Rethinking the Safety Case for Risk-Aware Social Embodied Intelligence

Jay Patrikar

CMU-RI-TR-25-26

May 2025

School of Computer Science
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Abstract

Achieving real-world robot safety requires more than avoiding risk—it demands embracing and managing it effectively. This thesis presents a safety case for risk-aware decision-making and behavior modeling in complex, multi-agent environments such as aviation and autonomous driving. We argue that safety stems from an agent's ability to anticipate uncertainty, reason about intent, and act within operational boundaries defined by prior knowledge, rules of the road, social context, and historical precedent.

To enable safe and interpretable decision-making, we integrate MCTS with logic specification to improve rule adherence into learned policies for both single and multi-agent settings. We develop symbolic rule-mining methods using inductive logic programming, extracting interpretable constraints from both trajectories and crash reports. To address out-of-distribution risk, we propose a fusion framework that combines neural imitation learning with symbolic rule-based systems. Finally, to mitigate modelling risks we will talk about combining RAG with crash reports for grounded action arbitrations in complex settings.

To support learning from real-world behavior within aviation, we introduce three datasets: TrajAir, a social aerial navigation dataset; TartanAviation, a time-synced multimodal dataset for intent inference; and Amelia-48, a large-scale airport surface movement dataset across U.S. airports, enabling predictive analytics in air traffic management.

Together, these contributions and the tools developed along the way enable autonomous systems to reason under uncertainty, incorporate diverse priors, and operate reliably in complex, real-world settings.

"Hello, babies. Welcome to Earth. It's hot in the summer and cold in the winter. It's round and wet and crowded.. There's only one rule that I know of, babies — 'God damn it, you've got to be kind."

— Kurt Vonnegut

To Mom and Dad

Acknowledgment

This dissertation marks the culmination of a journey that would not have been possible without the support, guidance, and encouragement of many individuals.

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Sebastian Scherer, for his unwavering support, insightful feedback, and patient mentorship throughout my PhD. Basti, your guidance has not only shaped this research but also profoundly influenced my growth as a researcher and engineer. Thank you also for supporting and encouraging my aviation journey. I look forward to continuing to work with you!

I am also deeply thankful to my thesis committee members, Dr. Jean Oh, Dr. Andrea Bajcsy, and Dr. Hamsa Balakrishnan, for their valuable time, constructive feedback, and encouragement that helped refine and strengthen this work. I am fortunate to have worked alongside brilliant collaborators and labmates in the AirLab and BIG Lab, whose technical insights, stimulating discussions, and camaraderie made the research process both productive and enjoyable. Special thanks to Brady Moon, Ingrid Navarro, Parv Kapoor, Cherie Ho, Thiago A. Rodrigues, Nikhil Varma Keetha, Arnav Choudhry, Sourish Ghosh, Pablo Ortega-Kral, Alonso, Milad Moghassem Hamidi, Jack Wang, Rebecca Martin, Jacob Feldgoise, Bastian Wagner, João P. A. Dantas, Yafei Hu, Rohan Baijal, Jasmine Jerry Aloor, Marcelo Jacinto, Vishal Dugar, Benjamin Stoler, Jimin Sun, T. Yoneyama, Atharva Chandak, Yaoyu Hu, Yao He, and Jong Hoon Park for their collaboration and critical input.

I would also like to acknowledge the support of my internship mentors and collaborators at NVIDIA: Dr. Sushant Veer, Dr. Apoorva Sharma, Dr. Boris Ivanovic, and Dr. Marco Pavone, whose expertise and perspective enriched this work and opened new directions for exploration. At Microsoft Research, I am deeply grateful to Dr. Rogerio Bonatti and Dr. Ashish Kapoor for providing me with the opportunity

to intern at Microsoft. The experience was transformative, offering a unique blend of academic rigor and practical application. Collaborating with leading researchers and contributing to projects with real-world impact was both inspiring and invaluable for my professional development.

Aviation has always been a source of inspiration and joy throughout my PhD journey. Flying provided me with a unique perspective, not just literally but also metaphorically, allowing me to approach research challenges with renewed vigor and clarity. I would like to thank my Certified Flight Instructor, Dr. Tim Weber from the University of Pittsburgh, for his unwavering belief in my abilities. Your mentorship and guidance have been instrumental in my growth as a pilot. The Flying Club, co-founded with Basti and Azarakhsh Keipour, has evolved into a vibrant community of aviation enthusiasts. To all its members and especially its board—Jack Wang, Ben Kroop, Advait Madhekar, and Dr. Conrad Tucker—thank you for your dedication and enthusiasm. I hope The Flying Club continues to inspire a new generation of pilots at CMU and beyond.

I would also like to extend my heartfelt appreciation to the staff at the Robotics Institute. Your support, both administrative and technical, has been invaluable.

As I reflect on this journey, I am overwhelmed with gratitude for the unwavering support of my family and friends. To my parents, your endless encouragement and sacrifices have been the bedrock of my aspirations; your belief in me has been my guiding light. To my sister Prajakta, her husband Pranav, and their children Preet and Praveer, your constant cheer and love have been a source of immense strength. To my friends beyond Pittsburgh—Suchita, Hardik, Kalyani, Yash, Shantanu, Utkarsh, Madhura, Yukta, and my peers from IIT Kanpur—your unwavering support and belief in my potential have been invaluable. To my Pittsburgh family— Nihar, Prakhar, Shrey, Shruti, Parth, Rutuja, Malvika, Sai, Bela, Aniket, Aishwarya, Tarika, Rohit, Mahabod, Abhishek, Riti, Ceci, Mononito, Ritika, Ryan, and many others—your companionship has been a sanctuary, providing joy and solace amidst the challenges.

This achievement is not mine alone; it is a testament to the collective love, patience, and encouragement that I have been fortunate to receive. From the depths of my heart, thank you for being my pillars of strength and for walking this path with me.

Keep on keeping on..

Funding

This research was supported by US Department of Energy's Vehicle Technologies Office, award number DE-EE0008463, Army Futures Command Artificial Intelligence Integration Center (AI2C), Mitsubishi Heavy Industries (MHI) project #A025279. Work was also supported by Boeing (award #2022UIPA422). This work used Bridges-2 at Pittsburgh Supercomputing Center through allocation cis220039p from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

This material also is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE1745016.

We would like to thank the Allegheny County Airport Authority and the Office of Manager of Pittsburgh-Butler Regional Airport for their support. We would like to thank the pilots of Condor Aero Club and ABC Flying Club for their support in user studies.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Achieving effective human-aware autonomy requires more than avoiding risk—it demands embracing and managing it. Autonomous systems that need to operate alongside humans must do so by adhering to established rules, behavioral expectations, and safety practices. Although generalist agents trained on internet-scale data can leverage common-sense priors to bootstrap reasonable behavior, deploying them introduces challenges that call for a careful balance between robust safety guardrails and preserving model expressiveness. One way to achieve this is by *rethinking* the *Absence of Unreasonable Risk* (AUR) principle [54] for embodied intelligence systems. Central to safety standards like ISO-26262 [2] and DO-178C [1], AUR ensures that even when failures or malfunctions occur, a system's operation does not result in an unacceptable level of harm. All this leads to a fundamental research question: How can we ensure AUR while deploying data-driven autonomous agents in the real-world? In everyday life, humans continuously assess and mitigate risks, striking a balance between safety and performance through an implicit understanding of their environment. This ability stems not only from acquired knowledge learned from others (akin to behavior cloning) - but also from personal experience gained through interaction with the world (akin to reinforcement learning). In addition, humans develop an understanding of laws, rules, and regulations, applying them contextually to different situations. This thesis focuses on equipping autonomous agents that interact with humans in shared spaces with similar capabilities. We explore safety-critical domains: from aviation navigation algorithms for pilotless AI aircraft [126] and self-driving vehicles [138]— united by the goal of achieving safe and seamless human-aware autonomy. Aviation is especially critical with the planned introduction of autonomous agents in an increasingly dense airspace. Recognizing that risk emerges from both known and unknown factors, I have decomposed the AUR framework into four knowledge-aware quadrants. The risks associated with known uncertainties in human behavior **(known-unknowns)**, failures to contextualize established rules **(known-knowns)**, inherent "baked-in" system failure modes **(unknown-knowns)**, and unforeseen black swan events **(unknown-unknowns)**, each of which requires distinct mitigation strategies to balance safety and performance.

As robots leave the deterministic world of factory floors and take a leap into the chaotic real world, roboticists are facing newer and harder challenges in modeling and handling the inherent stochasticity of these new arenas. The feasibility of this transition hinges on how well these robots perceive and reason in real-life situations. Developing risk-aware and safe algorithms is critical if these machines are to coexist alongside uncertain human actors while operating in a world designed for humans. Humans have the uncanny ability to adaptively balance safety and performance while navigating alongside other humans. They build behavior models of other humans with the assumption that the other human will act in a cooperative manner without breaking social and legal rules. Humans continuously monitor the adherence of the other agents to these models. Higher trust enables them to take riskier but potentially rewarding actions while low trust prompts conservative behaviour.

For autonomous agents to integrate within the current infrastructure, autonomous systems need to act in a manner that is indistinguishable from humans. Endowing them with these attributes necessitates developing systems that can predict human behavior, comprehend and ground abstract rules in the action space, and embrace the uncertainty that is inherent in cooperative decision-making alongside humans. This thesis provides pathways to achieve this by using data-driven methods that use historical data to synthesis strategies for human-robot collaboration within the multiple domain.

1.2 Safe and Seamless Full-Scale Aerial Autonomy

The content of this thesis is applicable to multiple robotics domains. Although results are presented on established domains like autonomous vehcile and unamnned aerial systems, this thesis also presents some pioneering work in the understudied domain of full-scale autonomous aircraft.

Advanced Aerial Mobility (AAM) is an inclusive term that covers urban (UAM), regional (RAM), intraregional (IAM), and suburban air mobility (SAM) solutions [12]. All of these proposed solutions have one thing in common: They all envision a future where autonomous or semi-autonomous aerial vehicles are seamlessly integrated into the current airspace system. AAM solutions open doors to significant socio-economic benefits while at the same time presenting challenges in the seamless integration of these systems with human-operated aircraft and controlling agencies.

Today, manned and unmanned vehicles are separated, limiting the utility and flexibility of operations and reducing efficiency. Human-operated aircraft follow one of the two rules for operation: Visual Flight Rules (VFR) or Instrument Flight Rules (IFR). Flying under VFR or IFR is typically a function of weather conditions. Under VFR, an aircraft is flown just like driving a car within the Federal Aviation Administration (FAA) established rules of the road. On the other hand, IFR flying is typically associated with flying aircraft under degraded weather conditions where separation is provided by the Air Traffic Control (ATC). While the pursuit of integrating AAM can start either under IFR or VFR, automated-VFR operations often have better scalability than automated-IFR operations [120]. Another option involves Unmanned Aircraft System Traffic Management (UTM) solution [12], which in its current iteration only focuses on small unmanned aerial aircraft operating close to the ground (less than 700 ft) in uncontrolled airspace.

Mastering VFR operations for autonomous aircraft has significant operational advantages at unimproved sites and achievable traffic density compared to IFR or completely separated operations between manned and unmanned systems. For (semi-)autonomous aircraft to operate in tandem with human pilots and ATC controllers under VFR, technological advancements in certain key areas are required, specifically:

1. Unmanned aircraft should be able to guarantee self-separation even in a tightly-spaced terminal airspace environment;

2. Unmanned vehicles should be able to interpret high-level instructions by ATC to meet the expectations of a normal traffic flow;

3. Autonomous aircraft need to understand the expected and unexpected behavior of other aircraft;

4. Communications by other pilots and ATC need to be parsed, and corresponding

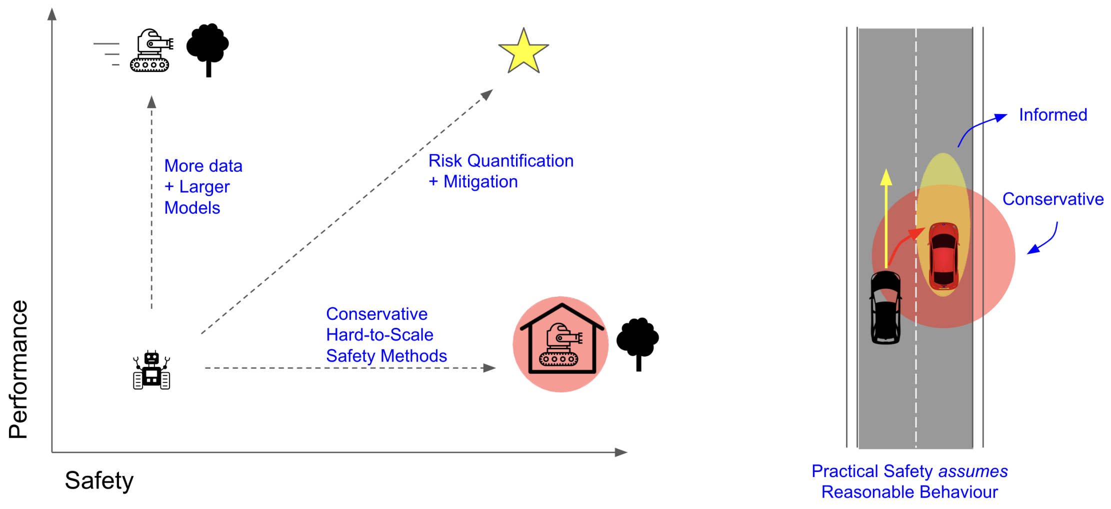


Figure 1.1: Figure shows the safety vs performance trade-off while advocating for informed risk quantification and mitigation strategies. Also shown is the motivating example showcasing practical safety in the case of two cars in parallel lanes on a highway. A conservative (red) bubble characterizes this as an unsafe encounter, as the car on the left can always swerve into the ego agent but humans perform this task every day with reasonable assumptions (yellow) on the behaviour of the other agent.

responses should be produced; and

5. Other aircraft need to be detected and estimated, and their future trajectories need to be predicted.

Many of these challenges have parallels in the self-driving industry, and technological improvements can be leveraged to produce domain-specific solutions for AAM. While this is promising, VFR-like AAM integration introduces newer challenges while pushing boundaries on the current state of technology. This thesis addresses some of these challenges.

1.3 Thesis Vision and Organization

We now introduce the statement of this thesis that frames the broader vision of this body of work.

Thesis Statement

This thesis contends that safe autonomy in social, safety-critical settings requires **rethinking safety as risk-aware reasonable social intelligence**—grounded in prediction, rule adherence, and planning under uncertainty—wherein **distinct mitigation strategies are crafted for each identified risk source.**

We decompose the safety case into 4 quadrants based on the sources of uncertainty. Commonly known as the Rumsfeld Matrix, we categorize sources of risk into four quadrants:

1. **Known-Unknowns:** Things we are aware of but do not understand. (eg behavior uncertainty)
2. **Known-Knowns:** Things we are aware of and understand. (eg rules-of-the-road)
3. **Unknown-Unknowns:** Things we are neither aware of nor understand. (eg black-swan events)
4. **Unknown-Knowns:** Things we understand but are not aware of. (eg biases in training data)

PART I: [Known-Unknowns] Modeling uncertainty in human behavior

We cannot eliminate uncertainty in human behavior, but we can quantify it. Recent approaches model human behavior using a combination of multi-modal trajectory data and deep social models [137] to learn the underlying uncertainty. But availability of high-quality data, generalizability and downstream use in decision-making remain a challenge. Prior work in socially aware trajectory prediction has focused on pedestrians and self-driving vehicles, with little exploration of aviation. This thesis pioneers human uncertainty estimation in aviation domain by developing a series of progressively advanced aviation trajectory prediction datasets and models. We started with TarjAir [137], the first opensource 3D trajectory prediction dataset using aircraft transponder data, and subsequently introduced TartanAviation [134], a multimodal dataset combining vision, speech, and transponder data. Towards the end, we introduce Amelia-48 [125], the largest public domain data set featuring multiple years of FAA surface movement data (70TB) from 48 US airports. We also introduce TrajAirNet, trajectory prediction method, to model agent-agent, agent-environment, and agent-context interactions. The future trajectories of agents around an airport are a function of the history of all the participating heterogeneous agents, their respective goals, the static airport environment, and the dynamic weather context. The proposed method uses the dataset to effectively capture all aspects of this behaviour to predict aircraft trajectories.

Part II: [Known-Knowns] Neurosymbolic Reasoning for Rule Adherence

Compliance with established rules minimizes risk. While distilling rules as logical specifications provides concise and interpretable representations [92], their scalability in complex contexts and their integration with learning-based systems remain a challenge. Learned policies often fail to capture the governing rules, making them vulnerable to covariate shifts. To address this, the thesis incorporates Signal Temporal Logic (STL) as spatio-temporal constraints into policy networks. In this work, STL specifications were integrated into a learned policy via MCTS, where STL robustness values guided the search towards rule-compliant trajectories. Unlike methods that embed STL during training [92], our online approach allows dynamic rule adjustments. This neurosymbolic approach achieved a 60% improvement over a learning-only baseline. Furthermore, we integrate our prediction models into downstream decision-making pipelines that distill pilot behavior into uncertainty-aware policies. Our approach leverages multi-agent MCTS that combines learned behavior with strict collision checking to achieve safer, more reliable plans.//

Part III: [Unknown-Unknowns] Managing out-of-distribution scenarios:

This is one of the hardest risk to manage because it is unforeseen requiring nuisance-free fallback safety mechanisms. Any system deployed in the real world will encounter out-of-distribution (OOD) scenarios necessitating mitigation strategies that gracefully degrade the system to a safe state. While learning-based planners excel in in-distribution (ID) settings, they can be unpredictable in OOD. To combine their strengths with the robustness of rule-based systems, I replaced the uniform prior in categorical evidential learning [34] with a rule-based informed Dirichlet prior. In our approach, a rule-based planner [176] generates a traffic-compliant prior over future trajectories, which a neural network refines into a data-driven posterior. This ensures that in OOD scenarios, the posterior remains close to the rule-based prior for safety, while in ID scenarios, the network captures nuanced driving behavior—yielding robust performance across both conditions.

