and Regression Trees as recursive partitioning methods for classification and regression [53]. A tree partitions feature space into leaves, and each leaf can store outcome frequencies or residual summaries. This structure is useful because it allows related contexts to share information while keeping the decision structure readable.

CART is particularly suitable for the thesis because interpretability matters. The components are not only predictive tools. They are also part of a behavioral baseline that should be auditable. A single tree

can show which context features are used to separate operational regimes. It can also support leaf-level acceptance criteria, such as minimum support and dominant-class probability.

Tree-based probability estimates still require caution. Provost and Domingos show that probability estimation trees require attention to leaf size and probability quality [54]. If a leaf is too small or poorly calibrated, its class probabilities may be unreliable. This supports the thesis decision to treat CART leaves as accepted only when they satisfy support and stability criteria.

3.5.4. Alternatives to CART

More flexible tree ensembles can improve predictive performance. Random forests average many decision trees and often perform well in classification and regression tasks [55]. Gradient boosting builds additive ensembles and can produce strong predictive models for complex nonlinear relationships [56]. Bayesian optimization and modern machine-learning workflows often use such flexible models when predictive accuracy is the only objective.

The thesis does not use these ensembles as the main behavioral estimator because interpretability and support diagnostics are central. A random forest may predict well, but its output is harder to connect to a small set of context-conditioned support rules. A boosted model may improve accuracy, but it can obscure whether a prediction is supported by repeated historical behavior. Rudin argues that interpretable models should be preferred in high-stakes settings when they can achieve adequate performance [57]. This argument supports the thesis preference for empirical distributions and CART estimators over black-box ensembles.

This does not mean that ensembles are irrelevant. They remain useful benchmarks or future extensions. They may also help identify feature interactions that single trees miss. The thesis uses CART because the final model must remain readable, auditable, and compatible with prediction-source diagnostics.

3.5.5. Probability Quality and Calibration

Many learned components produce probability distributions rather than single deterministic labels. Probability quality matters because a distribution can be accurate on average while being overconfident in sparse contexts. Calibration diagnostics are therefore necessary when class-probability estimates are used to represent behavior.

Guo et al. show that predictive accuracy and probability calibration can diverge in modern classifiers [58]. Provost and Domingos show that decision trees need care when used for probability estimation because the estimated probabilities depend on leaf composition and support [54]. Leathart et al. examine probability calibration for decision trees, which supports evaluating CART-stage probability quality rather than only class accuracy [59].

Entropy should be interpreted carefully. Predictive entropy measures uncertainty or sharpness in the predicted distribution. It is not a calibration metric. A low-entropy distribution may be confident but wrong. A high-entropy distribution may reflect genuine uncertainty among several supported path candidates or endpoint classes. This is why probabilistic predictive performance, calibration performance, and prediction-source diagnostics must be interpreted together.

3.5.6. Interpretability, Auditability, and Behavioral Evidence

Interpretability is not only a preference in this thesis; it is part of the model's evidential role. The model is intended to serve as an observed-behavior baseline that future users can inspect, compare, and challenge. A black-box model could be accurate but difficult to relate to airport operations. An interpretable model can expose which context cells, leaves, fallbacks, and deterministic rules produce the outputs.

Rudin argues that high-stakes decision domains should prefer interpretable models when comparable performance can be achieved [57]. This argument is relevant because airport-surface operations are safety-critical and operationally constrained. The thesis does not make automated operational decisions, but its outputs could influence how future studies define baseline behavior.

Auditability also affects how validation is interpreted. When a component output comes from an empirical context cell, the supporting historical observations can be counted and inspected. When a component output comes from a CART leaf, the leaf conditions and support can be examined. When a component

output comes from fallback, the result can be marked as executable but not learned. This structure makes the model more transparent than a single opaque predictor.

3.5.7. Fallback Rules and Executability

A support-aware model must decide what to do when learned support is insufficient. A fallback rule provides an executable output when neither empirical nor CART stages are accepted. For path choice, a fallback can provide graph-search geometry. For speed profiles, a fallback can use an edge-level baseline. For some discrete components, a fallback can use a supported prior or component-specific rule.

Fallback outputs should not be treated as learned behavior. They preserve executability. They do not provide the same evidence as empirical-stage or CART-stage outputs. This distinction is central to the thesis because the model is intended as an observed-behavior baseline, not only as a system that always returns an output.

Selective classification and reject-option literature supports separating predictions from cases where the model should abstain or fall back. Geifman and El-Yaniv show that selective classification evaluates the trade-off between coverage and reliability [60]. Hasan et al. discuss reject-option classification, where uncertain cases are treated differently from accepted predictions [61]. The thesis adapts this idea by reporting empirical, CART, fallback, deterministic, and unsupported prediction sources.

3.5.8. Prediction-source Diagnostics

Prediction-source diagnostics record where an output came from. An empirical-stage output indicates direct historical support in a context cell. A CART-stage output indicates accepted generalization from a tree leaf. A fallback output indicates executability without accepted learned support. A deterministic execution-rule output indicates that the result comes from a fixed safety or sequencing rule. An unsupported state indicates that no accepted output is available.

This diagnostic is part of the model result. A component with high F1 and high fallback share is less convincing as a learned behavioral component than a component with similar F1 and high empirical or CART share. Mitchell et al. discuss the importance of data lineage and provenance for understanding how outputs were produced [32]. The same principle applies here: output provenance is needed to interpret evidence strength.

Prediction-source diagnostics also help diagnose missing context. If a component relies heavily on CART, exact empirical context may be too sparse. If fallback use remains high, the available features may not capture important operational state. If unsupported cases appear, the component may need broader candidate sets, different support criteria, or new context variables.

3.5.9. Model Selection and Validation Discipline

Model selection must avoid leakage from the held-out test partition. Training data fit empirical distributions and CART estimators. Validation data guide feature-set screening, hyperparameter tuning, and selected configuration choice. Test data are used only after the selected configuration is fixed.

Wainer and Cawley discuss how model selection can produce optimistic estimates when validation and testing are not separated properly [62]. Guan et al. also show that validation-error-based selection can create bias if the same data are repeatedly used for selection and performance reporting [63]. These studies support the thesis use of a locked test partition after validation-stage model selection.

Bayesian optimization is useful because the validation objective is expensive and includes multiple support-aware terms. Snoek et al. present Bayesian optimization as a method for tuning expensive machine-learning objectives [64]. Shahriari et al. review Bayesian optimization and explain why it is useful when evaluating configurations is costly [65]. In this thesis, the objective is not only predictive accuracy; it also includes support, acceptance, fallback, and uncertainty diagnostics.

3.6. Path-choice Modeling

This section reviews literature and modeling choices related to path choice. It explains why path choice is treated as a Ground Control Agent behavior inferred from observed graph-aligned taxi trajectories. It also explains why canonical path families are used as path candidates rather than exact paths or shortest paths.

3.6.1. Path Choice as Inferred Operational Behavior

Path choice is the selection of a graph path between runway stopbar and apron boundary within an OD pair. In the data, the actual controller instruction is not observed. The selected path is inferred from the graph-aligned taxi trajectory. The component therefore estimates observed path-use behavior, not a directly observed controller command.

This distinction is common in route-choice studies. The analyst observes a realized route and infers a choice pattern from repeated observations. Train's discrete-choice framework supports the idea that observed choices can be modeled as outcomes from an alternative set and context [44]. In this thesis, the alternative set is the OD-specific path-choice set, and the context describes operating conditions such as runway configuration, demand, density, timing, and gate area.

3.6.2. Canonical Path Families as the Path-choice Unit

Canonical path families are used because exact observed paths are too sparse for direct class modeling. They preserve dominant graph structures while absorbing small reconstruction variations. This makes the path-choice target more stable and operationally interpretable.

Trajectory clustering and similarity literature supports grouping movement sequences when the goal is to reduce sparsity while preserving important structure. Churchill et al. show that surface trajectory clustering can reveal repeated airport ground-movement patterns [66]. Bian et al. review trajectory clustering methods and show that the choice of similarity representation affects the interpretation of clusters [50]. These findings support the thesis use of graph-edge overlap and path-length consistency to group exact observed paths.

The path-choice unit must still preserve operational differences. A path family should not merge paths that differ at a runway crossing, major taxiway branch, or apron-side approach if those differences matter operationally. This is why canonical grouping is treated as a support-aware cataloging step rather than as unsupervised clustering without domain constraints.

3.6.3. Path Choice Is Not Shortest-path Routing

Shortest paths are useful for fallback geometry, but observed path choice should not be assumed to follow shortest distance or shortest time. Airport path use may reflect controller strategies, runway configuration, crossing restrictions, traffic flows, standard taxi routes, apron access, and temporary procedures. A longer path can be operationally preferred when it avoids conflicts, supports queue formation, or follows a standard local corridor.

Taxi-route management concepts support this distinction. EUROCONTROL describes taxi routes as operationally managed surface routes rather than purely geometric shortest paths [48]. He et al. show that graph-search methods are useful for taxiway path finding, but graph search still depends on the chosen cost and constraints [34]. Therefore, graph search can support fallback and feasibility, but it cannot replace observed path-choice estimation.

3.6.4. Operational Context for Path Choice

Path choice depends on both static and time-varying context. The OD pair captures the broad runway–apron relation. Runway configuration describes the active runway-side structure. Gate area and apron context describe the apron-side access. Demand and density describe local traffic pressure. Time of day and runway-transition timing describe recurring operating patterns and periods close to runway changes.

This context structure is consistent with airport taxi studies that link surface outcomes to traffic state and airport configuration. Brownlee et al. discuss uncertainty in airport ground movement and show that operational context affects surface-movement predictions [67]. Wang et al. analyze taxi-time feature importance and show that operational variables contribute to predictive performance [29]. For path choice, these variables are not only predictive features; they are clues to the conditions under which certain path candidates become more or less likely.

3.6.5. Modeling Alternatives for Path Choice

Several modeling approaches could estimate path choice. Multinomial logit and nested logit models provide interpretable route-choice structures when the alternative set and attributes are well defined. Machine-

learning classifiers can predict observed path classes when enough labeled examples exist. Inverse reinforcement learning can infer route preferences from observed trajectories under a reward model.

Each approach has limitations for this thesis. Classical choice models require explicit path attributes and assumptions about utility. Flexible machine-learning classifiers can hide support limitations. Inverse reinforcement learning can be powerful but may introduce assumptions about reward structure that are hard to validate from surveillance data alone. Zhao et al. illustrate how inverse reinforcement learning can be used for route-choice modeling, but this thesis prioritizes a simpler support-aware baseline [47].

The empirical–CART–fallback approach is therefore a pragmatic choice. Empirical distributions preserve common observed path behavior. CART generalizes across related contexts when exact empirical support is sparse. Graph-search fallback preserves executable geometry when learned support is unavailable. The prediction-source diagnostic then records which source produced the selected output.

3.7. Endpoint-selection Modeling

Endpoint selection defines the runway stopbars and apron boundaries that bound the modeled taxiing scope. This section reviews why endpoint selection is a separate learned component, why arrival and departure endpoint targets differ, and why runway-side and apron-side endpoints require different context.

3.7.1. Endpoint Selection as a Taxiing-scope Boundary Problem

Endpoint selection is the problem of selecting the operational boundary points of the modeled taxi movement. These endpoints are not arbitrary coordinates. They are runway stopbars and apron boundaries that define where the taxiing-scope interval starts or ends. The endpoint-selection component therefore links raw trajectory extraction, graph reconstruction, and later path-choice sets.

The endpoint target is partly operational and partly reconstruction-dependent. The observed trajectory may show where an aircraft leaves a runway or enters an apron area, but the exact endpoint class must be inferred from graph alignment and operational boundary definitions. This is why endpoint selection is treated as a component rather than as a trivial lookup.

3.7.2. Runway-side Endpoint Evidence

Runway-side endpoints are affected by aircraft motion and runway use. For inbound movements, runway-exit behavior depends on landing speed, deceleration, runway exit availability, aircraft class, runway geometry, and local traffic conditions. Herrema et al. study runway-exit selection and show that aircraft performance and runway-exit geometry influence where aircraft leave the runway [68]. Porcayo et al. analyze runway and taxiway exits and show that surface decision-aid evaluation can benefit from endpoint-specific diagnostics [69]. These studies support the thesis use of runway-side motion and aircraft context for runway-stopbar endpoint inference.

For outbound movements, runway-entry endpoint selection is linked to departure procedure, active runway use, runway-entry availability, local restrictions, traffic state, and sequencing. SID information can be relevant because departures with similar procedures may be subject to additional separation or sequencing constraints. RECAT-EU wake category can also influence runway-departure sequencing and release timing. The KDC study on RECAT departures provides Schiphol-specific evidence for wake-category and SID-related departure timing factors [70].

3.7.3. Apron-side Endpoint Evidence

Apron-boundary endpoints are more closely tied to gate area, apron geography, aircraft type, and local density. Gate-assignment literature shows that gate and apron operations influence how aircraft connect to the taxiway system [71]. Airport surface trajectory analysis also shows that surface movement patterns depend on airport geometry and apron-side structure [49]. These findings support the thesis distinction between apron-boundary selection and runway-stopbar selection.

The thesis does not model stand-level endpoint assignment. Stand-level behavior belongs partly to apron-internal movement and gate-area operations, which are outside the main taxiing scope. Scheduled gate area can still be used as context. It helps explain which apron boundary is likely without redefining the model target as a stand-level prediction task.

3.7.4. Arrival and Departure Asymmetry

Inbound and outbound endpoint selection are asymmetric. An inbound aircraft reaches the taxiing scope after landing and runway exit. Its runway-side endpoint is linked to aircraft rollout and deceleration. Its apron-side endpoint is linked to destination apron and gate context. An outbound aircraft starts at an apron boundary and moves toward a runway-entry stopbar. Its runway-side endpoint is linked to active runway configuration, SID, operating state, and departure-side constraints.

This asymmetry explains why one generic endpoint classifier is insufficient. Different endpoint targets require different feature sets and different interpretations. A runway-exit error may reflect missing aircraft-motion or runway-exit availability context. A runway-entry error may reflect missing procedure or runway-side availability rules. An apron-boundary error may reflect missing gate-area or apron-demand context.

3.7.5. Modeling Alternatives for Endpoint Selection

Endpoint selection can be modeled as a categorical classification problem. Empirical distributions can work well when the same endpoint context appears frequently. CART estimators can generalize across related contexts when exact empirical support is sparse. More flexible classifiers could also be used, but they may obscure the support and operational interpretation of endpoint decisions.

The support-aware empirical-CART-fallback structure is appropriate because endpoint classes are often imbalanced. Common apron boundaries may have strong support. Rare runway stopbars may appear only in particular runway configurations or procedure contexts. Prediction-source diagnostics therefore matter as much as weighted F1, because they show whether endpoint predictions are learned from repeated observations or supplied by fallback rules.

3.8. Speed-profile Behavior Modeling

The speed-profile component represents aircraft-side movement execution along a selected graph path. This section reviews taxi-time prediction, edge-relative speed representation, baseline-plus-residual modeling, motion continuity, uncertainty diagnostics, and the relationship between nominal speed profiles and deterministic execution rules.

3.8.1. From Taxi Time to Speed Profiles

Taxi-time prediction is a well-studied airport-surface problem. Taxi-time models estimate the duration of taxi-out or taxi-in movements using variables such as runway configuration, aircraft type, demand, weather, distance, and queue state. Khadilkar and Balakrishnan model airport surface operations and show that taxi-time prediction must account for congestion and runway queues [72]. Ravizza et al. show that airport ground-movement models can combine statistical prediction with airport network structure [10]. Yin et al. review taxi-time prediction and emphasize that operational context strongly affects taxi-time outcomes [30].

Taxi time alone is not enough for this thesis. The integrated model requires a target speed profile along a selected graph path. Speed profiles support path-progress timing, protected-resource arrival times, conflict-event generation, following-and-merge guards, and runway-departure release diagnostics. A taxi movement with the correct total taxi time can still be locally unrealistic if the speed profile places stops or slowdowns at the wrong graph locations.

3.8.2. Edge-relative Speed Representation

Graph alignment enables speed profiles to be represented relative to directed edges. Each edge traversal can be mapped onto normalized edge progress. This makes speed behavior comparable across observations that traverse the same directed edge. It also allows speed profiles to be assembled into a path-level target profile.

Edge-relative representation is useful because airport taxi speed is highly local. Turns, intersections, runway exits, apron boundaries, stopbars, and merge points all affect speed. A global taxi-time model may hide these effects. An edge-level profile can capture typical speed behavior at specific infrastructure elements.

3.8.3. Baseline-plus-residual Modeling

The thesis uses an edge-level baseline speed profile and models residual deviations from that baseline. This structure is common in time-series and regression settings where a simple baseline captures most systematic variation and a learned model estimates remaining deviations. Rasmussen and Williams present Gaussian process regression as a general framework for modeling residual structure and uncertainty, which illustrates the broader principle of learning deviations from a prior function [73]. The thesis does not use Gaussian processes as the main estimator, but the baseline-plus-residual logic is aligned with the idea of separating a mean structure from context-dependent deviations.

For taxi speed, the edge-level baseline captures typical behavior on a directed edge. The residual captures how the current aircraft deviates from that typical behavior. Previous-edge speed, aircraft class, apron context, airline, and local movement state can explain these residuals. This makes the speed-profile component more interpretable than a direct black-box model of absolute speed.

3.8.4. Operational Context for Speed Profiles

Speed behavior depends on aircraft characteristics, local geometry, nearby traffic, and recent motion state. Aircraft class affects acceleration, braking, and taxiing practices. Operator or airline variables may reflect fleet composition, apron use, or standard operating patterns. Apron context can capture local geography and surface constraints. Previous-edge speed captures whether the aircraft approaches the current edge from a slow, fast, accelerating, or recently constrained state.

Gaikwad et al. review taxiing behavior and show that taxi movement depends on aircraft, traffic, and airport conditions [74]. Pham et al. analyze taxi trajectories and show that trajectory-derived variables can support taxi-behavior modeling [75]. These studies support the thesis choice to derive speed-profile features from graph-aligned trajectory data rather than relying only on schedule fields.

3.8.5. Stops, Acceleration, and Deterministic Interaction Rules

A learned target speed profile does not guarantee safe integrated movement. Two aircraft may receive plausible nominal speed profiles but still violate separation if they follow the same edge too closely or enter a merge area at the same time. Deterministic execution rules are therefore needed after nominal speed-profile construction.

Gipps-style car-following provides a classical basis for deterministic following and braking behavior [76]. Newell's car-following model also shows how simple rules can describe spacing and following dynamics in traffic streams [77]. These models are road-traffic models rather than airport-surface models, but they provide useful principles for deterministic following, safe-speed limits, and braking envelopes.

In this thesis, deterministic following and merge rules are not learned speed behavior. They are execution constraints. This separation preserves interpretability: the speed-profile component represents observed aircraft-side movement tendencies, while the following-and-merge guard preserves safe and executable integrated model behavior.

3.8.6. Uncertainty and Continuous Diagnostics

Speed-profile evaluation requires continuous diagnostics. RMSE penalizes large errors strongly. MAE gives a more direct measure of typical absolute error. Edge-level residual diagnostics localize where the model is weak. Prediction intervals provide uncertainty information when the model stores residual quantiles.

Gneiting and Raftery argue that probabilistic forecasts should be evaluated with proper scoring rules, which reward both calibration and sharpness [78]. Chen et al. discuss prediction intervals and show that interval evaluation should consider both coverage and width [79]. These references support the thesis use of interval score, empirical coverage, interval width, and coverage gap for speed-profile uncertainty diagnostics.

3.9. Local Conflict-resolution Modeling

Local conflict resolution represents movement-derived interaction behavior at protected taxi resources. This section reviews conflict representation, protected resources, free-motion counterfactuals, priority labels, resolution-action labels, and the limits of surveillance-only conflict interpretation.

3.9.1. Local Interactions on the Airport Surface

Aircraft taxi operations contain local interactions at intersections, merge points, apron boundaries, and crossings. Two aircraft may compete for the same protected resource or may arrive at a crossing within a short time window. The operation must maintain safe ordering and separation. The observed trajectory therefore contains evidence of how conflicts were resolved, even when the actual controller instruction is not available.

Airport conflict detection literature shows that interactions can be represented through predicted positions, resource occupancy, or trajectory intersections. Zhou et al. discuss conflict detection for airport surface operations and illustrate the importance of identifying potential interactions before they become unsafe [80]. Landry et al. study conflict prevention and show that surface safety requires anticipating interactions between aircraft movements [81]. These studies support the thesis focus on protected-resource encounters.

3.9.2. Protected Resources and Free-motion Counterfactuals

A protected taxi resource is a location where aircraft ordering or separation must be represented. It can be a graph node, apron-boundary resource, merge resource, or reviewed geometric crossing. The protected-resource concept makes local conflict modeling graph-based rather than purely coordinate-based.

Free-motion counterfactuals define what would happen if aircraft followed selected paths and speed profiles without local conflict-resolution behavior. They are useful because they identify potential interactions that the observed operation may have resolved through slowing, holding, yielding, or priority decisions. The conflict event is then defined by comparing free-motion resource occupancy or passage times.

This approach has an important limitation. A free-motion counterfactual is a modeling construct, not an observed operational plan. It helps infer whether an interaction likely required resolution, but it cannot reveal controller intent. The labels should therefore be described as movement-derived rather than as controller commands.

3.9.3. Priority and Resolution-action Labels

The priority label identifies which aircraft passed the protected resource first in the observed trajectory. The resolution-action label describes the inferred action of the own aircraft, such as proceed, slow down, or hold short. The delay target measures the difference between observed and free-motion resource-passing time when this target is available.

Bao et al. study conflict priority in aircraft surface operations and show that priority can be treated as an important component of interaction modeling [82]. Sui et al. use Monte Carlo tree search for conflict-related decision-making, which illustrates another approach to reasoning about aircraft surface interactions [83]. These studies support the relevance of priority and resolution as operational concepts, even though this thesis derives labels from movement data rather than from explicit commands.

Near-tie cases are especially difficult. If two aircraft reach a protected resource at nearly the same time, a small timing difference can change the inferred priority. A small map-matching error can also change the resource-passing time. This means that local conflict-resolution labels can be noisy even when the underlying operational concept is meaningful.

3.9.4. Controller Intent and Surveillance-only Evidence

Surveillance data show movement outcomes, not controller intent. They do not show whether a controller issued a hold-short instruction, whether a pilot slowed for comfort, or whether a slowdown resulted from another aircraft, a turn, or local procedure. Voice-to-safety research shows that operational communications can contain valuable safety and instruction information that is not present in trajectory data alone [84]. Zhuge et al. study control-command and taxi behavior, which illustrates how command information can improve interpretation when available [85].

This thesis does not use such instruction logs. It therefore limits its claim to movement-derived labels. This is a strength and a limitation. It allows large-scale analysis from surveillance data. It also means that the model should not be interpreted as recovering true controller intent.

3.9.5. Modeling Alternatives for Local Conflict Resolution

Local conflict resolution could be modeled through rule-based priority logic, optimization, game-theoretic interaction models, supervised classifiers, or learning from demonstrations. Lee et al. study joint interaction modeling, which shows that interaction outcomes can depend on the behavior of multiple agents rather than one isolated actor [86]. Chen et al. study conflict-free cooperation and show that multi-agent movement coordination can be framed through cooperative decision-making [87]. These methods are more ambitious than the movement-derived component used in this thesis.

The thesis uses a supervised support-aware component because the available evidence is trajectory-derived. Empirical distributions preserve repeated encounter patterns. CART generalizes over timing and geometry features. Fallback or deterministic safeguards preserve executability when learned support is insufficient. This is consistent with the overall observed-behavior baseline: the goal is to reproduce visible interaction patterns, not to solve a full controller decision-making problem.

3.10. Deterministic Execution Rules and Integrated Model-validation Diagnostics

Learned behavioral components estimate selected behaviors from historical observations. They do not by themselves guarantee safe or executable integrated model behavior. Deterministic execution rules are therefore needed for following, merge handling, and runway-departure release. This section reviews those rules and explains why integrated model validation is separate from component-level validation.

3.10.1. Deterministic Execution Rules in Hybrid Models

Hybrid models often combine learned components with fixed rules. The learned components capture observed behavior where data support it. The fixed rules enforce physical, safety, or sequencing constraints. In this thesis, deterministic execution rules include the following-and-merge guard and the runway-departure release rule.

This separation is important because deterministic rules have a different evidential status. A learned component output can be interpreted as empirical support or accepted generalization. A deterministic rule output cannot be interpreted as learned behavior. It is a modeling constraint introduced to keep the integrated model safe and executable.

3.10.2. Following and Merge Guards

Following interactions occur when aircraft travel along the same path or consecutive graph edges. Merge interactions occur when aircraft approach a shared downstream resource. These interactions require separation-preserving behavior during integrated model execution.

Gipps provides a classical car-following model that links safe speed to leader position, follower speed, reaction time, and braking assumptions [76]. Zhou et al. review intelligent-driver-style models and show that following models remain important for describing vehicle interaction dynamics [88]. These road-traffic references are not airport-specific, but they support the use of deterministic following principles when aircraft interact along shared movement paths.

For airport taxi operations, following and merge rules must also respect aircraft size, protected radii, graph edges, and resource occupancy. These rules preserve executable integrated movement. They should not be interpreted as a learned Aircraft Operator Agent behavior unless their parameters are explicitly estimated from observed following data.

3.10.3. Runway-departure Release

Runway-departure release represents the constraint that outbound aircraft cannot be released at runway-entry stopbars without appropriate separation. Queueing abstractions are common in runway and departure modeling. Simaiakis and Balakrishnan model departure queues and show how runway throughput and taxi-out processes interact [89]. Burgain et al. develop virtual-queue concepts for airport departure management, which supports the idea that runway-boundary queues can be represented separately from taxi movement [90].

Schiphol-specific departure separation may depend on wake category and SID. The KDC RECAT study provides operational support for departure-timing factors related to wake turbulence and same-SID effects

[70]. In this thesis, the runway-departure release rule is a deterministic runway-boundary constraint. It is not a learned runway-control clearance model.

3.10.4. Component Validation and Integrated Validation Answer Different Questions

Component-level validation evaluates whether a selected learned component reproduces its target on held-out data. It answers questions such as whether the path-choice component recovers the observed canonical path family, whether endpoint selection recovers the observed endpoint class, whether the speed-profile component matches observed edge-relative speeds, and whether the local conflict-resolution component recovers movement-derived labels.

Integrated model validation answers a different question. It asks whether selected outputs can be combined into a coherent graph-based execution sequence when deterministic rules are applied. A component can be accurate in isolation but still create timing interactions that alter conflict emergence, following behavior, or runway-departure release timing. Conversely, integrated timing errors may arise from deterministic rules even when nominal component outputs are accurate.

This distinction is supported by simulation verification and validation literature because model validity depends on the question being asked and the intended use of the model [91]. It is also supported by reactive simulation work because closed-loop model behavior can reveal issues that open-loop prediction metrics do not show [92]. The thesis therefore reports integrated model-validation diagnostics separately from component-level metrics.

3.10.5. Integrated Model Validation

Component-level validation evaluates isolated learned components. Integrated model validation evaluates whether selected component outputs and deterministic rules remain coherent when executed together. These are different questions. A component can perform well in isolation while still creating timing drift or interaction mismatches after coupling.

Sargent emphasizes that model verification and validation must be distinguished from model construction and from the intended use of the model [91]. This principle is central here because the thesis does not validate an external operational tool. It validates whether the observed-behavior baseline is internally coherent and whether the selected components can be combined in historical-scenario model execution.

Integrated model-validation diagnostics include planning completion, taxi-time error, protected-resource events, following-and-merge activations, runway-departure release diagnostics, queueing diagnostics, and trajectory-consistency checks. These diagnostics reveal coupling effects that component metrics cannot show. They also show where deterministic execution rules require refinement.

3.10.6. Boundary of the Integrated Claims

Integrated model validation should not be interpreted as operational deployment validation. The model does not observe all controller decisions. It does not include every airport-surface process. It uses deterministic simplifications for following, merge handling, and runway-departure release. The integrated diagnostics therefore test component interoperability and emergent behavior within the modeled scope.

This boundary is valuable rather than weak. It prevents overclaiming. It also makes future work clearer. Missing layout state, operational rules, protected-resource definitions, and richer movement-state context can be targeted because the validation structure shows where current simplifications matter.

3.11. Evaluation Design, Metrics, and Diagnostic Reporting

This section reviews evaluation design for support-aware taxi-behavior modeling. It covers temporal partitioning, validation/test separation, classification and probabilistic metrics, continuous diagnostics, data-volume sensitivity, and the final research gap addressed by the thesis.

3.11.1. Temporal Splitting and Operational Representativeness

Airport operations are temporally correlated. Movements close in time may share runway configuration, weather, traffic demand, local restrictions, and operational procedures. A flight-level random split can

place highly similar operating conditions in both training and test partitions. This can overstate generalization.

Similar-day grouping has been used in air traffic flow management to compare operating periods with related demand and operational characteristics [93]. K-medoids clustering is useful because each cluster is represented by an observed medoid rather than a synthetic centroid [94]. These references support the thesis use of complete ISO-week blocks and operational-profile clustering before assigning training, validation, and test partitions.

The split must serve both statistical and operational goals. It should avoid leakage. It should also preserve representative operating regimes across partitions. A partition that contains only common runway configurations cannot test rare or transitional regimes. A partition dominated by rare weeks would also distort validation. Week-blocked, cluster-stratified splitting addresses this balance.

3.11.2. Validation, Test Sets, and Model-selection Discipline

The validation partition is used for feature-set screening, hyperparameter tuning, and selected configuration choice. The test partition is used once after the configuration is fixed. This separation protects the credibility of held-out results.

Model-selection bias is a known problem when the same data are used repeatedly for tuning and evaluation. Wainer and Cawley discuss the risk of biased performance estimates when model selection and evaluation are not separated [62]. Guan et al. show that repeated selection on validation error can affect reported performance estimates [63]. These concerns justify the thesis emphasis on locked test evaluation.

3.11.3. Classification, Probability, and Calibration Metrics

Classification metrics must match the target and class imbalance. Weighted F1 reflects performance on classes according to their test support. Macro F1 treats classes equally and can reveal rare-class weakness. Confusion matrices show which classes are substituted for each other and are especially useful when errors have operational meaning.