Part IV: [Unknown-Knowns] Mitigating Latent Failure Modes

Embodied AI systems can harbor hidden risks that emerge only after deployment, making runtime mitigation particularly challenging. Operational Design Domains (ODDs) define the conditions under which these systems operate safely. However, establishing an ODD is itself challenging, as it requires anticipating potential failure modes in advance. Motivated by the "Swiss-Cheese" model, to mitigate unknown common-mode failures, I explored the use of multiple reasoning engines concurrently, each with distinct approaches: some based on learned data, others purely rule-based. However, this raises the challenge of runtime arbitration between the different models. To address this, I focused on using LLMs to adjudicate actions from different planners. Defining 'safe' behavior is inherently complex, as overly simplistic definitions can lead to unintended consequences. To address this, I modified GraphRAG [68] to mine crash reports for safety-critical counterfactual reasoning. These human-generated reports served as a rich source of grounded guardrails in realistic, context-sensitive terms, helping us define and ensure meaningful safety metrics for autonomous systems. In addition to this, previous works rely on handcrafted logic specifications, limiting both their complexity and soundness. Although LLMs can directly ground the rules in data, they struggle with inductive reasoning on their own. To address this, I propose combining Inductive Logic Programming (ILP) [42] with LLM synthesis for rule mining. ILP decomposes rule learning into generate, test, and constrain stages, progressively pruning the hypothesis space. Using crash reports as a rich source of rule information, I use LLMs to extract relevant information to feed into the ILP system. Using rules generated from the ILP system, the system can serve as runtime monitors to identify safety critical cases.

1.4 Bibliographic Notes

Most of the work in this thesis has been published or presented in the following peer-reviewed venues:

1. **RuleFuser: An Evidential Bayes Approach for Rule Injection in Imitation Learned Planners for Robustness under Distribution Shifts**
   Jay Patrikar, Sushant Veer*, Apoorva Sharma*, Marco Pavone & Sebastian Scherer
   International Symposium of Robotics Research (ISRR) [2024]

2. **AmeliaTF: A Large Model and Dataset for Airport Surface Movement Forecasting**
   Jay Patrikar*, Ingrid Navarro*, Pablo Ortega-Kral*,Haichuan Wang, Zelin Ye, Jong Hoon Park, Jean Oh & Sebastian Scherer
   AIAA AVIATON 2024 [2024]

3. **TartanAviation: Image, Speech, and ADS-B Trajectory Datasets for Terminal Airspace Operations**
   Jay Patrikar, Joao Dantas, Brady Moon, Milad Hamidi, Sourish Ghosh, Nikhil Keetha, Ian Higgins, Atharva Chandak, Takashi Yoneyama, & Sebastian Scherer
   Nature Scientific Data [2024]

4. **Learned Tree Search for Long-Horizon Social Robot Navigation in Shared Airspace**
   J Patrikar*, Ingrid Navarro*, J Dantas, R Baijal, I Higgins, Jean Oh, S Scherer
   IEEE Robotics and Automation Letters (RA-L) [2023]

5. **Follow The Rules: Online Signal Temporal Logic Tree Search for Guided Imitation Learning in Stochastic Domains**
   J Patrikar*, Jasmine Aloor*, Parv Kapoor, Jean Oh, S Scherer
   International Conference on Robotics and Automation (ICRA) [2023]

6. **Predicting Like A Pilot: Dataset and Method to Predict Socially-Aware Aircraft Trajectories**
   J Patrikar, B Moon, Jean Oh, S Scherer
   International Conference on Robotics and Automation (ICRA) [2022]

Additional Peer-Reviewed Papers

Below we list additional papers that are related to but fall outside the primary focus of this dissertation.

1. **Demonstrating ViSafe: Vision-enabled Safety for High-speed Detect and Avoid**
Jay Patrikar*, Parv Kapoor*, Ian Higgins*, Nikhil Varma Keetha*, Brady Moon, Zelin Ye, Yao He, Ivan Cisneros, Changliu Liu, Eunsuk Kang, Sebastian Scherer
Robotics: Science and Systems [2025]

2. **Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis**
Hu, Y., Xie, Q., Jain, V., Francis, **J., Patrikar**, J., Keetha, N., ... & Bisk, Y.
The International Journal of Robotics Research [Submitted] [2024]

3. **FoundLoc: Vision-based Onboard Aerial Localization in the Wild**
Y He, I Cisneros, N Keetha, **J Patrikar**, Ye Z, I Higgins, Y Hu, P Kapoor, S Scherer
Preprint ArXiv [2023]

4. **Pegasus Simulator: An Isaac Sim Framework for Multiple Aerial Vehicles Simulation**
Jacinto, M., Pinto, **J., Patrikar**, J., Keller, J., Cunha, R., Scherer, S., Pascoal, A.
International Conference on Unmanned Aircraft Systems 2024 [2023]

5. **AirTrack: Onboard Deep Learning Framework for Long-Range Aircraft Detection and Tracking**
Sourish Ghosh, **J Patrikar**, B Moon, Milad Hamdi, S Scherer
International Conference on Robotics and Automation (ICRA) [2023]

6. **Quantification of Viable Drone Flight Hours Due to Weather Conditions**
A Sharma, **J Patrikar**, B Moon, C Samaras, S Scherer
Journal of Transport Findings [2022]

7. **Challenges in Close-Proximity Safe and Seamless Operation of Manned and Unmanned Aircraft in Shared Airspace**
Jay Patrikar, et al.
International Conference on Robotics and Automation (ICRA Workshop) [2022]

8. **CVaR-based Flight Energy Risk Assessment for Multirotor UAVs using a Deep Energy Model**
J Patrikar*, B Moon*, A Choudhry*, C Samaras, S Scherer
International Conference on Robotics and Automation (ICRA) [2021]

9. **Adaptive Tube Library for Safe Online Planning Under Unknown Tracking Performance**
C Ho, **J Patrikar**, R Bonatti, S Scherer
Workshop, Robotics: Science and Systems [2021]

10. **In-flight positional and energy use dataset of package delivery quadcopter UAVs**
T Rodrigues, **J Patrikar**, A Choudhry, J Feldgoise, V Arcot, A Gahlaut, S Lau, B

Moon, B Wagner,
S Matthews, S Scherer, C Samaras
Nature Scientific Data [2020]

11. **Wind and the City: Utilizing UAV-Based In-Situ Measurements for Estimating Urban Wind Fields**
J Patrikar, B Moon, S Scherer
International Conference on Intelligent Robots and Systems (IROS) [2020]

12. **Real-time Motion Planning of Curvature Continuous Trajectories for Urban UAV Operations
in Wind**
J Patrikar, V Dugar, V Arcot, S Scherer
International Conference on Unmanned Aircraft Systems (ICUAS) [2020]

Part I

[Known-Unknowns] Modeling Uncertainty in Human Behavior

Chapter 2

TrajAir And TarjAirNet: Dataset and Method For Social Aircraft Trajectory Prediction

2.1 Introduction

General Aviation (GA) comprises all civil flights except scheduled passenger airline services. More than 90% of the roughly 220,000 civil aircraft registered in the United States (US) are GA aircraft [56]. In contrast with airline service aircraft, which operate with two pilots in a structured higher-altitude operational envelope, GA aircraft are often individually piloted in a more unstructured lower-altitude environment. This makes the pilotage challenging in the best of circumstances. According to the Federal Aviation Administration (FAA), for every commercial airline accident in 2015, there were approximately 36 accidents in GA, with 77 % of non-fatal accidents in terminal airspaces [123]. This low altitude environment is also where a bulk of the next generation of Unmanned Aerial Vehicles (UAVs) are expected to operate [143]. These UAVs are expected to seamlessly interact with other UAVs and crewed air traffic operating in this shared airspace. Nowhere is this interaction more pronounced than in low-altitude terminal airspace around airports.

All flights typically begin and end in airspace surrounding airports known as terminal airspace. Low altitudes, multi-agent close-proximity interactions, dynamically changing conditions, and rapid decision making are hallmarks of this type of airspace.

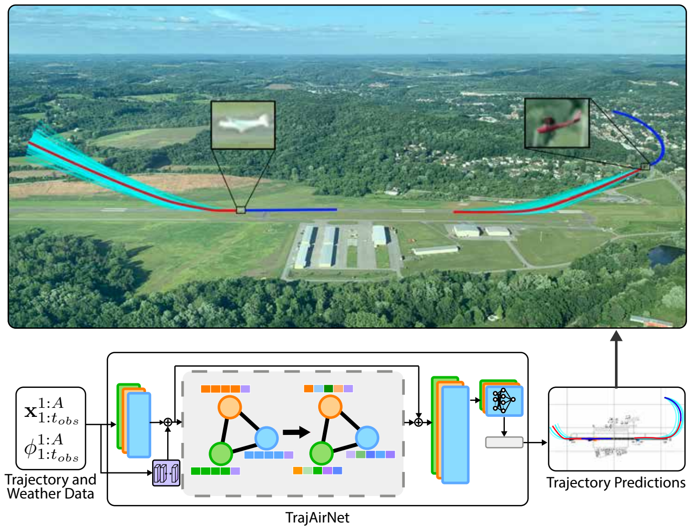


Figure 2.1: *TrajAir* dataset includes recorded ADS-B trajectories of aircraft interacting in a non-towered general aviation airport and the weather context from METAR strings. *TrajAirNet* predicts multi-future samples (cyan) of trajectories for all agents by conditioning each on the history (blue) of all the agents and the weather context.

As compared to en-route airspace, where agents are typically well-separated, agents in terminal airspace are at a higher collision risk. Out of the nearly 20,000 active airports [81] in the US, only around 4% are *towered*, meaning that a control tower is present as a centralized authority ensuring separation. This indicates that most GA airports are *non-towered*, implying that the pilots must directly communicate with other pilots and take decentralized actions. Pilots operating in non-towered airspace are solely responsible for guiding aircraft to safety.

In the context of social navigation, where only implicit rules are assumed [116], GA offers a unique case where all aircraft operating within terminal airspace are expected to self-organise to follow guidelines established by the FAA, which act like rules-of-the-road for aircraft. These guidelines ensure separation and smooth flow by standardising a rectangular traffic pattern around the runways [55]. More details on these guidelines are provided in future sections. While not enforced, pilots are expected to adhere to the traffic pattern, but deviations arise due to the lack of specificity, pilot's experience level, type of aircraft, weather, and position of other agents. This design flexibility improves efficiency and ensures safety while accommodating the needs of various agents using the same infrastructure. Only the traffic pattern's general shape, direction, and altitude are established. This leaves room for each pilot to make their own decisions to maintain separation while respecting personal minimums and aircraft

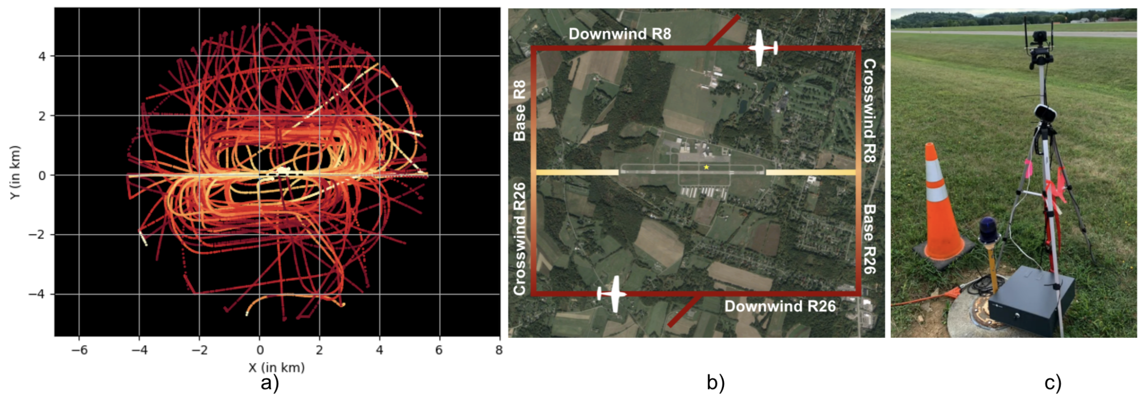


Figure 2.2: Figure shows the dataset and its collection setup at the Pittsburgh-Butler Regional Airport (KBTP)—a non-towered GA airport that serves as a primary location for the dataset. Lighter color indicates lower altitude. (a) Shown is a snippet of the processed dataset with aircraft trajectories showing clear lobes for traffic patterns for both runways. (b) The left traffic pattern and nomenclature for the runways at the airport. (c) Picture of the data-collection setup.

capabilities.

Airports can have multiple agents using the same airspace where all agents are expected to follow socially-compliant trajectories while loosely following the traffic pattern guidelines. Using a diverse set of inputs such as radio, transponder data, weather, and vision, each pilot constructs their airspace situational awareness to decide an entry or exit from the pattern. To avoid collisions, therefore, it is critical for each pilot to predict the future trajectory predictions of other agents in that airspace.

In this work, we address the problem of learning how to predict the actions of other agents (aircraft) in a static environment with the dynamic context needed to navigate an aircraft in non-towered terminal airspace safely.

This work presents a novel dataset, *TrajAir*, that provides recorded trajectories of multiple aircraft operating around a standard non-towered airport while also providing the weather conditions during these operations. While several datasets exist for ground navigation or pedestrian trajectories, few datasets are available for aerial navigation that imposes unique challenges including the need to handle regulations and weather conditions.

This work also provides a baseline trajectory prediction method, *TrajAirNet*, to model agent-agent, agent-environment, and agent-context interactions. The future trajectories of agents around an airport are a function of the history of all the participating heterogeneous agents, their respective goals, the static airport environment, and the dynamic weather context. The proposed method uses the dataset to effectively capture all aspects of this behaviour to predict aircraft trajectories.

The major contributions of this work are as follows:

1. We provide the first publicly-available large-scale processed trajectory and weather data with multiple aircraft socially interacting in a non-towered GA airport.
2. We provide an open-source end-to-end attention-based baseline method to predict socially-aware multi-future trajectories in a static environment with a dynamic context.

2.2 Related Work

2.2.1 Aircraft Trajectory Prediction

Previous work on aircraft trajectory prediction can be split into macro-level predictions in high-altitude en-route airspace and micro-level predictions for terminal airspace. En-route long-range predictions are often a function of weather conditions [13, 133, 58, 132]. For terminal airspace, previous work has focused on larger airports with commercial aviation (CA) traffic. CA aircraft usually follow strict approach procedures for entering and exiting terminal airspace and are often guided by air traffic control, making the trajectory prediction problem akin to learning the statistical variation in following these procedures [193, 18, 7]. Non-towered terminal airspace, on the other hand, while still following FAA guidelines, has a much higher complexity and often expects air traffic to follow socially compliant behaviour. The lack of a centralised authority gives the pilots more freedom with the choice of actions to achieve a particular goal. To the best of the authors' knowledge, no publicly available dataset or open-source trajectory prediction methods exist for non-towered GA traffic, which forms the bulk of the aviation infrastructure.

2.2.2 Trajectory Prediction Datasets

The majority of the research in socially-aware human-robot interaction has focused on pedestrians and autonomous vehicles (AVs) [43, 110, 111, 158, 192]. The arena is often crowded spaces like shopping malls, college campuses, or city streets. Pedestrian datasets like UCY [90], ETH [141], and the Stanford Drone Dataset [154], among others [155], have long been the dominant datasets for evaluating pedestrian trajectory prediction tasks. Within the AV domain, Argoverse [33], KITTI [62] and nuScenes [28] have been more popular.

With *TrajAir*, we introduce the new 3D domain of general aviation within the paradigm of social navigation. The dataset differs from previously published datasets due to the spatial dependence of the agent trajectories within a static environment. *TrajAir* trajectories are conditioned not only on the relative location of agents but also on their absolute locations. In addition, trajectories are explicitly goal-directed. This enables benchmarking models that use goal- or intention-driven predictions. Due to the FAA guidelines, the trajectories loosely follow the same semantic structure opening up the field of structured predictions in social navigation. Another point of difference is the presence of a global context in the form of weather data directly affects the trajectories and goals of all agents. This allows benchmarking algorithms that can use contextual clues to aid social behaviour prediction.

2.2.3 Trajectory Prediction Algorithms

Social trajectory prediction algorithms typically use three separable modules to generate trajectory predictions. A sequential module like recurrent units [89] or convolutions [130] is first used to encode the observed trajectories of each agent to generate a vector in latent space. A social module then uses a method like pooling [5] or attention [119] to encode the social context. A generative module then decodes the socially-aware representation into an output that is either the relative coordinates [89], a distribution on the relative coordinates [197] or accelerations [149]. Due to these algorithms' the domain specific nature they do not generalize well to GA aircraft. The use of relative or ego-centric coordinate systems, suitable for pedestrians and AV domains, renders these algorithms unsuitable for domains like GA, where the decisions are a function of the absolute spatial coordinates. While algorithms like Trajectron++[158] have been proposed to support domain independence with support for dynamics, its extension to 3D space, airport maps and double-integrator dynamics is non-trivial. The proposed baseline method, *TrajAirNet*, which uses absolute coordinates and global contexts, builds on similar structuring to provide a trajectory prediction algorithm that combines the state-of-the-art in each of the aforementioned modules.

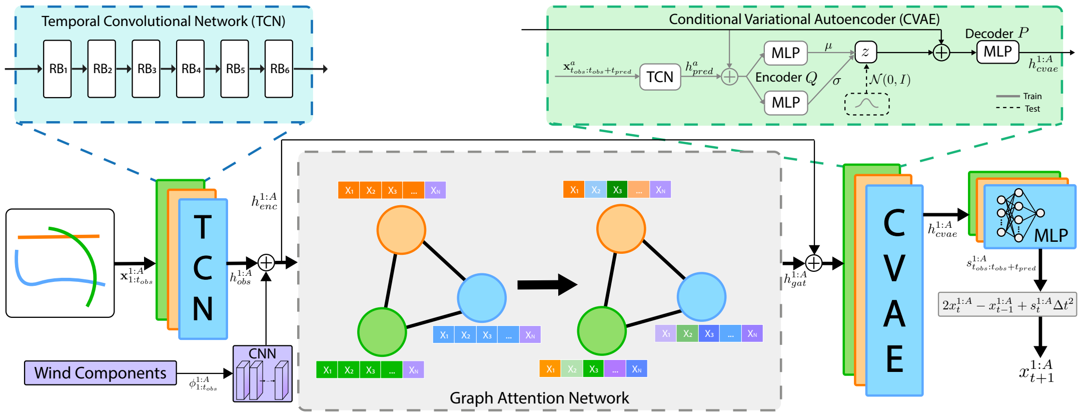


Figure 2.3: Our proposed *TrajAirNet* baseline model for aircraft trajectory prediction in static environment with a dynamic context. The model uses Temporal Convolutional Networks (TCNs) to encode the 3-D trajectory. The dynamic weather context (wind vectors) are encoded using Convolutional Neural Networks (CNNs), which are appended to the encoded trajectory. To encode the social context, we use a Graph Attention Network (GAT) that uses attention to combine data from different agents. Finally, we use Conditional Variational Autoencoders (CVAEs) to produce multi-future acceleration commands, which are then used in a forward Verlet integration to produce future aircraft trajectories for all agents.

2.3 *TrajAir* Dataset

The *TrajAir* dataset is collected at the Pittsburgh-Butler Regional Airport (ICAO:KBTP), a single runway GA airport, 10 miles North of the city of Pittsburgh, Pennsylvania. Additional information about KBTP is available online[1]. Aircraft entering and leaving non-towered airspace need to follow guidelines established by the FAA to ensure the safety and efficiency of all participating agents. KBTP has Left Traffic patterns for both runways, meaning the patterns are rectangular-shaped with left-handed turns relative to the direction of landing or take-off. Figure 2.2b shows the traffic pattern for Runway 8 and 26 around KBTP with the corresponding direction of traffic flow. Aircraft usually take-off or land into the wind; hence the nomenclature follows this sequence. When an aircraft takes off, it is on an upwind leg. A left turn puts it on a crosswind leg, followed by turns into downwind leg and base leg. The final left turn puts the aircraft on the final leg for a touch-down. FAA also establishes that an entry into the pattern should be at a 45-degree angle to the downwind leg.

[1] http://www.airnav.com/airport/kbtp

2.3.1 Dataset Overview

The trajectory data is recorded using an on-site setup (see Figure 2.2c). Data is provided starting on 18 Sept 2020 and continues till 23 Apr 2021. It includes a total of 111 days of data discounting downtime, repairs and bad weather days with no traffic. Data is collected from 01:00 AM local time to 11:00 PM local time. The dataset can be accessed at https://theairlab.org/trajair/. More information about the dataset, including the file structure and dataloaders, is also provided.

2.3.2 Trajectory Data

The dataset uses an Automatic Dependent Surveillance-Broadcast (ADS-B) receiver [44] placed within the airport premises to capture the trajectory data. The ADS-B In receiver receives data directly broadcasted by other aircraft with ADS-B Out. For aircraft that do not have an ADS-B Out, the Traffic Information Service-Broadcast (TIS-B) takes the position and altitude of aircraft using radar and converts that information into a format that's compatible with ADS-B. It then broadcasts the information to our receiver. The receiver uses both the 1090 MHz and 978 MHz frequencies to listen to these broadcasts. The ADS-B uses satellite navigation to produce accurate location and timestamp for the targets, which is recorded on-site using our custom setup.

2.3.3 Weather Data

The weather data is obtained post-hoc using the METeorological Aerodrome Reports (METAR) strings generated by the Automated Weather Observing System (AWOS) system at KBTP. We use the Iowa State METAR repository [71] to gather all the weather data during the trajectory collection time frame. The raw METAR string is then appended to the raw trajectory data by matching the closest UTC timestamps.

2.3.4 Data Processing

The data obtained from the ADS-B receiver and the METAR strings is processed to make it suitable for training networks. The following steps are performed:

- Removal of data points that have corrupt or no location fields.

- Removal of duplicate data points with the same aircraft identifier and location fields.
- Removal of data points where the altitude is more than 6000 feet MSL, and distance is more than 5 km from one end of the runway.
- Transforming the data to a local Cartesian coordinate frame in SI units. The origin is at the end of the runway, with the horizontal x-axis pointing along the runway.
- Processing raw METAR strings to get wind velocity and direction along and across the runway in the local Cartesian frame in SI units.
- Interpolating trajectory data every second for all agents.
- Segmenting the data into scenes with at least one active aircraft in the airport vicinity.

The raw data and processed data are provided as part of the data release. A snippet of the processed data is shown in Fig 2.2a. Lighter color indicates lower altitude. Both left-hand patterns are visible as distinct lobes on both sides of the runway. The variability in following the pattern is also evident, as are the trajectories before entering and after leaving the pattern.

2.4 *TrajAirNet* Prediction Network

We propose *TrajAirNet*, an end-to-end network to provide aircraft trajectory predictions within the terminal airspace. The proposed baseline method provides us with the ability to do the following: 1) condition the prediction on ego agent's absolute position history in the static environment, 2) condition the prediction on the absolute position of other agents within the terminal airspace weighted according to their relative importance to the ego-agent, 3) condition the prediction on the dynamic environmental context like wind, 4) predict multi-modal multi-future trajectories of all the agents, and 5) operate invariant to the number of agents in the scene without using zero-padding. To achieve this, the method uses multiple components in an end-to-end fashion using a combined loss function. Section 2.4.1 defines the nomenclature and provides a formal mathematical definition of the problem. Section 2.4.2 provides details on each model component, and Section 2.4.3 provides details on the implementation and the loss function.

2.4.1 Problem Formulation

We consider the problem of predicting the multi-modal distribution of future trajectories of all the agents given the past history and environmental context. Let $\mathbf{x}_t^a = (x_t^a, y_t^a, z_t^a)$ and ϕ_t^a denote the position and context of the a^{th} agent at time t, respectively. While the context may include terrain, maps, weather and wind conditions; for the purposes of this work, we only focus on wind as the context. Let $\mathbf{x}_{t_1:t_2}^{1:A}$ and $\phi_{t_1:t_2}^{1:A}$ denote the trajectories and context over a $\{t_1, \ldots, t_2\}$ time-horizon for all $\{1, \ldots, A\}$ agents in that scene. We then define our trajectory prediction problem as finding distribution of future trajectories $\hat{\mathbf{x}}_{t_{obs}:t_{pred}+t_{obs}}^{1:A}$ conditioned on the past trajectories $\mathbf{x}_{1:t_{obs}}^{1:A}$ and context $\phi_{t:t_{obs}}^{1:A}$, where, t_{obs} is the observation time window, and t_{pred} is the prediction time horizon. Mathematically,

$$\hat{\mathbf{x}}_{t_{obs}:t_{pred}+t_{obs}}^{1:A} \sim p(\hat{\mathbf{x}}_{t_{obs}:t_{pred}+t_{obs}}^{1:A} \mid \mathbf{x}_{1:t_{obs}}^{1:A}, \phi_{t:t_{obs}}^{1:A}) \tag{2.1}$$

2.4.2 Model Details

We propose a baseline trajectory prediction algorithm, *TrajAirNet*, that takes as input the past trajectories of all the agents, along with the weather context, to predict multi-future trajectories of all agents. The network combines elements from Temporal Convolutional Networks (TCNs), Graph Attention Network (GATs), and Conditional Variational Autoencoder (CVAEs). Each component addresses various needs of the underlying problem. The entire architecture is shown in Figure 2.3.

The TCN layers are used to encode the spatio-temporal trajectory into a latent vector without losing the causal relations in the underlying data. This latent representation can be used for other downstream tasks like trajectory prediction. While LSTMs have been a popular choice to encode the trajectory space, our choice of TCNs is largely because they have been shown to perform better or at least as good as LSTMs [15].

The GAT layer is used to encode the influence of other agents on the predicted trajectory of a particular agent. While max-pooling has been a traditional choice to incorporate the effect of other agents, the lack of interpretability in the latent encoded output makes max-pooling a rather black-box choice [197]. Using an attention mechanism as in GATs, on the other hand, has become a popular choice because attention can be directed selectively at particular agents in a scalable manner in terms of the number of agents [179]. This becomes especially important in the shared-airspace domain, where the relative importance of an agent is not only based on their proximity to the ego agent but also on their absolute location in space. Another advantage of using GATs is that it makes the algorithm permutation-invariant as well as invariant to the number of agents without a need to pad zeros.

The CVAE layer serves as the backbone of the multi-future trajectory prediction. There is inherent stochasticity in the data, and CVAEs are well suited to capture this distribution. They have been used successfully in both pedestrian and AV trajectory prediction networks [158]. The CVAEs encode the underlying distribution in its latent space, which can then be sampled at run-time to provide samples in the trajectory space. In order to incorporate the dynamics of the vehicle, we decided to a generalized dynamics formulations in the form of a forward Verlet integration [149] that provides a constant velocity model for a zero output. Instead of predicting the positions directly, we predict the Verlet acceleration and then calculate the absolute positions of each agent.

2.4.3 Implementation Details

TrajAirNet codebase is available open-source at https://github.com/castacks/trajairnet. A modified dataloader breaks the scene into sequences of length $t_{obs} + t_{pred}$ with a certain minimum number of agents constant across each sequence. The number of agents can change from sequence to sequence. For each agent in a given scene, the raw trajectory in absolute coordinates is encoded using the same TCN layers.

$$h^a_{obs} = TCN_{obs}(\mathbf{x}^a_{1:t_{obs}}) \ \forall \ a \in \{1, \ldots, A\} \tag{2.2}$$

To include the environmental context, wind velocities along and across the runway in our case; the raw context is encoded using a standard CNN layer and the output is concatenated to the TCN encoded trajectories. The choice of CNN over MLP was motivated to enable spatial contexts in later revisions of the algorithm.

$$h^a_{enc} = h^a_{obs} \oplus CNN(\phi^a_{1:t_{obs}}) \ \forall \ a \in \{1, \ldots, A\} \tag{2.3}$$

The concatenated encoded trajectories and encoded context for all agents are then stacked together as the input to the GAT layers. Each agent acts as a node in the GAT graph structure. We use a standard GAT structure with multi-head attention [178].

$$h^{1:A}_{gat} = GAT(h^{1:A}_{enc}) \tag{2.4}$$

The output for each agent from the GAT is then segregated and concatenated with the full encoded output h^a_{enc} for each agent a. This concatenated vector then serves as a conditional input to the CVAE layers. The CVAE is characterized by an encoder $Q(\cdot)$ and a decoder $P(\cdot)$. For all $a \in \{1, \ldots, A\}$,

$$\begin{aligned} h^a_{pred} &= TCN_{pred}(\mathbf{x}^a_{t_{obs}:t_{obs}+t_{pred}}) \\ z &\sim Q(z \mid h^a_{pred}, h^a_{enc} \oplus h^a_{gat}), \quad \text{for training} \\ z &\sim \mathcal{N}(0, I), \quad \text{for testing} \\ h^a_{cvae} &\sim P(h^a_{cvae} \mid z, h^a_{enc} \oplus h^a_{gat}) \end{aligned} \tag{2.5}$$

Finally, the sampled CVAE output is passed through a MLP layer to get the correct dimension for the acceleration output.

$$s^a_{t_{obs}:t_{obs}+t_{pred}} = MLP(h^a_{cvae}) \tag{2.6}$$

The acceleration output is then converted to absolute positions for all $t \in \{t_{obs}, \ldots, t_{obs} + t_{pred}\}$, using,

$$x^{1:A}_{t+1} = 2x^{1:A}_t - x^{1:A}_{t-1} + s^{1:A}_t \Delta t^2 \tag{2.7}$$

The entire pipeline uses a combination of loss functions.

$$\mathcal{L}_{total} = \mathcal{L}_{traj} + \mathcal{L}_{cvae} \tag{2.8}$$

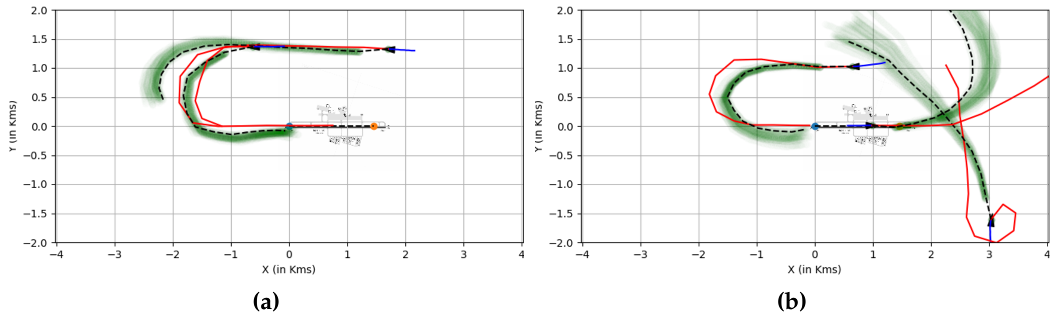
The $\mathcal{L}_{traj}$ measures how close the predicted trajectory is to the ground-truth trajectory using a mean squared error (MSE) loss.

$$\mathcal{L}_{traj} = MSE(\mathbf{x}^a_{t_{obs}:t_{obs}+t_{pred}}, \hat{\mathbf{x}}^a_{t_{obs}:t_{obs}+t_{pred}}) \tag{2.9}$$

The $\mathcal{L}_{cvae}$ measures the KL-Divergence between the sampling distribution of the latent variable, the easiest choice being $\mathcal{N}(0, I)$, to the distribution of latent variable that we learn during training.

$$\mathcal{L}_{cvae} = D_{kl}(Q(z \mid h^a_{pred}, h^a_{enc} \oplus h^a_{gat}) \mid\mid \mathcal{N}(0, I)) \tag{2.10}$$

For training, we use the Adam optimiser with a learning rate of $1e - 4$.



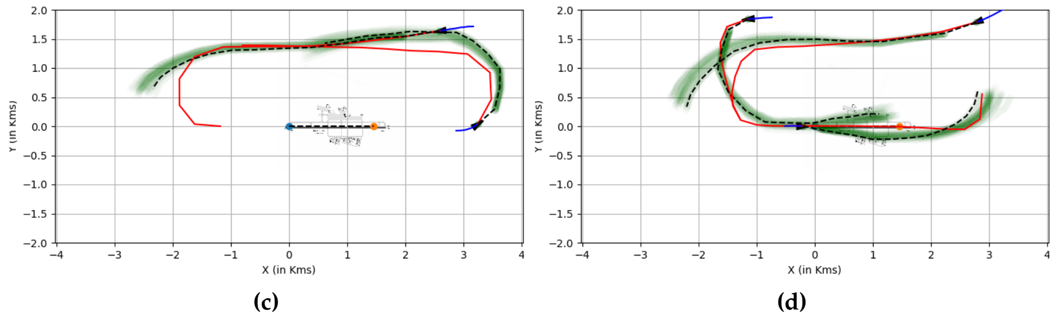


Figure 2.4: Figure shows the qualitative results for the *TrajAirNet* framework. Four independent scenarios are chosen to showcase the performance. Blue is the observation trajectory (t_{obs} = 11 sec). Green are the sampled trajectories from the *TrajAirNet* output. Black shows the output closest to the ground truth (t_{pred} = 120 sec), which is shown in Red. Also shown is the airport diagram to scale.

2.5 Evaluations and Discussion

Evaluations for the proposed network are carried out on a 28 day subset of data from the *TrajAir* dataset that contains 4-sets of 7 consecutive days of data in different months. Results from our network along with comparative methods are shown for all four sets. In order to focus on long-horizon predictions, we use $t_{obs} = 11$ sec and $t_{pred} = 120$ sec. The choice of the observation and prediction horizons are chiefly motivated to match the scale of decision horizons in GA. We compare our results with the following baselines:

1. Constant Velocity [158]: Trajectories are predicted by setting acceleration to zero in the Verlet integration.
2. Nearest Neighbour: We use a nearest neighbour search to find the closest absolute trajectory in the training set to the queried trajectory using an L2 metric.

Algorithm	**7Days-1**	**7Days-2**	**7Days-3**	**7Days-4**
Const. Vel. [158]	1.79/4.08	1.90/4.31	1.92/4.30	1.82/4.16
Nearest Neigh.	3.13/2.70	1.92/1.99	3.41/2.69	2.59/2.58
STG-CNN [119]	1.19/2.35	1.36/2.70	1.33/2.67	1.17/2.29
TransformerTF [64]	1.58/3.85	1.69/4.10	1.97/4.36	1.79/4.19
TrajAirNet (ours)	**0.73/1.42**	**0.81/1.63**	**0.86/1.72**	**0.71/1.41**

Table 2.1: Table shows the quantitative results ADE/FDE (in Km) for *TrajAirNet* baseline along with comparative results.

3. STG-CNN [119]: Uses a spatio-temporal graph convolutional neural network for trajectory prediction.

4. Transformer-TF [64]: Standard transformer implementation for trajectory prediction.

We chose not to include extensive comparative results from other social trajectory prediction algorithms. Our experiments found that the top-performing algorithms for pedestrian and AV benchmarks either exploit the problem's underlying structure or require non-trivial modifications, making them unsuitable for a domain transfer. A simple change in hyper-parameters very often did not provide optimal performance as can be seen from the short comparative result section. We use two popular metrics [197] for evaluating the performance of the proposed network: Average Displacement Error (ADE) and Final Displacement Error (FDE). Results for the best of N trajectories are used where the network is queried N times, and the best ADE/FDE scores are recorded. We nominally use N = 5.