Probabilistic metrics are needed because the learned components output distributions. The Brier score evaluates predicted probabilities against the realized class. Predictive entropy summarizes uncertainty or sharpness. Expected calibration error compares predicted confidence with empirical accuracy across confidence bins.

These metrics answer different questions. F1 asks whether the predicted class is correct. Brier score asks whether the probability vector is accurate. Entropy asks whether the distribution is concentrated or diffuse. Calibration asks whether confidence corresponds to empirical frequency. A support-aware model should report these diagnostics together rather than relying on one aggregate metric.

3.11.4. Continuous and Interval Diagnostics

Speed-profile behavior requires continuous-output metrics. RMSE penalizes large errors strongly. MAE measures typical absolute error. Edge-level maps localize where speed-profile errors occur on the airport layout. Distributional summaries show whether errors are concentrated in a few difficult graph locations or spread broadly.

Forecast-evaluation literature supports using multiple metrics because different error measures emphasize different parts of the error distribution [95]. Prediction-interval literature supports evaluating coverage and width together because an interval can cover many observations by being too wide [79]. Proper scoring-rule literature supports using scores that reward both sharpness and calibration when uncertainty is reported [78].

For this thesis, the speed-profile component is evaluated through point error, residual behavior, interval diagnostics, and prediction-source shares. This is necessary because a speed profile is a graph-local continuous object, not a single class label.

3.11.5. Training-data Sensitivity

Training-data sensitivity is important because the model is support-aware. More data may improve performance. It may also reduce fallback dependence without changing headline error metrics. A model can

therefore benefit from additional data by increasing learned support even when F1 or RMSE changes only slightly.

Stratified fraction subsets test how much representative data is needed. Chronological windows test whether broader historical coverage improves support and generalization. These two views answer different questions. A representative subset measures sample-size dependence under similar context coverage. A chronological window measures the value of additional operating regimes and longer temporal history.

Data-volume sensitivity helps diagnose model limitations. If performance improves mainly with chronological coverage, the model may need regime diversity more than raw sample count. If fallback remains high even with all data, the feature set may be missing important operational context. If unsupported cases persist, the candidate set or acceptance criteria may need revision.

3.11.6. Synthesis of the Literature Gap

The reviewed literature establishes four important points. First, ASM Ops are coupled systems in which aircraft taxi behavior depends on infrastructure, runway use, traffic demand, local procedures, human decisions, and shared resources. Second, surface-surveillance trajectories provide valuable evidence but require filtering, taxiing-scope extraction, graph-constrained reconstruction, and context joining before they can support behavior modeling. Third, path candidates, endpoint targets, speed profiles, and conflict-resolution labels have different operational meanings and should be modeled as separate learned behavioral components. Fourth, evaluation must report not only predictive performance but also support, calibration, prediction-source diagnostics, and integrated model behavior.

The gap is therefore not the absence of any airport-surface model. Optimization models, taxi-time predictors, map-matching methods, route-choice models, and simulation tools already exist. The gap is the lack of a reproducible, graph-based, support-aware observed-behavior baseline that uses surface-surveillance data to infer selected taxi-operation behaviors while preserving the distinction between learned behavior, fallback outputs, deterministic execution rules, and integrated validation diagnostics.

This thesis addresses that gap by combining graph-aligned taxi trajectories, canonical path families, context features, empirical distribution estimators, CART-based generalization, fallback rules, prediction-source diagnostics, and historical-scenario integrated model validation. The contribution lies in the combination and operationalization of these elements for aircraft taxi operations at a large airport. The following scientific article and appendices implement and evaluate that baseline for Amsterdam Airport Schiphol.

Research Definition

4.1. Research Gap

The literature review shows that airport surface movement operations (ASM Ops) have been studied through optimization, forecasting, map matching, simulation, multi-agent modeling, and data-driven trajectory analysis. These research streams provide strong methods for specific tasks. Optimization studies can compute efficient paths or schedules, map-matching studies can reconstruct movement on a graph, and simulation studies can evaluate operational concepts under assumed behavior. However, these streams do not yet provide a complete methodology for constructing a support-aware observed-behavior baseline from confidential surface-surveillance data at a large airport.

This gap matters because many airport-surface studies require assumptions about current taxi behavior. These assumptions may concern endpoint use, path choice, taxi speed, local interactions, queueing, or deterministic execution rules. If every study defines these assumptions independently, results become difficult to compare and conclusions become tied to case-specific modeling choices. A reusable observed-behavior baseline should instead be derived from operational data, represented on a common airport layout graph, and evaluated with diagnostics that show where the model is supported by historical observations.

Surface-surveillance data provide the empirical basis for such a baseline, but they are not directly usable for behavior identification. The raw records must be filtered, scoped to aircraft taxi operations, reconstructed on the airport layout graph, grouped into meaningful path candidates, and linked to operational context. These steps are not only preprocessing details. They determine whether the subsequent learned behavioral components receive reliable inputs and whether their outputs can be interpreted as observed taxi-operation behavior.

The methodological gap addressed in this thesis is therefore the absence of an integrated pipeline that combines graph-constrained reconstruction, canonical path-family construction, context-feature definition, support-aware learned behavioral components, prediction-source diagnostics, and integrated model validation. Existing methods address parts of this pipeline, but they rarely connect all steps in a way that separates learned behavior from fallback rules and deterministic execution rules.

A support-aware methodology is needed because predictive performance alone is not enough. A component may appear accurate while relying on fallback outputs or deterministic rules that do not provide evidence of learned behavior. The model must therefore report where outputs come from: empirical support, CART-based generalization, fallback rules, deterministic execution rules, or unsupported states. Without this distinction, an integrated model may appear plausible while weak learned support remains hidden.

The thesis addresses this gap through a data-driven MAS methodology for aircraft taxi operations at Amsterdam Airport Schiphol. The methodology uses VEMMIS operational records and surface-surveillance trajectories to construct graph-aligned taxi trajectories, exact observed paths, canonical path families, context features, component-specific input datasets, learned behavioral components, and integrated model-validation diagnostics. The resulting model is not an optimizer and does not claim operational deployment validity. It is an observed-behavior baseline that identifies selected taxi-operation behaviors from historical data and exposes the limits of the available data, graph representation, operational context, and deterministic execution rules.

4.2. Research Objective

Research Objective

Develop and evaluate a data-driven MAS methodology for constructing a support-aware observed-behavior baseline of aircraft taxi operations at a large airport, using Schiphol operational and surface-surveillance records to reconstruct graph-aligned taxi trajectories, identify learned behavioral components, and assess both component-level performance and integrated model behavior.

This objective is methodological and empirical. The methodological part concerns the design of the data-processing, graph-reconstruction, behavior-identification, calibration, and validation pipeline. The empirical part concerns the Schiphol case study, where the methodology is applied to historical VEMMIS operational records and surface-surveillance trajectories. The output is a structured observed-behavior baseline for runway-stopbar–apron-boundary aircraft taxi operations.

4.3. Research Questions

The research questions follow the logic of the thesis workflow. They move from data reconstruction, to behavior representation, to support-aware calibration and validation, to integrated model behavior, and finally to the interpretation limits revealed by additional case studies.

Research Question 1

How can operational records and surface-surveillance trajectories be transformed into reliable taxi trajectories, graph-aligned taxi trajectories, exact observed paths, canonical path families, path-choice sets, and component-specific input datasets for aircraft taxi operations?

This question addresses the data foundation of the thesis. It concerns taxiing-scope extraction, airport layout graph representation, graph-constrained map matching, path-catalog construction, and context-feature linking. The purpose is to determine whether confidential raw operational records can be converted into reliable graph-based data products for behavior identification.

Research Question 2

How can observed aircraft taxi behavior be represented within a MAS model that separates learned behavioral components from deterministic execution rules?

This question addresses the model structure. It concerns the allocation of endpoint selection, path choice, speed-profile behavior, local conflict resolution, following-and-merge behavior, and runway-departure release within a MAS representation. The question also concerns the separation between the historical data-processing pipeline and the executed MAS model: historical data are used to infer model properties, while the MAS model organizes the learned components and deterministic execution rules during model validation.

Research Question 3

How can endpoint selection, path choice, speed-profile behavior, and local conflict resolution be learned, calibrated, and evaluated while keeping empirical support, CART generalization, fallback use, and unsupported cases visible?

This question addresses support-aware model identification. It concerns the shared empirical, CART, and fallback inference procedure, the definition of component observations and target variables, the selection of context features, the calibration of support and acceptance criteria, and the evaluation of prediction-source diagnostics. The goal is to ensure that learned behavior is distinguished from fallback-based executability.

Research Question 4

What does integrated model validation reveal about the consistency, interoperability, and emergent timing behavior of combining selected learned behavioral components with deterministic execution rules?

This question addresses integrated model behavior. Component-level validation shows whether isolated learned components recover held-out labels, speed profiles, or probability distributions. Integrated model validation asks a different question: whether selected component outputs can be combined into coherent graph-based model execution, and whether deterministic execution rules introduce timing, queuing, or conflict-event effects that are not visible in isolated component metrics.

Research Question 5

How do missing operational context, known procedure constraints, construction-state changes, and apron-area movement characteristics affect the interpretation and transferability of the methodology?

This question addresses the limits of the observed-behavior baseline. It concerns whether selected context features are operationally meaningful, where prediction-source support changes, which errors suggest missing context, and how additional case studies expose limitations of the static graph representation, missing operational rules, and transfer to apron-area movements.

4.4. Research Scope and Boundaries

The main scope of the thesis is aircraft taxi operations between runway stopbars and apron boundaries at Amsterdam Airport Schiphol. The main model does not cover the full airside surface operation. Apron-internal movement, gate-area movement, pushback, de-icing, private-aviation movements, eastern maintenance-area movements, runway 22/04, and full tower-clearance behavior are outside the primary modeled taxiing scope.

The thesis models selected behaviors from trajectory-derived observations. It does not claim that actual ATC or pilot commands are observed. Endpoint-selection, path-choice, speed-profile, and local conflict-resolution outputs are inferred from historical trajectories and operational context. The local conflict-resolution labels are movement-derived labels based on observed timing and free-motion counterfactuals, not direct records of controller intent.

The deterministic execution rules are included to preserve safe and executable integrated model behavior. They are not learned behavioral components. This applies especially to the following-and-merge guard and runway-departure release rule. Their role is to support integrated model validation, not to claim a complete representation of aircraft-side execution or runway-control clearance behavior.

The Schiphol case study uses VEMMIS operational records and surface-surveillance trajectories from January 2024 to December 2025. Because raw VEMMIS records cannot be distributed, reproducibility depends on the documented methodology, derived data definitions, aggregate diagnostics, and generated figures and tables. The methodology can be applied to other comparable surface-surveillance data sources when sufficient trajectory, timing, layout, and operational-context information is available.

The appendices extend the main scientific paper without changing the core objective. Appendix A (Model Calibration and Tuning) documents calibration evidence and context-feature definitions. Appendix B (Verification and Validation) provides detailed reconstruction, component-level, and integrated model-validation diagnostics. Appendix C (Operational Interpretation of Selected Model Configurations) interprets the final selected features and prediction-source support. Appendix D (Construction-state Effects on Path Choice), Appendix E (Operational Constraints and Availability Rules), and Appendix F (Methodology Transfer to Apron-area Taxi Movements) examine specific limitations and transfer cases.

4.5. Planning

This section translates the research objective and research questions into a work-package plan. The plan follows the structure of the thesis: the literature review and research definition establish the academic basis; the data-processing and graph-reconstruction pipeline creates the taxi-operation data products; the learned behavioral components form the support-aware MAS model; validation and case studies assess the selected model configurations; and the final thesis report integrates the scientific article with supporting appendices.

The work packages are grouped into five phases. Baseline work packages are required for the core thesis contribution. Intermediate work packages strengthen validation, interpretation, and robustness. Advanced work packages extend the methodology to additional operational cases and clarify transferability.

4.5.1. Work-package Overview

Table 4.1: Work-package overview

Work age	pack-	Phase	Main objective	Priority
WP1		Academic framing and re-search definition	Establish the literature basis, research gap, research objective, research questions, scope, and terminology used throughout the thesis.	Baseline
WP2		Data processing and graph-based representation	Transform VEMMIS operational records and surface-surveillance trajectories into scoped taxi trajectories, graph-aligned taxi trajectories, exact observed paths, canonical path families, and context-linked data products.	Baseline
WP3		Support-aware learned behavioral components	Define and calibrate the learned endpoint-selection, path-choice, speed-profile, and local conflict-resolution components within the MAS model.	Baseline
WP4		Calibration, validation, and integrated model evaluation	Select final component configurations, evaluate held-out component performance, and assess integrated model behavior in historical validation scenarios.	Baseline
WP5		Operational interpretation and case studies	Interpret selected model configurations, identify missing context, and examine construction-state, operational-constraint, and apron-area transfer cases.	Intermediate / Advanced
WP6		Thesis and scientific paper finalization	Harmonize the scientific paper, literature review, research definition, appendices, nomenclature, figures, tables, references, and table-of-contents mappings.	Baseline

4.5.2. Detailed Work Packages

Table 4.2: Detailed work-package tasks and deliverables

Work pack- age	Main tasks	Main deliverables
WP1	Review literature on ASM Ops, surface-surveillance data, graph reconstruction, MAS modeling, data-driven behavior identification, and model validation. Define the research gap, objective, research questions, and research scope.	Literature review; research definition; research objective; research questions; scope and boundaries section.
WP2	Extract the modeled taxiing scope from VEM-MIS operational records. Construct the airport layout graph. Perform graph-constrained map matching. Build exact observed paths, canonical path families, path-choice sets, and context-feature definitions.	Scoped taxi trajectories; graph-aligned taxi trajectories; airport layout graph; path catalog; context-feature pool; component-specific input datasets.
WP3	Define the MAS model and agent categories. Implement the shared empirical, CART, and fallback inference structure. Calibrate endpoint selection, path choice, speed-profile behavior, and local conflict resolution as learned behavioral components.	MAS model definition; learned behavioral components; deterministic execution-rule boundary; prediction-source diagnostic definitions.
WP4	Construct the week-blocked, cluster-stratified split. Perform feature-set selection and Bayesian hyperparameter optimization. Evaluate selected components on the held-out test partition. Run integrated historical validation scenarios.	Selected component configurations; Bayesian optimization traces; training-data sensitivity diagnostics; component-level validation results; integrated model-validation diagnostics.
WP5	Analyze selected context features and prediction-source support. Examine missing construction state, operational availability rules, and apron-area transfer. Interpret limitations and future context needs.	Appendix C operational interpretation; Appendix D construction case study; Appendix E operational-constraints case study; Appendix F apron-area transfer case study.
WP6	Finalize the scientific paper and thesis report. Harmonize terminology, nomenclature, acronyms, figures, tables, captions, references, and appendix titles. Check compilation and table-of-contents mapping.	Final scientific paper; final thesis report; final appendices; updated nomenclature; compiled report.

4.5.3. Milestones

The work packages are connected through milestone gates. Each milestone marks a point where the next phase depends on the previous one being stable enough to support downstream work.

Table 4.3: Thesis milestones

Milestone	Completion criterion	Main deliverables
M1	Literature review, research gap, research objective, research questions, and scope are consistent.	Literature review; research definition; planning section.
M2	Taxiing-scope extraction, airport layout graph, and graph-constrained map matching produce a usable graph-aligned taxi dataset.	Taxi trajectories; graph-aligned taxi trajectories; reconstruction diagnostics.
M3	Exact observed paths, canonical path families, path-choice sets, context features, and component-specific input datasets are available.	Path catalog; context-feature definitions; component-specific input datasets.
M4	Learned behavioral components are calibrated on training and validation data, while the held-out test partition remains unused.	Selected component configurations; Bayesian optimization traces; training-data sensitivity diagnostics.
M5	Component-level validation and integrated model-validation scenarios are completed and interpreted.	Held-out component diagnostics; integrated model-validation diagnostics; prediction-source diagnostics.
M6	Construction-state, operational-constraints, and apron-area transfer case studies are completed.	Appendices D–F.
M7	Scientific paper, literature review, research definition, appendices, nomenclature, references, and TOC mappings are finalized.	Final thesis report and scientific article.

4.5.4. Indicative Timeline

The indicative timeline follows the dependency structure of the work packages. Earlier phases produce the data products and model definitions; later phases evaluate, interpret, and document the final methodology.

Table 4.4: Indicative thesis timeline

Period	Main activities	Milestone
November–December 2025	Literature review, research definition, terminology alignment, initial data access, and taxiing-scope definition.	M1
December 2025–January 2026	Data ingestion, preprocessing, airport layout graph preparation, taxiing-scope extraction, and graph-constrained map matching.	M2
January–February 2026	Path-catalog construction, context-feature definitions, component-specific input datasets, and MAS model allocation.	M3
February–March 2026	Development and calibration of endpoint-selection, path-choice, speed-profile, and local conflict-resolution components.	M4
March–April 2026	Held-out component validation, prediction-source diagnostics, training-data sensitivity analysis, and integrated model-validation scenarios.	M5
April–May 2026	Construction-state, operational-constraints, and apron-area transfer case studies.	M6
May–July 2026	Scientific paper finalization, literature-review expansion, appendix writing, report harmonization, nomenclature, figure/table checks, and final compilation.	M7

This planning supports the research objective by keeping the thesis measurement-first, support-aware, and validation-driven. The thesis first establishes reliable graph-aligned taxi trajectories and context-rich component inputs. It then identifies selected taxi-operation behaviors through learned behavioral components and evaluates them with prediction-source diagnostics. Finally, integrated model validation and

case-study appendices are used to interpret component interoperability, emergent timing effects, missing operational context, and transferability.

References

- [1] International Civil Aviation Organization. *Advanced Surface Movement Guidance and Control Systems (A-SMGCS) Manual*. Doc 9830 AN/452. 2004.
- [2] EUROCONTROL. *EUROCONTROL Specification for Advanced Surface Movement Guidance and Control System (A-SMGCS) Services*. Tech. rep. EUROCONTROL-SPEC-171. Edition 2.0. EUROCONTROL, 2020. URL: <https://www.eurocontrol.int/sites/default/files/2020-04/eurocontrol-specification-a-smgcs-v-2-0.pdf>.
- [3] Sabine Wilke et al. "Airport Surface Operations: A Holistic Framework for Operations Modeling and Risk Management". In: *Safety Science* 63 (2014), pp. 18–33. DOI: 10.1016/j.ssci.2013.10.015. URL: <https://doi.org/10.1016/j.ssci.2013.10.015>.
- [4] Husni R. Idris et al. "Queuing Model for Taxi-Out Time Estimation". In: *Air Traffic Control Quarterly* 10.1 (2002), pp. 1–22. DOI: 10.2514/ATCQ.10.1.1. URL: <https://doi.org/10.2514/ATCQ.10.1.1>.
- [5] Ioannis Simaiakis et al. "Impact of Congestion on Taxi Times, Fuel Burn, and Emissions at Major Airports". In: *Transportation Research Record* 2184.1 (2010), pp. 22–30. DOI: 10.3141/2184-03. URL: <https://doi.org/10.3141/2184-03>.
- [6] Regina Clewlow et al. "Impact of Arrivals on Departure Taxi Operations at Airports". In: *AIAA Guidance, Navigation, and Control Conference*. 2010. DOI: 10.2514/6.2010-7698. URL: <https://doi.org/10.2514/6.2010-7698>.
- [7] Angel G. Marin. "Airport Management: Taxi Planning". In: *Annals of Operations Research* 143 (2006), pp. 191–202. DOI: 10.1007/s10479-006-7381-2. URL: <https://doi.org/10.1007/s10479-006-7381-2>.
- [8] Geoff L. Clare et al. "Optimization of Taxiway Routing and Runway Scheduling". In: *IEEE Transactions on Intelligent Transportation Systems* 12.4 (2011), pp. 1000–1013. DOI: 10.1109/TITS.2011.2131650. URL: <https://doi.org/10.1109/TITS.2011.2131650>.
- [9] Dimitris Bertsimas et al. "Unified Optimization of Traffic Flows Through Airports". In: *Transportation Science* 50.1 (2016), pp. 77–93. DOI: 10.1287/trsc.2015.0590. URL: <https://doi.org/10.1287/trsc.2015.0590>.
- [10] Stefan Ravizza et al. "A Combined Statistical Approach and Ground Movement Model for Improving Taxi Time Estimations at Airports". In: *Journal of the Operational Research Society* 64.9 (2013), pp. 1347–1360. DOI: 10.1057/jors.2012.123. URL: <https://doi.org/10.1057/jors.2012.123>.
- [11] Charles M. Macal et al. "Tutorial on Agent-Based Modelling and Simulation". In: *Journal of Simulation* 4.3 (2010), pp. 151–162. DOI: 10.1057/jos.2010.3. URL: <https://doi.org/10.1057/jos.2010.3>.
- [12] Malte von der Burg et al. "Modelling and Analysis of Autonomous Airport Surface Movement Operations Based on Multi-Agent Planning: Explorative Case Study at Amsterdam Airport Schiphol". In: *European Journal of Transport and Infrastructure Research* 25.1 (2025). DOI: 10.59490/ejtir.2025.25.1.7459. URL: <https://doi.org/10.59490/ejtir.2025.25.1.7459>.
- [13] Jacco Hoekstra et al. "BlueSky ATC Simulator Project: An Open Data and Open Source Approach". In: *Proceedings of the 7th International Conference on Research in Air Transportation*. 2016. URL: <https://research.tudelft.nl/en/publications/bluesky-atc-simulator-project-an-open-data-and-open-source-approach/>.

- [14] Ewout ter Hoeven et al. "Mesa 3: Agent-Based Modeling with Python in 2025". In: *Journal of Open Source Software* 10.107 (2025), p. 7668. DOI: 10.21105/joss.07668. URL: <https://doi.org/10.21105/joss.07668>.
- [15] Ingrid Navarro et al. "Amelia: A Large Dataset and Benchmark for Airport Surface Movement Forecasting". In: *arXiv* (2024). arXiv: 2407.21185 [cs.LG]. URL: <https://arxiv.org/abs/2407.21185>.
- [16] Federal Aviation Administration. *Airport Surface Detection Equipment, Model X (ASDE-X)*. 2023. URL: https://www.faa.gov/air_traffic/technology/asde-x.
- [17] Edwin A. Bloem et al. *Advanced Data Fusion for Airport Surveillance*. Tech. rep. NLR-TP-2002-052. National Aerospace Laboratory NLR, 2002. URL: <https://reports.nlr.nl/server/api/core/bitstreams/0101be88-55dc-48ca-a996-0c1085f4dc40/content>.
- [18] Stephen Pledgie et al. "Methods for Aircraft State Estimation from Airport Surface Surveillance". In: *AIAA Guidance, Navigation, and Control Conference*. 2009. DOI: 10.2514/6.2009-5645. URL: <https://doi.org/10.2514/6.2009-5645>.
- [19] Martin Strohmeier et al. "Crowdsourced Air Traffic Data from the OpenSky Network 2019–2020". In: *Earth System Science Data* 13 (2021), pp. 357–366. DOI: 10.5194/essd-13-357-2021. URL: <https://doi.org/10.5194/essd-13-357-2021>.
- [20] Matthias Schäfer et al. "OpenSky Report 2018: Assessing the Integrity of Crowdsourced Mode S and ADS-B Data". In: *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. 2018, pp. 1–9. DOI: 10.1109/DASC.2018.8569833. URL: <https://doi.org/10.1109/DASC.2018.8569833>.
- [21] Thanh-Nam Tran et al. "A Map-Matching Algorithm for Ground Movement Trajectory Representation using A-SMGCS Data". In: *2020 International Conference on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT)*. 2020, pp. 1–8. DOI: 10.1109/AIDA-AT48540.2020.9049181. URL: <https://doi.org/10.1109/AIDA-AT48540.2020.9049181>.
- [22] Xavier Olive et al. "Detection and Identification of Significant Events in Historical Aircraft Trajectory Data". In: *Transportation Research Part C: Emerging Technologies* 119 (2020), p. 102737. DOI: 10.1016/j.trc.2020.102737. URL: <https://doi.org/10.1016/j.trc.2020.102737>.
- [23] Xavier Olive et al. "Filtering Techniques for ADS-B Trajectory Preprocessing". In: *Journal of Open Aviation Science* 2.2 (2025). DOI: 10.59490/joas.2024.7882. URL: <https://doi.org/10.59490/joas.2024.7882>.
- [24] Navid Mirmohammadsadeghi et al. "Taxi Event Extraction from Surveillance for Surface Performance Evaluation". In: *Journal of Air Transportation* 28.2 (2020), pp. 28–35. DOI: 10.2514/1.D0160. URL: <https://doi.org/10.2514/1.D0160>.
- [25] Michael Schultz et al. "Data-Driven Airport Management Enabled by Operational Milestones Derived from ADS-B Messages". In: *Journal of Air Transport Management* 99 (2022), p. 102164. DOI: 10.1016/j.jairtraman.2021.102164. URL: <https://doi.org/10.1016/j.jairtraman.2021.102164>.
- [26] Michael Bloem et al. "Filtering of Aircraft Surface Trajectories". In: *AIAA AVIATION 2020 Forum*. 2020. DOI: 10.2514/6.2020-2884. URL: <https://doi.org/10.2514/6.2020-2884>.
- [27] Xavier Olive et al. *Filtering Aircraft Surface Trajectories Using Information on the Taxiway Structure of Airports*. ICRAAT 2024 contribution. 2024. DOI: 10.21256/ZHAW-34908. URL: <https://doi.org/10.21256/ZHAW-34908>.
- [28] Erik Vargo et al. "Airport Taxi Time Prediction and Alerting: A Convolutional Neural Network Approach". In: *arXiv preprint arXiv:2111.09139* (2021). URL: <https://arxiv.org/abs/2111.09139>.
- [29] Xinyu Wang et al. "Aircraft Taxi Time Prediction: Feature Importance and Their Implications". In: *Transportation Research Part C: Emerging Technologies* 124 (2021), p. 102892. DOI: 10.1016/j.trc.2020.102892. URL: <https://doi.org/10.1016/j.trc.2020.102892>.

- [30] Jie Yin et al. "Prediction and Analysis of Airport Surface Taxi Time: Classification, Features, and Methodology". In: *Applied Sciences* 14.3 (2024), p. 1306. DOI: 10.3390/app14031306. URL: <https://doi.org/10.3390/app14031306>.
- [31] RDA FAIR Curation Working Group. *10 Things for Curating Reproducible and FAIR Research*. Tech. rep. Research Data Alliance, 2022. DOI: 10.15497/RDA00074. URL: <https://doi.org/10.15497/RDA00074>.
- [32] Sonia Natalie Mitchell et al. "FAIR Data Pipeline: Provenance-Driven Data Management for Traceable Scientific Workflows". In: *arXiv preprint arXiv:2110.07117* (2021). URL: <https://arxiv.org/abs/2110.07117>.
- [33] Enrico Spinielli et al. "Towards Reproducibility in ANS Performance". In: *SESAR Innovation Days*. 2019. URL: https://www.sesarju.eu/sites/default/files/documents/sid/2019/papers/SIDs_2019_paper_70.pdf.
- [34] Siyuan He et al. "Verification of an Airport Taxiway Path-Finding Algorithm". In: *2020 IEEE/AIAA 39th Digital Avionics Systems Conference (DASC)*. 2020. DOI: 10.1109/DASC50938.2020.9256698. URL: <https://doi.org/10.1109/DASC50938.2020.9256698>.
- [35] Kartik A. Pant et al. "SAFE-TAXI: A Hierarchical Multi-UAS Safe Auto-Taxiing Framework with Runtime Safety Assurance and Conflict Resolution". In: *arXiv preprint arXiv:2503.04942* (2025). URL: <https://arxiv.org/abs/2503.04942> (visited on 07/02/2026).
- [36] Brooks A. Butler et al. "Safe Reference Tracking and Collision Avoidance for Taxiing Aircraft Using an MPC-CBF Framework". In: *arXiv preprint arXiv:2410.03890* (2024). URL: <https://arxiv.org/abs/2410.03890> (visited on 07/02/2026).
- [37] Mohammed A. Quddus et al. "Current Map-Matching Algorithms for Transport Applications: State-of-the Art and Future Research Directions". In: *Transportation Research Part C: Emerging Technologies* 15.5 (2007), pp. 312–328. DOI: 10.1016/j.trc.2007.05.002. URL: <https://doi.org/10.1016/j.trc.2007.05.002>.
- [38] Mohammed A. Quddus et al. "Integrity of Map-Matching Algorithms". In: *Transportation Research Part C: Emerging Technologies* 14.4 (2006), pp. 283–302. DOI: 10.1016/j.trc.2006.08.004. URL: <https://doi.org/10.1016/j.trc.2006.08.004>.
- [39] Paul Newson et al. "Hidden Markov Map Matching Through Noise and Sparseness". In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. New York, NY, USA: Association for Computing Machinery, 2009, pp. 336–343. DOI: 10.1145/1653771.1653818. URL: <https://doi.org/10.1145/1653771.1653818>.
- [40] Yin Lou et al. "Map-Matching for Low-Sampling-Rate GPS Trajectories". In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. New York, NY, USA: Association for Computing Machinery, 2009, pp. 352–361. DOI: 10.1145/1653771.1653820. URL: <https://doi.org/10.1145/1653771.1653820>.
- [41] Xinyu Wang et al. "An Interval Type-2 Fuzzy Logic Based Map Matching Algorithm for Airport Ground Movements". In: *IEEE Transactions on Fuzzy Systems* 31.2 (2023), pp. 582–595. DOI: 10.1109/TFUZZ.2022.3221793. URL: <https://doi.org/10.1109/TFUZZ.2022.3221793>.
- [42] Zhihao Gao et al. "A Novel Semantic Representation of Airport Surface Trajectory for Taxiing Pattern Recognition". In: *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*. 2023. DOI: 10.1109/DASC58513.2023.10311315. URL: <https://doi.org/10.1109/DASC58513.2023.10311315>.
- [43] Peter E. Hart et al. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136. URL: <https://doi.org/10.1109/TSSC.1968.300136>.
- [44] Kenneth E. Train. *Discrete Choice Methods with Simulation*. 2nd ed. Cambridge University Press, 2009. URL: <https://eml.berkeley.edu/books/choice2.html>.

- [45] Carlo Giacomo Prato. “Route Choice Modeling: Past, Present and Future Research Directions”. In: *Journal of Choice Modelling* 2.1 (2009), pp. 65–100. DOI: 10.1016/S1755-5345(13)70005-8. URL: [https://doi.org/10.1016/S1755-5345\(13\)70005-8](https://doi.org/10.1016/S1755-5345(13)70005-8).
- [46] Rui Yao et al. “Experiments on Route Choice Set Generation Using a Large GPS Trajectory Set”. In: *arXiv preprint arXiv:2006.04536* (2020). URL: <https://arxiv.org/abs/2006.04536>.
- [47] Zhan Zhao et al. “A Deep Inverse Reinforcement Learning Approach to Route Choice Modeling with Context-Dependent Rewards”. In: *Transportation Research Part C: Emerging Technologies* 149 (2023), p. 104079. DOI: 10.1016/j.trc.2023.104079. URL: <https://doi.org/10.1016/j.trc.2023.104079>.
- [48] EUROCONTROL Experimental Centre. *Standard Airport Taxi Route Study*. Tech. rep. EEC Technical/Scientific Report No. 2009-002. EUROCONTROL Experimental Centre, 2009. URL: https://www.eurocontrol.int/sites/default/files/library/002_Standard_airport_taxi_route_study.pdf.
- [49] Martin Schlosser et al. “Enhancing Aircraft Ground Trajectories through Map-Matching and Stochastic Pavement Roughness Modeling”. In: *Journal of Open Aviation Science* 2.2 (2025). DOI: 10.59490/joas.2024.7898. URL: <https://doi.org/10.59490/joas.2024.7898>.
- [50] Jiang Bian et al. “A Survey on Trajectory Clustering Analysis”. In: *arXiv preprint arXiv:1802.06971* (2018). URL: <https://arxiv.org/abs/1802.06971>.
- [51] Danlei Hu et al. “Spatio-Temporal Trajectory Similarity Measures: A Comprehensive Survey and Quantitative Study”. In: *IEEE Transactions on Knowledge and Data Engineering* 36.5 (2024), pp. 2191–2212. DOI: 10.1109/TKDE.2023.3323535. URL: <https://doi.org/10.1109/TKDE.2023.3323535>.
- [52] Larry Wasserman. *All of Nonparametric Statistics*. New York: Springer, 2006. DOI: 10.1007/0-387-30623-4. URL: <https://doi.org/10.1007/0-387-30623-4>.
- [53] Leo Breiman et al. *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group, 1984.
- [54] Foster Provost et al. “Tree Induction for Probability-Based Ranking”. In: *Machine Learning* 52.3 (2003), pp. 199–215. DOI: 10.1023/A:1024099825458. URL: <https://doi.org/10.1023/A:1024099825458>.
- [55] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324. URL: <https://doi.org/10.1023/A:1010933404324>.
- [56] Jerome H. Friedman. “Greedy Function Approximation: A Gradient Boosting Machine”. In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451. URL: <https://doi.org/10.1214/aos/1013203451>.
- [57] Cynthia Rudin. “Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215. DOI: 10.1038/s42256-019-0048-x. URL: <https://doi.org/10.1038/s42256-019-0048-x>.
- [58] Chuan Guo et al. “On Calibration of Modern Neural Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1321–1330. URL: <https://proceedings.mlr.press/v70/guo17a.html>.
- [59] Tim Leathart et al. “Probability Calibration Trees”. In: *arXiv preprint arXiv:1808.00111* (2018). URL: <https://arxiv.org/abs/1808.00111>.
- [60] Yonatan Geifman et al. “Selective Classification for Deep Neural Networks”. In: *arXiv preprint arXiv:1705.08500* (2017). URL: <https://arxiv.org/abs/1705.08500>.
- [61] Mehedi Hasan et al. “Survey on Leveraging Uncertainty Estimation Towards Trustworthy Deep Neural Networks: The Case of Reject Option and Post-Training Processing”. In: *arXiv preprint arXiv:2304.04906* (2023). URL: <https://arxiv.org/abs/2304.04906>.

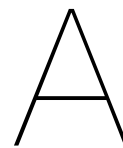
- [62] Jacques Wainer et al. "Nested Cross-Validation When Selecting Classifiers Is Overzealous for Most Practical Applications". In: *Expert Systems with Applications* 182 (2021). Preprint available as arXiv:1809.09446, p. 115222. DOI: 10.1016/j.eswa.2021.115222. URL: <https://doi.org/10.1016/j.eswa.2021.115222>.
- [63] Laying Guan. "Test Error Estimation after Model Selection Using Validation Error". In: *arXiv preprint arXiv:1801.02817* (2018). URL: <https://arxiv.org/abs/1801.02817> (visited on 07/02/2026).
- [64] Jasper Snoek et al. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems*. Vol. 25. 2012. URL: https://papers.nips.cc/paper_files/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html.
- [65] Bobak Shahriari et al. "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175. DOI: 10.1109/JPROC.2015.2494218. URL: <https://doi.org/10.1109/JPROC.2015.2494218>.
- [66] Andrew Churchill et al. "Hierarchical Clustering of Aircraft Surface Trajectories". In: *AIAA AVIATION 2020 Forum*. 2020. DOI: 10.2514/6.2020-2870. URL: <https://doi.org/10.2514/6.2020-2870>.
- [67] Alexander E. I. Brownlee et al. "A Fuzzy Approach to Addressing Uncertainty in Airport Ground Movement Optimisation". In: *Transportation Research Part C: Emerging Technologies* 92 (2018), pp. 150–175. DOI: 10.1016/j.trc.2018.04.020. URL: <https://doi.org/10.1016/j.trc.2018.04.020>.
- [68] Floris Herrema et al. "A Machine Learning Model to Predict Runway Exit at Vienna Airport". In: *Transportation Research Part E: Logistics and Transportation Review* 131 (2019), pp. 329–342. DOI: 10.1016/j.tre.2019.10.002. URL: <https://doi.org/10.1016/j.tre.2019.10.002>.
- [69] Alex Porcayo et al. "Data-Driven Runway and Taxiway Exits Prediction of Landing Aircraft: A Case Study at Hartsfield-Jackson Atlanta International Airport". In: *arXiv preprint arXiv:2606.11017* (2026). URL: <https://arxiv.org/abs/2606.11017>.
- [70] Ander Okina et al. *RECAT-EU for Departures at Schiphol*. Tech. rep. FWY-2020-01. Version V1.0; executed by FerWay and MovingDot. Knowledge and Development Centre Mainport Schiphol, 2020. URL: <https://kdc-mainport.nl/wp-content/uploads/2021/01/FWY-2020-01-RECAT-EU-for-departures-v1.0.pdf>.
- [71] Chendong Li. "Airport Gate Assignment: A Hybrid Model and Implementation". In: *arXiv preprint arXiv:0903.2528* (2009). URL: <https://arxiv.org/abs/0903.2528> (visited on 06/30/2026).
- [72] Harshad Khadilkar. "Analysis and Modeling of Airport Surface Operations". PhD thesis. Massachusetts Institute of Technology, 2011. URL: <https://dspace.mit.edu/handle/1721.1/68939>.
- [73] Carl Edward Rasmussen et al. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL: <https://gaussianprocess.org/gpml/>.
- [74] Prashant Gaikwad et al. "Developing a Computer Vision Based System for Autonomous Taxiing of Aircraft". In: *Aviation* 27.4 (2023), pp. 248–258. DOI: 10.3846/aviation.2023.20588. URL: <https://doi.org/10.3846/aviation.2023.20588>.
- [75] Duc-Thinh Pham et al. "A Generative Adversarial Imitation Learning Approach for Realistic Aircraft Taxi-Speed Modeling". In: *IEEE Transactions on Intelligent Transportation Systems* 23.3 (2022), pp. 2509–2522. DOI: 10.1109/TITS.2021.3119073. URL: <https://doi.org/10.1109/TITS.2021.3119073>.
- [76] P. G. Gipps. "A Behavioural Car-Following Model for Computer Simulation". In: *Transportation Research Part B: Methodological* 15.2 (1981), pp. 105–111. DOI: 10.1016/0191-2615(81)90037-0. URL: [https://doi.org/10.1016/0191-2615\(81\)90037-0](https://doi.org/10.1016/0191-2615(81)90037-0).
- [77] G. F. Newell. "A Simplified Car-Following Theory: A Lower Order Model". In: *Transportation Research Part B: Methodological* 36.3 (2002), pp. 195–205. DOI: 10.1016/S0191-2615(00)00044-8. URL: [https://doi.org/10.1016/S0191-2615\(00\)00044-8](https://doi.org/10.1016/S0191-2615(00)00044-8).

- [78] Tilmann Gneiting et al. "Strictly Proper Scoring Rules, Prediction, and Estimation". In: *Journal of the American Statistical Association* 102.477 (2007), pp. 359–378. DOI: 10.1198/016214506000001437. URL: <https://doi.org/10.1198/016214506000001437>.
- [79] Haoxian Chen et al. "Learning Prediction Intervals for Regression: Generalization and Calibration". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Vol. 130. Proceedings of Machine Learning Research. 2021, pp. 820–828. URL: <https://proceedings.mlr.press/v130/chen21b.html> (visited on 07/02/2026).
- [80] Hang Zhou et al. "Research on Taxiway Path Optimization Based on Conflict Detection". In: *PLOS ONE* 10.7 (2015), e0134522. DOI: 10.1371/journal.pone.0134522. URL: <https://doi.org/10.1371/journal.pone.0134522>.
- [81] Steven J. Landry et al. "A Decision Support Methodology for Dynamic Taxiway and Runway Conflict Prevention". In: *Decision Support Systems* 55.1 (2013), pp. 165–174. DOI: 10.1016/j.dss.2013.01.016. URL: <https://doi.org/10.1016/j.dss.2013.01.016>.
- [82] Jie Bao et al. "A Dynamic Control Method for Airport Ground Movement Optimization Considering Adaptive Traffic Situation and Data-Driven Conflict Priority". In: *Journal of Air Transport Management* 125 (2025), p. 102753. DOI: 10.1016/j.jairtraman.2025.102753. URL: <https://doi.org/10.1016/j.jairtraman.2025.102753>.
- [83] Dong Sui et al. "A Conflict Resolution Strategy at a Taxiway Intersection by Combining a Monte Carlo Tree Search with Prior Knowledge". In: *Aerospace* 10.11 (2023), p. 914. DOI: 10.3390/aerospace10110914. URL: <https://doi.org/10.3390/aerospace10110914>.
- [84] Yutian Pang et al. "From Voice to Safety: Language AI Powered Pilot-ATC Communication Understanding for Airport Surface Movement Collision Risk Assessment". In: *arXiv preprint arXiv:2503.04974* (2025). DOI: 10.48550/arXiv.2503.04974. URL: <https://arxiv.org/abs/2503.04974> (visited on 06/30/2026).
- [85] Jingchang Zhuge et al. "Aircraft Ground Taxiing Deduction and Conflict Early Warning Method Based on Control Command Information". In: *arXiv preprint arXiv:2211.02320* (2022). URL: <https://arxiv.org/abs/2211.02320> (visited on 06/30/2026).
- [86] Donsuk Lee et al. "Joint Interaction and Trajectory Prediction for Autonomous Driving using Graph Neural Networks". In: *arXiv preprint arXiv:1912.07882* (2019). URL: <https://arxiv.org/abs/1912.07882> (visited on 07/02/2026).
- [87] Chaoyi Chen et al. "Conflict-Free Cooperation Method for Connected and Automated Vehicles at Unsignalized Intersections: Graph-Based Modeling and Optimality Analysis". In: *arXiv preprint arXiv:2107.07179* (2021). URL: <https://arxiv.org/abs/2107.07179> (visited on 07/02/2026).
- [88] Shirui Zhou et al. "Twenty-Five Years of the Intelligent Driver Model: Foundations, Extensions, Applications, and Future Directions". In: *arXiv preprint arXiv:2506.05909* (2025). DOI: 10.48550/arXiv.2506.05909. URL: <https://arxiv.org/abs/2506.05909> (visited on 07/02/2026).
- [89] Ioannis Simaiakis et al. "A Queuing Model of the Airport Departure Process". In: *Transportation Science* 50.1 (2016), pp. 94–109. DOI: 10.1287/trsc.2015.0603. URL: <https://doi.org/10.1287/trsc.2015.0603>.
- [90] Pierrick Burgain et al. "Collaborative Virtual Queue: Fair Management of Congested Departure Operations and Benefit Analysis". In: *arXiv preprint arXiv:0807.0661* (2008). URL: <https://arxiv.org/abs/0807.0661> (visited on 07/02/2026).
- [91] Robert G. Sargent. "Verification and Validation of Simulation Models". In: *Journal of Simulation* 7.1 (2013), pp. 12–24. DOI: 10.1057/jos.2012.20. URL: <https://doi.org/10.1057/jos.2012.20>.
- [92] Luca Bergamini et al. "SimNet: Learning Reactive Self-driving Simulations from Real-world Observations". In: *arXiv preprint arXiv:2105.12332* (2021). URL: <https://arxiv.org/abs/2105.12332> (visited on 07/02/2026).

- [93] Kenneth D. Kuhn. “A Methodology for Identifying Similar Days in Air Traffic Flow Management Initiative Planning”. In: *Transportation Research Part C: Emerging Technologies* 69 (2016), pp. 411–429. DOI: 10.1016/j.trc.2016.05.014. URL: <https://doi.org/10.1016/j.trc.2016.05.014>.
- [94] Leonard Kaufman et al. *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: John Wiley & Sons, 1990.
- [95] Vitor Cerqueira et al. “Evaluating Time Series Forecasting Models: An Empirical Study on Performance Estimation Methods”. In: *arXiv preprint arXiv:1905.11744* (2019). URL: <https://arxiv.org/abs/1905.11744>.

Part IV

Appendices



Model Calibration and Tuning

A.1. Purpose and Relation to the Scientific Paper

The scientific paper summarizes the model-training, tuning, and evaluation workflow in Section IV and reports the selected component configurations in Section V. This appendix provides the calibration evidence behind that summary. It documents the week-blocked operational-profile split, defines the context-feature pool used during calibration, shows the Bayesian hyperparameter optimization traces, and analyzes training-data sensitivity.

The appendix does not introduce alternative model specifications, additional learned components, or implementation-specific export details. Its role is to explain how the selected learned behavioral components were calibrated before locked test evaluation. The context-feature definitions clarify what information was available to the empirical and CART stages, while the optimization and sensitivity analyses show how support, prediction-source behavior, fallback dependence, and training-data availability affected the selected configurations.

This separation keeps the scientific paper focused on the main methodology and results. Appendix B (Verification and Validation) evaluates the selected components on the held-out test partition, and Appendix C (Operational Interpretation of Selected Model Configurations) interprets the operational meaning of the final selected context features and context-conditioned performance patterns.

A.2. Training, Validation, and Test Partitioning

The scientific paper defines the pooled 2024-2025 week-blocked, cluster-stratified training, validation, and test split. This appendix expands that design by showing how the operational weeks were grouped before stratified assignment. The goal is to verify that the split preserves the main operating regimes in the two-year dataset, rather than assigning weeks only by calendar order or random sampling.

The split is week-blocked because aircraft taxi operations are temporally correlated. Movements within the same week often share runway configurations, traffic levels, weather conditions, demand patterns, and local operational constraints. Keeping complete weeks in one partition reduces leakage between training, validation, and test sets. The additional clustering step then ensures that each partition contains a representative mix of operational regimes.

Each ISO week is summarized by a weekly operational profile. The profile combines variables that describe the operating conditions of that week, including traffic per day, inbound share, local 30-minute density, mean taxi time, mean path length, runway-use entropy, OD-pair entropy, and temperature. These feature groups are normalized before clustering so that no single group dominates the operational-distance matrix. K-medoids clustering is used because each cluster is represented by an observed week, or medoid, rather than by a synthetic centroid. This makes the clusters easier to interpret operationally.

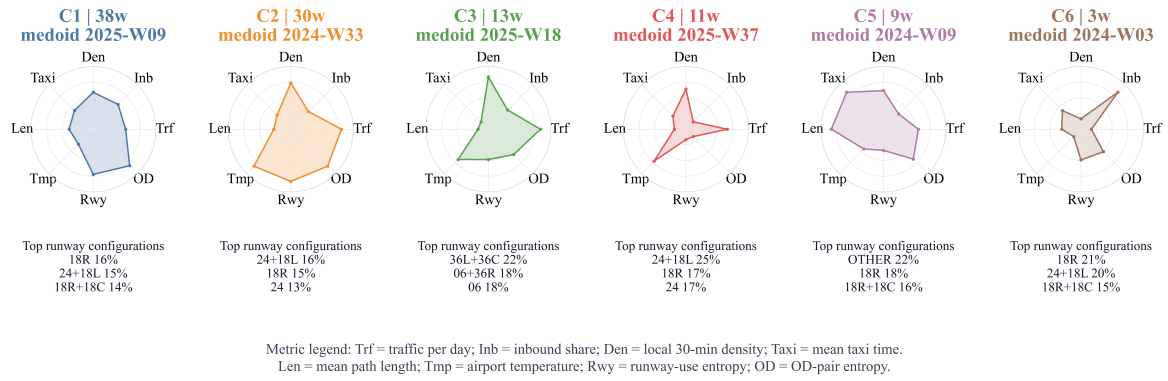
**Figure A.1:** Weekly operational-profile clusters

Figure A.1 identifies six weekly operational-profile clusters. The largest clusters represent common operating regimes, while the smaller clusters preserve rarer weeks with unusual traffic, runway-use, or directional patterns. This is important for model calibration: if rare regimes were concentrated in only one partition, validation results could reflect partition bias rather than model generalization.

Table A.1: Interpretation of weekly operational-profile clusters

Cluster	Size and medoid	Dominant runway configurations	Operational interpretation
C1	38 weeks; 2025-W09	18R; 24+18L; 18R+18C	Largest baseline cluster. It represents common mixed operations with moderate traffic, density, path length, taxi time, and runway-use diversity.
C2	30 weeks; 2024-W33	24+18L; 18R; 24	High-activity warm-period cluster. It combines relatively high traffic, density, runway-use diversity, and OD-pair diversity, making it important for calibration under busy representative conditions.
C3	13 weeks; 2025-W18	36L+36C; 06+36R; 06	Distinct runway-use regime dominated by 36/06 configurations. It captures weeks with strong traffic and density signals but a different runway-configuration structure from the larger baseline clusters.
C4	11 weeks; 2025-W37	24+18L; 18R; 24	Concentrated high-traffic regime. The profile emphasizes traffic and temperature while showing more concentrated runway-use and OD structure than the broad high-activity clusters.
C5	9 weeks; 2024-W09	Other; 18R; 18R+18C	Long-path and long-taxi regime. The cluster captures weeks with larger path-length and taxi-time signatures and a higher share of less common runway-configuration patterns.
C6	3 weeks; 2024-W03	18R; 24+18L; 18R+18C	Small rare-regime cluster. It is inbound-heavy and lower in traffic and density, but is retained in the stratification so that unusual operating weeks are not excluded from validation coverage.

Table A.1 interprets the six clusters in terms of their operational role. Clusters C1 and C2 cover most of the two-year sample and provide the main representative operating conditions. Clusters C3 and C4 preserve runway-configuration regimes that differ from the broad baseline. Cluster C5 captures longer-path and longer-taxi weeks with less common runway-configuration patterns. Cluster C6 is small but important because it keeps rare inbound-heavy, lower-density weeks visible during split construction.

The stratified assignment uses these clusters when assigning complete weeks to the training, validation, and test partitions. The target is not to make the partitions identical, but to avoid a split in which one partition contains only common weeks while another contains rare or operationally distinct weeks. This supports the calibration workflow in the rest of this appendix: empirical and CART stages are trained on a broad operating sample, validation tuning is exposed to representative operational variation, and the locked test partition remains a meaningful check of generalization. The same split is used by the

path-choice, endpoint-selection, speed-profile, and local conflict-resolution components, so the calibration diagnostics in this appendix are aligned with the component-level validation results in Appendix B (Verification and Validation).

A.3. Context Feature Definitions

The scientific paper reports the selected empirical and CART feature sets and describes how candidate feature sets are screened during model training. This section defines the context-feature pool from which those candidate sets were constructed. The purpose is to clarify which operational information was available during calibration and how derived context features were calculated from processed trajectory or operational records. The operational interpretation of the final selected features is provided in Appendix C.

Context features are restricted to information available before, or at the time of, the represented behavior. This restriction avoids using future outcomes to explain a decision or movement pattern. Endpoint-selection and path-choice features describe the movement context before the endpoint or path is inferred. Speed-profile features describe conditions available before estimating an edge-level residual speed profile. Local conflict-resolution features describe the encounter state before assigning a movement-derived priority or resolution-action label.

A.3.1. Movement, Endpoint, and OD Features

Movement, endpoint, and OD features define the operational identity of a taxi movement. They include movement direction, runway stopbar, apron boundary, OD pair, and OD group. These variables are derived from taxiing-scope extraction and graph alignment. Movement direction distinguishes inbound from outbound taxi movements. The runway stopbar and apron boundary define the endpoints of the modeled taxiing scope. Together, movement direction, runway stopbar, and apron boundary define the OD pair used by the path-choice component and by OD-specific diagnostics.

An OD group is a coarser runway-apron grouping used when exact OD-pair diagnostics would be too sparse. It preserves the main spatial relation between runway side and apron side while reducing the number of categories. In the calibration workflow, OD-related variables can serve as conditioning variables, grouping variables, or target-specific context, depending on the learned behavioral component. They are not all used as features for every component.

A.3.2. Schedule, Aircraft, and Procedure Features

Schedule, aircraft, and procedure features are derived from flight-level and schedule records linked to each taxi movement. They include aircraft class, wake category, airline or operator, Standard Instrument Departure (SID), and scheduled gate area. These variables describe aircraft, operator, apron-side planning, and departure-procedure context available before the modeled taxi movement or decision is inferred.

Aircraft class and wake category provide size and performance context. Airline or operator information can capture repeated operational patterns, fleet composition, and apron usage. SID information is relevant for outbound runway-stopbar selection and runway-departure release behavior because departures with similar procedures may be subject to additional sequencing constraints. Scheduled gate area provides apron-side context without treating the exact stand as an endpoint target, since stand-level endpoint assignment is outside the modeled taxiing scope.

A.3.3. Runway and Operating-State Features

Runway and operating-state features describe the airport configuration active around the taxi movement. They include runway configuration, runway role, BZO phase, and runway-configuration transition features. Runway configuration identifies the active arrival and departure runway setup. Runway role describes how a runway participates in the operation, for example as an arrival, departure, or mixed-use runway. BZO phase is a Schiphol operational phase variable that summarizes higher-level operating state.

Runway-transition features describe the temporal proximity of a taxi movement to a runway-configuration change. The “minutes since runway-configuration change” feature is calculated as the elapsed time between the taxi movement and the most recent runway-configuration transition. The “minutes until runway-configuration change” feature is calculated as the time until the next transition in the historical operating-state sequence. In retrospective model identification, this sequence is taken from historical operational

records. In prospective use, the same type of feature would require a planned runway-configuration schedule; otherwise, only elapsed-time information can be used without future-state knowledge.

A.3.4. Temporal Features

Temporal features encode recurring daily, weekly, and seasonal structure. They include taxi start minute of day, time-of-day category, weekday, and broader calendar indicators. Taxi start minute of day is derived from the timestamp at which the taxiing-scope interval starts. Time-of-day categories group this continuous timestamp into operational periods such as early morning, morning peak, midday, afternoon peak, evening, and other low-volume periods.

These features allow the empirical and CART stages to distinguish repeated operating patterns that are not fully captured by runway configuration or traffic counts. They are useful when operational procedures, traffic demand, airline schedules, or runway-use patterns vary systematically over the day or week.

A.3.5. Demand and Density Features

Demand and density features summarize nearby traffic conditions around the evaluated taxi movement. They include 30-minute direction-demand count, 30-minute local-density count, same-apron demand, and local density. These features are derived from time-aligned taxi movements rather than from a single flight record.

The 30-minute direction-demand count is the number of taxi movements with the same movement direction in the relevant 30-minute operating window around the evaluated movement. It represents directional traffic pressure. The 30-minute local-density count is the number of nearby taxi movements in the same operating window and represents surface-traffic intensity. Same-apron demand counts movements linked to the same apron area and captures local apron-side pressure. Local density is a discretized or grouped version of nearby movement intensity used when a component benefits from categorical density context rather than a raw count.

These features are included because many taxi-operation behaviors depend on nearby traffic, not only on the aircraft's own flight record. They help distinguish isolated movements from movements taking place under congested or highly interactive operating conditions.

A.3.6. Trajectory-Derived Motion Features

Trajectory-derived motion features are calculated from processed taxi trajectories or graph-aligned taxi trajectories. They include exit speed at threshold, exit speed at fixed downstream distances, and previous-edge speed. These variables describe motion state before the target behavior is estimated.

For inbound runway-stopbar selection, runway-exit speed features are sampled from the early post-landing trajectory. The threshold feature records the speed at the runway threshold reference, while the +250 m, +500 m, and +1000 m features record speed at fixed distances downstream along the observed runway-exit trajectory. These features describe how the aircraft decelerates after landing and are therefore relevant for runway-side endpoint inference.

For the speed-profile component, previous-edge speed describes local motion continuity before the current edge-level residual is estimated. It is derived from the accepted speed observation on the preceding directed edge of the selected or observed path. This feature allows the residual-speed estimator to account for whether the aircraft approaches the current edge from a slow, fast, accelerating, or recently constrained state.

A.3.7. Path, Graph, and Protected-Resource Features

Path, graph, and protected-resource features are derived from the airport layout graph, the path catalog, and protected-resource definitions. They include path candidate, path-choice set, directed edge, resource group, and protected taxi resource. These variables connect the learned behavioral components to the graph-based representation used throughout the methodology.

For path choice, the path-choice set defines the available canonical path families for the OD pair. For speed-profile behavior, the directed edge defines the local graph element on which an edge-relative speed profile or residual profile is estimated. For local conflict resolution, resource group and protected

taxi resource identify the graph node, apron-boundary resource, merge resource, or geometric crossing involved in a pairwise encounter.

These graph-derived features are not raw data fields. They are created after graph-constrained map matching, path-catalog construction, and protected-resource extraction. They are included because the learned behavioral components estimate behavior on the graph representation, not directly on raw coordinate traces.

A.3.8. Conflict-Event Features

Conflict-event features describe the pairwise state of two aircraft approaching a protected taxi resource. They include relative estimated time of arrival (ETA), relative ETA bin, own direction, other direction, own distance to resource, other distance to resource, and resource group. These features are derived from free-motion trajectories and protected-resource geometry.

Relative ETA is the difference between the free-motion arrival times of the two aircraft at the protected taxi resource. It indicates which aircraft would reach the resource first if no local conflict-resolution behavior were applied. Relative ETA bin is a discretized version of this timing difference and is used when the empirical stage requires categorical context. Own and other direction describe the movement directions of the two aircraft. Own and other distance to resource measure the remaining path distance from each aircraft to the protected taxi resource at the encounter-evaluation time.

These features provide the context needed to infer movement-derived priority and resolution-action labels. They do not represent observed controller commands. They describe the encounter geometry and timing from which the historical movement-derived labels are inferred.

This section defines the calibration input space used by the learned behavioral components. The scientific paper describes how these features are screened and tuned during model selection, while Appendix C interprets the final selected feature sets and their context-conditioned validation behavior.

A.4. Context Feature Coverage and Dataset Profile

The previous section defined the context-feature pool available during calibration. This section describes how those features are distributed in the final graph-aligned taxi dataset. The purpose is to check whether the calibration input space covers a broad range of operational conditions, not to interpret the final selected feature sets. The operational meaning of the selected features after model selection is discussed in Appendix C.

Context-feature coverage matters because the learned behavioral components are support-aware. Empirical-stage outputs require enough repeated observations in comparable contexts, while CART-stage outputs require leaves with sufficient support and stability. If important operating regimes are underrepresented, the model may rely more heavily on CART generalization or fallback rules. A dataset profile is therefore needed to understand which operational conditions are well represented before interpreting component-level validation results.

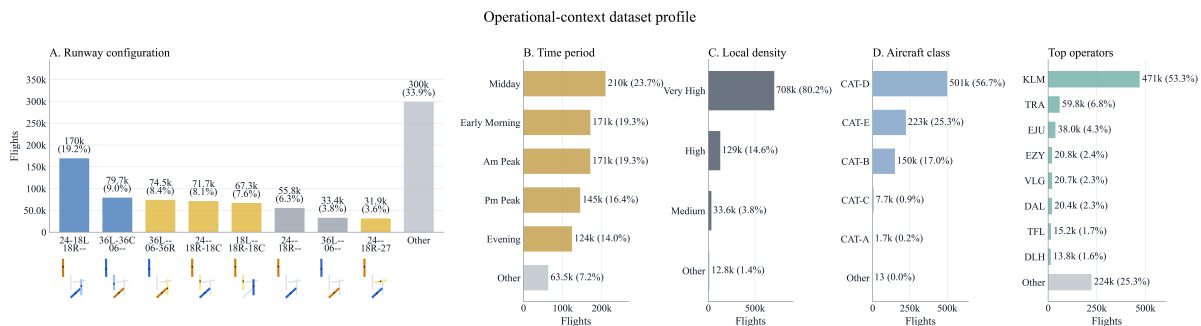


Figure A.2: Context-feature coverage profile

Figure A.2 summarizes the main context-feature distributions in the final graph-aligned taxi dataset. The profile covers endpoint volumes, runway configurations, aircraft and operator composition, temporal

context, and density or demand state. These dimensions correspond to the feature families defined in Section A.3.

The endpoint-volume panels show how taxi movements are distributed over runway stopbars and apron boundaries. This distribution is important because endpoint-selection and path-choice targets are not equally supported across all OD pairs. High-volume endpoints provide strong empirical support, while rare runway stopbars or apron boundaries may require CART generalization or fallback behavior during model execution.

The runway-configuration and temporal panels show the operating regimes represented in the dataset. Runway configuration is central for path choice and endpoint selection because it constrains the active runway-side structure of the operation. Time-of-day and weekday patterns capture recurring schedule and demand structure. These distributions confirm that the dataset contains both dominant operating patterns and less frequent regimes, which supports the week-blocked, cluster-stratified split described in Section A.2.