2.5.1 Results and Discussions

Figure 2.4 shows the qualitative results for the *TrajAirNet* framework. Four independent scenarios are showcased to highlight the various behaviours captured by *TrajAirNet*. Figure 2.4(a) shows a scenario with two agents on the downwind leg of the pattern. The algorithm correctly predicts a longer turn to base for the trailing aircraft to improve spacing between agents. Figure 2.4(b) shows three aircraft with one aircraft trying to enter the pattern. While the algorithm correctly predicts the entry, it fails to predict the entering aircraft's roundabout turn to improve spacing. This highlights the diversity of maneuvers in the dataset and the difficulty in predicting them. The third scenario shows an aircraft entering downwind with another trailing aircraft already in the pattern. The network predicts that the trailing aircraft extends the crosswind leg and falls in line behind the aircraft entering the pattern. Lastly, the fourth scenario in Figure 2.4(d) shows three aircraft in the traffic pattern around the airport with the aircraft on downwind-to-base turning early to improve spacing. Table 2.1 shows the quantitative results for the *TrajAirNet* framework. As can be seen, *TrajAirNet* dominantly outperforms prediction baselines across all the sub-datasets.

2.6 Conclusions

This work presents a novel dataset for socially-aware trajectory prediction in the aviation domain. Additionally, it also presents a baseline trajectory prediction algorithm for this static environment with a dynamic context. To the best of the authors' knowledge, this is the first publicly available dataset and method in the domain of general aviation. A major motivation of this work is to open up this domain to the wider robotics and automation community. With the recent advances in self-driving and social robotics, this publication aims to encourage a more in-depth look into the similar and unique problems faced by the autonomous aviation community and propose solutions in this relatively under-explored domain.

Chapter 3

TartanAviation : Multi-modal Datasets for Terminal Area Operations

3.1 Background

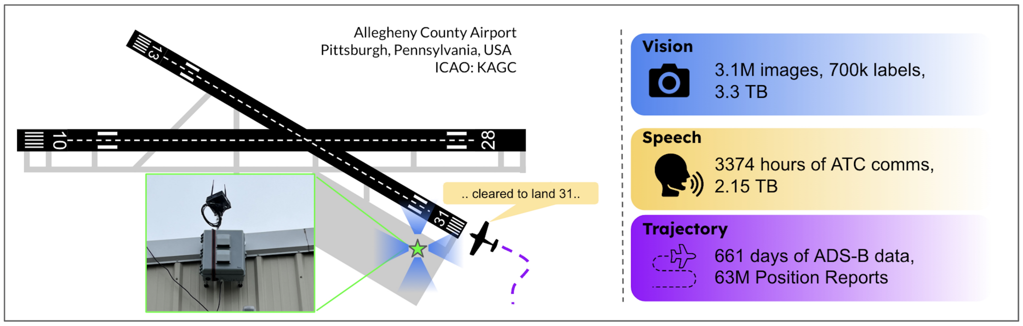

Allegheny County Airport
Pittsburgh, Pennsylvania, USA
ICAO: KAGC
13
10
28
31
.. cleared to land 31..
Vision
3.1M images, 700k labels, 3.3 TB
Speech
3374 hours of ATC comms, 2.15 TB
Trajectory
661 days of ADS-B data, 63M Position Reports

Figure 3.1: Our custom data collection setup installed at the Allegheny County Airport with its approximate location within the airport premises with respect to the runway geometry. The setup recorded images, audio, and aircraft trajectory data.

In 2023, an average of 45,000 flights per day served over 2.9 million passengers in the US [3], which is nearly at the saturation capacity of the current US air traffic control system [120]. The expected introduction of Advanced Aerial Mobility (AAM) operations within the National Airspace System heralds a further increase in flight operations, necessitating a need to increase the airspace capacity [10]. Compared to en-route airspace, terminal areas experience a higher air traffic density as nearly all aerial operations start or end within these areas. The effective future management of this high-risk airspace necessitates insights into various aspects of air traffic management. In addition to manned AAM operation, data is needed to achieve advances in fully or partially autonomous AAM agents. Curated datasets within this domain not only enable analytical research into understanding how the system is currently managed but also help accelerate the development of novel technologies for future operations. More generally, the aviation domain presents newer challenges to widely applied technologies like vision-based object detection, speech-to-text translation, and time-series analytics. Advancements in AI and machine learning can potentially revolutionize air traffic control systems, ensuring safer and more efficient coordination for the ever-increasing number of flights.

In this work, we introduce TartanAviation, a multi-modal dataset collected at towered and non-towered terminal areas within the US. The TartanAviation dataset covers three primary concurrently collected modalities of data: trajectory positions for capturing the spatial and temporal information of aircraft movements, video flight sequences collected with static cameras installed within terminal areas, and audio communications to document the voice interactions between pilots and air traffic controllers. While prior datasets in the aviation domain have focused on specific modalities like speech [99, 200] or vision [167, 4], TartanAviation aims to provide a more holistic view of terminal airspace operations across various data modalities. Additionally, while previous datasets focus on large commercial airports, TartanAviation focuses on smaller regional airports within the Greater Pittsburgh area. Regional airports serve a multitude of different aircraft and mission profiles, providing a richer and more diverse data stream. We specifically focus on two airports: the towered Allegheny County Airport (ICAO:KAGC) and the non-towered Pittsburgh-Butler Regional Airport (ICAO:KBTP).

Human vision is the last line of defense against mid-air collisions, making it critical for aviation safety [185]. With recent advances in computer vision technologies, visual detect-and-avoid (DAA) systems have shown promising results in detecting airborne objects at greater distances [63]. Prior datasets for DAA either lack diversity [4], are computer-generated [167], or are relatively small [52, 113]. In this context, the TartanAviation vision dataset is a large-scale real-world dataset collected by placing static cameras within the terminal area. TartanAviation currently offers 3.1 million images covering challenging scenarios like snow, mist, rain, varying cloud cover, and diverse aircraft types. The data is augmented with the associated weather and air traffic trajectory data. With over 700k aircraft labels, these challenging real-world scenarios promise to enable more research in robust computer vision techniques for long-range object detection.

Trajectory data represents the time-series information of aircraft positions operating within the terminal airspace. This data is collected by recording Automatic Dependent Surveillance-Broadcasts (ADS-B) using receivers placed within the terminal area. ADS-B is an aviation surveillance technology where aircraft automatically broadcast their position and other data using GNSS satellite navigation, enabling tracking by air traffic control and nearby aircraft without ground interrogation. It is a key component of global air traffic systems and is mandatory in several countries for enhanced safety and efficiency. Prior studies have looked into the performance of ADS-B including its accuracy, integrity, continuity and availability [6] as well as its use in sequencing the optimal landing order for arriving flights.[188] Our prior work, TrajAir [136], provided 111 days of ADS-B data at KBTP. TartanAviation extends this tally to a total of 661 days of data at both KBTP and KAGC. Beyond the aviation domain, trajectory data can enable more research in time-series forecasting, social trajectory prediction, and anomaly detection.

Radio communications enable pilots and Air Traffic Controllers (ATC) to share time-sensitive intent and instructions over dedicated frequencies. The information included in these communications helps pilots and ATC to build situational awareness, enabling the safe operation of aircraft on the ground and in the air. Radio communication within aviation use both High-Frequency (HF) as well as Very-High-Frequency (VHF) bands. While HF is mostly used for long-range communication, for example in oceanic flights, VHF is preferred for short to medium range communication. A VHF radio operates on frequencies between 118.0 megahertz (MHz) and 136.975 MHz [189]. While most speech datasets are unimodal [65], recent multi-modal datasets with text and trajectory [67] have focused exclusively on commercial aircraft operations at large airports. TartanAviation provides first-of-its-kind speech data at relatively smaller airports, including both towered and untowered airfields. Along with the diversity in speech data, TartanAviation also provides concurrent trajectory data, enabling novel research in multi-modal speech-to-text translation and speech-to-intent predictions [168] conditioned on external context.

All of the data in TartanAviation was collected with administrative support and prior authorization at both the KAGC and KBTP airports. TartanAviation represents our

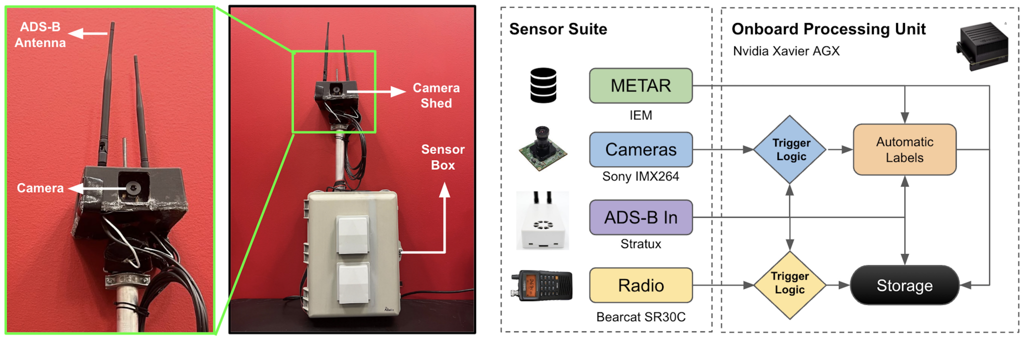


Figure 3.2: Our custom setup hardware with the camera and ADS-B antenna mounts *(left)*. We also showcase the data collection pipeline with the associated sensor suite and automatic logic that triggers camera and speech recordings *(right)*.

first step at building a centralized comprehensive multi-modal repository for aircraft data within the terminal airspace.

3.2 Methods

This section briefly overviews the equipment and protocols used to collect the datasets. Figure 3.2 shows the setup and an overview of the data collection pipeline. An waiver from Carnegie Mellon University IRB was obtained prior to data collection.

3.2.1 Vision Data Acquisition

The vision data was exclusively collected at the KAGC airport. The setup was mounted at 40.351422 latitude, -79.923939 longitude, as shown in Figure 7.1. The vision setup uses an array of Sony IMX 264 cameras connected to a NVIDIA Xavier AGX 32GB dev kit via a LI-JXAV-MIPI-ADPT-6CAM camera adaptor. The cameras collect data at 2048×2448 resolution at 24 Frames Per Second (FPS). The camera recordings were triggered using ADS-B data by enforcing an 8 km fence around the setup. If an aircraft crosses the threshold, the camera automatically starts recording. The recording stops either at the end of a 250-sec timer or if the aircraft exits the 8 km fence. A new recording is activated if the recording times out before the aircraft exits. This process repeats till the aircraft leaves the geo-fence. To ensure seasonal diversity, data collection occurred in multiple phases from December 14, 2021, to February 23, 2023, from sunrise to sunset.

Post-Processing

As shown in Table 3.2, along with the raw camera output, TartanAviation offers various useful meta-data, including ADS-B and bounding box labels.

For ADS-B, each recorded video sequence has a PKL file that contains the GPS coordinates of all flying aircraft in the area at the specific time of recording. The ADS-B data also provides the unique flight ID and N-number (if provided by the pilot to the ADS-B) per aircraft. The unique flight ID and the N-number of the aircraft are used to parse the FAA N-number registry inquiry website. Furthermore, we obtain the ADS-B data and interpolate it to obtain ground truth information matching the frequency of the camera recording, i.e., 24 Hz.

For the bonding box labels, each recorded video sequence has a zip file containing the label files for individual video frames. The labels are text files, where each row provides the bounding box information for a flying aircraft in the video. In particular, the label files include the bounding box coordinates for the aircraft in the image, a unique track ID for each aircraft, the range of the aircraft, the manufacturer, type, and model.

We leverage two main modules to generate the bounding box labels: (a) Auto-labeling and (b) Manual label verification/generation. The Auto-labeling module is performed in two stages. The first stage uses our deep learning-based aircraft detection & tracking system, AirTrack [63], to obtain initial bounding box hypotheses. Despite not having perfect recall, this stage makes labeling bounding boxes easier and reduces the amount of manual labor required downstream. The second stage projects the 3D ADS-B data into the image frame using the calibrated extrinsics to provide an initial estimate of the aircraft position in the image. This projection helps filter false positives from the detector and provides the initial airborne object location to the labeler. Once the auto-labeling procedure is finished, we manually verify the image labels frame-by-frame and make any necessary corrections using our custom labeling software.

The system used in the Auto-labeling module, AirTrack [63], consists of modules for ego-aircraft motion estimation, detection and tracking, and secondary classification to filter out false positives. The inputs are two successive grayscale image frames

$I_t, I_{t-1} \in \mathbb{R}^{H\times W}$ where $H \times W$ are the dimensions of the input frames. We utilize the full image resolution of 2048×2448 during inference to maximize the chances of detection at long ranges. The outputs are a list of tracked objects with bounding box coordinates, track ID, 2D Kalman filtered state, estimated range, angular rate, and time to closest point approach (tCPA). We then filter out false positive detections by projecting the ADS-B data into the image frame and ensuring that the projected ADS-B coordinates lie within a 100 pixel radius of the predicted image bounding box. Whenever we have false negatives from the model, we manually add in the bounding box label if the aircraft is visible in the image.

For the ADS-B data projection, we assume that the 4-camera setup is located at the origin of a north-east-down (NED) world reference frame where the positive z-axis points towards the earth's center and the positive x-axis points towards True North. Given all the relevant information, such as GPS coordinates and altitudes, we can compute the 3D position of aircraft using geodesics. The corresponding image frame location in homogeneous coordinates for camera i ($i = 1, \ldots, 4$) is given by the following perspective projection: $\mathbf{p}_i = \mathbf{K}_i[\mathbf{R}_i \mid \mathbf{t}_i]\mathbf{P}$, where $\mathbf{K}_i$ is a known 3×3 intrinsic matrix for camera i, $[\mathbf{R}_i|\mathbf{t}_i]$ constitutes the 3×4 extrinsic matrix of camera i, and $\mathbf{P} \in \mathbb{R}^3$ is the aircraft 3D position in the world reference frame. In particular, we use the PnP-RANSAC algorithm to estimate $[\mathbf{R}_i|\mathbf{t}_i]$. Using time synchronization, the 2D correspondences for the recorded 3D observations are manually labeled using custom labeling software. Since the setup is static, this is a one-time calibration step that gives us the projection matrix for camera i, with which we can project new 3D aircraft data.

3.2.2 Trajectory and Weather Data Acquisition

The trajectory and weather data are collected at both the KAGC and KBTP airports. Data collection follows similar procedures as our prior ADS-B dataset [136]. We use a Stratux ADS-B receiver capable of receiving position reports on both the 1090 MHz and 978 MHz frequencies. The receiver was installed within the terminal area of each airport. The recording operation ran from 1:00 AM to 11:00 PM local time, a period selected to encapsulate the full range of aviation operations during both peak and off-peak hours. Data collection at KBTP started on September 18, 2020 and concluded on October 27, 2022. Data collection at KAGC began on October 31, 2021 and ended on February 17, 2023. Data was collected in discrete phases, resulting in 381 days or 36 million raw position reports of data at KBTP and 280 days or 27 million raw position reports of data at KAGC. Weather reports for the corresponding airports are collected post hoc in the form of METeorological Aerodrome Reports (METAR) strings. We use the Iowa State METAR repository [71] to compile reports for the duration of the ADS-B data. The total file size of the audio dataset is 1.9 GB compressed and 12 GB uncompressed.

Post-processing: The raw ADS-B data is first post-processed to remove corrupted and duplicate data points. The data is then filtered for altitude and distance from the airport. We nominally chose 6000 ft MSL and a 5 km radius around the airport. Once filtered, the data is transformed from a global to a local Cartesian coordinate frame in SI units with the origin at the end of the runway. The raw METAR strings are also processed to get wind velocity and direction relative to the runway in the local frame. Finally, we interpolate the trajectory data every second for all agents. TartanAviation provides the processed data along with the raw trajectory and weather data.

3.2.3 Speech Data

The speech data is collected at both the KAGC and KBTP airports. The setup uses a Bearcat SR30C radio receiver capable of receiving aviation radio frequencies. For towered KAGC, the radio is tuned to receive 121.1 MHz, the air traffic control tower frequency for KAGC operations. KAGC has an active tower 24 hours a day. For KBTP, the radio was tuned to 123.05 MHz, the Common Traffic Advisory Frequency for KBTP operations. The audio is recorded at a rate of 44.1k samples per second onboard the system. The record trigger mechanism for the speech setup is similar in construction to the vision setup and uses ADS-B data to trigger recordings. The trigger threshold was set to 10 km. The corresponding raw ADS-B data is also provided for the communication recordings spanning multiple years for both airports. The communication data from KBTP starts on September 6 2020 and ends on October 27 2022. After filtering, this data comprises 278 days of data, corresponding to 790 GB of disk space. The communications data for KAGC starts on October 31, 2021 and ends on February 17, 2023. The KAGC data comprises 392 days of data, corresponding to 1358 GB of disk space. In total, the dataset contains 670 days of raw communications recordings. The total file size of the audio dataset is 2.15 TBs uncompressed and 505.2 GB compressed.

Post-processing: We filtered the raw recordings using two conditions: audio clips must be longer than 1 second and have a maximum decibel level above -20 db. This removes erroneous recordings and clips that don't contain any spoken words. The KAGC audio has 33289 audio files comprising 2131.9 hours of audio, with 327.714

Extension	Nomenclature	Content
.zip	`<camera_id>_<timestamp>`	Zip folder containing sequence data
↪ .mp4	`<camera_id>_<timestamp>`	Video File
↪.avi	`<camera_id>_<timestamp>_sink_verified`	Video File with embedded labels
↪.srt	`<camera_id>_<timestamp>_subtitle`	Raw timestamps of the recorded video
↪.pkl	`<camera_id>_<timestamp>_sink_adsb`	Raw ADS-B dictionary
↪.pkl	`<camera_id>_<timestamp>_acft_sink`	Raw aircraft type data
↪.zip	`<camera_id>_<timestamp>_labels`	Zip folder containing the image labels
↪↪.label	`<frame_number>.label`	Text file containing label data

Table 3.1: File structure information for TartanAviation image data

hours above -20 db. The KBTP audio has 8534 files and 1242.9 hours of audio, with 149.9 hours above -20 db. This gives a total of 41823 files, 3374.8 hours of audio, and 477.6 hours of audio above -20 db.

3.3 Data Records

The dataset [134] is stored on open access servers maintained by Carnegie Mellon University School of Computer Science and is available using the download scripts hosted on https://github.com/castacks/TartanAviation.git [**DOI**: 10.5281/zenodo.14699102]. We provide Python scripts to enable data download of each modalities, including support for sample and partial data. These scripts provide a platform-agnostic way to download these datasets. Each of the modalities also have documentation that list any dependencies required to execute the scripts.