The aircraft, operator, and demand panels describe the traffic composition available to the learned behavioral components. Aircraft class and operator composition support aircraft-side and operator-related context in endpoint selection and speed-profile behavior. Density and demand features distinguish isolated taxi movements from movements occurring under high traffic pressure. This is important for path choice, speed-profile residual estimation, and local conflict-resolution observations.

The dataset profile shows that the final graph-aligned taxi dataset is broad enough to support context-conditioned calibration, while still being dominated by common Schiphol operating patterns. This imbalance is expected in operational data. The calibration workflow therefore combines empirical support, CART generalization, and fallback rules so that high-support contexts can preserve observed behavior while sparse contexts remain executable.

A.5. Bayesian Hyperparameter Optimization

The scientific paper summarizes Bayesian hyperparameter optimization as the validation-stage step that selects the final component configurations. This appendix expands that summary by showing how the validation objective evolved during tuning. The purpose is to demonstrate that the selected configurations were chosen from a structured search over support-aware model settings, not from manual adjustment after test evaluation.

Bayesian hyperparameter optimization is applied after the candidate empirical and CART feature sets have been defined. At this stage, the feature structure of each candidate configuration is fixed, while the tuning process adjusts component-specific hyperparameters. These include empirical support thresholds, temporal-coverage requirements, CART leaf-acceptance criteria, CART complexity settings, and fallback settings. The exact hyperparameter ranges differ by component because endpoint selection, path choice, speed-profile behavior, and local conflict resolution have different targets and support requirements.

The validation objective combines predictive performance with prediction-source diagnostics. This is necessary because high predictive performance alone does not show whether a component is supported by historical observations. A configuration that performs well but relies heavily on fallback rules is less informative as a learned behavioral component than a configuration with comparable performance and stronger empirical or CART support. The validation objective therefore favors configurations that balance predictive quality, learned support, fallback control, and executable output availability.

The objective values are component-specific. They should not be compared across components as physical metrics, because each component uses a different target variable, metric combination, and support diagnostic. Their role is to rank candidate configurations within the same component family before the held-out test partition is evaluated.

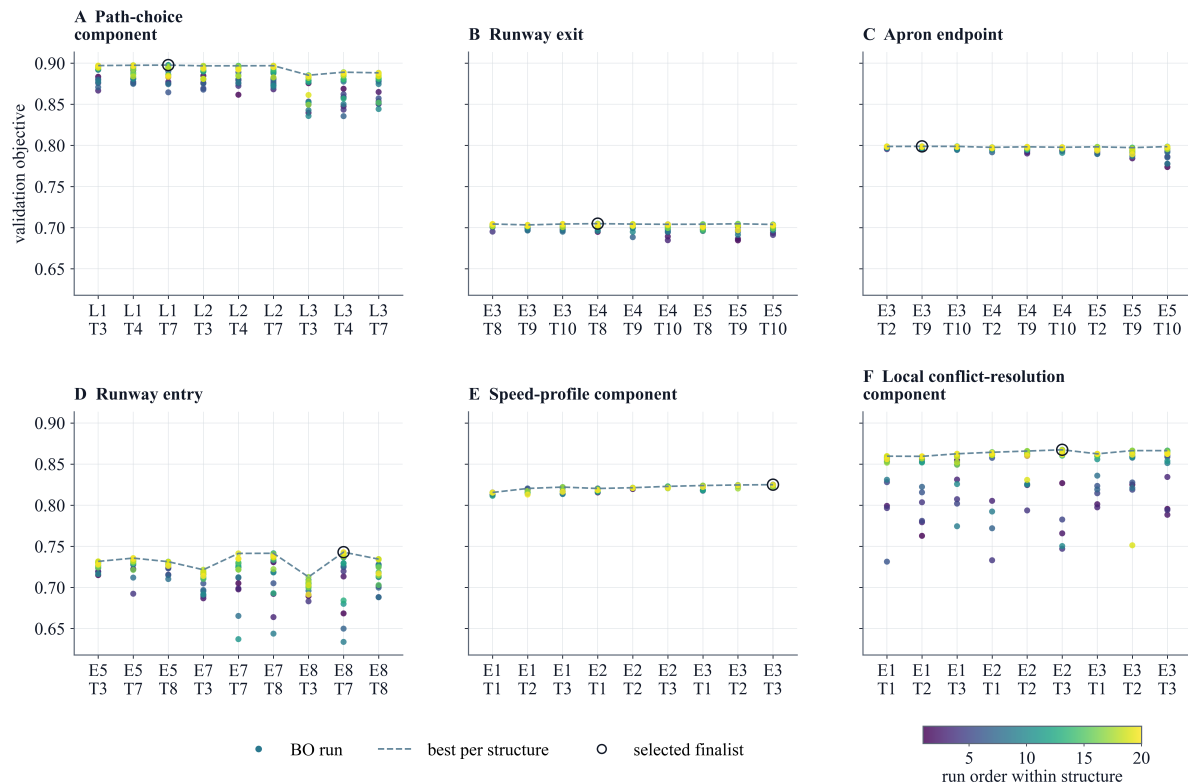


Figure A.3: Bayesian optimization validation traces

Figure A.3 shows the validation objective values explored during Bayesian hyperparameter optimization. The selected finalists lie near the upper envelope of their respective candidate families, indicating that the search identified configurations with strong validation-stage performance under the component-specific objective. The traces also show why Bayesian optimization is useful in this setting: the objective surface is irregular because each trial changes both predictive performance and prediction-source behavior.

The optimization step preserves the train/validation/test separation. Training observations fit the empirical and CART stages, validation observations guide hyperparameter selection, and the held-out test partition remains unused until the selected configuration is fixed. The selected configurations are therefore locked before the component-level validation reported in Appendix B (Verification and Validation) and the integrated model-validation scenarios discussed in the scientific paper.

A.6. Training-Data Sensitivity Analysis

The scientific paper summarizes the training-data sensitivity design. This section adds the diagnostic interpretation: how much the selected learned behavioral components depend on training-data amount and temporal coverage. The purpose is not to reselect the models, but to test whether the selected calibration remains stable when the available training history is reduced or expanded.

The selected feature sets, acceptance structure, and tuning settings remain fixed. Only the training observations available to the empirical and CART stages are varied. The validation population used for the sensitivity diagnostics remains fixed, and the locked test partition is not used for this analysis. This design isolates the effect of training-data availability from changes in the evaluation population.

Two training-subset sequences are compared. The fraction sequence uses nested, stratified subsets with 5%, 10%, 25%, 50%, 75%, and 100% of the training observations. This sequence tests how much representative training data is needed. The chronological sequence uses expanding historical windows of 7, 14, 30, 61, 92, 183, 274, 365, 548, and 730 days. This sequence tests whether broader historical coverage improves validation behavior as more operating conditions become available.

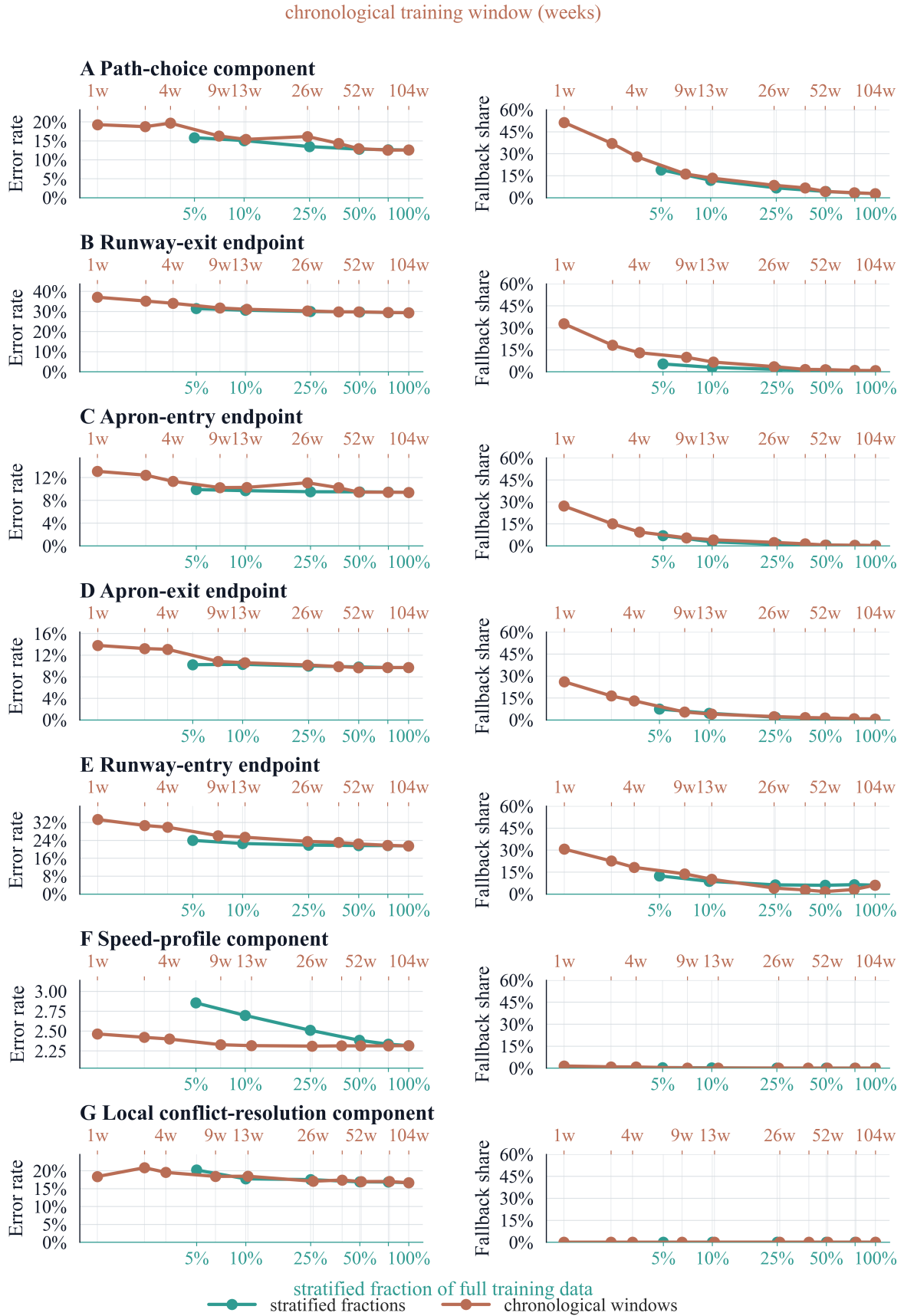


Figure A.4: Training-data sensitivity diagnostics

Figure A.4 shows that additional training data generally improves learned support and reduces fallback dependence. The effect is strongest for components whose targets are sparse across OD pairs, runway stopbars, or rare operational contexts. Path choice benefits from broader historical coverage because many OD-specific path candidates have limited support. Endpoint-selection fallback decreases most clearly for runway-stopbar targets, where the operational context is more varied than for apron-boundary targets.

The speed-profile and local conflict-resolution components are less sensitive to small training subsets in terms of learned support, but their errors remain context-dependent. Speed-profile residuals depend on local motion state and graph location, while local conflict-resolution labels are sensitive to near-tie timing and protected-resource geometry. Additional data therefore improves these components less by changing the overall support level and more by refining behavior in specific operating contexts.

The sensitivity analysis adds two calibration conclusions. First, more data mainly improves the reliability of learned support and reduces unnecessary fallback use. Second, the value of additional data depends on the operational structure of the target. Sparse choice sets and rare endpoint classes need broader historical coverage, while dense motion and interaction datasets may already support the main learned stages with smaller subsets.

A.7. Calibration and Tuning Takeaways

This appendix provides the calibration evidence behind the selected learned behavioral components reported in the scientific paper. Its main purpose is to show that the final configurations were selected through a controlled validation-stage process, not through test-set adjustment or manual tuning after evaluation.

The first calibration takeaway is that the split design protects the validity of the evaluation. Complete ISO weeks are kept together, and weekly operational-profile clustering preserves variation in runway use, traffic volume, movement direction, density, demand, and temporal operating conditions. This gives the empirical and CART stages a broad training base while keeping the validation and test partitions operationally meaningful.

The second takeaway is that context features define the information available to the learned behavioral components. Movement, schedule, runway, temporal, demand, motion, graph, and conflict-event features provide the calibration input space. Derived features such as demand counts, runway-transition timing, previous-edge speed, and relative ETA translate raw operational records and graph-aligned trajectories into model-ready context. These definitions are needed before feature selection and tuning can be interpreted.

The third takeaway is that Bayesian hyperparameter optimization tunes support-aware model behavior, not only predictive accuracy. The validation objective combines performance with prediction-source diagnostics, so selected configurations are favored when they preserve empirical or CART support and limit unnecessary fallback dependence. This is important because fallback outputs preserve executability but do not provide evidence of learned behavior.

The fourth takeaway is that training-data sensitivity is component-dependent. Additional training data generally improves learned support and reduces fallback dependence, but the benefit depends on the structure of the target. Sparse path candidates and rare endpoint classes need broader historical coverage, while speed-profile and local conflict-resolution components already have broad support under smaller training subsets and mainly benefit through context-specific refinement.

Taken together, Appendix A shows how the final learned behavioral components were calibrated before locked test evaluation. Appendix C then interprets what the final selected context features reveal about the operational structure of the modeled taxi behaviors, while Appendix B evaluates the selected components on the held-out test partition.

Verification and Validation

B.1. Purpose and Relation to the Scientific Paper

The scientific paper reports compact verification and validation results in Section V. This appendix provides the supporting evidence behind those results. It documents the reconstruction quality of the graph-aligned taxi dataset, the validity of the path catalog, the detailed metric definitions, the held-out component diagnostics, and the integrated model-validation scenario diagnostics.

The appendix is organized as an evidence layer, not as a second methodology section. The paper explains how the model is built and reports the headline results. This appendix shows the diagnostic detail that supports those claims: whether the processed trajectories are reliable enough for behavior identification, whether path-catalog compression preserves useful path candidates, how component-level metrics are defined, and how selected component outputs behave when combined with deterministic execution rules.

The validation evidence is deliberately separated by level. Data reconstruction and path-catalog diagnostics verify the quality of the derived data products. Component-level diagnostics evaluate the selected learned behavioral components on the held-out test partition. Integrated model-validation diagnostics evaluate whether selected outputs and deterministic execution rules remain coherent during historical-scenario model execution. These levels answer different questions and should not be interpreted as the same type of validation.

B.2. Data Reconstruction and Map-Matching Validation

The first validation layer checks whether surface-surveillance trajectories can be converted into graph-aligned taxi trajectories on the airport layout graph. This matters because all downstream components use graph-derived objects: exact observed paths, canonical path families, edge-relative speed profiles, endpoint targets, and protected-resource events. If the reconstruction were unreliable, the learned behavioral components would inherit that error.

The reconstruction quality is evaluated with matching mode, point-to-path projection error, high-error exclusions, and spatial concentration of reconstruction failures. These diagnostics test both aggregate quality and local representation limits of the airport layout graph.

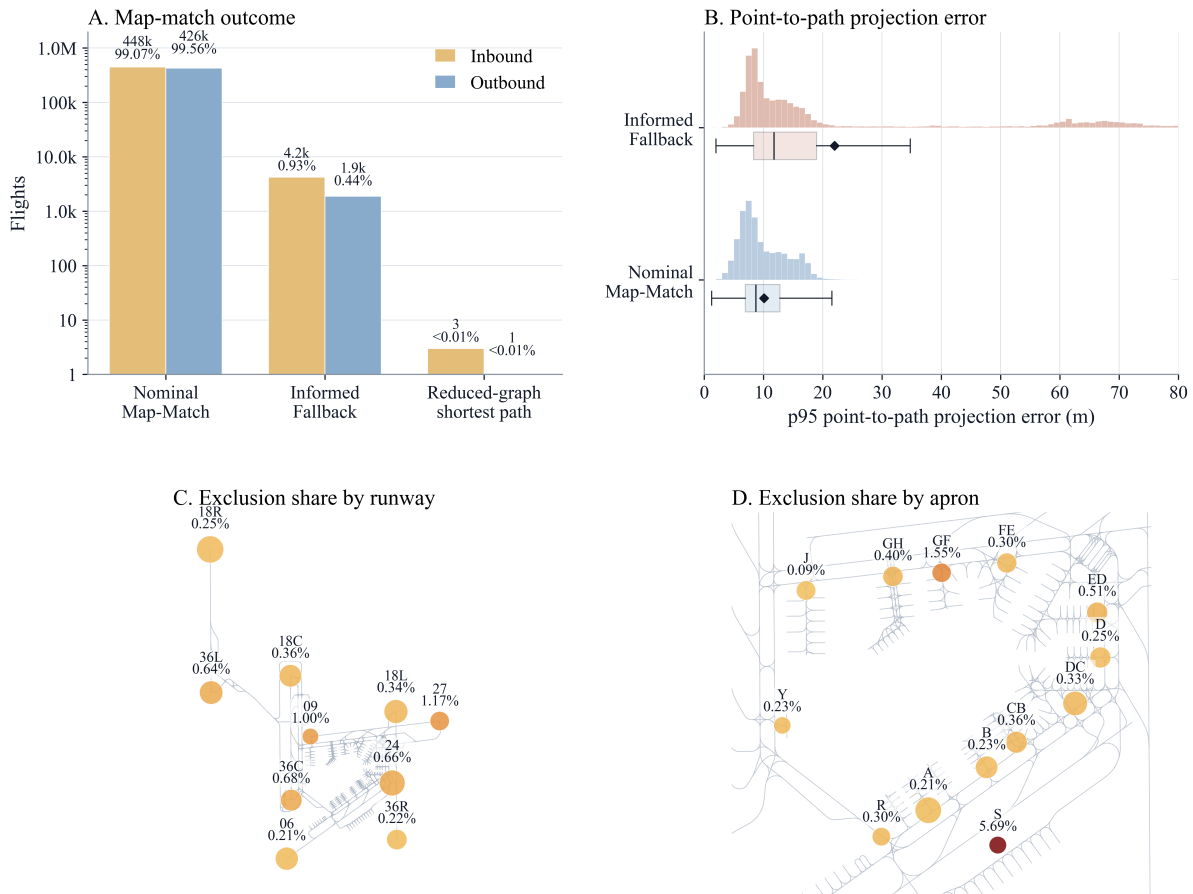


Figure B.1: Map-matching validation diagnostics

Figure B.1 shows that most retained taxi trajectories are reconstructed with low projection error. The scientific paper reports a median 95th-percentile point-to-path projection error of 8.71 m for nominal graph-aligned taxi trajectories. Fallback reconstruction is used for less than 1% of trajectories, reduced-graph shortest-path recovery is needed in only four cases, and high-error exclusions with 95th-percentile error above 80 m account for 0.46% of the retained set.

The important diagnostic result is that reconstruction problems are localized rather than global. The high-error concentrations around runway 27, runway 09, and apron S indicate local representation or surveillance-data issues. They are therefore treated as limitations of the static graph and data source, not as evidence that graph-constrained map matching fails for the modeled taxiing scope as a whole.

B.3. Path Catalog and Canonical Path-Family Diagnostics

After map matching, exact observed paths are grouped into canonical path families. This step reduces tactical and reconstruction-level variation while preserving the main path candidates available within each OD pair. The grouping used in this study applies mutual directed-edge-count coverage and path-length consistency. Two paths are grouped when they meet the 70% mutual directed-edge-count coverage criterion and the 15% length-difference threshold described in the scientific paper.

The purpose of this validation layer is to check whether the catalog reduces sparsity without collapsing meaningful path differences. A catalog with too many exact paths would make path choice sparse and unstable. A catalog with too few families would remove operationally relevant path candidates.

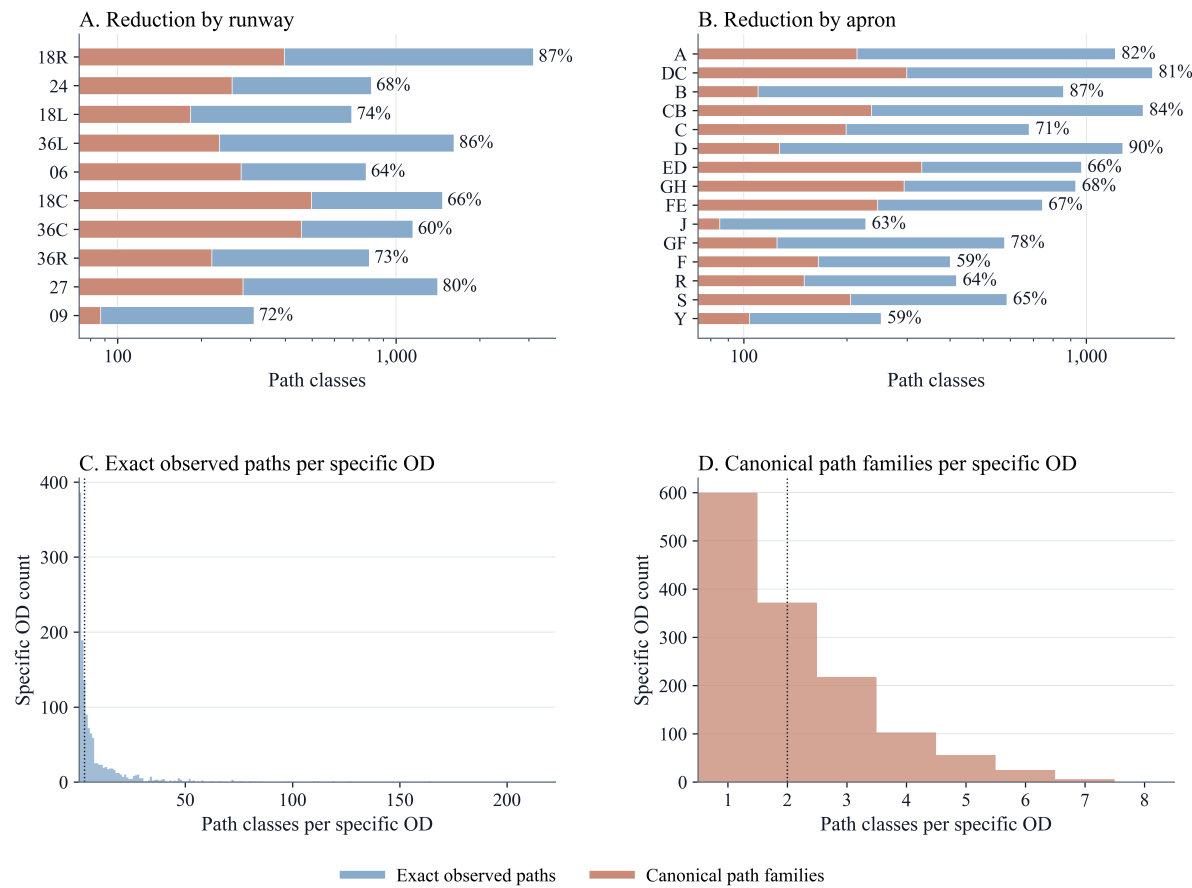


Figure B.2: Path-catalog compression diagnostics

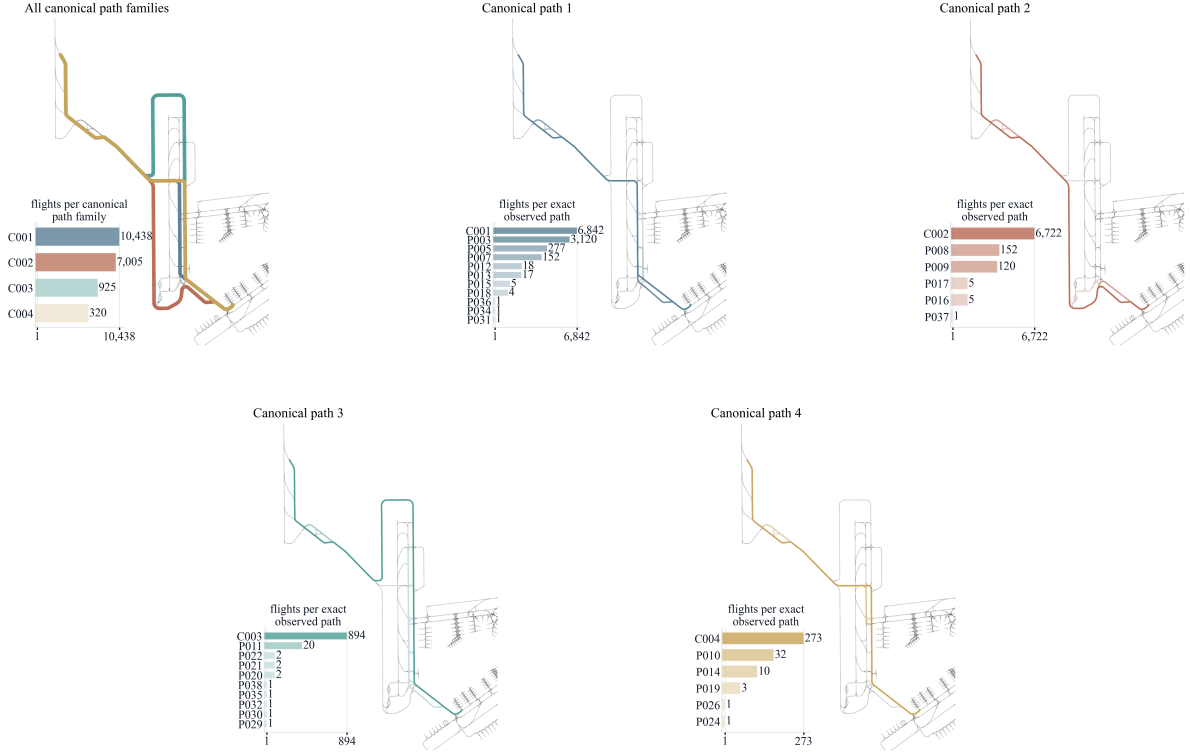


Figure B.3: Representative canonical path families

Figure B.2 shows the reduction from exact observed paths to canonical path families across OD structures. The final path catalog contains 883,771 graph-aligned taxi trajectories, 12,735 exact observed paths, and 3,002 canonical path families. The average number of path candidates per OD pair decreases from 8.81 exact observed paths to 2.09 canonical path families. This confirms that the catalog reduces sparsity substantially.

Figure B.3 shows a representative high-traffic OD pair. The canonical path families preserve the dominant path structures while absorbing small local deviations and map-matching variation. This supports the use of canonical path families as path candidates for the path-choice component.

B.4. Validation Metrics and Diagnostic Definitions

The scientific paper reports metric families and headline values. This section defines the detailed diagnostics used in the appendix figures and tables. Let y_i be the observed class label for observation i , let $\hat{p}_{ik}$ be the predicted probability of class k , and let K_i be the number of classes available to observation i .

The normalized Brier score used for classification diagnostics is

$$B_i = \frac{1}{2} \sum_{k=1}^{K_i} (\hat{p}_{ik} - \mathbf{1}(y_i = k))^2. \quad (\text{B.1})$$

The factor $1/2$ places the multiclass Brier score on a common 0–1 scale for one-hot class labels, where 0 indicates a perfect probabilistic prediction.

The normalized predictive entropy is

$$H_i = -\frac{1}{\log K_i} \sum_{k=1}^{K_i} \hat{p}_{ik} \log (\hat{p}_{ik}). \quad (\text{B.2})$$

It measures the sharpness of the predicted class-probability distribution. The diagnostic is evaluated for cases with $K_i > 1$; single-class cases are treated as trivial for entropy-based interpretation.

Expected calibration error (ECE) compares confidence and empirical accuracy over confidence bins b :

$$\text{ECE} = \sum_b \frac{n_b}{N} |\text{acc}(b) - \text{conf}(b)|. \quad (\text{B.3})$$

Here, n_b is the number of observations in bin b , N is the total number of observations, $\text{acc}(b)$ is the empirical accuracy in the bin, and $\text{conf}(b)$ is the mean predicted confidence.

For the speed-profile component, the central 80% prediction-interval score uses lower and upper bounds ℓ_i and u_i , observation y_i , and $\alpha = 0.20$:

$$S_{\alpha,i} = (u_i - \ell_i) + \frac{2}{\alpha}(\ell_i - y_i)\mathbf{1}(y_i < \ell_i) + \frac{2}{\alpha}(y_i - u_i)\mathbf{1}(y_i > u_i). \quad (\text{B.4})$$

The score rewards narrow intervals, but penalizes intervals that miss the observed value.

The empirical coverage gap is

$$G_{80} = \left| \frac{1}{N} \sum_{i=1}^N \mathbf{1}(\ell_i \leq y_i \leq u_i) - 0.80 \right|. \quad (\text{B.5})$$

It reports the absolute difference between achieved coverage and the nominal 80% target.

Integrated model-validation timing error is reported as model-produced taxi time minus the processed historical taxi-time reference:

$$\Delta t_i = t_i^{\text{model}} - t_i^{\text{hist}}. \quad (\text{B.6})$$

Positive values indicate that the model-produced movement is later than the processed historical reference.

Conflict-pair agreement is computed on unordered aircraft pairs detected as conflict events in the model execution and in the historical reference extraction. Pair precision and recall are

$$P_{\text{pair}} = \frac{|\mathcal{P}^{\text{model}} \cap \mathcal{P}^{\text{hist}}|}{|\mathcal{P}^{\text{model}}|}, \quad R_{\text{pair}} = \frac{|\mathcal{P}^{\text{model}} \cap \mathcal{P}^{\text{hist}}|}{|\mathcal{P}^{\text{hist}}|}, \quad (\text{B.7})$$

where $\mathcal{P}^{\text{model}}$ and $\mathcal{P}^{\text{hist}}$ are the model-produced and historical conflict-pair sets. The pair-level F1 score is

$$F1_{\text{pair}} = \begin{cases} \frac{2P_{\text{pair}}R_{\text{pair}}}{P_{\text{pair}} + R_{\text{pair}}}, & P_{\text{pair}} + R_{\text{pair}} > 0, \\ 0, & P_{\text{pair}} + R_{\text{pair}} = 0. \end{cases} \quad (\text{B.8})$$

These diagnostics are interpreted together with prediction-source shares. A prediction produced by a fallback or deterministic execution rule can preserve executability, but it does not provide the same evidence of learned behavior as an empirical-stage or CART-stage prediction.