Additional information about the raw and processed data for all the modalities is available at https://theairlab.org/tartanaviation/. The following sections present details on the data formats and provide information on file organization.

3.3.1 Image Data

The image dataset is split across 550 independent sequences. We define a sequence as all of the data recorded during a single event where the camera recordings were started and stopped. The vision data folder contains multiple zipped files, each associated with a particular camera recording for that sequence. Each zipped sequence

Raw Data Fields			Processed Data Fields		
Field	**Units**	**Description**	**Field**	**Units**	**Description**
ID	#	ADS-B Aircraft ID	Frame	#	Relative Timestep
Time	HH:MM:SS.ss	Time of observation	ID	#	ADS-B Aircraft ID
Date	MM/DD/YYY	Date of observation	x	km	Local X Cartesian Position
Altitude	Feet	Aircraft Altitude (Mean Sea Level)	y	km	Local Y Cartesian Position
Speed	Knots	Aircraft Speed	z	km	Local Z Cartesian Position
Heading	Degrees	Aircraft Heading	w_x	m/s	Component of wind along the dominant runway
Lat	Decimal Degrees	Latitude of the aircraft			
Lon	Decimal Degrees	Longitude of the aircraft	w_y	m/s	Component of wind across the dominant runway
Age	Seconds	Time since last observation			
Range	km	Distance from airport centre			
Bearing	Degrees	Bearing Angle with respect to North			
Tail		Aircraft Registration Number			
AltisGNSS	boolean	Indicator flag for Altitude Measurement			

Table 3.2: Variable description for the TartanAviation ADS-B trajectory data.

folder has multiple files, as presented in Table 3.1. Further information regarding the nomenclature and file contents is also shown in Table 3.1. In addition to the video files and labels, we also provide ADS-B data for each sequence.

3.3.2 Trajectory and Weather Data

TartanAviation provides both raw and processed data for each airport. Raw data is separated into individual folders for each day of collection. Each raw data folder has CSVs with fields detailed in Table 3.2. The processed files are available as comma-separated TXT files with fields described in 3.2.

3.3.3 Speech Data

Both the raw and filtered audio files are included in the dataset. The filtered data is organized in a directory structure by location, year, month, and day. Each day is individually zipped, contains audio files in the WAV format, and has an accompanying text file that contains the audio clip's start, end, and total time.

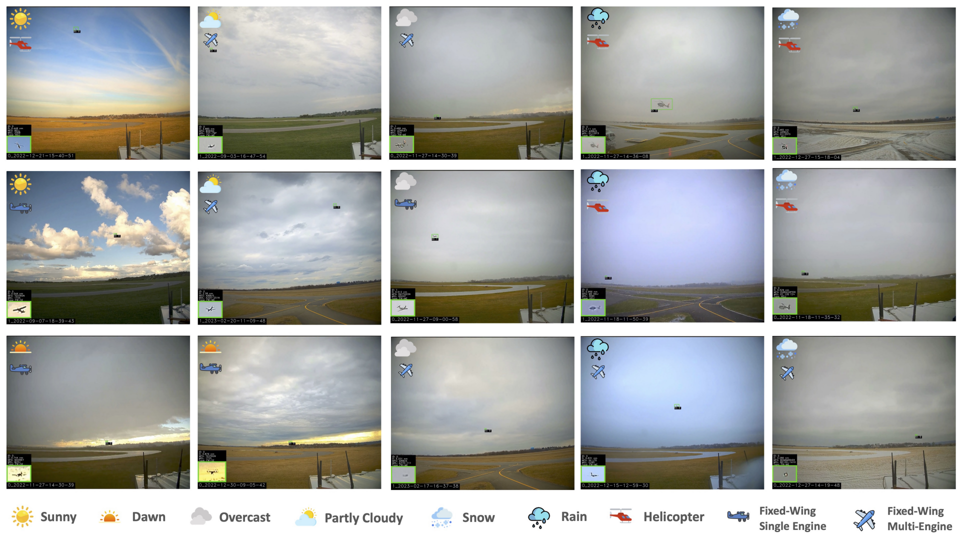


Figure 3.3: Qualitative samples from the TartanAviation Image dataset showcasing the diversity of the collected images in different lighting conditions, seasons, cloud covers, and aircraft types.

3.4 Technical Validation

The data collected were assessed to ensure the reliability of the data provided for each modality individually.

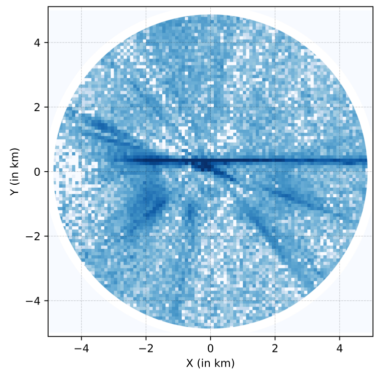


Figure 3.4: Log-normed trajectory histograms from ADS-B aircraft position reports at Allegheny County Airport.

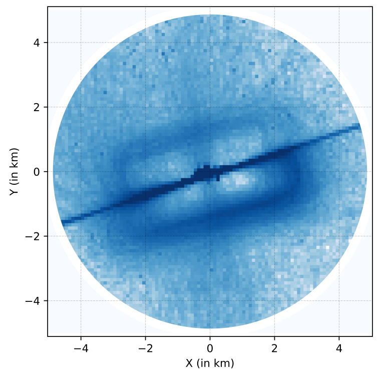


Figure 3.5: Log-normed trajectory histograms from ADS-B aircraft position reports at Pittsburgh-Butler Regional Airport.

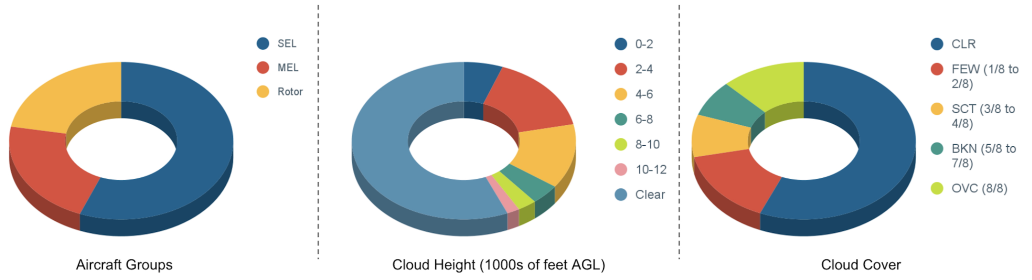


Figure 3.6: Quantitative diversity from the TartanAviation Image dataset showcasing the distribution of the collected images with respect to aircraft groups, cloud heights, and cloud cover.

3.4.1 Image Data

Figure 3.3 shows example images from the dataset. The subset highlights the variation in lighting conditions, seasons, cloud covers, cloud heights, and aircraft types. The tightly labeled bounding boxes provided with each image are also shown. The scale of the bounding boxes with respect to the total image size highlights the challenge faced by object detection algorithms trying to reliably detect aircraft in a cluttered background. Figure 3.6 shows quantitative results for the entire dataset. To showcase the diversity in aircraft, we group the images based on single-engine land (SEL), multi-engine land (MEL), and Rotorcraft (rotor). This shows an almost equal distribution in the dataset with SEL as the major class with 56%. Further, we group the images based on the recorded cloud height. Cloud height has a direct impact on the available lighting. While 56.4% of the images have no clouds in them, 5.6% of images have very low cloud layers below 2000 feet above ground level (AGL). Grouping the images by cloud coverage, we observe that while 56.4% of data has no clouds, 12.2% of data has overcast skies. In addition to this, 6.4% of data has active precipitation while 12.4 % of data has visibility less than 10 statute miles.

3.4.2 Trajectory Data

Figure 3.5 and Figure 3.4 showcase trajectory histograms for the KAGC and KBTP Airports. The histograms represent the aircraft occupancy frequencies on a log scale around both airports. The airport is at the center in both images, with the geographic north pointing upwards. The effect of runway geometries is clearly visible. Figure 3.5 shows the 08/26 KBTP runway while Figure 3.4 shows the crossing smaller 13/31 and

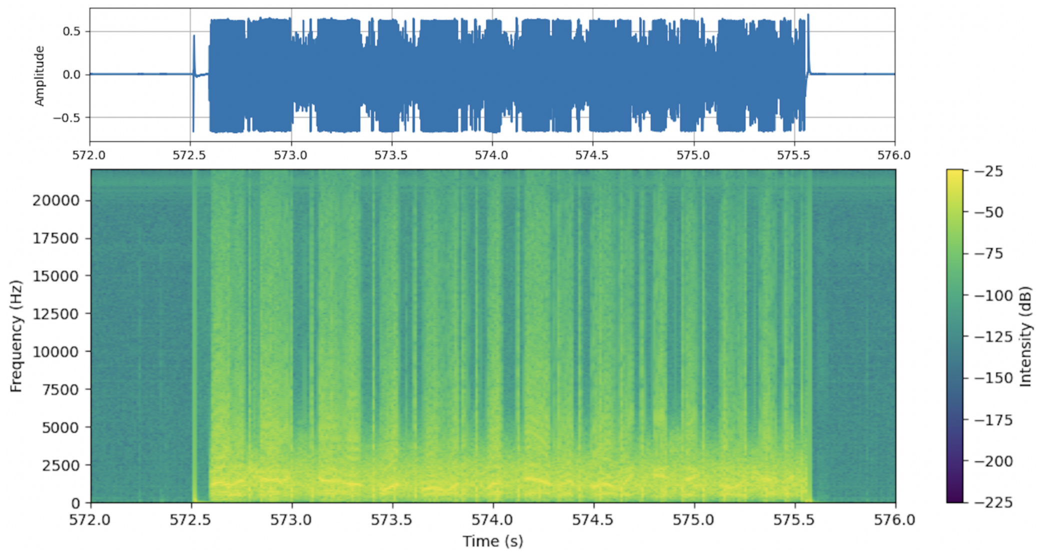


Figure 3.7: A portion of the spectrogram and waveform for audio file 6.wav from KBTP on November 2, 2020. The spectrogram shows the frequency of the audio signal versus time. In the time period of this figure, it is clear when the pilot is speaking, which can be seen as the higher intensity portion of the spectrogram and the spikes in amplitude in the waveform.

larger 10/28 runways at KBTP. Also clearly visible are the different types of operations at both airports. KBTP is an un-towered airport and home to a few flight schools. The left traffic patterns for both runways are clearly visible, highlighting the adherence to FAA guidelines when operating in an un-towered airfield. KAGC, on the other hand, is a towered airfield that hosts medical evacuation helicopters and business jet traffic in addition to flight schools. This leads to more straight-line arrivals and departures from the airport runways, as reflected in the figure.

3.4.3 Speech Data

Our filtering of the audio data ensures that empty files are discarded while any that could contain speech data are included. The threshold of -20 db was chosen to preserve any audio louder than radio silence. Figure 3.7 shows a portion of the spectrogram and waveform of one of the audio files from our dataset, specifically 6.wav from KBTP on November 2, 2020. It is apparent in the spectrogram where the pilot is speaking and where there is radio silence. Furthermore, some low-intensity noise during radio silence occurs at clear intervals and shows up as vertical lines with even spacing. During the speaking portion of the audio, there is a clear spike in intensity between all bands, with the highest being between 0-3 kHz. The human ear is highly sensitive in frequencies between 1-4 kHz, with the sensitivity dropping off steeply to the limit of around 20 kHz [166]. Research in the interpretability of speech finds that the values below 4 kHz yield high accuracy in articulation [59]. The spectrogram shows these vital frequencies for speech data are recorded sufficiently.

Chapter 4

Amelia-48: Large Scale Terminal Surface Movement Dataset

4.1 Background

Surface area operations refer to aircraft and vehicle movements on or near the airport surface. In this work, we restrict the surface movement definition to include aircraft landing, taking off, or taxiing within the marked airport movement area along with vehicles. We are interested in predicting agent behavior within this area and therefore, we aim to capture any aircraft and vehicle trajectory that enters it. The scope ends when the agent either exits the area or crosses the non-movement boundary markings on the airport surface. To use the collected data for training a trajectory forecasting model, we transform the raw position reports into smooth trajectories that include agent metadata. The processed data format is compatible with existing dataloaders for motion forecasting.

NASA's Sherlock Data Warehouse [9] is a comprehensive aviation data platform developed by the Aviation Systems Division at NASA Ames Research Center. It integrates extensive flight, air traffic management, and weather datasets to support research in airspace operations, safety, and efficiency. While Sherlock is a powerful tool for aviation research, access to its full dataset is restricted and not available for public use. Only authorized researchers and collaborators can utilize the complete resources of Sherlock.

Toward this goal, we first introduce Amelia-48, a large-scale dataset, collected using the Federal Aviation Administration's (FAA) System Wide Information Management (SWIM) Program. Amelia-48 comprises trajectory data from 48 U.S. airports collected starting Dec 1st of 2022. In addition to the SWIM data, we collect, post-process, and release airport surface maps in the form of easy-to-use lightweight graphs with semantic markings for important attributes such as taxiways, runways, hold-short lines, etc. To the best of our knowledge, Amelia-48 is the largest dataset of its kind and is geared towards use by researchers even beyond those interested in airport motion forecasting.

4.2 Trajectory Data Collection

4.2.1 Raw Data

To build Amelia-48, we leverage the System Wide Information Management (SWIM) Program [152], an information system that provides a repository of aviation data spanning the National Airspace System. We utilize the SWIM Terminal Data Distribution System (STDDS), which aggregates terminal data from various sources like Airport Surface Detection Equipment – Model X (ADSE-X), and Airport Surface Surveillance Capability into a single data stream providing position reports for aircraft and vehicles operating within a few miles of the airport, covering *approaching* and *ground movement* events. The raw dataset is a collection of SMES messages covering Terminal Radar Approach Control Facilities (TRACONs) within the NAS.

Our data collection started on Dec 1st, 2022, and represents **over 50TB** of raw trajectory data recorded and stored on our in-house server. We ensure redundancy against server downtime and networking snags in our collection process through concurrent Advanced Message Queuing Protocol [**?**] connections. Each connection records data for one hour and then offloads the data to storage as a tarball.

4.2.2 Processed Data

To process the raw position messages, we further develop data pre-processing scripts following [137]. Here, we focus on 48 airports in the US[1], for which we produce clean

[1]List of currently supported airports: ameliacmu.github.io/amelia-dataset/.

Table 4.1: List of fields with corresponding units and descriptions in the pre-processed Amelia-48 dataset.

Field	Units	Description
`Frame`	#	Timestamp
`ID`	#	STDDS Agent ID
`Altitude`	feet	Agent Altitude (Mean Sea Level)
`Range`	km	Distance from airport datum
`Bearing`	rads	Bearing Angle w.r.t. North
`Lat`	decimal degs	Latitude of the agent
`Lon`	decimal degs	Longitude of the agent
`Speed`	knots	Agent Speed
`Heading`	degrees	Agent Heading
`x`	km	Local X Cartesian Position
`y`	km	Local Y Cartesian Position
`Type`	int	Agent Type [0:aircraft, 1:vehicle, 2:unknown]
`Interp`	boolean	Interpolated data point flag

interpolated data in formats suitable for most modern ML-based trajectory forecasting dataloaders. We do so by defining a 3D geographical fence around the airport of interest, and then filtering the position reports that fall inside it and within 2000ft above ground level. The data is padded with reports before and after the requested time to ensure continuity in the processed data.

We also extract relevant metadata from each position report which we then interpolate and resample at 1Hz. We produce CSV files corresponding to one hour of data for each airport. Table 4.1 shows the various fields in each CSV with their units and descriptions. To get the local cartesian coordinates, we pick an arbitrary origin within the airport surface.

4.3 Data Records

4.3.1 Raw Data

The raw data contains everything captured by the SMES SWIM system for the airports in the United States. To download the raw data, please follow the instructions in AmeliaSWIM repository on how to download the data for a user specified time duration.

4.3.2 Processed Data

To convert the raw data into CSV files, please follow the instructions in AmeliaSWIM on how to use the processing script. We provide the processed trajectory data used for our trajectory forecasting experiments, which contains 1 month of data for each of the 10 airports:

1. Boston-Logan Intl. Airport - Jan 2023
2. Newark Liberty Intl. Airport - Mar 2023
3. Ronald Reagan Washington Natl. Airport - April 2023
4. John F. Kennedy Intl. Airport - April 2023
5. Los Angeles Intl. Airport - May 2023
6. Chicago-Midway Intl. Airport - June 2023
7. Louis Armstrong New Orleans Intl. Airport - July 2023
8. Seattle-Tacoma Intl. Airport - Aug 2023
9. San Francisco Intl. Airport - Sept 2023
10. Ted Stevens Anchorage Intl. Airport - Nov 2023

4.4 Conclusion

This work contributes Amelia-48, a large-scale dataset for airport surface movement which comprises trajectory information and graph-based representations of the airport maps. We validate our dataset and data processing pipelines, providing statistical analyses, visualizations, and insights for 10 major U.S. airports. To the best of our knowledge, this is the **largest dataset of its kind in the public domain**, specifically geared toward training next-generation data-driven predictive models for airport surface operations.

The work was extended to Amelia-TF, a large transformer-based trajectory forecasting model trained on $\sim$9.4 billion tokens of the processed dataset. We **open-source all data and model tools** at: ameliacmu.github.io.

Part II

[Known-Knowns] Neurosymbolic Reasoning for Rule Adherence

Chapter 5

Signal Temporal Logic Tree Search for Guided Imitation Learning

5.1 Introduction

In many real-world domains, the ability to learn from multiple interactions with the environment is either prohibitively expensive or comes with safety concerns. Providing embodied AI agents with the ability to learn effectively from offline datasets of human demonstrations is thus critical in our pursuit to deploy them reliably in real-world domains. Prior works have demonstrated limited success when Learning from Demonstrations (LfD) is treated as a purely supervised learning task [14]. Expert demonstrations are often noisy, incomplete, or both. Thus, pure LfD policies, like Behavior Cloning (BC), perform suboptimally when deployed in the real world. They also often fail to *distill* the underlying rules and constraints that guide the behaviors of the expert agent. These issues are especially more pronounced when using LfD to train policies for safety critical systems when a goal is to be achieved while adhering to rules. We thus ask the question:

Can we encode rules as high-level task specifications to improve the online performance of the LfD policies?

For continuous dynamical systems, there often exist spatio-temporal constraints that define the rules of operation of the system. However, traditional LfD methods do not provide a formal way to bias the system behavior for the satisfaction of rules. Moreover,

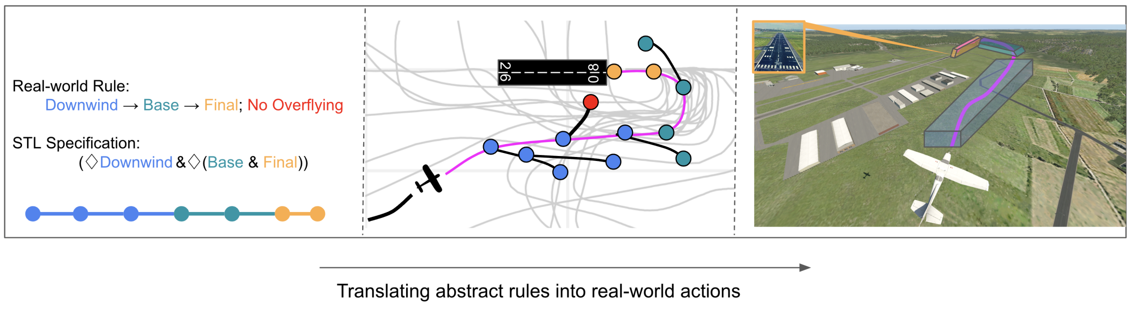

Real-world Rule:
Downwind → Base → Final; No Overflying
STL Specification:
(◇Downwind & ◇(Base & Final))
26
08
Translating abstract rules into real-world actions

Figure 5.1: Figure shows the proposed approach in a prototypical scenario to plan for an aircraft landing in a non-towered airfield. The expert/pilot demonstrations (grey) are used offline to train an LfD policy; online, we use the Signal Temporal Logic specifications in the MCTS expansion to ensure rule compliance.

the rules are often expressed in ambiguous natural language with partial specifications. Temporal logics such as Signal Temporal Logic (STL) provide a mathematically robust representation to encode such spatio-temporal constraints. They can be used to logically specify desired behavior translated from requirements expressed in natural language. STL is used to specify properties over real-valued dense time signals often generated by continuous dynamical systems [106]. Quantitative semantics associated with STL provides a real value called robustness which quantifies the degree of satisfaction or violation. This property enables the designer to encode domain-specific constraints and quantitatively measure their satisfaction with motion planning and control. In order to infuse the STL specifications to improve the base LfD, we propose using a variant of Monte Carlo Tree Search (MCTS) that uses an offline pre-trained network to generate simulated roll-outs. MCTS is a powerful heuristic search algorithm often deployed for long-horizon decision-making tasks [164]. MCTS, when used with a UCT heuristic, [86] has properties like anytime convergence to the best action. This makes it well suited to be used as a long-horizon goal-directed planner within large state spaces with a time budget.

In this work, we present a novel decision-making method that uses MCTS to build a tree structure that guides the offline pre-trained LfD policies online with STL specifications. We achieve this by biasing the MCTS heuristic sampling towards branches with higher STL satisfactions. The primary insight is that while the LfD policy decides on the low-level executions in line with the expert demonstrations, STL encourages the satisfaction of high-level objectives. This hierarchical approach provides a method to encode rules while allowing the agent to choose how to satisfy the constraints in a learning-enabled framework. The STL specifications thus provide a

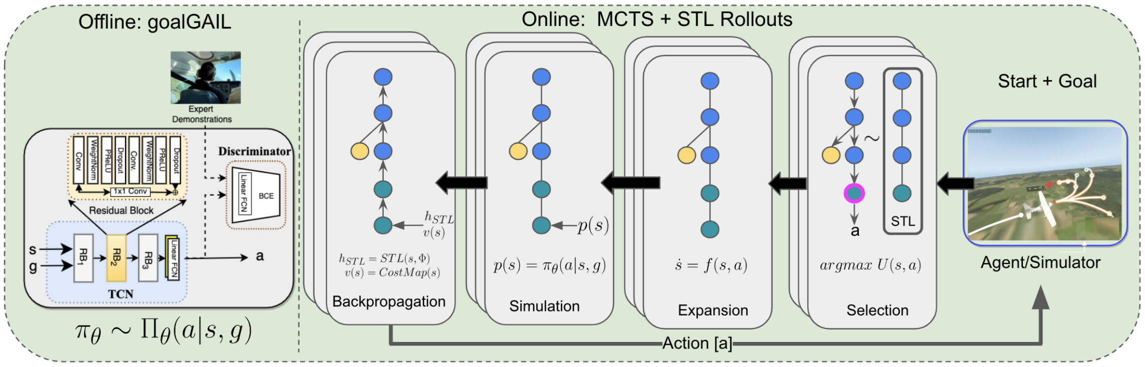


Figure 5.2: Overview of the approach: Offline, we train an LfD policy using datasets, which are used Online in a Monte-Carlo Tree Search (MCTS). The online expansion uses a modified heuristic that uses robustness values from Signal Temporal Logic (STL) specification to guide the search toward higher rule conformance.

guide rail against the low-bandwidth noise in the expert demonstrations.

In order to showcase the efficacy of the proposed algorithm, we use the prototypical case study of planning for a General Aviation (GA) aircraft operating at a non-towered airfield. GA aircraft operating under Visual Flight Rules (VFR) in non-towered airspaces are allowed to operate without a central authority while following some general rules established by the FAA. While not enforced, pilots are expected to adhere to these rules, but deviations arise due to a multitude of factors. Transponder data is available to observe and learn from this behavior, but it is often noisy as each pilot follows a variation of the rules. The proposed algorithm uses this data to train a behavior cloning policy and ensures rule compliance by augmenting the MCTS using STL specifications derived from the rules.

The contributions of this work are as follows:

1. We propose a novel method to incorporate high-level rules expressed in STL specifications in online MCTS simulation that augments any base pre-trained LfD policy.
2. We showcase results on a challenging real-world problem that uses human demonstration data with experimental evaluations performed on a simulator and show improvements over the base policy.

The chapter is organized as follows: In Section 5.2, we provide an overview of the previous approaches to solve sequential decision-making using imitation learning, MCTS, and STL. In Section 5.3, we introduce the approach and our algorithm. In Section 5.4, we provide implementation details for the GA problem statement. Section 5.5 provides comparative results with established baselines. Section 5.6 provides concluding remarks and outlines future work.

5.2 Related Work

Guiding LfD robot behavior using specifications expressed in formal logic has been previously explored in literature. Previous methods have leveraged LTL [76, 180] or, more recently, STL [91, 83, 190, 147] to encode the desired robot behavior to enable planning for autonomous systems. Most prior work focuses on offline backpropagating STL robustness along with imitation learning loss to improve the trained policy's constraint satisfaction. These proposed offline methods that learn from either a margin based on the lower bound of STL satisfaction [39], reward functions [144, 145], vector representation [69], or risk metrics [96].

While offline learning has led to improved STL satisfaction, there are no guarantees that the resulting controller will produce satisfying trajectories [93] nor can it accommodate post hoc specification changes. To this end, methods that use constraints online in the form of Control Barrier Functions [102] and one-step state feedback [190] have been proposed, but neither uses expert demonstrations. [75] takes an alternate approach to improving the efficiency and applicability of LfD-generated policies using beam search with goal generation as a hierarchical approach. The closest to our work is the recently proposed method [93] that uses STL and expert demonstrations to synthesize a trajectory-feedback controller. The offline component uses an LSTM-based controller whose parameters are modified on-the-fly. Our proposed method has no such requirements for using a particular architecture, nor does it require updating the base policy's parameters.

While a majority of the algorithms showcase results on either grid-based worlds [82], simple reach-avoid problems [102], or deterministic settings [83], our work showcases results using real-world, noisy expert demonstrations.

5.3 Methodology

This section details the problem statement and the proposed framework.

5.3.1 Preliminaries

Given a continuous-space dynamic system of the form $\dot{s} = f(s, a)$, we define a discrete-time Markov Decision Process without rewards, (MDP \R). Let $\mathcal{M} = (S, A, T, \rho_0, G)$ where S is the set of states $s \in S$, $a \in A$ is the discrete set of actions or motion primitives that follow $f(\cdot), T : S \times A \Rightarrow S$ is the transition function, $\rho_0 \in S$ is an initial state distribution, and G is the goal distribution.

The task is to produce a policy $\pi(\theta)$ from a start location $s_0 \in \rho_0$ to goal location $g \in G$ that leads to a trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, g)$. We also assume access to expert trajectories $\mathcal{D} = \{(s_0^j, a_0^j, s_1^j, a_1^j, \ldots, g^j)\}_{j=0}^{D}$ and high-level STL specification Φ that encodes any rules we expect the system to follow. An STL formula Φ can be built recursively from predicates using the following grammar

$$\Phi := \top \mid \mu_c \mid \neg\Phi \mid \Phi \wedge \Psi \mid \Diamond_{[a,b]}\Phi \mid \Box_{[a,b]}\Phi \mid \Phi_1 \mathcal{U}_{[a,b]} \Phi_2 \tag{5.1}$$

where Φ_1, Φ_2 are STL formulas, $\top$ is the Boolean True, μ_c is a predicate of the form $\mu(s) > c$, $\neg$ and $\wedge$ the Boolean negation and AND operators, respectively, and $0 \leq$ a $\leq$ b $< \infty$ denote time intervals. The temporal operators $\Diamond, \Box$ and $\mathcal{U}$ are called "eventually", "always", and "until" respectively. The quantitative semantics of a formula with respect to a signal $\vec{x_t}$ can be used to compute robustness values [46] for the specifications used in our application.

5.3.2 Framework

The overall framework is shown in Fig. 5.2 and Algorithm 1. We first train an LfD policy by formulating the problem as finding a distribution of future actions $\hat{a}_t$ conditioned on the past trajectories $s_{t-t_{obs}:t}$ and the goal g where t_{obs} is the observation time horizon.

$$\hat{\pi}_\theta \sim \Pi_\theta(\hat{a}_t \mid s_{t-t_{obs}:t}, g) \tag{5.2}$$

The MCTS (see Algorithm 2) uses the policy $\hat{\pi}_\theta$ to generate simulations. Each simulation starts from the root state and iteratively selects moves that maximize the STL-modified UCT heuristic.

For each state transition, we maintain a directed edge in the tree with an action value $Q(s,a)$, prior probability $P(s,a)$, STL heuristic $H(s,a)$, and a visit count $N(s,a)$. The total heuristic value is calculated as

$$U(s,a) = Q(s,a') + \frac{c_1 P(s,a')\sqrt{N(s)}}{1 + N(s,a')} + c_2 H(s,a') \tag{5.3}$$

where c_1 and c_2 are the hyperparameters controlling the degree of exploration and STL heuristic's weight. Starting with the initial state $s_{(0)}$, at each time step, we calculate the action to take, which maximizes $U(s,a)$ (Line 12). If the next state already exists in the tree, we continue our simulation, else a new node is created in our tree, and we initialize its $P(s,\cdot) = \vec{p}_\theta(s)$ from our policy $\hat{\pi}_\theta$ (Line 8). The expected reward $v = v_\theta(s)$ can be provided by the user as a learned value function or as a cost-map (Line 9). The heuristic h_{STL} is calculated using the STL specification (Line 14). We initialize $Q(s,a)$, $H(s,a)$ and $N(s,a)$ to 0 for all a. We then propagate the cost v and the STL heuristic h_{STL} back up the MCTS tree, updating all the $Q(s,a)$ and $H(s,a)$ values seen during the simulation, and start again from the root.

After running forward simulations of the MCTS, the $N(s,a)$ values provide a good approximation for the optimal stochastic process from each state. Hence, the action we take is randomly sampled from a distribution of actions with probability proportional to $N(s,a)$. We expand the search space until we exhaust the planning budget time $planHorizon$. After each action is selected, the MCTS tree is reinitialized from the actual trajectory followed by the agent. The planner is terminated when the goal is reached, or the maximum number of steps is reached, whichever comes first.

5.4 Experiments

In our experiments, we tackle the case of planning for a general aviation aircraft at a non-towered airport. This section provides the necessary background, problem definition, and implementation details.

Algorithm 1 Plan (θ,Φ)

```
$s \leftarrow Sample(\rho_0)$

$g \leftarrow Sample(G)$

**while** $s \neq g$ **and** not maxSteps **do**
    **while** $timeElapsed \leq planHorizon$ **do**
        $N(\cdot) \leftarrow MCTS(s_{0:t}, g, \theta, \Phi, 0)$
    **end while**
    $a \leftarrow choice_{a'}(N(s, a'))$

    $s \leftarrow T(s, a)$
**end while**
```

5.4.1 Background

In an environment without a centralized controlling authority, such as near a non-towered airspace, the FAA establishes rectangular airport traffic patterns. These patterns facilitate the smooth flow of aircraft entering and leaving the airspace, similar to roadways for automobiles. However, different pilots follow distinct paths [135], leading to variations in the expert demonstration trajectories. This variation arises due to the unique combinations of aircraft type, weather, pilots' experience, visual observations, and the non-strict regulations for following the pattern. A single fixed set of waypoints to follow would not suffice to cover all possibilities for the aircraft. With such a broad set of rules, a range of possible trajectories can be followed to land an aircraft safely.

To illustrate with an example, the high-level goal of landing on a runway through traffic patterns can usually be divided into three stages [51]. The three stages can be represented by three individual regions to be occupied by the aircraft in a particular order (See Fig. 5.3). The first stage is termed the downwind leg, which involves flying in the opposite direction to the intended landing runway at 1000 ft above ground level. Once the pilot is at a 45° angle from the approach end of the runway, the second stage begins, in which a perpendicular turn is made to the base leg until it reaches the runway approach's extended center line. Once the turn is complete, the third and final stage begins, which is a descending path to the point of touchdown. We provide a mechanism to encode such traffic patterns and utilize LfD to implement the sequence of actions that can land an aircraft from a given start to the runway, just like a human

Algorithm 2 MCTS ($s_{0:t}$,g,θ,Φ,h_{STL})

```
if s ∈ G then
    return s == g, $h_{STL}$
end if
if s ∉ Tree then
    $Tree \leftarrow Tree \cup s$

    $Q(s,a) \leftarrow 0$

    $N(s,a) \leftarrow 0$

    $P(s,a) \leftarrow \hat{\pi}_\theta(s)$

    $v(s) \leftarrow CostMap(s)$

    return $v(s), h_{STL}$

else
    $a \leftarrow argmax_{a'}\left[Q(s,a') + \frac{c_1 P(s,a')\sqrt{N(s)}}{1+N(s,a')} + c_2 H(s,a')\right]$

    $s' \leftarrow T(s,a)$

    $h_{STL} \leftarrow STL(s' + s_{0:t}, \Phi)$

    $v, h_{STL} \leftarrow MCTS(s' + s_{0:t}, g, \theta, \Phi, h_{STL})$

    $N(s,a) \leftarrow N(s,a) + 1$

    $Q(s,a) \leftarrow \frac{N(s,a)Q(s,a)+v}{1+N(s,a)}$

    $H(s,a) \leftarrow \frac{N(s,a)*H(s,a)+h_{STL}}{1+N(s,a)}$

    return $v, h_{STL}$
end if
```

pilot.

5.4.2 Problem Definition

Consider a fixed-wing aircraft at time t, let $s_t = (x_t, y_t, z_t, \chi_t) \in \mathcal{R}^3 \times SO(2)$ denote the position and heading of the agent. We also define the system $\dot{s} = f(s, a)$ (5.4) as:

$$\dot{x} = v_{2D} \cos \chi \tag{5.4a}$$

$$\dot{y} = v_{2D} \sin \chi \tag{5.4b}$$

$$\dot{z} = v_h \tag{5.4c}$$

$$\dot{\chi} = \frac{g_e \tan \phi}{v_{2D}} \tag{5.4d}$$

$$v = \sqrt{v_{2D}^2 + v_h^2} \tag{5.4e}$$

where v is the aircraft's inertial speed, v_{2D} is the speed in the 2-D plane, ϕ is the roll-angle, and v_h is vertical speed. Finally, g_e is the acceleration due to gravity. We ignore the effects of wind. The action space $A = \{(v^j, v_h^j, \phi^j)\}_{j=0}^{A}$ is a fixed library of 30 motion primitives that discretize each of the control inputs. We use the inertial velocities (v) 70 and 90 knots, the vertical velocities (v_h) +500 ft/min and -500 ft/min, and the bank angle (ϕ) discretization such that χ changes by $45°$ and $90°$ heading over the chosen time-horizon of 20 sec. The goal distribution G is a one-hot vector representation of the final goal of a particular agent as the eight cardinal directions along with two runway ends, as shown in Fig. 5.3. The set G is represented as $G = \{N, NE, E, SE, S, SW, W, R08, R26\}$ with each element representing the final region the airplane is desired to reach as shown in Fig. 5.3. For simplicity, we also set the start states to one of these regions, i.e. $\rho_0 = G$.

5.4.3 Implementation Details

The implementation details are split between online and offline components.

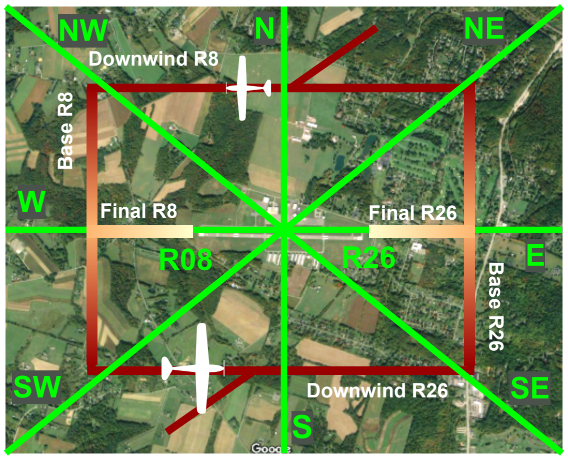


Figure 5.3: Goal representation is in the form of a one-hot vector where each region is the respective goal element in the goal vector G. The maroon rectangle shows airport traffic patterns.