B.5. Path-Choice Component Validation

The path-choice component is evaluated in two ways. The exact observed path view measures graph-geometry recovery through directed-edge overlap. The canonical path-family view measures whether the selected path candidate matches the observed canonical family. These views are complementary: a predicted path can be geometrically close even when the exact canonical label differs, and a correct family label can still contain local geometric deviations.

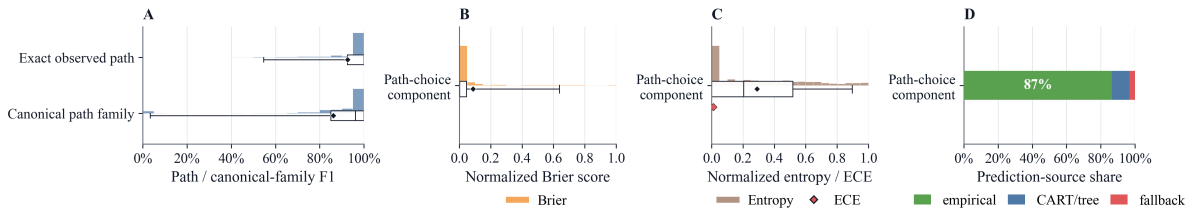


Figure B.4: Path-choice validation diagnostics

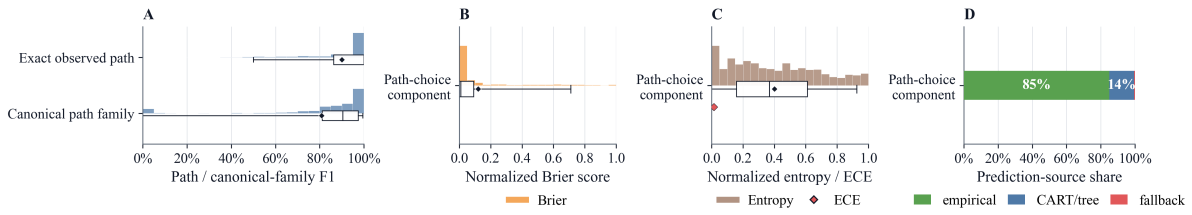


Figure B.5: Non-trivial path-choice diagnostics

Across all 132,760 held-out test cases, the mean edge-weighted exact observed path F1 is 0.926 and the mean canonical path-family weighted F1 is 0.861. Prediction-source shares are 86.53% empirical, 10.17% CART, and 3.30% fallback. These values show that most path-choice outputs are supported by observed empirical context or CART generalization.

Figure B.5 restricts the evaluation to OD pairs with more than one relevant path candidate. In this subset, the exact observed path F1 decreases to 0.900 and the canonical path-family weighted F1 decreases to 0.808. This confirms that part of the all-case performance comes from structurally simple OD pairs, while genuine path-choice cases remain more difficult. The non-trivial subset is therefore the more informative diagnostic for operational path-choice behavior.

B.6. Endpoint-Selection Component Validation

Endpoint selection is evaluated separately for the inbound runway stopbar, inbound apron boundary, outbound apron boundary, and outbound runway stopbar targets. The separation is important because these targets have different operational mechanisms. Runway-side targets are affected by aircraft state, runway use, departure procedures, and runway-side availability, while apron-boundary targets are more strongly linked to apron and gate geography.

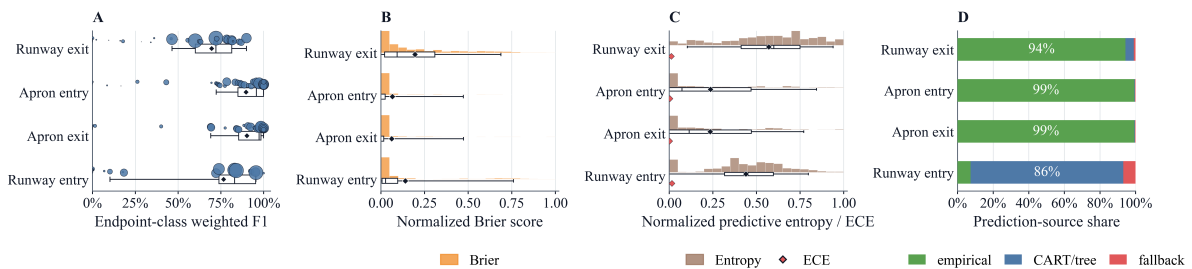


Figure B.6: Endpoint-selection validation diagnostics

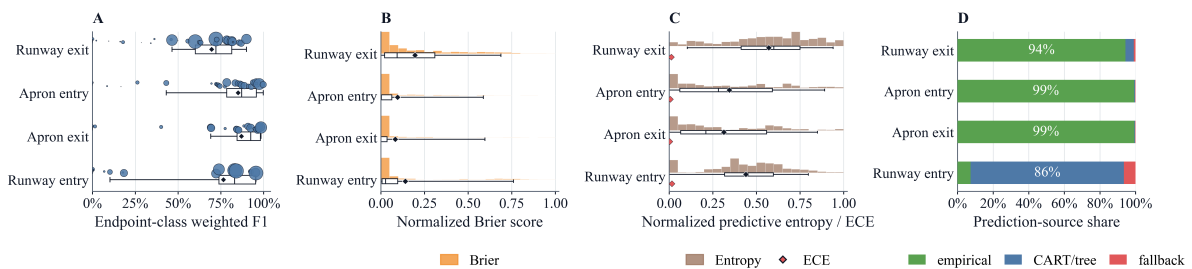


Figure B.7: Non-trivial endpoint-selection diagnostics

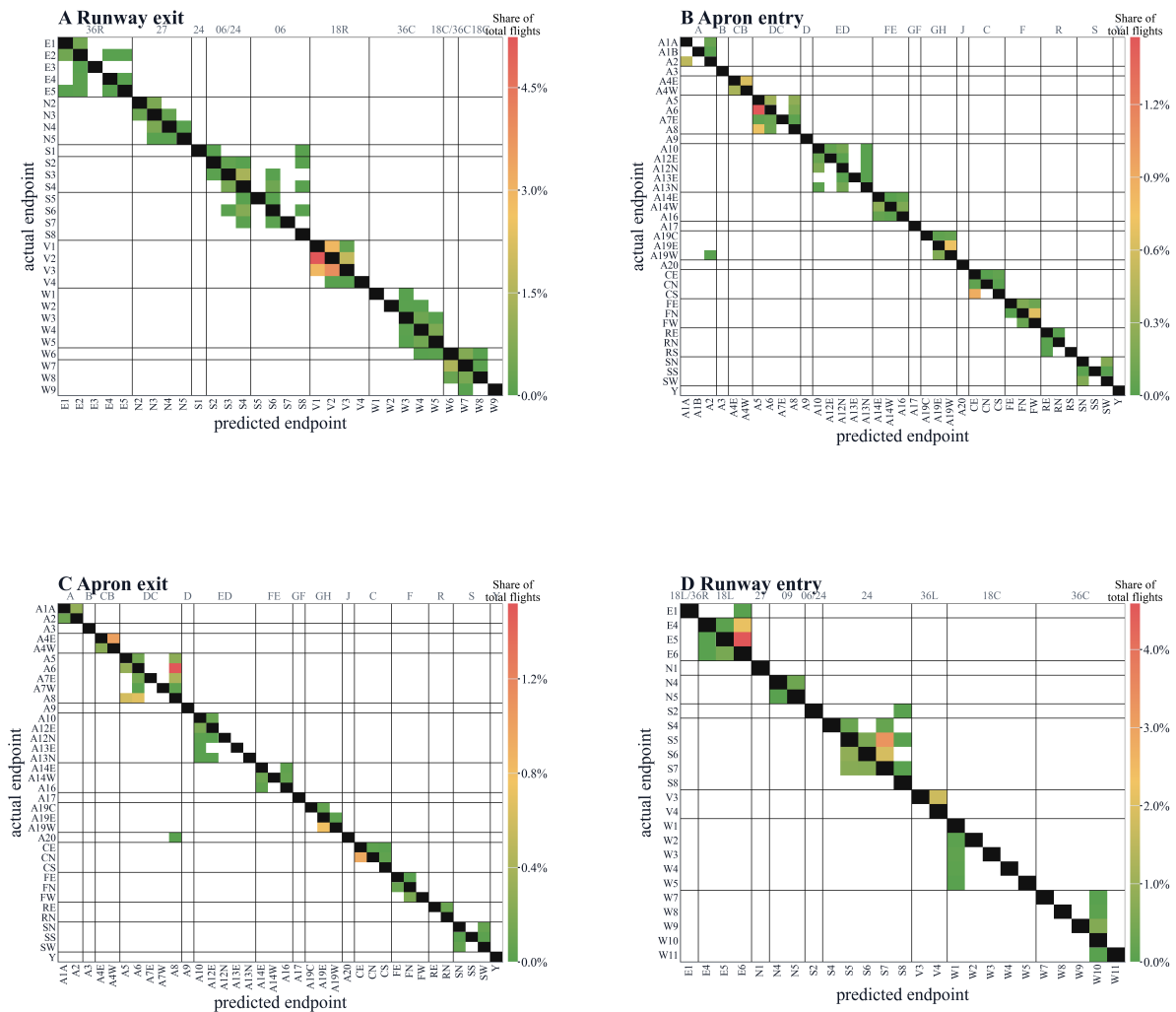


Figure B.8: Endpoint-selection confusion matrices

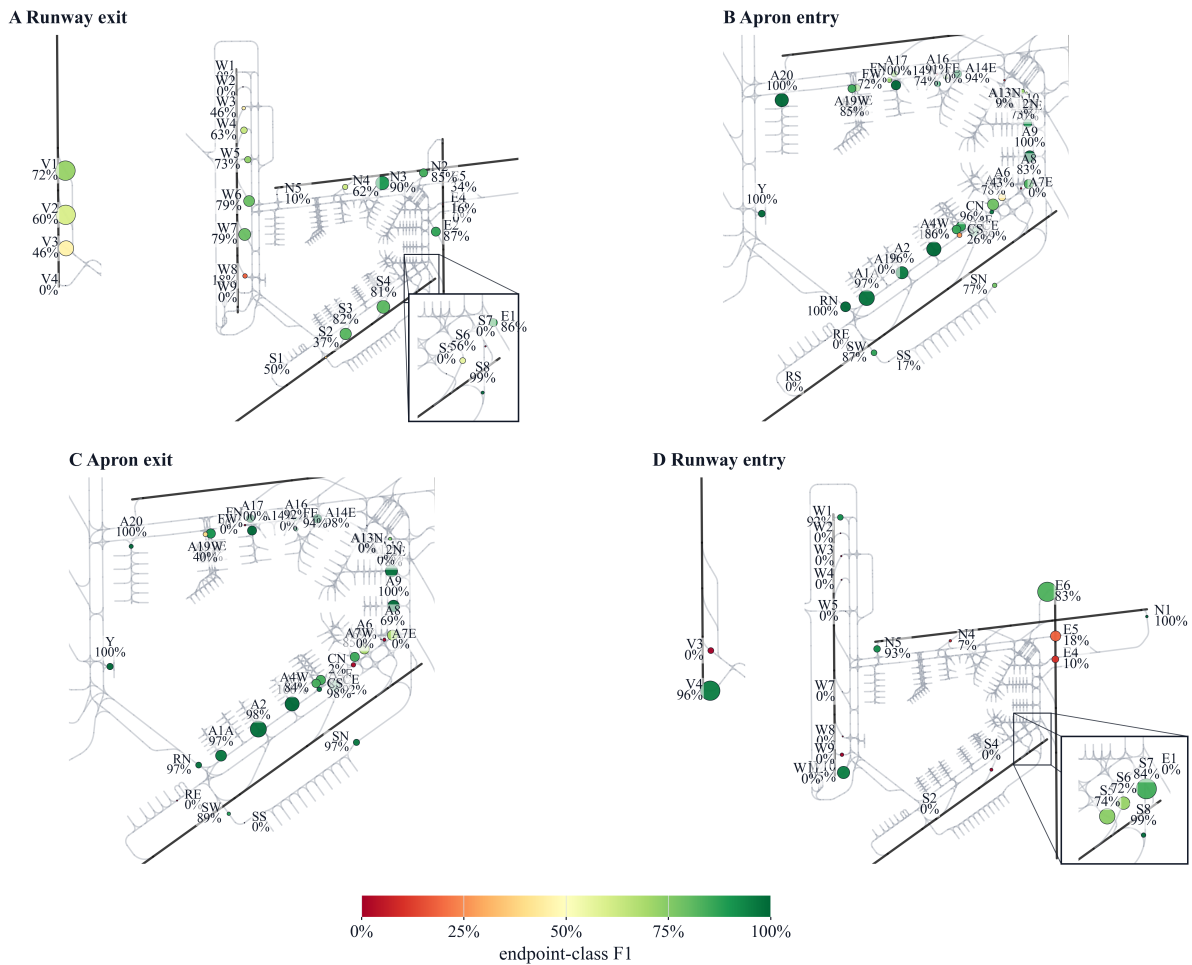


Figure B.9: Spatial endpoint-selection diagnostics

The held-out weighted F1 values are 0.696 for inbound runway-stopbar selection, 0.898 for inbound apron-boundary selection, 0.902 for outbound apron-boundary selection, and 0.766 for outbound runway-stopbar selection. The corresponding fallback shares are 1.01%, 0.32%, 0.30%, and 6.90%, with no unsupported cases.

The diagnostic pattern is consistent with the operational structure of the targets. Apron-boundary targets are easier because they are strongly tied to gate and apron area. Runway-stopbar targets are more variable. Inbound runway-stopbar selection reflects aircraft-side landing and exit behavior, while outbound runway-stopbar selection reflects departure-side procedures, runway configuration, and local availability. The spatial diagnostics show where endpoint classes have weaker recovery and help distinguish rare-class difficulty from broader component failure.

B.7. Speed-Profile Component Validation

The speed-profile component estimates residual speed profiles around edge-level baseline speed profiles. Its validation therefore focuses on regression error, prediction-interval quality, prediction-source shares, and the spatial distribution of speed error. Unlike the discrete learned components, the speed-profile component does not return a class-probability distribution over alternatives.

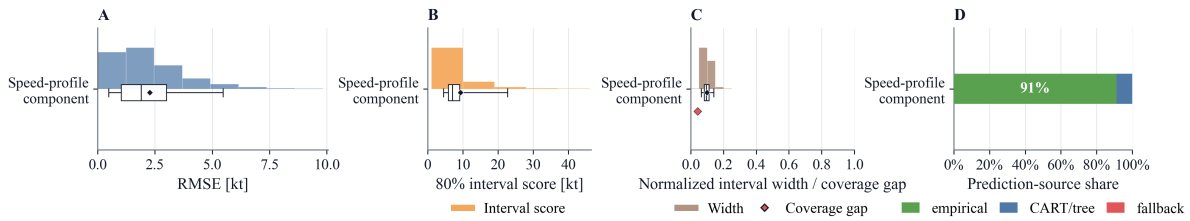


Figure B.10: Speed-profile validation diagnostics

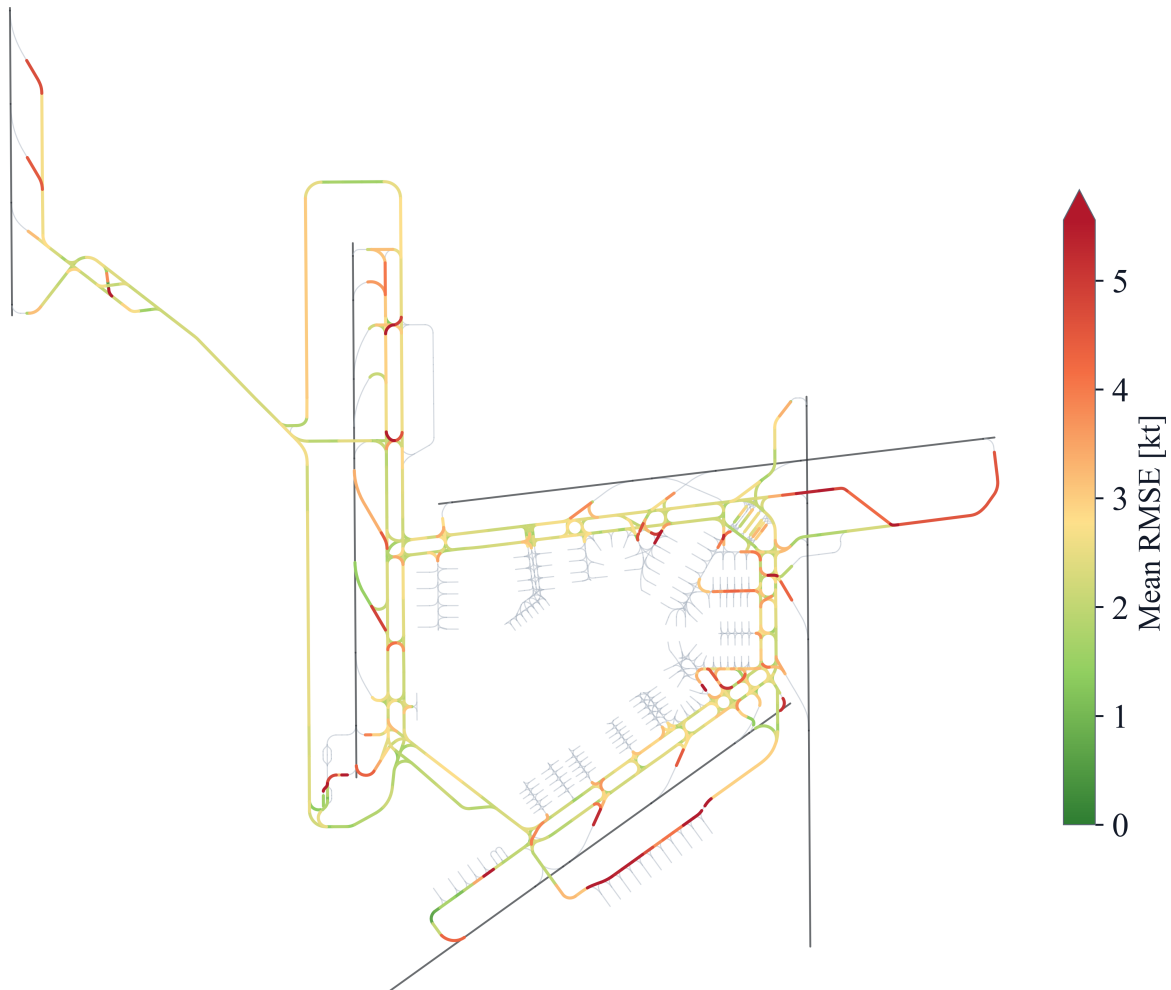


Figure B.11: Spatial speed-profile error diagnostics

The final speed-profile component is evaluated on 4,138,913 edge-speed profiles. The mean RMSE is 2.279 kt, the mean 90th-percentile absolute error is 2.962 kt, and the mean signed bias is 0.089 kt. The central 80% interval coverage is 84.21%, giving a 4.21 percentage-point coverage gap. Prediction-source shares are 91.10% empirical, 8.87% CART, 0.033% fallback, and 0.0002% unsupported.

These values show that the edge-level baseline plus residual structure is effective for aggregate speed behavior. The slight over-coverage of the central 80% interval indicates conservative uncertainty estimates, which is preferable to under-coverage for a support-aware baseline. The spatial RMSE diagnostic localizes the remaining error and shows where geometry, local traffic conditions, or edge-specific movement behavior require further refinement.

B.8. Local Conflict-Resolution Component Validation

The local conflict-resolution component is evaluated on labels inferred from observed trajectories and free-motion counterfactuals. These labels are movement-derived; they are not ATC instruction logs. The priority label records which aircraft passed the protected taxi resource first, while the resolution-action label for aircraft i uses the classes PROCEED, SLOW_DOWN, and HOLD_SHORT to describe the inferred movement response.

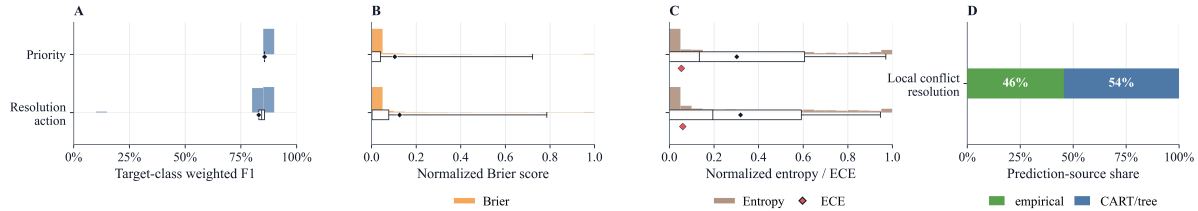


Figure B.12: Local conflict-resolution validation diagnostics

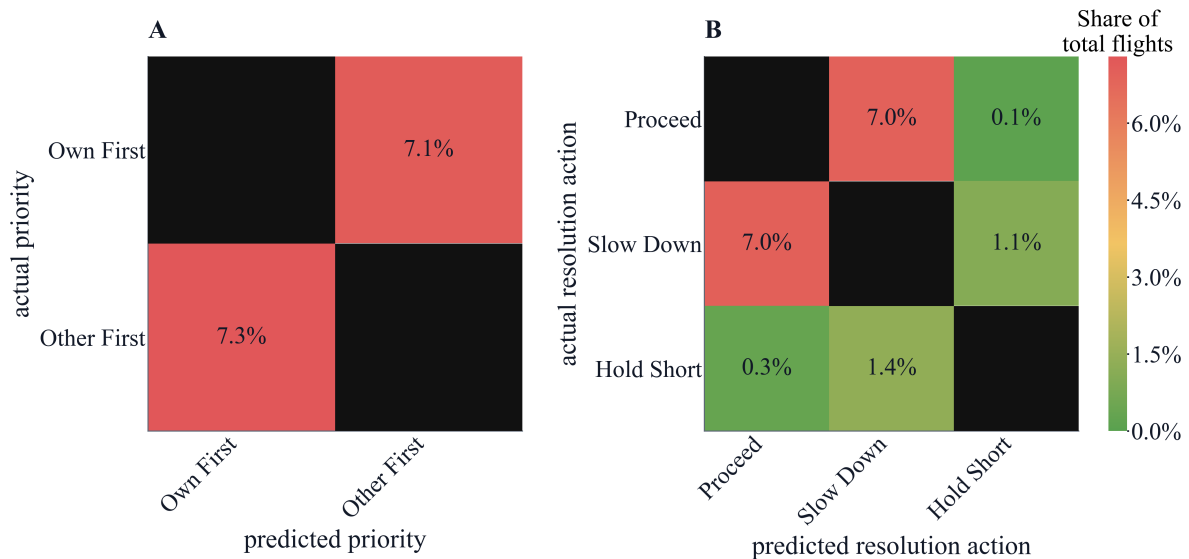


Figure B.13: Local conflict-resolution confusion matrices

The locked test set contains 125,832 conflict-event observations. Weighted F1 is 0.856 for priority labels and 0.830 for movement-derived resolution-action labels. Mean normalized Brier scores are 0.104 and 0.126, while ECE values are 0.054 and 0.060. Prediction-source shares are 45.62% empirical and 54.38% CART, with no fallback or unsupported cases in held-out component validation.

The component recovers the main priority and resolution-action patterns, but the confusion matrices show that action-level distinctions are harder than priority recovery. Proceed and slow-down cases can be difficult to separate from trajectory data alone, especially when the observed movement response is gradual. Hold-short behavior is less frequent and therefore remains more sensitive to support and labeling uncertainty. These results should be interpreted as validation of movement-derived interaction labels, not as validation of recovered controller intent.

B.9. Integrated Model-Validation Scenario Diagnostics

Integrated model validation combines the selected learned behavioral components with deterministic execution rules. This validation level checks whether component outputs remain coherent when path choice, speed-profile behavior, local conflict resolution, protected-resource monitoring, following-and-merge constraints, and runway-departure release are executed together.

The representative historical-scenario runs use historical endpoints. This isolates timing, path, speed, conflict-event, following, and runway-release behavior from endpoint-selection errors, which are evaluated separately at component level. The integrated diagnostics therefore evaluate component interoperability and emergent behavior, not isolated component accuracy.

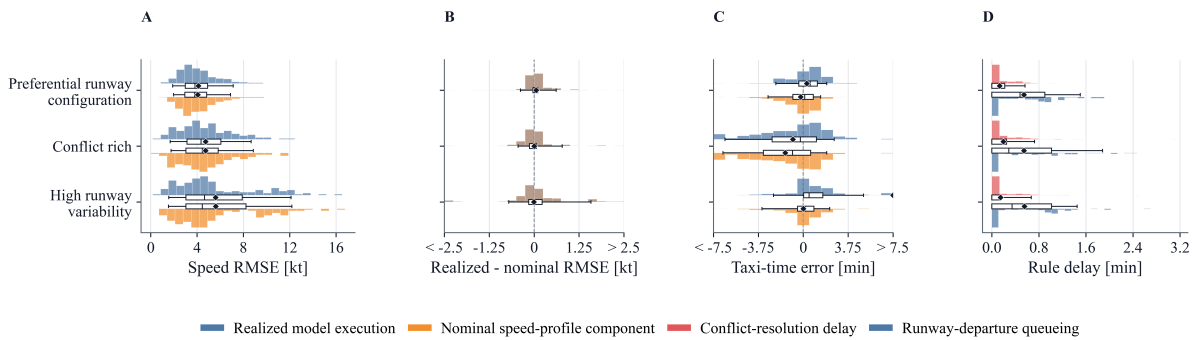


Figure B.14: Integrated motion diagnostics

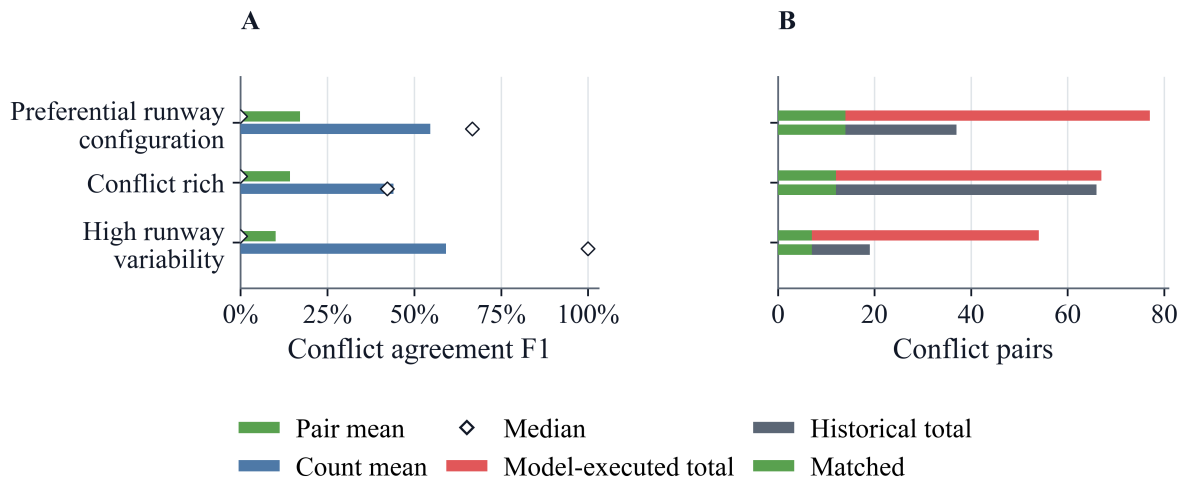


Figure B.15: Integrated conflict-emergence diagnostics

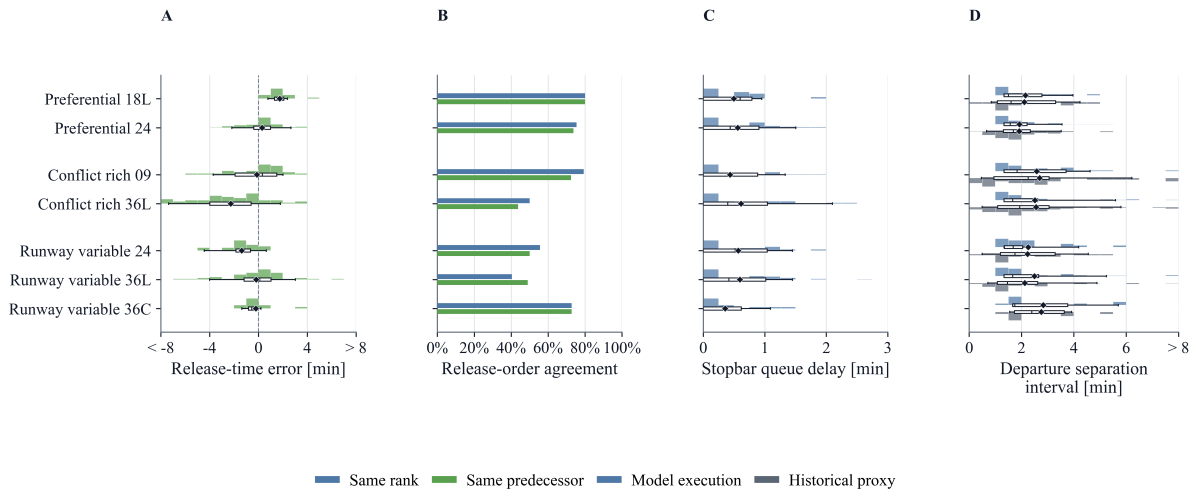


Figure B.16: Runway-departure queueing diagnostics

Table B.1: Integrated scenario diagnostics

Historical validation scenario	Movements	Speed RMSE, kt	Taxi-time error, min	Conflict and runway-departure release diagnostics
Preferential runway configuration	192	Realized 4.12; nominal 4.06	Realized +0.30; nominal -0.21	Conflict-pair F1 24.6%; 76 compared runway releases; mean absolute release-time error 1.26 min.
Conflict-rich	142	Realized 4.74; nominal 4.74	Realized -0.87; nominal -1.50	Conflict-pair F1 18.0%; 77 compared runway releases; mean absolute release-time error 2.45 min.
High runway variability	165	Realized 5.61; nominal 5.62	Realized +7.67; nominal +0.01	Conflict-pair F1 19.2%; 76 compared runway releases; mean absolute release-time error 1.61 min.

Table B.1 shows that the selected component outputs can be combined into complete graph-based movement plans for all three scenarios. Pointwise speed-profile RMSE changes little between nominal and realized model execution, but taxi-time errors can change substantially after deterministic execution rules are applied. The high-runway-variability scenario is the main outlier: realized mean taxi-time error increases to +7.67 min even though nominal timing is close to the processed historical reference.

Table B.2: Runway-departure release diagnostics

Scenario	Runway	Releases	Release-time MAE, min	Same-rank share	Queueing and throughput notes
Preferential runway configuration	18L	15	1.74	0.80	Median queue delay 0.60 min; 15-min throughput MAE 0.44.
Preferential runway configuration	24	61	1.15	0.75	Median queue delay 0.44 min; 15-min throughput MAE 0.67.
Conflict-rich	09	29	1.62	0.79	Median queue delay 0.00 min; 15-min throughput MAE 0.22.
Conflict-rich	36L	48	2.95	0.50	Median queue delay 0.40 min; 15-min throughput MAE 0.89.
High runway variability	24	18	1.60	0.56	Median queue delay 0.52 min; 15-min throughput MAE 0.22.
High runway variability	36L	47	1.77	0.40	Median queue delay 0.42 min; 15-min throughput MAE 1.22.
High runway variability	36C	11	0.95	0.73	Median queue delay 0.00 min; 15-min throughput MAE 0.00.

Table B.2 shows that runway-departure release behavior varies by scenario and runway. The conflict-rich scenario has the largest release-time error on runway 36L, while the high-runway-variability scenario shows lower same-rank agreement for the main departure runways. These differences help explain why integrated timing errors cannot be inferred from speed-profile RMSE alone.

Conflict-pair agreement remains limited in all three scenarios. This does not directly measure local conflict-resolution component accuracy. Instead, it measures whether the integrated execution produces the same protected-resource encounters after path choice, speed-profile behavior, following-and-merge constraints, and runway-departure release have all affected the movement sequence. Small timing changes can therefore change the detected conflict pairs even when component-level labels are accurate.

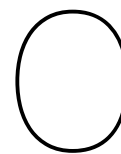
B.10. Verification and Validation Takeaways

The first validation takeaway is that the reconstructed taxi dataset is suitable for graph-based behavior identification within the modeled taxiing scope. Map-matching errors are generally low, fallback reconstruction is rare, and high-error cases are localized. This supports the use of graph-aligned taxi trajectories as the foundation for path catalogs, speed-profile observations, and local conflict-event extraction.

The second takeaway is that canonical path-family grouping provides a necessary sparsity reduction for path choice. The path catalog preserves the main OD-specific path candidates while reducing the fragmentation caused by exact observed path variation. This makes the path-choice component learnable without discarding the underlying graph-sequence structure.

The third takeaway is that component-level validation must be interpreted together with prediction-source diagnostics. Path choice, endpoint selection, speed-profile behavior, and local conflict resolution each show useful held-out performance, but the strength of the evidence depends on whether outputs come from empirical support, CART generalization, fallback, or unsupported states.

The fourth takeaway is that integrated model validation reveals coupling effects that are invisible in isolated component metrics. Deterministic execution rules have little effect on pointwise speed-profile error, but they can strongly affect timing, protected-resource encounters, and runway-departure queueing. These diagnostics identify where the integrated model is coherent and where future refinement is needed.



Operational Interpretation of Selected Model Configurations

C.1. Purpose and Relation to the Scientific Paper

The scientific paper reports the final selected component configurations, the selected empirical and CART context features, and the main component-level and integrated model-validation results. Appendix A (Model Calibration and Tuning) defines the available context-feature pool, describes the context-feature coverage of the graph-aligned taxi dataset, and documents the calibration evidence behind the selected configurations. Appendix B (Verification and Validation) provides the held-out component-level and integrated model-validation diagnostics. This appendix starts after those steps: it interprets what the final selected features and prediction-source diagnostics reveal about the operational structure of the selected learned behavioral components.

The purpose is therefore not to describe all available context features, repeat the dataset profile, or re-report the validation metrics. Instead, this appendix asks whether the final selected features are operationally plausible, how empirical-stage and CART-stage feature sets differ by component, where prediction-source support changes across components, and which operational limitations are suggested by context-conditioned validation diagnostics. The analysis supports the scientific paper by connecting selected feature sets and support diagnostics to operational reasoning.

The interpretation is descriptive rather than causal. A context group with lower performance, higher error, stronger CART dependence, or higher fallback share identifies where the selected model configuration has weaker support or where behavior is harder to distinguish from the available data. It does not prove that the context variable itself causes the error. The goal is to localize model strengths, weaknesses, and future context needs in operational terms.

C.2. Selected Context Features by Component

The scientific paper reports the selected empirical and CART feature sets in the model-training results. This section repeats those selected feature sets only as a reference point for the operational-context interpretation in this appendix. It does not re-train, re-tune, or re-select any component. The feature definitions and available feature pool are documented in Appendix A; the focus here is on the final selected configurations.

The selected features differ by learned behavioral component because each component represents a different taxi-operation behavior. The path-choice component estimates a distribution over path candidates. Endpoint selection estimates runway-stopbar and apron-boundary targets. The speed-profile component estimates residual speed behavior around an edge-level baseline. Local conflict resolution estimates movement-derived priority and resolution-action labels for pairwise protected-resource encounters.

The distinction between empirical-stage and CART-stage features is important. Empirical-stage feature sets are compact because they rely on direct historical support in matching context cells. CART-stage feature sets can include richer context because the tree partitions related observations into accepted

leaves. The selected feature sets therefore show both where direct empirical support was sufficient and where context-conditioned generalization was needed.

Table C.1: Selected empirical and CART context features

Component	Target	Empirical-stage selected context	CART-stage selected context
Path choice	All OD pairs	Runway configuration	Runway configuration; scheduled gate area; 30-min direction-demand count; 30-min local-density count; taxi start minute of day; minutes since runway-configuration change; minutes until runway-configuration change
Endpoint selection	Inbound runway stopbar	Exit speed +1000 m; aircraft class; local density; exit speed +500 m	Exit speed +1000 m; aircraft class; local density; exit speed +500 m; exit speed at threshold; time of day; exit speed +250 m; airline
Endpoint selection	Apron boundary	Aircraft class; gate area; local density	Aircraft class; gate area; local density; runway role; time of day; weekday; BZO phase; same-apron demand; airline
Endpoint selection	Outbound runway stopbar	Aircraft class; SID; BZO phase; time of day; local density; weekday; runway configuration; airline	Aircraft class; SID; BZO phase; time of day; local density; weekday; runway configuration
Speed-profile component	Edge-relative speed profile	Previous-edge speed; aircraft class; apron	Previous-edge speed; airline; aircraft class
Local conflict resolution	Priority and resolution-action labels	Relative ETA bin; own direction; resource group; other direction	Relative ETA; own distance to resource; other distance to resource; own direction

Table C.1 mirrors the selected feature sets reported in the scientific paper. The apron-boundary row applies to the apron-boundary endpoint targets, which share the same selected feature structure. The table also shows that the empirical and CART stages do not serve the same role. The empirical stage uses compact context to preserve direct observed support, while the CART stage introduces additional context where more detailed generalization is useful.

The main pattern is that the CART-stage feature sets are equal to or richer than the empirical-stage feature sets for most components. Path choice adds gate-area, demand, density, timing, and runway-transition features in the CART stage. Endpoint-selection targets use different feature families depending on whether the target is runway-side or apron-side. The speed-profile component keeps both stages compact because the edge-level baseline already captures much of the typical speed behavior. Local conflict resolution uses categorical timing and direction features empirically, while the CART stage uses continuous timing and distance-to-resource features for more detailed encounter-state separation.

The following section interprets these selected features by learned behavioral component. The aim is to assess whether the final selected configurations are operationally plausible and what they reveal about the modeled taxi behaviors, not to repeat the calibration procedure from the scientific paper.

C.3. Operational Interpretation by Learned Behavioral Component

This section interprets the selected context features from an operational perspective. The selected features should not be read as proof of causal mechanisms. They are validation-supported signals that repeatedly helped separate observed taxi behavior under the available data and model structure. Their

value is that they show which operational conditions contain information about each learned behavioral component.

C.3.1. Path-choice Component

The path-choice component uses runway configuration as the empirical-stage context and adds gate area, demand, density, taxi start time, and runway-configuration transition timing in the CART stage. This structure suggests that a large share of observed path choice is organized around the active runway system. Once the OD pair is fixed, runway configuration captures much of the repeated runway-side structure of the operation and provides enough support for many common path candidates.

The richer CART-stage context identifies where path choice becomes more context-sensitive. Scheduled gate area represents apron-side access and the likely approach to or from the terminal area. The 30-minute direction-demand count and local-density count represent surface-traffic pressure. Taxi start time captures recurring daily operating patterns, while the time since and until runway-configuration changes captures transitional periods in which path use may shift before or after the active runway setup changes.

The operational trend is therefore twofold. Under common operating conditions, path choice follows repeated canonical path families that are mainly structured by OD pair and runway configuration. Under non-trivial or sparse OD conditions, additional context is needed to explain deviations from those repeated patterns. This supports the use of canonical path families as the main path candidates while also showing that path choice is not purely geometric: it is affected by apron access, traffic pressure, and runway-transition state.

C.3.2. Endpoint-selection Component

The endpoint-selection component shows the clearest difference between target types. Apron-boundary targets use aircraft class, gate area, and local density in the empirical stage, with additional runway role, time of day, weekday, BZO phase, same-apron demand, and airline information in the CART stage. This indicates that apron-boundary selection is primarily an apron-side and schedule-linked problem. Gate area and same-apron demand describe where traffic is expected to enter or leave the apron side of the modeled taxiing scope, while time, weekday, and operating-state variables capture repeated patterns in how apron access is used.

Inbound runway-stopbar selection has a different structure. The selected features include runway-exit speed at fixed downstream distances, aircraft class, and local density. These variables describe the aircraft's early post-landing motion and the surrounding traffic state. Their selection suggests that inbound runway-side endpoint inference is linked to aircraft rollout and deceleration behavior, not only to runway configuration or schedule context. The CART stage adds threshold and intermediate exit-speed features, time of day, and airline, which indicates that runway-exit behavior may vary with aircraft performance, operator/fleet patterns, and time-dependent operating conditions.

Outbound runway-stopbar selection is more procedure- and configuration-dependent. The selected features include aircraft class, Standard Instrument Departure (SID), BZO phase, time of day, local density, weekday, runway configuration, and airline. This is consistent with departure-side operations, where runway entry depends on active runway use, departure procedure, traffic state, and operating phase. The fact that the empirical feature set is already detailed suggests that repeated outbound endpoint behavior requires more specific context than broad runway or apron grouping alone.

The operational insight is that endpoint selection is not one homogeneous classification problem. Apron-boundary selection is mainly spatial and apron-side. Inbound runway-stopbar selection reflects aircraft rollout and runway-exit behavior. Outbound runway-stopbar selection reflects departure procedure, runway configuration, and operating state. This explains why runway-stopbar targets are harder to recover than apron-boundary targets and why missing operational constraints can appear as endpoint-selection errors.

C.3.3. Speed-profile Component

The speed-profile component estimates residual speed behavior around an edge-level baseline. The selected features should therefore be interpreted as explaining deviations from typical edge speed, not the full absolute speed pattern. The edge-level baseline already captures much of the recurring speed behavior associated with each directed edge.

Previous-edge speed is selected in both the empirical and CART stages. This is the strongest operational signal in the speed-profile feature set. It indicates that taxi speed is locally continuous: the speed on a directed edge depends strongly on how the aircraft approached that edge. If the aircraft arrives from a slow, recently constrained, accelerating, or high-speed state, the residual speed profile on the next edge changes accordingly.

Aircraft class is also selected in both stages, which indicates that residual speed behavior varies with aircraft group. This is operationally plausible because aircraft size, acceleration behavior, braking behavior, and typical taxiing practices differ between aircraft classes. Apron context appears in the empirical stage, while airline appears in the CART stage. These features suggest that local geography and operator or fleet composition explain part of the remaining speed variation after the edge-level baseline is removed.

The operational trend is that speed behavior is more local than global. Broad airport-wide variables such as runway configuration or overall demand are not the dominant selected signals for the final speed-profile component. Instead, the selected features describe motion continuity, aircraft type, local apron context, and operator-related patterns. This supports the baseline-plus-residual formulation: the graph edge defines the typical speed pattern, and the selected context explains how an aircraft deviates from that pattern.

C.3.4. Local Conflict-resolution Component

The local conflict-resolution component uses features that describe the pairwise encounter state before a movement-derived label is assigned. The empirical stage uses relative ETA bin, own direction, other direction, and resource group. The CART stage uses continuous relative ETA, own distance to the protected resource, other distance to the protected resource, and own direction. This split is consistent with the different roles of the two stages: the empirical stage uses robust categorical encounter patterns, while the CART stage uses more detailed timing and distance information for generalization.

The selected features show that local conflict resolution is mainly a local timing-and-geometry problem. Relative ETA describes which aircraft would reach the protected taxi resource first under free motion. Own and other distance to the resource describe how close both aircraft are to the interaction point. Movement direction describes the encounter geometry, such as crossing, converging, or opposing approach patterns. Resource group distinguishes the type of protected taxi resource, which affects how priority and resolution-action labels are inferred.

The operational insight is that global traffic demand alone is not sufficient to explain local conflict-resolution labels. The relevant information is pairwise and resource-specific: who is expected to arrive first, how close each aircraft is, which directions they approach from, and what kind of resource they share. This supports the use of protected-resource-based conflict-event construction.

The selected features also reveal a limitation. Near-tie encounters are sensitive to small timing changes. A small change in speed, path progress, or resource-passing time can change the inferred priority or resolution-action label. This is especially important because the labels are movement-derived from trajectory behavior and free-motion counterfactuals, not observed ATC commands. The component can therefore reproduce local interaction patterns visible in the trajectories, but it should not be interpreted as recovering controller intent directly.

C.3.5. Operational Knowledge from the Selected Features

Across components, the selected features reveal a structured view of aircraft taxi operations. Path choice is governed first by runway configuration and OD structure, then refined by apron access, demand, density, and transition timing. Endpoint selection separates into different mechanisms for apron boundaries, inbound runway stopbars, and outbound runway stopbars. Speed behavior is mainly local and residual, depending on previous motion state and aircraft/operator context. Local conflict resolution is driven by pairwise timing and protected-resource geometry.

These patterns support the overall modeling approach. The learned behavioral components are not using arbitrary statistical signals; they rely on context dimensions that correspond to recognizable operational mechanisms. At the same time, the selected features expose missing context. Temporary construction state, runway availability restrictions, local procedures, controller-sector practices, and detailed surface-condition variables are not fully represented. When such factors influence observed behavior, the

model can only approximate them indirectly through the available context features. This helps explain why some errors appear in specific endpoints, OD pairs, operating states, or protected-resource situations.

C.4. Prediction-source Support by Component

Prediction-source diagnostics show how the selected components produce their outputs during held-out evaluation. An empirical-stage output indicates that the queried context matched a sufficiently supported historical context cell. A CART-stage output indicates that exact empirical support was insufficient, but that the component could still generalize from an accepted tree leaf. A fallback output preserves executability when neither learned stage is accepted. An unsupported state indicates that no accepted output is available.

These sources are important for operational interpretation. Two components can have similar predictive performance but different evidential strength. A prediction supported by the empirical or CART stage provides evidence that the behavior is represented in the observed data or in an accepted generalization of it. A fallback output does not provide the same behavioral evidence; it only ensures that model execution can continue.

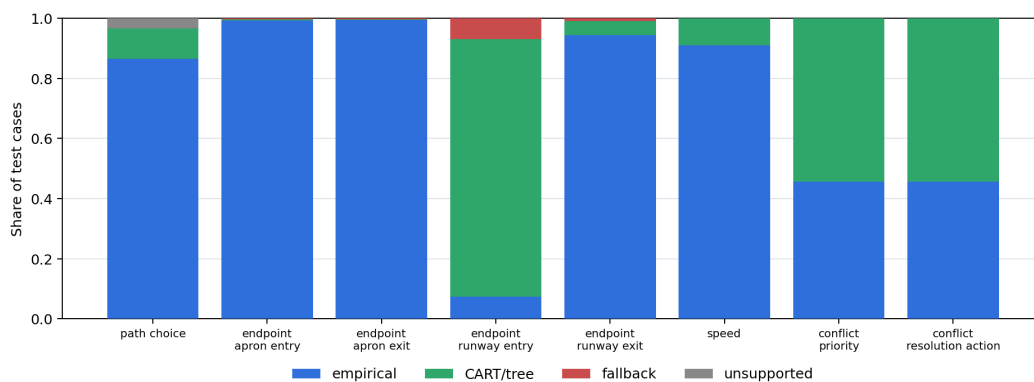


Figure C.1: Prediction-source shares by component

Figure C.1 shows that the selected components rely on prediction sources in different ways. Path choice is dominated by empirical support, with a smaller CART share and limited fallback. This indicates that many OD-specific path-choice situations repeat often enough in the historical data to support direct empirical distributions. Operationally, this is consistent with the presence of repeated canonical path families under common runway configurations and OD pairs.

Endpoint selection shows stronger differences between endpoint targets. Apron-boundary targets are mainly supported by empirical observations, which suggests that apron-side endpoint behavior is relatively stable when gate, apron, aircraft, and density context are known. Outbound runway-stopbar selection is the main exception. It relies more strongly on CART generalization and shows the largest endpoint fallback share. This suggests that runway-entry behavior is less directly repeated in exact empirical context cells and is more sensitive to procedure, runway configuration, operating phase, SID, traffic state, and runway-side availability.

The speed-profile component has broad learned support. This is operationally plausible because graph alignment creates many accepted edge-speed observations, and the edge-level baseline already captures much of the repeated speed behavior. The prediction-source shares therefore indicate that most residual speed estimates are supported by observed edge-level motion patterns rather than by fallback behavior.

Local conflict resolution is divided between empirical and CART support. This reflects the structure of protected-resource encounters. Some encounter patterns repeat in categorical form, such as direction combinations and resource groups, while others require more detailed timing and distance information. The CART share therefore indicates that local conflict-resolution behavior is not only a repeated-class problem; it often requires context-conditioned generalization over pairwise encounter geometry.

The operational trend is that stable, repeated behaviors are mostly empirical, while sparse, procedure-dependent, or timing-sensitive behaviors require CART generalization. Fallback use highlights where the available context and historical support are still insufficient. This supports one of the main conclusions of the scientific paper: prediction-source diagnostics are part of the model result because they show where the observed-behavior baseline is supported by historical data and where it remains dependent on executability rules.

C.5. Context-conditioned Validation Insights

This section interprets where the final selected model configurations are weaker after model selection. The diagnostics are computed on locked validation or test outputs and are not used to reselect features, retune hyperparameters, or change the final components. Their purpose is to identify whether errors and weak support are randomly distributed, or whether they are associated with recognizable operational structures.

The section uses three diagnostic views. OD-pair and OD-group diagnostics localize errors across runway–apron structures. Selected context-feature diagnostics show how performance changes across the final selected feature dimensions. Weekly operational-profile diagnostics group performance by broader operating regimes. Together, these views help explain what the final model configurations reveal about aircraft taxi operations and where missing context may still limit the observed-behavior baseline.

C.5.1. OD-pair and OD-group Error Concentration

OD structure remains useful for validation even when it is not selected as a direct feature for every component. An OD-conditioned view asks whether errors concentrate around specific runway–apron combinations, endpoint contexts, or protected-resource situations. This matters because a component can perform well in aggregate while remaining weak for a small number of operationally important OD structures.

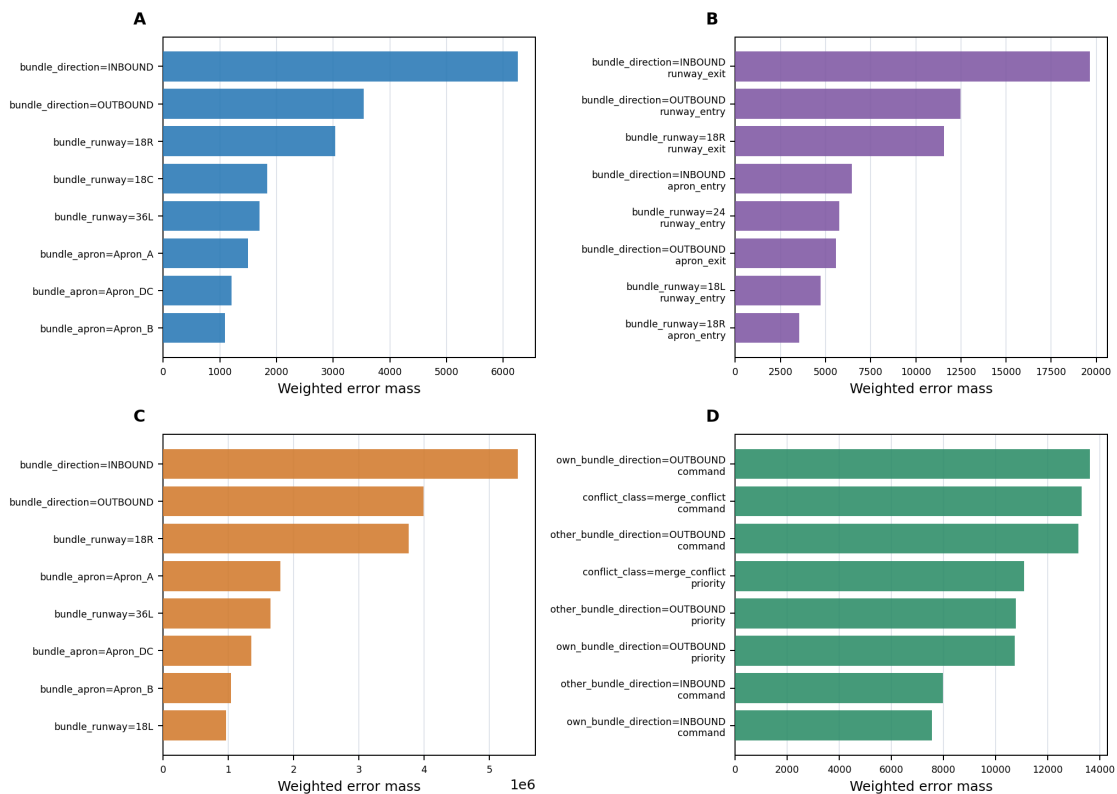


Figure C.2: OD-conditioned error concentration

Figure C.2 shows that error mass is localized rather than evenly distributed across the operation.

Path-choice degradation is concentrated in selected runway and apron groupings. Endpoint-selection degradation is concentrated around specific gate-area and runway-side endpoint contexts. Local conflict-resolution errors also concentrate in particular direction and protected-resource combinations.

The operational implication is that aggregate validation metrics should not be interpreted alone. A high overall F1 score can coexist with weaker behavior for specific runway–apron structures or rare end-point classes. These localized weaknesses can indicate sparse historical support, missing operational rules, temporary restrictions, or local procedures that are not represented explicitly in the selected context features.

C.5.2. Selected Context-feature Validation Patterns

Selected context-feature diagnostics compare grouped validation behavior against the component baseline. They show which final selected context dimensions are associated with lower performance, higher error, weaker learned support, or greater reliance on CART generalization or fallback. These diagnostics do not imply that a feature causes the error; they identify where the final configuration is operationally more difficult.

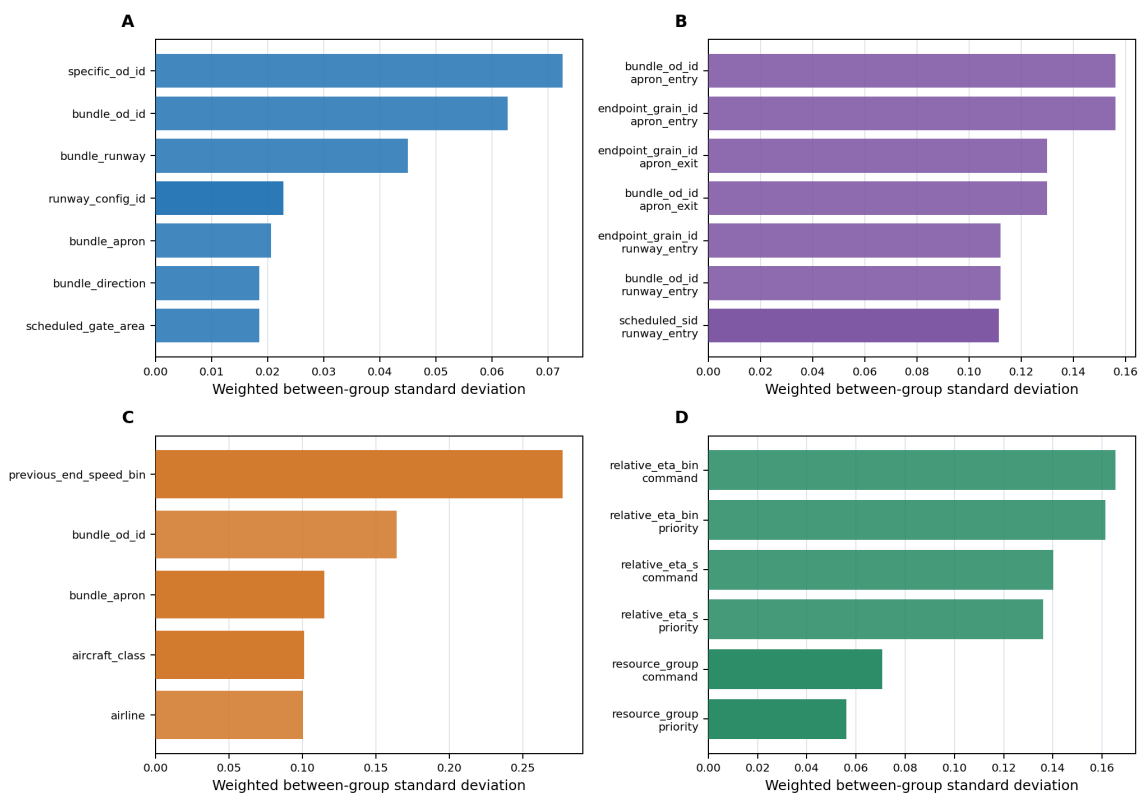


Figure C.3: Selected context-feature validation patterns

Figure C.3 shows four main patterns. Path-choice degradation is associated with selected runway and OD groupings, including periods close to runway-configuration changes. This suggests that path choice is less stable during transition periods or in OD structures with multiple plausible path candidates.

Endpoint-selection degradation appears in apron and runway endpoint contexts where class support is uneven. Some contexts are dominated by one endpoint class, while others contain rare endpoint classes with limited historical support. This supports the interpretation that endpoint selection is partly constrained by operational rules and availability conditions that are only indirectly represented in the available context.

Speed-profile RMSE is most sensitive to movement-state continuity and local geography. Previous-edge speed and apron context identify whether an aircraft enters an edge from a slow, fast, accelerating, or recently constrained state. This confirms that residual speed behavior is local: after the edge-level

baseline is known, the remaining error depends more on motion continuity and local operating context than on broad airport-wide variables.

Local conflict-resolution errors concentrate in near-tie encounters where the relative ETA between aircraft is small. These cases are operationally difficult because small differences in speed, path progress, or resource-passing time can change the inferred priority or movement-derived resolution-action label. Lower performance in these cases does not necessarily indicate a wrong interaction concept; it reflects the ambiguity of deriving local conflict-resolution labels from trajectories without ATC instruction logs.

C.5.3. Weekly Operational-profile Validation Patterns

Weekly operational-profile diagnostics group validation behavior by broader operating regimes. These regimes combine runway use, traffic volume, OD structure, density, timing, and seasonal or weather-related conditions. They therefore provide a complementary view to the selected feature-level diagnostics.

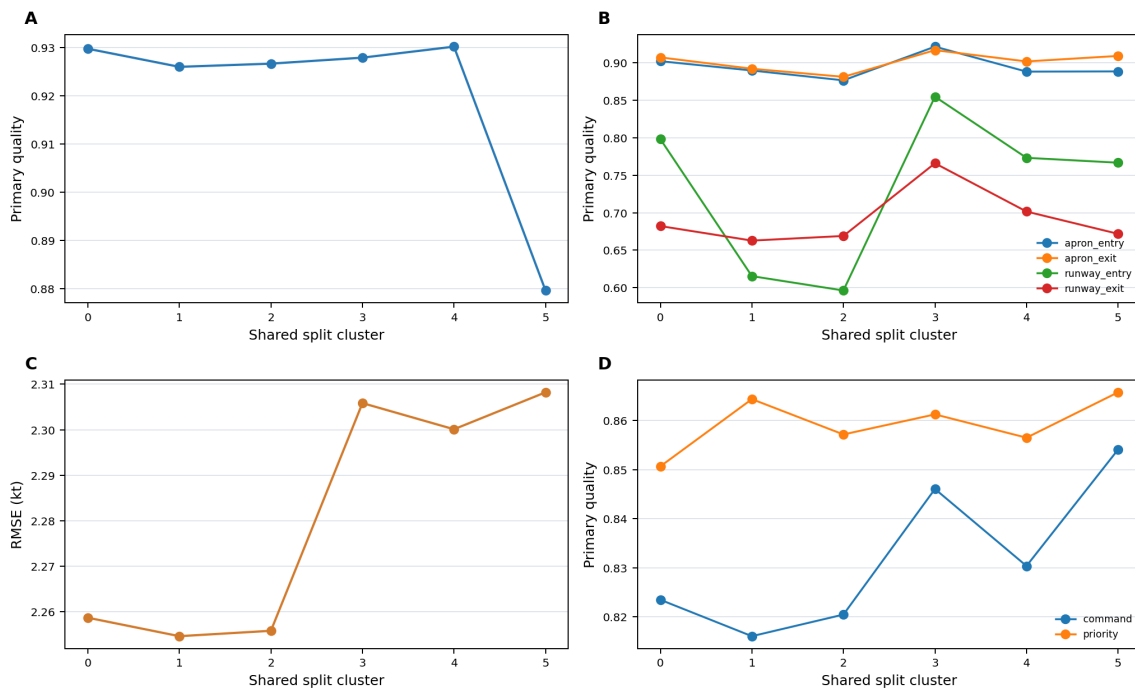


Figure C.4: Weekly operational-profile validation patterns

Figure C.4 shows whether component performance changes across broader operating regimes. Cluster-level differences are expected because each weekly profile combines several sources of operational variation. A weak cluster result may therefore reflect the combined effect of runway configuration, traffic volume, OD mix, local density, weather, seasonal patterns, or temporary operating conditions rather than one selected feature.