Dataset

We use the *TrajAir*[1] dataset, which provides recorded trajectories of aircraft operating at the Pittsburgh-Butler Regional Airport (ICAO:KBTP) [135]. The dataset contains 111 days of transponder data, processed to obtain the local x, y, and z coordinates of aircraft at every second. This is used to extract expert demonstrations from pilots as they navigate the un-towered airspace. The dataset trajectories are smoothed using a B-spline (basis-spline) approximate representation of order 2. The trajectories are not biased toward satisfying the STL rules as there are variations in landing patterns exhibited by the dataset.

Offline LfD Policy Details

The LfD policy takes as input the past trajectories of the agent to predict its possible action distribution. While the method can use any LfD policy, we use a goal-conditioned generative adversarial imitation learning (GoalGAIL) method [45]. The GoalGAIL is modified to use Temporal Convolutional Layers (TCNs) to process the sequential trajectory data. TCN layers encode a trajectory's spatio-temporal information into a latent vector without losing the underlying data's temporal (causal) relations [16]. We use TCNs as an alternative to using LSTMs [198] for encoding the trajectories.

[1]http://theairlab.org/trajair/

Algorithm	**Takeoff Specification Φ_T**				**Landing Specification Φ_L**				**Total**
	N	S	E	W	N	S	E	W	
BC + MCTS	0.2 / 0.2	0.0 / 0.1	0.3 / 0.1	0.3 / 0.1	0.0 / 0.0	0.1 / 0.4	0.1 / 0.4	0.3 / 0.4	0.1 / 0.2
GoalGAIL + MCTS	0.0 / 0.3	0.1 / 0.4	0.0 / 0.3	0.0 / 0.4	0.3 / 0.6	0.1 / 0.2	0.3 / 0.7	0.1 / 0.5	0.1 / 0.4
BC + MCTS + STL	1.0 / 0.9	1.0 / 0.9	0.6 / 0.3	0.7 / 0.4	0.2/ 0.5	0.2 / 0.5	0.2 / 0.5	0.2 / 0.5	0.5 / 0.6
GoalGAIL + MCTS+STL	**1.0 / 0.9**	**1.0 / 0.9**	**0.8 / 0.7**	**1.0 / 0.9**	**0.7 / 0.9**	**0.8 / 0.9**	**0.3 / 0.8**	**0.5 / 0.9**	**0.7 / 0.8**

Figure 5.4: Table shows the quantitative results with randomly sampled start and goal states for two LfD policies with ablation studies with the STL heuristic. Results show the Success Rate ↑ / STL Score ↑ for two vanilla LfD policies and their ablations with the STL heuristic. Results show both Landing $X \Rightarrow R$ and Takeoff $R \Rightarrow X$ scenarios for each of the cardinal directions X.

We break the trajectories in a scene into sequences of length $t_{obs} + t_{pred}$ where $t_{obs} = 11sec$ and $t_{pred} = 20sec$. In a given scene, the raw trajectory in absolute coordinates of the agent is encoded using the TCN layers as h_{obs}. The agent's goal $g \sim G$ is encoded through an MLP layer, φ_1, and is concatenated with the encoded trajectory vector. Equations 5.5 show the encoding sequence.

$$h_{obs} = TCN_{obs}(s_{1:t_{obs}}) \tag{5.5a}$$

$$h_g = \varphi_1(g) \tag{5.5b}$$

$$h_{enc} = h_{obs} \oplus h_g \tag{5.5c}$$

$$\hat{s}_{t_{obs}:t_{obs}+t_{pred}} = \varphi_2(h_{enc}) \tag{5.5d}$$

The $\mathcal{L}_{act}$ measures how close the predicted trajectory is to the expert trajectory using a mean squared error (MSE) loss.

$$\mathcal{L}_{act} = MSE(s_{t_{obs}:t_{obs}+t_{pred}}, \hat{s}_{t_{obs}:t_{obs}+t_{pred}}) \tag{5.6}$$

A discriminator D_ψ is trained to distinguish expert transitions $(s, a, g) \sim \tau_{\text{expert}}$, $E(s_{1:t_{obs}})$, from policy transitions $(s, a, g) \sim \tau_{\text{policy}}$, $\hat{s}_{t_{obs}:t_{obs}+t_{pred}}$.

The discriminator is trained to minimize,

$$\begin{aligned}\mathcal{L}_{goalGAIL}\left(D_\psi,\right) = &\mathbb{E}_{(s,a,g)\sim\text{policy}}\left[\log D_\psi(a, s, g)\right] + \\ &\mathbb{E}_{(s,a,g)\sim\text{expert}}\left[\log\left(1 - D_\psi(a, s, g)\right)\right]\end{aligned} \tag{5.7}$$

The combination of these two loss functions is used to train the model.

$$\mathcal{L}_{total} = \mathcal{L}_{act} + \mathcal{L}_{goalGAIL} \tag{5.8}$$

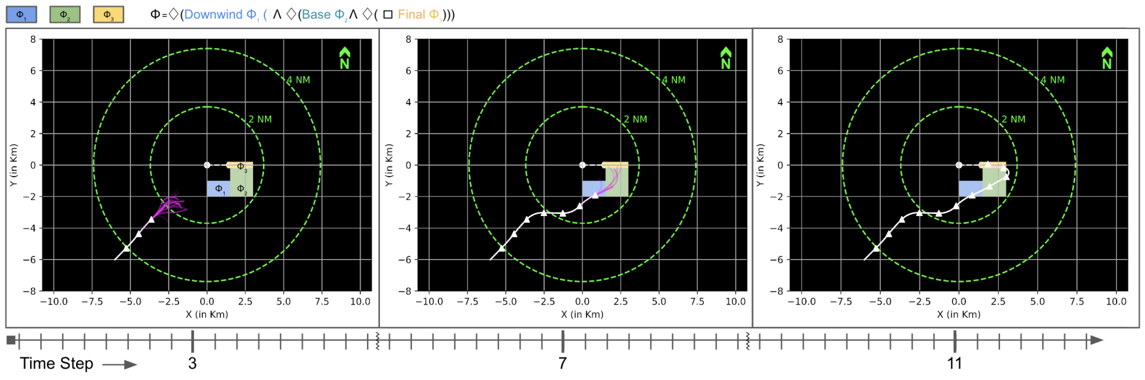


Figure 5.5: Figure shows a qualitative example from one of the cases where the aircraft starts from the South-West and needs to land at one of the runways (R26). The specifications Φ_1, Φ_2 and Φ_3 are shown as rectangles. White marked lines show the aircraft trajectory, and the magenta shows the MCTS tree. The runway threshold for R08 (+x-axis) is at the center.

In order to convert $\hat{s}$ to $\hat{a}$, we match the generated trajectories from the control inputs in the library A using a weighted L2 Euclidean error distance over (x, y, z) points on the trajectory. For training, we use the AdamW optimizer with a learning rate of $3e-3$.

Signal Temporal Logic Specifications

We evaluate the performance of our agent based on reaching the goal while adhering to the airport traffic pattern. The goal objective, as well as traffic pattern compliance, are both encoded using STL specifications. We use the three stages for the landing pattern as defined in (IV-A). Φ_1, Φ_2, and Φ_3 represent the STL formulas encoding occupancy of regions corresponding to the downwind, base, and final stages, respectively. The landing STL specification becomes:

$$\Phi_L = \diamondsuit(\Phi_1 \;\wedge\; (\diamondsuit(\Phi_2 \;\wedge\; (\diamondsuit\square\Phi_3)))) \tag{5.9}$$

$\diamondsuit(\Phi)$ can be interpreted as "Eventually" being in a region represented by Φ. The nested $\diamondsuit$ operators encode a sequential visit of regions represented by Φ_1, Φ_2, and Φ_3. Similarly, the takeoff STL specification is defined based on the goal regions reached by the aircraft. By defining reaching a goal region $g \sim G$ by an STL formula Φ_4, we get the takeoff specification:

$$\Phi_T = \diamondsuit(\Phi_4) \tag{5.10}$$

The robustness values of the STL specifications are evaluated on the state trajectory traces generated by the search tree. The first element of the traces is the tree root node, and the last element of the trace is the node whose value is currently computed.

Online Monte Carlo Tree Search

The MCTS is implemented as a recursive function where each iteration ends with a new leaf that corresponds to an action in the trajectory library. The implementation uses a normalized costmap $v(s)$ that is built by counting the frequencies of the aircraft in the TrajAir dataset at particular states s.

5.5 Evaluations

Evaluation of the proposed approach is done using a custom simulator that follows the dynamics defined in Eq. 5.4. The network implementations are done on PyTorch. For calculating STL robustness values, we formulate our specifications using the `rtamt` [129] package, an online monitoring library. To showcase real-time online evaluations, simulations are performed on an Intel NUC computer with Intel® Core™ i7-8559U CPU @ 2.70GHz × 8. The complete implementation details and parameter details are in the open-sourced code-base[2] and the associated Readme.

5.5.1 Qualitative results

Figure 5.5 shows an example scenario where the aircraft starts from the South-West and is tasked with landing at R26 while following the standard FAA traffic pattern. The rules of the traffic pattern are encoded as STL specifications. The white-marked line shows the aircraft's position at each step. At every step, the MCTS replans, and the resulting tree is shown in magenta. The STL sub-specifications are shown as rectangles. As can be seen, the aircraft manages to reach the runway while satisfying the specifications. The size of the search tree is a function of available $planHorizon$.

[2] Codebase: https://github.com/castacks/mcts-stl-planning

5.5.2 Comparative results

In order to perform quantitative studies, we compare the performance of the proposed algorithm with vanilla LfD policies. We uniformly sample 100 start-goal pairs randomly from ρ_0, G. G is truncated to N, S, E, W, R where R represents both R08 and R26 to condense the results. We then provide these to Algorithm 1, which plans for the agent. Based on the selected start and goal pair, a relevant STL specification is chosen. Each episode ends when the agent reaches the goal or if the maximum steps are exceeded. In addition to the goalGAIL policy, we also train a pure Behavior Cloning (BC) policy. The BC policy uses a TCN to encode the history and outputs an action without a goal vector or a discriminator. Comparisons were carried out with both these policies integrated into the MCTS framework with ablation on the STL heuristic to show the impact of the STL specification.

We define our evaluation metrics as follows:

- Success Rate: Fraction of episodes that were successful in reaching their goal locations.
- STL Score: Average of the normalized fraction of the STL robustness value satisfied over all the episodes. A higher value indicates better satisfaction.

Table 5.4 shows the quantitative results. Our proposed algorithm performs significantly better than the baselines for all start-goal pairs. We get an almost perfect success rate in the aircraft takeoff scenarios. The aircraft landing cases are more challenging due to following the specific landing patterns when incoming from different sides of the runway, which is reflected in the success rates shown. Additionally, we observe the baseline BC performs similarly to GAIL on the success metric but not on the STL robustness. This indicates that while BC policies are comparable in reaching the goals, the transient performance of GoalGAIL is better. Incorporating STL improves the robustness values for both LfD policies.

5.6 Conclusions

In this work, we present a novel method that improves the online robustness of offline pre-trained LfD policies using MCTS to fuse STL specifications. To the best of the authors' knowledge, this is the first method that combines high-level STL specifications with low-level LfD policies through MCTS. Our experimental evaluation targets the real-world problem of autonomous aircraft planning and exhibits the feasibility of our techniques for similar challenging decision-making problems. We show our method outperforms vanilla LfD methods on a number of successful missions for multiple complex objectives using real-world data.

Chapter 6

Multi-Agent Tree Search for Data Driven Social Navigation

6.1 Introduction

A social robot strives to synthesize decision policies that enable it to seamlessly interact with humans, ensuring social compliance while attaining its desired goal. While marked progress has been made in social navigation and motion prediction [116, 155, 170], achieving seamless navigation among humans while balancing social and self-interested objectives remains challenging.

Social navigation can be formulated as a Partially Observable Stochastic Game (POSG) [156, 151]. Deep Reinforcement Learning (DRL) methods [114] explicitly formulate the POSG reward to derive a policy from simulated self-play experiments. While such techniques are promising in sparse-data domains where the robot is easily distinguishable from human, tuning reward parameters for homogeneous navigation among humans is not trivial [118]. DRL policies are also a function of the underlying simulator, which often translates to undesirable behavior with the sim-to-real transfer owing to the lack of compatibility with real-world scenarios.

On the other hand, data-driven approaches are common in social trajectory prediction. They aim to directly characterize human interactions observed in the data [155], eliminating the need for reward shaping and accurate simulations. Recent sequence-to-sequence models, for instance, have achieved promising results in intent prediction [158, 137, 124]. However, using these models for downstream navigation is difficult as they often suffer from prediction failures [53] which hurt their generalization capabilities. This potential for unsafe behavior prompts the need for robustifying models deployed in the real world.

In social navigation, the actions of one agent influence those of another and vice-versa [156, 73]. This temporally recursive decision-making intuition has been used for modeling human-like gameplay [165, 78]. Leveraging this insight, we propose using a recursive *search*-based policy to robustify offline-trained models with downstream *social* navigation objectives. Specifically, we use Monte Carlo Tree Search (MCTS) [85] as our *search* policy which provides long-horizon recursive simulations, collision checking and goal conditioning. We combine it with a *socially*-aware intent prediction model to provide short-horizon agent-to-agent context cues and motion naturalness. We use MCTS to fuse these short-horizon cues with long-horizon planning by including global reference paths to guide the tree expansion. We refer to our framework as *Social Robot Tree Search (SoRTS)*.

The growing operations of Unmanned Aerial Vehicles are leading to a demand for using airspace concurrently with human pilots [11, 66]. We, therefore, select the domain of general aviation (GA) to showcase our approach. GA was recently framed within the paradigm of social navigation [137, 124], where pilots are *expected* to follow flying guidelines to coordinate with each other and respect other's personal space to ensure safe operations. This is analogous to following etiquette in human crowds and vehicular settings.

Safety-critical domains, like GA, demand high competence to guarantee seamless and safe operations. This entails developing trustworthy robots which are able to understand and follow navigation norms but also understand long-term dynamic interactions to avoid causing danger or discomfort to others. In this paper, we separate these aspects into two axes, *navigation performance* and *safety*. We center our framework design and evaluations around these two axes.

Statement of contributions:

1. We introduce and open-source SoRTS, an MCTS-based algorithm for long-horizon navigation that robustifies offline learned socially-aware intent prediction policies for downstream navigation.
2. We introduce X-PlaneROS, a high-fidelity simulation environment for navigation in shared aerial space, and;

6.2 Related Work

6.2.1 Social Navigation Algorithms

Social navigation has a rich body of work focused on pedestrian and autonomous driving domains [116]. In pedestrian settings, classical model-based approaches [22, 115] have been proposed and remain prominent baselines. Yet, their extension to other domains is non-trivial. Recent DRL methods [114, 38, 35] that use handcrafted safety-focused reward functions [173] have produced promising results in these domains. However, shortcomings in simulator design [23], and domain-specific reward functions limit real-world performance [173]. Achieving scalability and robustness is challenging, often requiring expensive retraining. Similar to our approach, [151] introduces a DRL method that uses MCTS to train and deploy policies. While their method relies on pre-defined reward functions and simulator training, our work extends these methods to use offline expert-based policies, providing domain-specific treatment for social navigation.

Data-driven approaches focus on learning policies from datasets that record interactions between agents [173, 116]. As these models do not need explicit reward construction, they can capture the rich, joint social dynamics. However, these methods are challenging to deploy directly owing to noisy and missing demonstrations [19, 40]. To alleviate this, [156] used the gradients of a Q-value function for Model Predictive Control, and [73] proposed a generalization to this method using dual control for belief state propagation. These methods rely on Inverse Reinforcement Learning as an additional step to generate the Q-value functions. Using gradients from sequence models directly in optimizations has also been proposed [160], but the convergence properties were not examined. Our method is more direct in its use of sequential models and calculating gradients or Q-values is not required. Instead, we transform the model's outputs into action distributions for the downstream planning task.

6.2.2 Social Navigation Evaluation

Different metrics have been considered for the evaluation of social robot navigation [116, 61, 155]. Some of the main axes of analysis for evaluation include behavior naturalness based on a reference trajectory or irregularity of the executed path [117, 159], performance and efficiency [101, 50, 98], and notions of physical personal space or discomfort [171, 37]. User studies are often conducted to evaluate more subjective

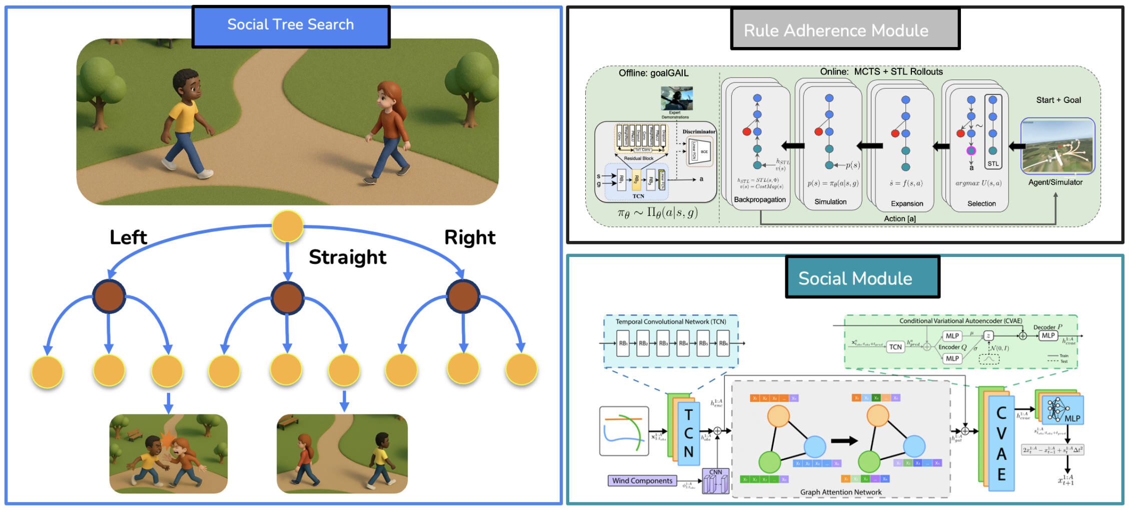


Figure 6.1: An overview of SoRTS, a Monte Carlo Tree Search (MCTS)-based planner for social robot navigation whose tree search is guided by three components; a Social Module, a Reference Module and a Cost Map. The Social Module leverages an offline-trained intent prediction model to characterize agent-to-agent interactions and predict a set of possible future states for the ego-agent. The Reference Module provides the agent with the closest reference state from a global plan which embodies navigation guidelines the agent must follow. The Cost Map encodes a global visitation map to further guide navigation. MCTS uses all these modules to provide online collision checking and long-horizon simulations by searching for choosing between different decision modalities.

aspects such as the perceived discomfort and trust that a robot induces during an interaction [27]. Following prior works, we focus on *navigation performance* to measure our agent's smoothness and ability to follow navigation guidelines, and *safety* to judge its ability to respect others' personal space. We also conduct a user study where we ask experienced pilots to interact with our algorithm in a realistic flight setting and rate the robot's performance, perceived safety and trust.