The cluster view helps separate two types of model weakness. If performance degrades only for a narrow feature category, the issue may be sparse support or a missing local rule. If performance degrades for an entire weekly operating regime, the issue may be broader: a combination of runway use, demand structure, and operational state that is not fully captured by the selected feature set.

Together, the OD-conditioned, selected-feature, and weekly-profile diagnostics show that the final model configurations are operationally interpretable but not uniformly strong. The selected features explain many repeated taxi-operation patterns, while the remaining error concentrations point to missing context such as temporary restrictions, construction state, runway-side availability rules, local procedures, or more detailed protected-resource definitions.

C.6. Missing Context and Operational Interpretation Limits

The selected context features provide operationally meaningful information, but they do not make the final model configurations context-complete. The context-conditioned validation diagnostics show that some remaining errors are localized in specific endpoints, OD structures, runway states, protected resources, or timing conditions. These patterns suggest that part of the remaining error is caused by missing operational state rather than by the learned components alone.

The first missing-context category is time-dependent layout state. The main model uses a static airport layout graph, while the real operation can change because of construction, temporary closures, surface works, or local availability restrictions. When the graph does not represent the active layout state, the path-choice component may assign probability to path candidates that are geometrically valid in the static graph but operationally unlikely or unavailable during the affected period. This limitation is especially relevant for construction-related path changes.

The second category is explicit operational-rule context. Some endpoint-selection and path-choice errors are likely associated with local runway procedures, runway-side availability rules, temporary restrictions, or procedure-dependent endpoint use. If these rules are not encoded, the model can only approximate them indirectly through historical patterns in runway configuration, SID, time of day, BZO phase, density, or airline/operator features. This can produce plausible aggregate behavior while still leaving localized errors around specific runway stopbars or path candidates.

The third category is refined movement-state context. The speed-profile component already uses previous-edge speed to represent local motion continuity, but speed residuals can also depend on nearby resource occupancy, queue state, recent holds, aircraft interaction history, and edge-specific geometry. These factors are only partly represented by the current context features. Their absence helps explain why residual speed errors can remain localized even when the edge-level baseline and previous-edge speed capture the main speed behavior.

The fourth category is richer protected-resource context. Local conflict resolution uses relative ETA, movement direction, distance to the protected resource, and resource group. These features capture the main pairwise timing and geometry of an encounter, but near-tie cases remain difficult. Additional information about local traffic state, resource geometry, visibility of close non-intersecting edges, and expert-reviewed conflict categories could help distinguish genuinely ambiguous encounters from cases with a clearer movement-derived priority.

Table C.2: Future context directions

Component	Suggested future context direction
Path choice	Add time-dependent layout state, construction state, temporary closure information, and explicit runway-procedure availability where reliable operational records are available.
Endpoint selection	Add sharper runway-entry and runway-exit availability features, especially for rare or procedure-constrained runway-stopbar alternatives.
Speed-profile component	Add refined movement-state and local-geometry descriptors, such as queue state, recent holds, downstream occupancy, and edge-specific constraint indicators.
Local conflict resolution	Add richer protected-resource descriptors, local traffic-state features, and expert-reviewed encounter categories, especially for near-tie ETA cases.

Table C.2 summarizes the context directions suggested by the selected-feature interpretation and validation diagnostics. The common theme is that future improvements should add operational state that is currently only implicit in the trajectory data. Adding this information would make the learned behavioral components easier to interpret and could reduce unnecessary CART dependence or fallback use in sparse and operationally constrained contexts.

These limits also define how the selected model configurations should be interpreted. The final components provide an observed-behavior baseline supported by available trajectory, graph, schedule, and context data. They do not recover every operational rule, controller intention, temporary restriction, or

local procedure. Their value is that they reveal where observed behavior is strongly supported and where additional operational context is needed.

C.7. Operational Interpretation Takeaways

Appendix C interprets the final selected model configurations after calibration and validation. The selected empirical and CART feature sets are operationally coherent: they use runway configuration, OD structure, apron-side context, demand, density, timing, aircraft class, motion continuity, and protected-resource geometry in ways that match the modeled taxi-operation behaviors.

The first takeaway is that path choice is structured by both stable geometry and time-varying operational state. Common OD pairs and runway configurations produce repeated canonical path families, which explains the strong empirical support. Non-trivial path candidates require richer CART-stage context, including gate area, demand, density, time of day, and runway-transition timing. This indicates that path choice is not only a geometric shortest-path problem; it also reflects apron access, traffic pressure, and runway-configuration transitions.

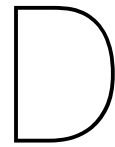
The second takeaway is that endpoint selection contains different operational mechanisms. Apron-boundary selection is mainly linked to gate area, apron geography, aircraft class, and local density. In-bound runway-stopbar selection is connected to aircraft rollout and deceleration behavior through runway-exit speed features. Outbound runway-stopbar selection depends more on SID, runway configuration, BZO phase, time of day, density, and airline/operator context. Endpoint selection should therefore be interpreted as a set of target-specific behaviors rather than as one generic endpoint classifier.

The third takeaway is that speed-profile behavior is mainly local and residual. The edge-level baseline captures much of the typical speed behavior on each directed edge. The selected context features then explain deviations from that baseline, especially through previous-edge speed, aircraft class, apron context, and airline/operator effects. This supports the baseline-plus-residual formulation used by the speed-profile component.

The fourth takeaway is that local conflict resolution is driven by pairwise timing and protected-resource geometry. Relative ETA, distance to resource, movement direction, and resource group describe the encounter state more directly than global demand variables. This confirms that movement-derived priority and resolution-action labels are local interaction labels. Their interpretation remains limited because the labels are inferred from trajectories and free-motion counterfactuals, not from ATC instruction logs.

The fifth takeaway is that prediction-source diagnostics are essential for interpreting the observed-behavior baseline. Empirical outputs indicate repeated historical support, CART outputs indicate accepted context-conditioned generalization, and fallback outputs indicate executable behavior without the same evidential strength. The final configurations are therefore not evaluated only by predictive performance, but also by where their outputs come from.

Overall, the selected context features and support diagnostics show that the final model configurations capture recognizable operational structure in aircraft taxi operations. They also identify where the current data representation remains incomplete. Missing layout state, operational restrictions, local procedures, refined movement-state variables, and richer protected-resource descriptors are the most important directions for improving future versions of the observed-behavior baseline.



Case Study A: Construction-state Effects on Path Choice

D.1. Purpose and Relation to the Scientific Paper

The scientific paper identifies missing operational context as one of the main limitations of the observed-behavior baseline. This case study examines that limitation through a construction-period path-choice diagnostic. The main methodology uses a static airport layout graph and context features such as runway configuration, demand, density, aircraft class, and timing. Temporary construction can violate this representation because it changes which physical taxiway segments are available or likely, while the static graph and selected context features may still treat those paths as valid candidates.

The case study focuses on a known August 2025 construction effect in one of the main taxiways parallel to runway 09/27. The operational question is whether path-choice errors appear when the selected path-choice component can still choose path candidates through a taxiway edge that was operationally avoided during construction. The analysis is diagnostic: it does not introduce a retrained construction-aware component, but tests whether a simple construction-corridor restriction would change selected paths and improve path agreement.

This appendix therefore supports the paper's discussion of static graph limitations and missing context. It also connects to Appendix C, where construction state is identified as a future context direction for path choice and endpoint selection.

D.2. Construction Scenario and Diagnostic Design

The diagnostic uses two levels of evidence. First, August 2024 and August 2025 are compared for two focused runway-direction cases: inbound taxi movements from runway 36R and outbound taxi movements to runway 36L. This comparison shows whether the observed paths changed during the known construction period in August 2025. The August 2025 focused cases are useful for identifying the construction pattern, but they are treated as in-sample diagnostic evidence because they belong to the shared weekly training split used by the selected path-choice component.

Second, a separate held-out week-35 comparison is used to test the same construction-corridor idea on the locked test partition. This held-out diagnostic applies a post-hoc path-candidate restriction: when the selected path uses the construction corridor and a same-OD candidate avoiding that corridor exists, the selected path is replaced by the same-OD non-corridor alternative. This is a transparent counterfactual, not a retrained model. Its purpose is to estimate whether explicit construction-state information could improve path-choice behavior for the affected operational subset.

D.3. Observed Construction-period Path Change

The first step is to inspect whether the historical movement data show a visible construction-period change. Figures D.1 and D.2 compare the focused runway-direction cases in August 2024 and August 2025.

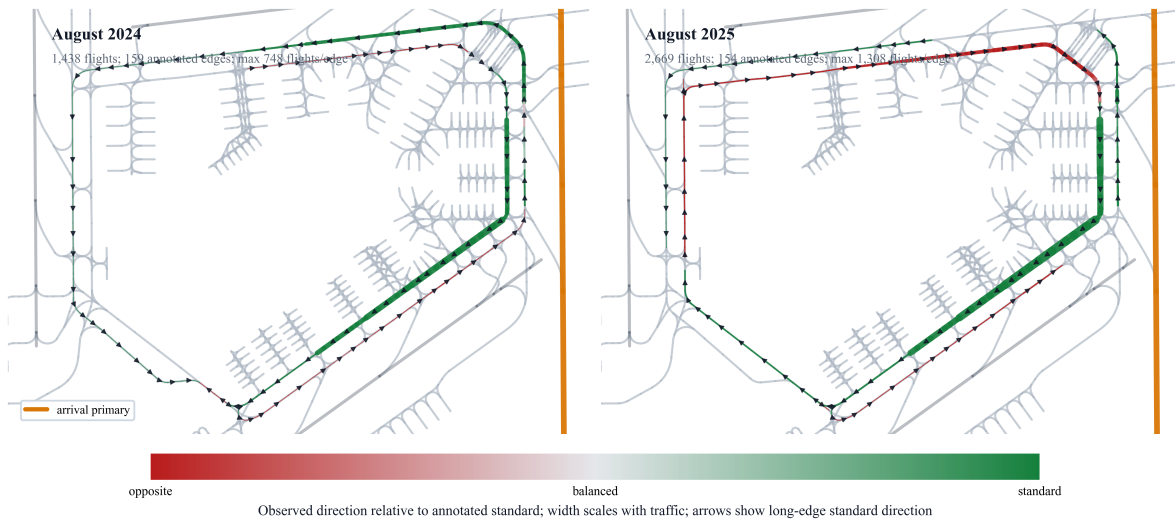


Figure D.1: Inbound 36R construction-period path usage

For inbound 36R movements, the August 2025 pattern shifts away from the affected inner corridor compared with August 2024. This supports the interpretation that the observed operation adapted to a temporary layout or availability change. Because the static path catalog still contains historically observed path candidates through that corridor, a path-choice component without construction state can still assign probability to paths that were no longer used during the construction period.

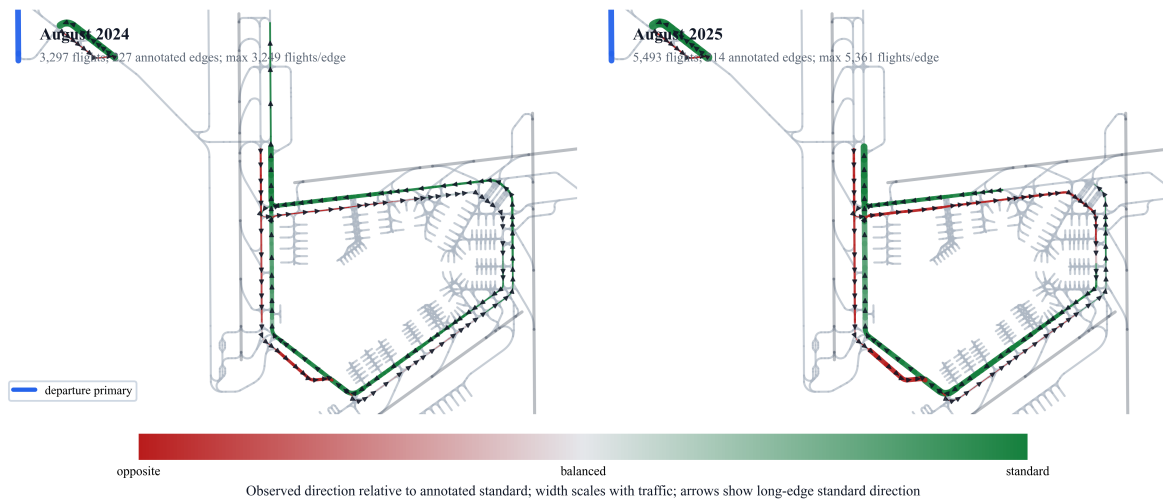


Figure D.2: Outbound 36L construction-period path usage

The outbound 36L comparison shows a related but weaker construction-period change. This difference between inbound and outbound cases is operationally important: the construction effect is not uniform across all movements using the area. It affects a targeted subset of path-choice situations, which is why aggregate path-choice validation metrics can remain strong while specific construction-period path candidates remain problematic.

D.4. August 2025 Path-choice Diagnostic

The selected path-choice component used in the main validation does not include a construction-state feature or a time-varying graph-availability layer. It therefore samples from the same OD-specific path-choice set even during periods when a corridor is operationally avoided. Figure D.3 compares the ob-

served August 2025 paths with the paths selected by the baseline component and with the post-hoc construction-corridor restriction.

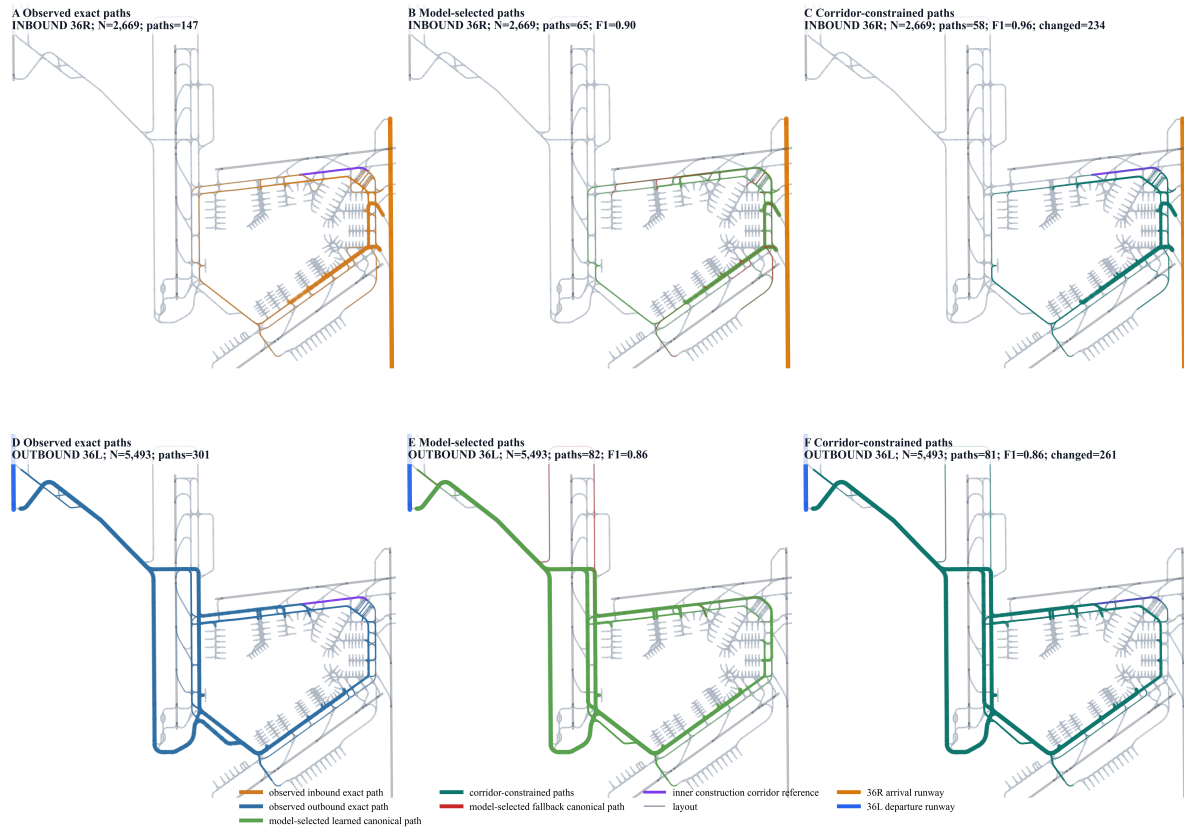


Figure D.3: August 2025 construction-corridor path-choice diagnostic

For the August 2025 focused diagnostic, the baseline selected output uses the inner construction corridor in 507 of 8,162 cases, or 6.21%, even though the observed paths do not use it. Applying the same-OD construction-corridor restriction reduces the inner-corridor share to 0.15%, changes 495 cases, and increases mean length-weighted path F1 from 0.872 to 0.892. The improvement is concentrated in inbound 36R movements, where mean F1 increases from 0.903 to 0.958.

Table D.1: August 2025 construction-corridor summary

Group	Taxi move- ments	Baseline inner- corridor share	inner- corridor share	Constrained inner- corridor share	Changed share	Mean path F1 change
All focused cases	8,162	6.21%		0.15%	6.06%	+0.020
Inbound 36R	2,669	8.80%		0.04%	8.77%	+0.055
Outbound 36L	5,493	4.95%		0.20%	4.75%	+0.004

Table D.1 shows that the construction signal is strongest for inbound 36R. The outbound 36L group still contains some baseline corridor use, but the F1 improvement is much smaller. This confirms that construction state is not a general correction for all path-choice outputs. It is a localized missing-context variable that matters most where the affected corridor was part of a previously common path candidate.

D.5. Held-out Week-35 Counterfactual

The held-out week-35 diagnostic applies the same post-hoc corridor restriction to locked test cases. This test is stricter than the August 2025 diagnostic because it uses held-out data, but it is still not a retrained

construction-aware model. The comparison asks whether a same-OD construction-corridor restriction changes selected paths and improves path agreement when evaluated outside the in-sample August diagnostic.

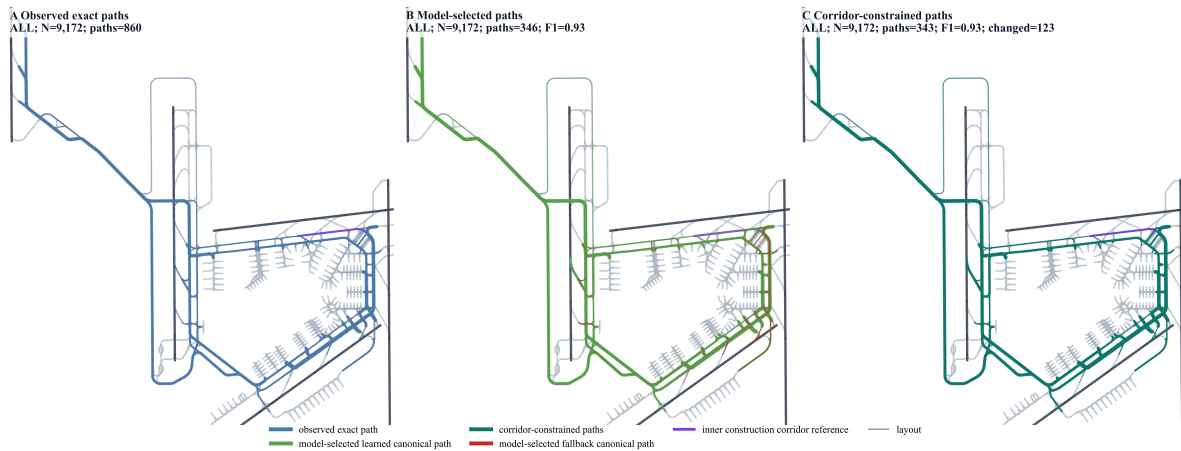


Figure D.4: Held-out week-35 construction-corridor diagnostic

Figure D.4 shows the observed paths, the baseline selected paths, and the corridor-constrained same-OD alternatives for the held-out comparison. The diagnostic reports both an inner-corridor indicator and a broader full-corridor indicator. The inner-corridor indicator isolates the most directly affected construction corridor, while the full-corridor indicator captures a broader set of path candidates using the surrounding corridor structure.

Table D.2: Held-out week-35 construction-corridor summary

Scope/group	Taxi move- ments	Baseline corridor share	Constrained cor- ridor share	Changed share	Mean path F1 change
Inner corridor, all	9,172	1.35%	0.01%	1.34%	+0.004
Inner corridor, out- bound 18L	1,391	8.70%	0.07%	8.63%	+0.025
Full corridor, all	9,172	14.69%	13.24%	1.45%	+0.004
Full corridor, outbound 18L	1,391	95.97%	87.13%	8.84%	+0.023

The held-out aggregate effect is smaller than the August 2025 in-sample effect because relatively few test cases are assigned to the construction corridor by the baseline component. However, the affected subset still shows a clear signal. For outbound 18L movements using the inner-corridor definition, the constraint changes 8.63% of cases and increases mean path F1 by 0.025. Under the broader full-corridor definition, outbound 18L changes 8.84% of cases and increases mean path F1 by 0.023.

The held-out result therefore supports the same interpretation as the August diagnostic, but with a narrower conclusion. Construction-state information is not expected to change the full path-choice population strongly. Its value is concentrated in operational subsets where the static path catalog still contains path candidates through a temporarily avoided corridor.

D.6. Construction-state Takeaways

This case study shows that a static airport layout graph and static path-choice set can be insufficient when temporary construction changes taxiway availability. The selected path-choice component can still

choose historically common path candidates that are geometrically valid in the graph but operationally implausible during the construction period.

The diagnostic also shows why aggregate component performance is not enough to identify this limitation. The construction effect is localized. It is strong for specific runway-direction and corridor combinations, but small in the full held-out path-choice population. This means that high aggregate path-choice performance can coexist with operationally important errors in a small construction-affected subset.

A construction-aware extension could address this limitation in two complementary ways. The first option is a time-varying graph-availability layer that removes unavailable physical taxiway segments from the path-choice set during construction periods. The second option is a construction-state context feature that allows the empirical and CART stages to learn construction-period preferences when enough observations are available. The graph-availability layer is more direct for hard closures, while the context-feature approach is more suitable for softer operational preferences or partial avoidance patterns.

The current case study should be interpreted as diagnostic evidence, not as a final construction-aware model. The August 2025 comparison is descriptive and in-sample. The week-35 comparison is held out, but still uses a post-hoc candidate restriction rather than a retrained model with construction state available during calibration. A final construction-aware experiment would require a predefined construction-state representation, a frozen training setup, and a locked evaluation period selected before the analysis.

Case Study B: Operational Constraints and Availability Rules

E.1. Purpose and Operational Question

The scientific paper identifies missing operational-rule context as one limitation of the observed-behavior baseline. This case study examines that limitation directly. It asks how explicit operational constraints or availability rules can complement trajectory-derived learned behavioral components when the rule describes whether a path candidate or endpoint should be available under a known operating state.

The case study is diagnostic. It does not retrain the selected path-choice or endpoint-selection components, and it does not replace the support-aware empirical, CART, and fallback structure used in the main model. Instead, it applies post-hoc operational constraints to selected held-out cases to test whether known rules can explain localized errors. Two cases are considered: center-runway active-state path procedures for traffic to and from 18R/36L, and a runway-24 S7 endpoint restriction when 18L or 36R is active.

The operational question is therefore not whether rules should replace learned behavior. It is whether known operational constraints should be represented as additional context or availability layers so that the learned components do not need to infer deterministic procedure effects only from trajectory data.

E.2. Center-runway Active-state Path Procedures

For aircraft traveling to or from 18R/36L, the active state of the center runway 18C/36C is associated with different procedure-like path regimes. When the center runway is inactive, movements can use a middle crossing. When 36C is landing or 18C is used for departures, traffic tends toward a north-bypass pattern. When 18C is landing or 36C is used for departures, traffic tends toward a south-bypass pattern.

This case first inspects held-out test trajectories under these center-runway states. It then applies a post-hoc same-OD path-candidate restriction based on manually defined standard corridors. The purpose is to test whether the selected path-choice component produces operationally plausible paths when an explicit center-runway procedure constraint is applied after model selection.

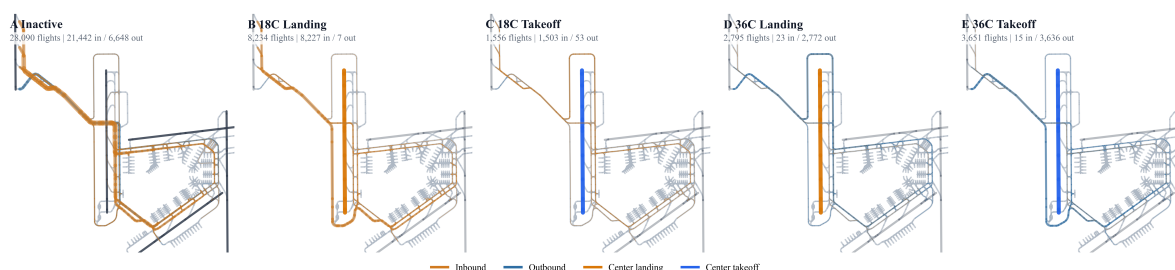


Figure E.1: Center-runway active-state trajectory regimes

Figure E.1 shows the three dominant trajectory regimes. The observed trajectories support the operational distinction between middle crossing, north bypass, and south bypass patterns. This means that the active state of the center runway is not only a background operating condition; it can change which path candidates are operationally plausible for the same broad runway–apron relation.

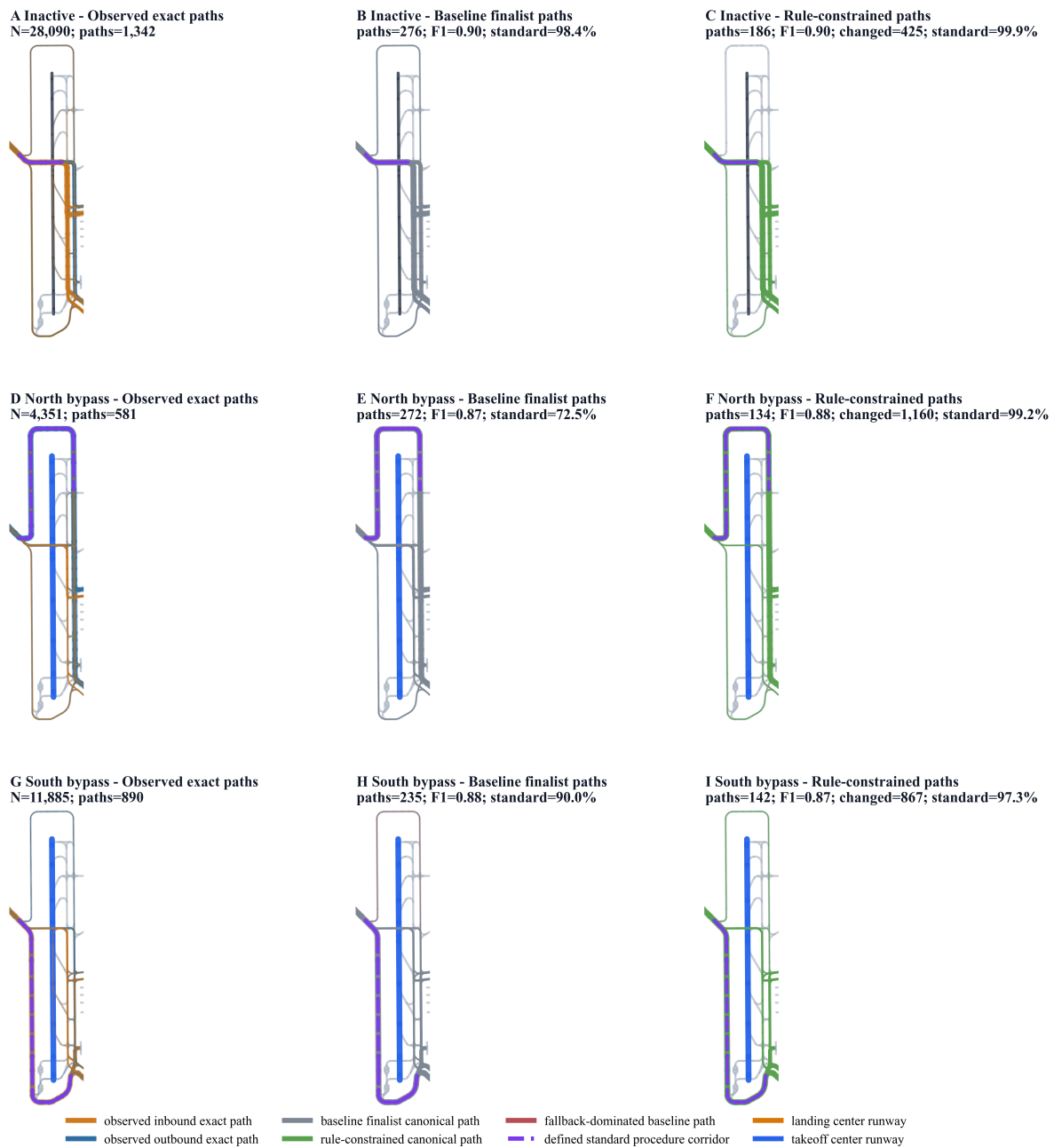


Figure E.2: Center-runway path-choice counterfactual

Figure E.2 compares observed paths, baseline model-selected paths, and the post-hoc standard-corridor constrained path candidates. The comparison shows that the baseline path-choice component often captures the dominant operational pattern, but that explicit corridor constraints can remove procedure-inconsistent path candidates in some regimes.

Table E.1: Center-runway standard-corridor counterfactual

Regime	Test ments	move-	Baseline standard-path share	Constrained standard-path share	Changed share	Mean F1 change
Inactive / middle crossing	28,090		98.36%	99.87%	1.51%	+0.002 (+0.22%)
North bypass / 36C landing or 18C takeoff	4,351		72.51%	99.17%	26.66%	+0.019 (+2.18%)
South bypass / 18C landing or 36C takeoff	11,885		90.03%	97.32%	7.29%	−0.010 (−1.12%)
All regimes	44,326		93.59%	99.12%	5.53%	+0.0004 (+0.05%)

Table E.1 shows that the rule-aware restriction has the clearest benefit in the north-bypass regime. In that regime, the constrained output increases the standard-path share from 72.51% to 99.17% and improves mean path F1 by 0.019. The inactive regime changes little because the baseline already follows the standard middle-crossing pattern in most cases.

The south-bypass regime is more ambiguous. The standard-path share increases, but mean F1 decreases slightly. This indicates that some observed paths in the held-out data do not follow the manually selected standard corridor. A rule-aware restriction can therefore improve operational plausibility, but only when the rule definition matches actual runway use and transition state. A blanket path mask can remove historically observed behavior if the operating-state label is too coarse or if the procedure has exceptions.

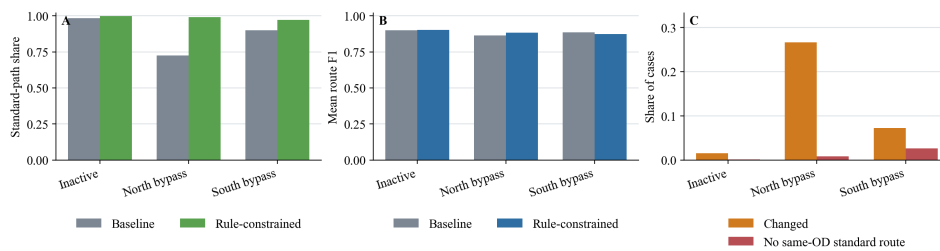
**Figure E.3:** Center-runway counterfactual summary

Figure E.3 summarizes the same finding. The center-runway active state is useful operational context, but it should be represented with enough temporal detail to account for transitions, mixed-use periods, and observed exceptions.

E.3. Canonical Path-family Availability under Procedure Constraints

The center-runway constraint can only select a compliant same-OD path candidate when such a candidate exists in the OD-specific path-choice set. For some OD/regime combinations, procedure-compliant exact observed paths exist in the historical catalog but do not reach the support threshold required to become their own canonical path family. For other combinations, no compliant exact observed path is present for that OD and regime.

Table E.2: OD groups without compliant canonical path families

Regime	OD label	Blocked test move- ments	Catalog compliant exact share	Dominant reason
South bypass	V2 to Entry_A20	128	0.51%	Compliant exact paths exist but each is below 1% OD share.
South bypass	V3 to Entry_A20	71	0.45%	Compliant exact paths exist but each is below 1% OD share.
South bypass	V1 to Entry_A20	55	0.11%	Compliant exact paths exist but each is below 1% OD share.
North bypass	V3 to Entry_RN	17	0.94%	Compliant exact paths exist but each is below 1% OD share.
Inactive / mid- dle crossing	V3 to Entry_A10	12	5.07%	Compliant exact path reaches 1%, but the canonical representative is non-compliant.

Table E.2 shows that operational constraints interact with path-catalog construction. The ordinary canonical path-family support threshold is useful for reducing sparsity, but it can remove rare path candidates that are operationally important under a specific procedure state. This suggests that future path catalogs should distinguish between ordinary statistical support and rule-critical support. A rare path may be worth retaining if it is the only path candidate that satisfies a known procedure constraint.

E.4. Runway 24 S7 Endpoint Restriction

The second case studies runway-24 departures while 18L or 36R is active. In this operating situation, S7 runway-entry stopbars are treated as unavailable because of jetblast-related procedure constraints. The endpoint-selection component is evaluated in its baseline form and under a post-hoc availability mask that removes S7 candidates without retraining.

This case tests a different kind of operational constraint from the center-runway example. The center-runway case restricts path candidates. The runway-24/S7 case restricts endpoint availability. Both cases ask whether explicit operational availability information can explain localized model errors.



Figure E.4: Runway-24/S7 surface-surveillance points

Figure E.4 checks whether the observed S7 endpoint labels correspond to the processed surface-surveillance trajectories. This matters because a rule-aware endpoint restriction should not be interpreted only as a model correction. It also tests whether the operational-state label used to define the rule matches the actual runway and endpoint use visible in the data.

Table E.3: Runway-24/S7 population denominators

Quantity	Taxi move-ments	Share of total endpoint test set	Share of runway-24 cases	Definition
Rule-scope move-ments	11,766	18.21%	44.52%	Runway-24 departures while 18L or 36R is active in the runway configuration.
Observed S7 move-ments	1,869	2.89%	7.07%	Rule-scope movements whose historical observed endpoint is S7E or S7W.

Table E.3 shows that the rule scope is large enough to affect endpoint-selection validation. The observed-S7 subset is smaller, but still operationally important because these cases determine whether the proposed availability rule is consistent with the observed data.

Table E.4: Runway-24/S7 endpoint-availability counterfactual

Group	Taxi move-ments	Baseline share	S7	Rule S7 share	Changed share	Weighted F1 change
All affected runway-24 cases	11,766	21.23%		0.00%	20.51%	−0.058
Observed endpoint is not S7	9,897	14.29%		0.00%	14.25%	+0.063
Observed endpoint is S7	1,869	58.00%		0.00%	53.67%	−0.728
18L active	11,636	20.46%		0.00%	19.73%	−0.063
36R active	130	90.00%		0.00%	90.00%	+0.561

Table E.4 shows that the availability mask improves cases whose observed endpoint is not S7. In those cases, the weighted F1 change is positive because the mask removes unavailable endpoint probability. However, the same mask necessarily fails cases whose observed endpoint is S7, because the observed class is removed from the candidate set. The aggregate result is therefore negative even though the non-S7 subset improves.

The interpretation depends on whether the observed-S7 cases are true exceptions to the procedure, procedure violations, or artifacts of using a scheduled runway-configuration state as a proxy for actual runway availability. This distinction is important. A deterministic endpoint-availability rule is only valid if the operating-state variable used to activate it matches the actual state of the operation at the minute level.

E.5. Historical Schedule Versus Observed Runway Use

The timing audit indicates that most observed-S7 cases occur near the edges of scheduled 18L/36R active windows or outside the observed first-to-last operation span for the blocking runway. Across 1,869 observed-S7 affected cases, 49.0% occur within 15 minutes of the scheduled configuration start, 11.7% within 15 minutes of the scheduled configuration end, and 75.7% are in a 15-minute edge band. Only 9.4% are inside the observed first-to-last 18L/36R operation span, while 68.3% are before the first observed operation and 16.0% have no observed blocking-runway operations in the window.

This supports a practical interpretation: the scheduled runway configuration is not a perfect minute-level proxy for actual runway availability. A scheduled active-state label may start before the first observed blocking-runway operation, extend after the last observed operation, or include transition periods during which endpoint availability differs from the nominal schedule. Operational constraints should therefore be linked to observed runway use, transition buffers, or more precise operational-state records when possible.

The same issue also explains why a rule-aware counterfactual can improve one subset and degrade another. If the schedule state is too coarse, a mask based on that state can remove endpoints that were

still operationally used during transition periods. This does not invalidate the operational rule, but it shows that availability constraints must be time-aware.

E.6. Operational Interpretation and Limitations

Both cases show that operational constraints can explain model errors that are not purely statistical. The center-runway case suggests that rule-aware path candidates can improve procedure consistency, especially in the north-bypass regime. It also shows that rare procedure-compliant paths may be lost during canonical path-family compression. The runway-24/S7 case shows that endpoint-availability rules can improve non-S7 cases, but that coarse schedule-state proxies can degrade cases occurring near runway-state transitions.

The main implication is that operational constraints should be represented as time-aware context or availability layers, not as broad masks applied only from scheduled runway configuration. For path choice, this may require a rule-aware path catalog that retains rare but procedure-critical path candidates. For endpoint selection, it may require endpoint availability features based on observed runway use, transition buffers, local procedures, or expert-reviewed rules.

This case study also clarifies the boundary of the learned behavioral components. The components can infer repeated operational patterns from trajectories, but they do not know which rules were active unless the relevant state is represented in the context features or candidate availability set. Explicit operational-rule context can therefore improve interpretability and support, but it must be encoded carefully enough to avoid replacing one missing-context problem with another.

E.7. Operational-constraints Takeaways

This case study supports three conclusions. First, known operational rules can explain localized path-choice and endpoint-selection errors that are difficult to resolve through statistical learning alone. Second, the effectiveness of rule-aware constraints depends on whether the relevant operational state is represented at the right temporal resolution. Third, canonical path-family construction and endpoint candidate sets should account for rare but operationally important rule-compliant alternatives.

The broader lesson is that the observed-behavior baseline should remain data-driven, but not context-blind. Learned behavioral components should be complemented with explicit operational-state and availability information where such information is reliable. This would allow future versions of the model to distinguish learned behavioral variation from deterministic procedure constraints more clearly.



Case Study C: Methodology Transfer to Apron-area Taxi Movements

F.1. Purpose and Transfer Question

The scientific paper focuses on aircraft taxi movements between runway stopbars and apron boundaries. This case study tests how far that methodology transfers into the apron area, where movements are shorter, more fragmented, and more strongly affected by stands, pushback, stops, local procedures, and surveillance occlusion near buildings.

The case study is a methodology-transfer audit, not a full retraining of apron-area learned behavioral components. It asks which parts of the runway-stopbar–apron-boundary workflow can be reused, which parts require adaptation, and which parts are not yet reliable enough for large-scale learned-component training in apron areas.

The transfer question is therefore: can the same data-processing logic, graph alignment, path-catalog construction, and component-input preparation be applied to apron-area taxi movements, and what changes are needed before the methodology becomes operationally meaningful inside the apron?

F.2. Data Subset and Apron Scope

Two apron cases are examined. Apron D is relatively ordered and has a clearer relation between stand-side movements and apron-boundary access. Apron ED is more complex because several apron-boundary candidates are close together and local movements have more path variants. These two cases provide a useful contrast: Apron D tests whether the method transfers in a simpler apron geometry, while Apron ED tests whether it remains robust in a denser and more ambiguous apron setting.

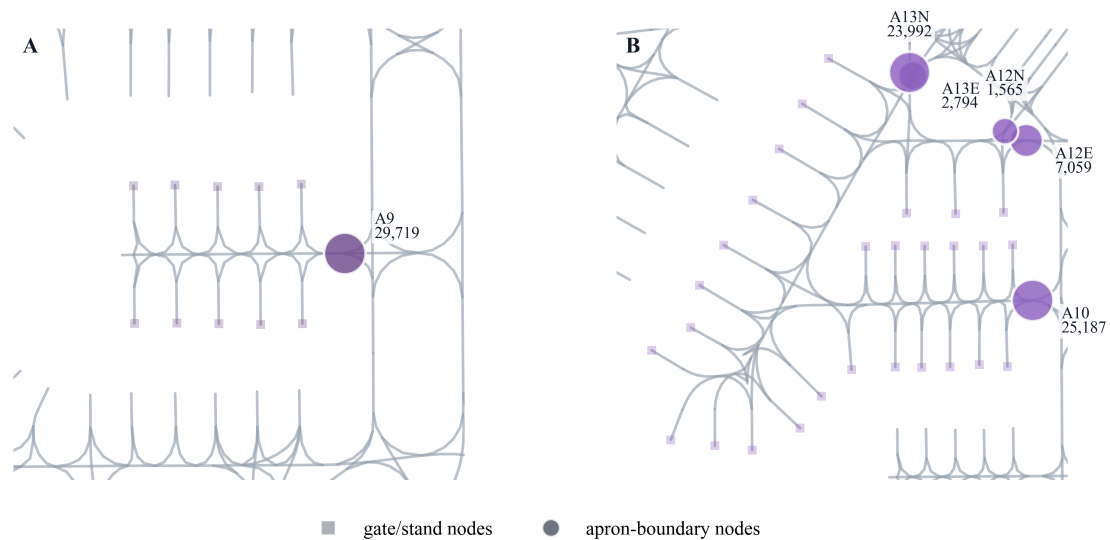


Figure F.1: Apron-area study scope

Figure F.1 shows the apron areas used for the transfer audit. The same airport layout graph is used as spatial reference, but the modeled movement phase changes. Instead of runway-stopbar–apron-boundary movements, the apron case focuses on movements between the gate or stand side and the apron boundary.

F.3. Taxiing-phase Extraction

The first transfer step is taxiing-phase extraction. In the main methodology, the scoped taxi trajectory is bounded by a runway stopbar and an apron boundary. In the apron case, the relevant phase is bounded by the gate or stand side and the apron boundary. This change makes boundary identification more difficult because apron boundary nodes can be close together and because stand-side movement may include pushback or waiting phases.

The initial extraction used direct proximity to boundary nodes. This was sufficient for Apron D, but rejected many Apron ED movements because adjacent boundary nodes were too close for robust identification. The adapted extraction adds topology-aware boundary disambiguation. Instead of selecting only the nearest boundary node, it also considers whether the candidate boundary is topologically consistent with the movement direction and nearby graph structure.

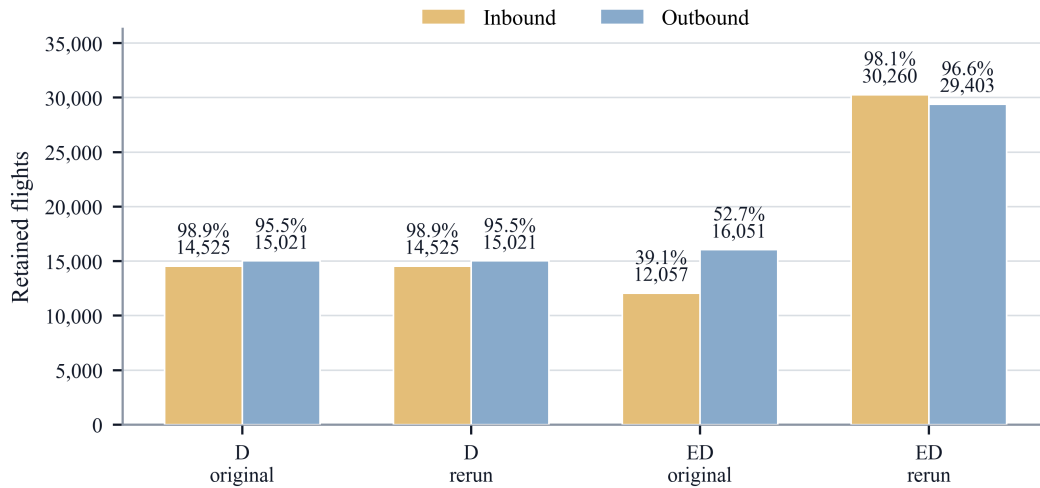


Figure F.2: Apron taxiing-phase extraction retention

Figure F.2 shows the retention improvement after topology-aware boundary disambiguation. The improvement is especially important for Apron ED, where geometric proximity alone cannot distinguish boundary candidates reliably.

Table F.1: Apron preprocessing bridge

Case	Stage	Inbound movements	move- ments	Outbound movements	Total movements	move- ments	Retained share of raw case
Apron D	Raw data	14,690		15,734	30,424		100.0%
Apron D	Track phase extracted	14,529		15,030	29,559		97.2%
Apron D	Map matched	14,487		14,764	29,251		96.1%
Apron D	Context attached	14,487		14,764	29,251		96.1%
Apron ED	Raw data	30,845		30,431	61,276		100.0%
Apron ED	Track phase extracted	30,211		29,292	59,503		97.1%
Apron ED	Map matched	25,315		19,037	44,352		72.4%
Apron ED	Context attached	25,315		19,037	44,352		72.4%

Table F.1 shows that taxiing-phase extraction transfers well after the topology-aware correction. Corrected Apron ED phase-extraction retention reaches 98.1% for inbound movements and 96.6% for outbound movements, compared with 39.1% and 52.7% before the correction. The main transfer lesson is that apron-scope extraction cannot rely only on nearest-boundary geometry; it needs topology-aware boundary logic.

F.4. State-machine Map Matching

The second transfer step is map matching. In the main taxiing scope, graph search connects runway stopbars and apron boundaries over taxiways. Inside the apron, a geometrically plausible shortest path can still be operationally invalid because apron movements include stand-entry edges, stand-side edges, pushback segments, and local one-way or phase-specific structures.

The adapted method adds a state-machine graph-search constraint. Inbound movements must finish through an expected stand-entry and stand-edge sequence near the gate. Outbound movements start from the gate or stand side, include pushback where appropriate, and then continue over taxiway edges

toward the apron boundary. The state machine does not replace map matching; it constrains the graph search so that the reconstructed path follows a plausible apron movement phase order.

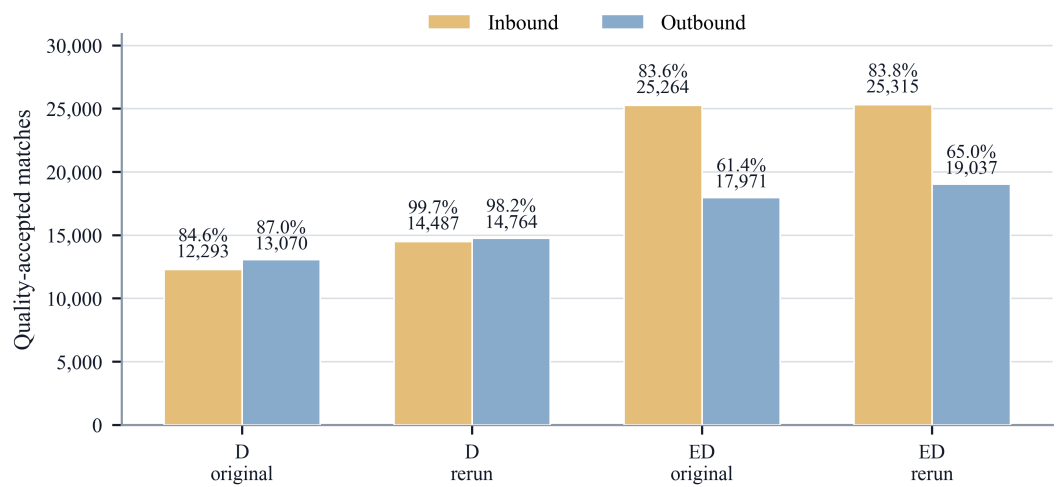


Figure F.3: Apron map-matching acceptance

Figure F.3 compares the original graph search with the state-machine constrained graph search. The main improvement is operational validity: the state-machine method removes reconstructed paths that are geometrically short but inconsistent with stand-entry, pushback, or apron movement ordering.

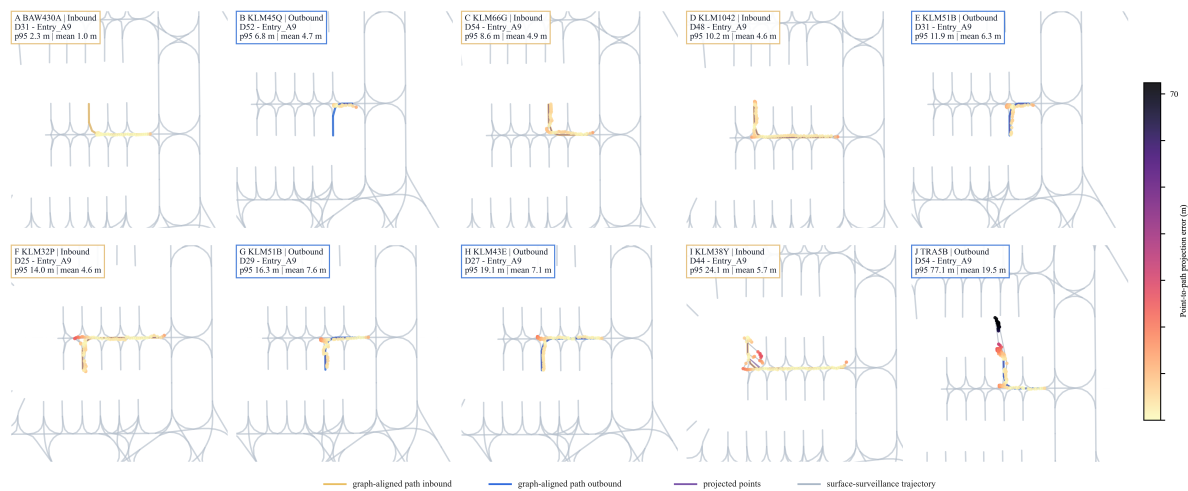


Figure F.4: Apron D map-matching examples

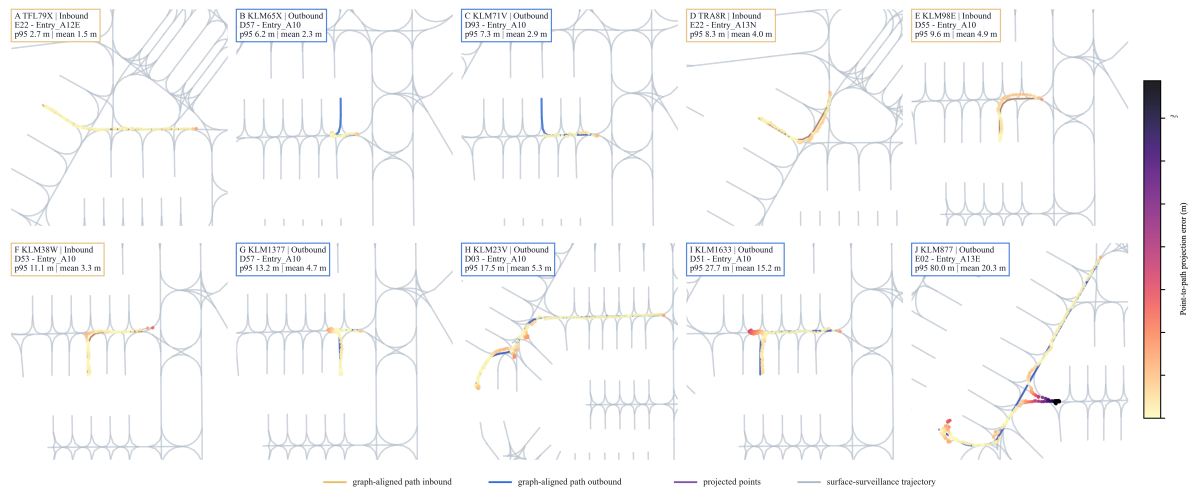


Figure F.5: Apron ED map-matching examples

Figures F.4 and F.5 show representative map-matching examples across the 95th-percentile point-to-path projection-error range. The examples illustrate why apron-area graph alignment is harder than runway-stopbar–apron-boundary alignment. Local surveillance noise, short edges, close boundary candidates, and stand-side geometry can all change whether the reconstructed path is operationally meaningful.

Table F.2: Apron map-matching summary

Case	Mode	Selected movements	Operationally valid share	Quality-accepted share	Error summary
Apron D	Original graph search	29,559	86.6%	85.8%	Mean error 4.64 m; median 95th-percentile error 9.04 m.
Apron D	State-machine graph search	29,559	100.0%	99.0%	Mean error 7.62 m; median 95th-percentile error 13.39 m.
Apron ED	Original graph search	59,503	95.4%	72.7%	Mean error 20.90 m; median 95th-percentile error 12.42 m.
Apron ED	State-machine graph search	59,503	100.0%	74.5%	Mean error 23.16 m; median 95th-percentile error 12.47 m.

Table F.2 shows the trade-off introduced by the state-machine constraint. For Apron D, operational validity increases from 86.6% to 100.0%, and quality-accepted matching increases from 85.8% to 99.0%. For Apron ED, operational validity also reaches 100.0%, while quality acceptance remains lower because ED contains more complex apron geometry and noisier surveillance patterns. The state-machine approach therefore improves operational validity, but it does not remove all spatial-quality limitations in complex apron areas.

F.5. Path Catalog and Component-input Preparation

After corrected phase extraction and state-machine map matching, the path-catalog step can be applied to the apron area. Exact observed apron paths are grouped into canonical path families using the same general idea as in the main methodology: reduce small reconstruction and tactical variations while preserving the main path candidates. The reduction is less pronounced in Apron ED than in Apron D because ED contains more gates, more OD combinations, and more local path variants.

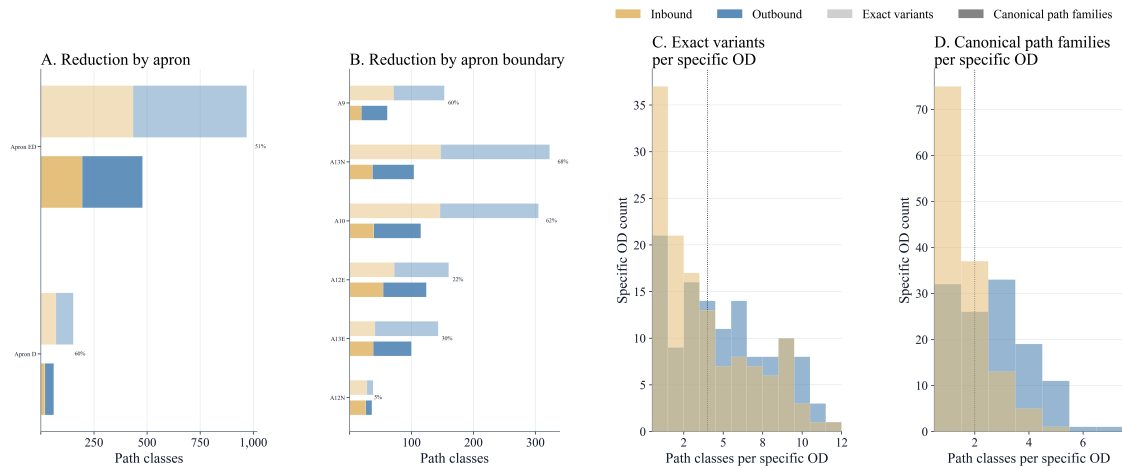


Figure F.6: Apron path-catalog preparation

Figure F.6 shows the reduction from exact observed paths to canonical path families for the apron cases. The diagnostic confirms that path-catalog construction remains meaningful inside the apron, but that local apron complexity reduces the amount of compression possible.

Table F.3: Apron path-catalog summary

Case	Taxi move- ments	Exact ob- served paths	Canonical path families	Specific groups	OD	Exact-to- canonical reduction
Apron D	29,249	153	61	20		60.1%
Apron ED	44,332	969	479	234		50.6%

Table F.3 shows that Apron D produces a compact catalog, while Apron ED remains more fragmented. The difference is operationally meaningful. Apron ED includes more local alternatives and more stand-side variation, so the same canonical grouping criteria cannot reduce the catalog as strongly without risking loss of meaningful path differences.

The component-input audit indicates that path-choice and endpoint-selection data products can be assembled after corrected graph alignment. For Apron D, 80.4% of movements lie in non-trivial OD groups for path choice. For Apron ED, the corresponding share is 53.7%. Endpoint boundary choice is trivial for Apron D in this case-study scope, while nearly all ED movements have a non-trivial boundary-choice setting after corrected extraction. This shows that apron-area component input preparation is possible, but that the difficulty and target structure vary strongly by apron geometry.

F.6. Speed-profile Preparation Limitations

Speed-profile preparation does not transfer as cleanly as path-catalog preparation. The main speed-profile preprocessing is designed for longer runway-stopbar–apron-boundary taxi movements. In that scope, edge traversals are generally longer, and stops can often be filtered without losing the movement structure. Apron movements are different. They include short edges, low speeds, pushback, push-and-pull behavior, and post-pushback waiting away from graph nodes.

Applying the original traversal criteria to apron movements can therefore reject or distort movements for reasons that are intrinsic to apron operations rather than data-quality failures. For example, a valid apron movement can contain a long wait after pushback, a short traversal over a stand-side edge, or a low-speed pushback segment. Treating all of these as ordinary taxi-speed observations would mix different movement phases into one speed-profile target.

The input audit reports 135,405 Apron D edge-traversal candidates, of which 128,533 are input-ready under the original traversal gates, corresponding to 94.9%. Apron ED has 416,182 candidates and 357,563 input-ready traversals, corresponding to 85.9%. These values show that many edge-speed observations can be prepared, but the high movement share with stops or waits remains a central limitation. For Apron D, 48.5% of movements include stops or waits.

The implication is that a future apron speed-profile component should separate movement phases before residual speed profiles are learned. Pushback, post-pushback release waiting, stand-side movement, and own-power taxiing should not be treated as one homogeneous edge-relative taxi-speed process. This is the main methodological difference between runway-stopbar–apron-boundary speed modeling and apron-area speed modeling.

F.7. Method-transfer Interpretation

The case study shows that the methodology transfers to apron areas in stages. Taxiing-phase extraction transfers after topology-aware boundary detection is added. Map matching transfers after state-machine constraints are added to represent apron-specific edge-type order. Path-catalog construction remains meaningful, although complex aprons produce more local path variants and therefore less compression. Component-input preparation is feasible for path choice and endpoint selection, but target difficulty depends strongly on apron geometry.

The main unresolved component is speed-profile behavior. Apron-area motion is not simply a shorter version of taxiway motion. It includes pushback, low-speed maneuvering, stand-side waiting, short-edge traversal, and local surveillance occlusion. These features require a movement-phase model before learned speed-profile behavior can be interpreted reliably.

This appendix therefore supports a focused conclusion: the runway-stopbar–apron-boundary methodology provides a useful base for apron-area analysis, but apron-area modeling requires explicit procedure phases and graph-state constraints before learned behavioral components should be trained at scale.

F.8. Apron-area Transfer Takeaways

The first takeaway is that the data-processing logic is transferable, but not unchanged. Apron-area scope extraction requires topology-aware boundary disambiguation because geometric proximity alone is unreliable when several apron-boundary candidates are close together.

The second takeaway is that graph alignment inside the apron must encode movement-phase order. State-machine map matching improves operational validity because it prevents geometrically plausible but operationally invalid stand-entry or pushback paths.

The third takeaway is that canonical path-family construction can still reduce path fragmentation, but complex apron geometry limits the amount of compression. Apron ED remains more fragmented than Apron D because it contains more gates, more OD combinations, and more local path variants.

The fourth takeaway is that speed-profile preparation requires a different movement-phase representation. Pushback, waiting, short-edge movement, and own-power taxiing should be separated before an apron-specific speed-profile component is trained.

Overall, the case study shows that the main methodology is a useful starting point for apron-area modeling, but that apron operations require additional boundary logic, state-machine graph constraints, and movement-phase definitions. This explains why the scientific paper keeps the main observed-behavior baseline focused on runway-stopbar–apron-boundary aircraft taxi operations.