6.3 Problem Formulation

We formulate social navigation as an approximate POSG, a framework for decentralized finite-horizon planning. For more details about POSGs and their use within social navigation, we refer the reader to [48, 151, 156].Following [124, 137], we assume M agents with $\mathbf{s}_t^i \in R^3$ representing state of agent i at time-step t and $\mathbf{a}_t^i \in \mathcal{A}$ is the discrete set of actions or motion primitives that follow $\dot{\mathbf{s}} = f(\mathbf{s}, \mathbf{a})$. Let $\mathbf{s}_0^i$ and $\mathbf{s}_g^i$ represent the start and goal states respectively. The system also has access to a set of offline *expert* demonstrations $\mathcal{D}$ and set of global reference trajectories τ_R. We omit superscripts to refer to the joint state for all agents.

Thus, given the set of start $\mathbf{s}_0$ and goal $\mathbf{s}_g$ states of M agents, the objective is to find a sequence of control inputs $\pi = \{\mathbf{a}_0, \ldots, \mathbf{a}_g\}$ such that the agents follow collision-free trajectories $\tau = \{\mathbf{s}, \cdots, \mathbf{s}_g\}$. The generated trajectories need to ensure $||\mathbf{s}_t^i - \mathbf{s}_t^j|| \geq d \; \forall i, j \in \{1, \ldots, M\} \; \forall t$ where d is the minimum separation distance to satisfy the safety objective. Furthermore, the trajectories also need to stay close to reference trajectories $\min ||\tau - \tau_R||$ to satisfy the navigation objective and follow $\tau \sim \mathcal{D}$ to satisfy the social objective. Without loss of generality, we assume first agent (i=0) to be the robot ego-agent and at each time step execute the optimal action.

6.4 Approach

We designed SoRTS along the axes of *navigation performance* and *safety*. The core insight of SoRTS is to use social modules and global reference paths to bias the MCTS to search for and choose between various decision modalities. Fig. 6.1 shows an example of two aircraft merging on a single path. Each aircraft can safely execute the merger by choosing from three actions: forward, speeding up, and slowing down. MCTS, in its forward simulations, not only prunes branches that lead to future collisions but also uses the social module to choose between cutting-in-front (socially undesirable) and yielding (socially desirable), thereby producing socially compliant and safe behavior.

6.4.1 Modules

SoRTS is a search-based planner which uses Monte Carlo Tree Search (MCTS) whose tree expansion is guided by three modules. First, a *Social Module* handles the short-horizon dynamics in the scene, characterizing social interactions and cues. Second, a *Reference Module* provides the agent with a global navigation guideline, *e.g.*, an airport traffic pattern. Finally, a *Cost Map* encodes global value map. MCTS uses these components together and provides collision checking and long-horizon socially-compatible simulations. Its corresponding pseudo-code is shown in Algorithm 3 and 4.

Social Module

We leverage an offline-trained intent prediction algorithm parameterized in θ, $P_\theta(\cdot)$, to account for the short-term agent-to-agent dynamics following the expert trajectories in $\mathcal{D}$.

$$p_\theta(\mathbf{s}_t^i, \mathbf{a}_t^i) \sim P_\theta(\mathbf{a}_t^i \mid \mathbf{s}_{t-t_{obs}:t}, \mathbf{s}_g^i) \tag{6.1}$$

where $p_\theta(\cdot)$ provides a distribution of future actions $\mathbf{a}_t^i$ for agent i conditioned on the past trajectories $\mathbf{s}_{t-t_{obs}:t}$ of all the agents and the goal $\mathbf{s}_g^i$ where t_{obs} is the observation time horizon.

Here, we use Social-PatteRNN [124], an algorithm which predicts multi-future trajectories from learned interactions that exploit motion pattern information in the data.

Reference Path Module

Given the start and goal state of agent i the algorithm samples a suitable reference trajectory $\tau_r^i \in \tau_R$, Algorithm 3 in Algorithm 3. Similar to Section 6.4.1, this trajectory is used to compute a *reference* action distribution $p_r(\cdot)$,

$$p_r(\mathbf{s}_t^i, \mathbf{a}_t^i) \sim P_r(\mathbf{a}_t^i | \mathbf{s}_t^i, \tau_r^i) \tag{6.2}$$

proportional to the L2 norm between $\mathbf{s}$ and τ_r at time t. In Algorithm 4, the reference action is obtained in Algorithm 4. τ_R can be drawn from expert distributions $\mathcal{D}$, global path planning algorithms like A-Star, logic specifications [8] or can also be handcrafted.

Cost Map

The algorithm also uses a cost map of the environment representing the *value* function, $v(\mathbf{s})$. For our case, we use the cost map to represent state visitation frequency to bias the search towards more desirable areas. Algorithm 4 uses the cost map in Algorithm 4. This value function can be either learned, *e.g.*, via self-play [151] or pre-computed from a prior distribution and captures the value of the joint state distribution.

6.4.2 Social Monte Carlo Tree Search

MCTS is a search-based algorithm that expands its tree search toward high rewarding trajectories. In principle, this is done by selecting nodes along the search that maximize upper confidence bound [85], which balances the degree of exploitation and exploration. SoRTS, leverages MCTS and uses the components presented in the previous section to guide its trajectory roll-outs. Formally, it follows the UCT shown below,

$$U(\mathbf{s},\mathbf{a}) = Q(\mathbf{s},\mathbf{a}) + c_1 \cdot S(\mathbf{s},\mathbf{a}) + c_2 \cdot R(\mathbf{s},\mathbf{a}) \tag{6.3}$$

where $Q(\mathbf{s},\mathbf{a})$ represents the expected value for taking action $\mathbf{a}$ at state $\mathbf{s}$; $S(\mathbf{s},\mathbf{a})$ is visitation normalized component according to the socially-aware module; $R(\mathbf{s},\mathbf{a})$ is the visitation normalized component according to the reference path; c_1 and c_2 are hyperparameters. We drop the time t subscript for ease of notation. In Algorithm 4 of Algorithm 4, these values follow the update:

$$Q(\mathbf{s},\mathbf{a}) = \frac{N(\mathbf{s},\mathbf{a}) \cdot Q(\mathbf{s},\mathbf{a}) + v(\mathbf{s})}{N(\mathbf{s},\mathbf{a}) + 1} \tag{6.4}$$

$$R(\mathbf{s},\mathbf{a}) = \frac{N(\mathbf{s},\mathbf{a}) \cdot R(\mathbf{s},\mathbf{a}) + p_r(\mathbf{s},\mathbf{a})}{N(\mathbf{s},\mathbf{a}) + 1} \tag{6.5}$$

$$S(\mathbf{s},\mathbf{a}) = \frac{\sqrt{N(\mathbf{s})}}{N(\mathbf{s},\mathbf{a}) + 1} \cdot p_\theta(\mathbf{s},\mathbf{a}) \tag{6.6}$$

where $N(\mathbf{s})$ is the state visitation count, and $N(\mathbf{s},\mathbf{a})$ the visitation given action $\mathbf{a}$. These updates are done iteratively by following the states within a time-budget, or until a new state is found. At each time-step, a new forward simulation tree is iteratively constructed by alternately expanding the agents' future states in a round-robin fashion. Branches that lead to collision states are pruned.[1] At the end of $planHorizon$, the ego agent takes the first action that maximizes the visitation count in Algorithm 3 of Algorithm 3. The tree is reset and the process continues till goal is reached.

[1]Note: In practice, for $M > 2$, we only use the ego-agent and the closest agent to the ego agent for tree expansion. While the tree is explicitly constructed only for two agents, p_θ provides the high-level social context for all the agents. This approximation preserves the real-time nature of the algorithm and is shown to perform well in practice.

Algorithm 3 SoRTS($\mathbf{s}_0, \mathbf{s}_g, \theta, \tau_R, v$)

```
1: $\tau_r \leftarrow$ GetReferencePaths($\mathbf{s}_0, \mathbf{s}_g, \tau_R$)
2: while $\mathbf{s}_t^0 \neq \mathbf{s}_g^0$ or $timeElapsed \leq planHorizon$ do
3:     $N(\cdot) \leftarrow$ SocialMCTS($\mathbf{s}_t, \mathbf{s}_G, \tau_r, \theta, v, 0$)
4:     $\mathbf{a}^0 \leftarrow \arg\max_{\mathbf{a}' \in \mathcal{A}} N(\mathbf{s}_t^0, \mathbf{a}')$
5:     $\mathbf{s}_{t+1}^0 \leftarrow f(\mathbf{s}_t^0, \mathbf{a}^0)$                ▷ Robot's transition model.
6: end while
```

Algorithm 4 SocialMCTS($\mathbf{s}_t, \mathbf{s}_G, \tau_r, \theta, v, p$)

```
1: while p $\leq$ M do
2:     if $\mathbf{s}_t^p \notin S(\cdot)$ then                          ▷ New tree node
3:         $v_s \leftarrow$ GetValue($v, \mathbf{s}_t$)
4:         $S(\cdot) \leftarrow$ GetSocialActionProbabilities($\mathbf{s}_t, \theta$)
5:         $p_r(\cdot) \leftarrow$ GetReferenceActionProbabilities($\tau_r^p, \mathbf{s}_t^p$)
6:         $N(\mathbf{s}_t^p) \leftarrow 1$
7:         return $v_s$
8:     end if
9:     $\mathcal{A}' \leftarrow$ CollisionCheck($\mathcal{A}, d, \mathbf{s}_t$)
10:    $\mathbf{a}^p \leftarrow \arg\max_{\mathbf{a}' \in \mathcal{A}'} U(\mathbf{s}_t^p, \mathbf{a}')$            ▷ See eq. 6.3
11:    $\mathbf{s}_{t+1}^p \leftarrow f(\mathbf{s}_t^p, \mathbf{a}^p)$
12:    p $\leftarrow$ p+1                              ▷ Choose next agent to expand
13:    SocialMCTS($\mathbf{s}, \mathbf{s}_g, \tau_r, \theta$, v, p)
14:    Update($Q, R, S, N$)                              ▷ See eq. 6.4 to 6.6
15: end while
16: return $N(\cdot)$
```

6.5 Experimental Setup

Our experiments focus on assessing SoRTS along our axes of interest, *i.e.*, *navigation performance* and *safety*. As such, this section describes the main aspects of our evaluation setup and implementation details for our case study.

6.6 XPlaneROS: High-fidelity Simulator for Autonomous Fixed Wing Operations

6.6.1 Background

Today, manned and unmanned vehicles are separated, limiting the utility and flexibility of operations and reducing efficiency. One area that is particularly challenging for autonomous aircraft are airport/heliport operations where conflicts between aircraft are common and need to be resolved. Mastering visual flight rules (VFR) operations for autonomous aircraft has significant operational advantages at unimproved sites, as well as in achievable traffic density compared to instrument flight rules (IFR) or completely separated operations between manned and unmanned systems.

As is the case with such systems, we need the ability to verify the safety of the algorithms before deploying them in the real world. There exist realistic simulators for testing driving cars (CARLA) and autonomous drones (AirSim) which makes testing and deployment more efficient and safe. A number of high-fidelity flight simulators exist, such as Microsoft Flight Simulator and X-Plane, but natively do not integrate with traditional deep learning and robotics pipelines.

In this work we present XPlaneROS that integrate a high-fidelity simulator with a state-of-the-art autopilot. The complete system enables the use of high-level or lower-level commands to control a general aviation aircraft in realistic world scenarios anywhere in the world. We chose X-Plane 11 as our simulator because of its open API and realistic aircraft models and visuals. For the lower-level control, we've integrated ROSplane as the autopilot. ROSplane is control stack for fixed-wing aircraft developed by the BYU MAGICC Lab.

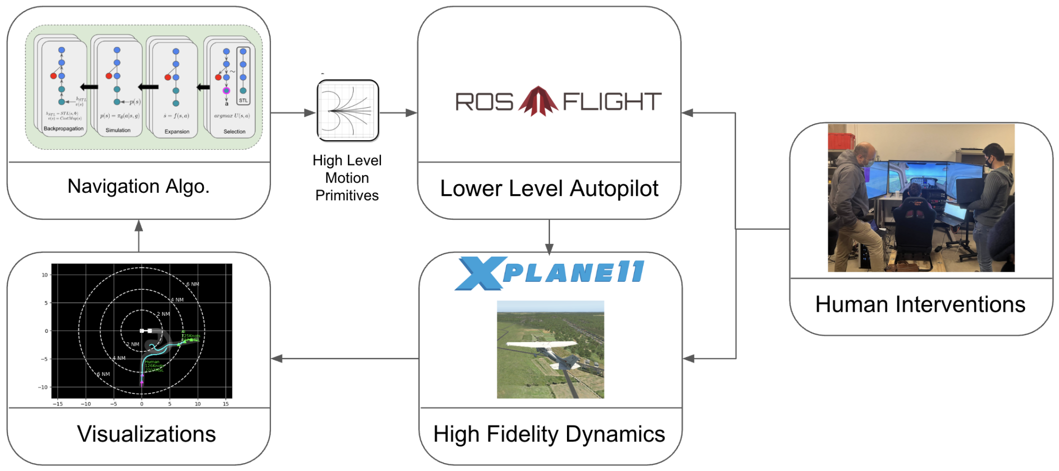


Figure 6.2: Overview of the XPlaneROS system. XPlaneROS builds on top of existing high-fidelity simulators like XPlane and use proven

6.6.2 System Overview

XPlaneROS interfaces with XPlane 11 using NASA's XPlaneConnect. With XPlaneROS, the information from XPlane is published over ROS topics. The ROSplane integration then uses this information to generate actuator commands for ailerons, rudder, elevator, and throttle based on higher-level input to the system. These actuator commands are then sent to XPlane through XPlaneConnect.

ROSplane uses a cascaded control structure and has the ability to follow waypoints with Dubin's Paths. XPlaneROS provides additional capabilities to follow a select set of motion primitives. There have also been some extensions to ROSplane like employing a proper takeoff, additional control loops for vertical velocity rates and a rudimentary autonomous landing sequence. Tuning the controllers can also be challenging and slow. To tackle this, a simple GUI utility is available via which users can give specific commands for roll, pitch etc and can tune the PID parameters based on the performance. Fig. 6.2 provides an overview of the simulator.

6.6.3 Code Availability

The repository for XPlaneROS can be found here. The accompanying README gives details on how to setup and run the codebase.

6.7 Conclusion

We present SoRTS, an algorithm for long-horizon social robot navigation. SoRTS is a MCTS-based planner which expands its search tree guided by an offline-trained intent prediction model and a global path which embodies navigation guidelines. We introduce X-PlaneROS, a high fidelity simulator for research in full-scale aerial autonomy. We use it to conduct a user study with experienced pilots to study our algorithm's performance in realistic flight settings. To the best of our knowledge, this is the first work in social navigation for general aviation and attempts to bring unique problems in general aviation within the purview of the larger robotics community.

Part III

[Unknown-Unknowns] Managing Out-of-Distribution Inputs

Chapter 7

RuleFuser: Injecting Rules in Evidential Networks

Citation:

Patrikar, Jay, Sushant Veer*[1], Apoorva Sharma*, Marco Pavone, and Sebastian Scherer. "RuleFuser: An Evidential Bayes Approach for Rule Injection in Imitation Learned Planners for Robustness under Distribution Shifts." International Symposium of Robotics Research (ISRR) 2024.

7.1 Introduction

Autonomous vehicles are increasingly venturing into complex scenarios that are common in dense urban traffic. Safely navigating these scenarios while interacting with heterogeneous agents like drivers, pedestrians, cyclists, etc. requires a sophisticated understanding of traffic rules and their impact on the behavior of these agents. However, many modern motion planners are developed by merely imitating trajectories from driving logs with no direct supervision on traffic rules. In fact, direct supervision of traffic rules is not practical as the training data would require examples of both traffic rule satisfaction and violation. Furthermore, the performance of such planners deteriorates in out-of-distribution (OOD) scenarios. On the other hand, rule-based

[1]*=equal advising

planners account for traffic rules and are more robust to distribution shifts, but they struggle with nuanced human driving behavior (which can often violate strict traffic rules) and multi-modal intents. In this paper, we develop a neural planning framework, RuleFuser, that combines the benefits of both learning-based and rule-based planners.

RuleFuser is inspired by the observation that, while learning-based planners can often outperform rule-based planners in in-distribution (ID) scenarios, they can behave much more erratically in out-of-distribution (OOD) scenarios. In order to address this, RuleFuser takes an evidential approach building on PosteriorNet [34]. Concretely, a rules-based planner (designed to comply with traffic rules) provides an informative prior over possible future trajectories an agent may take, which a learned neural network model subsequently updates to yield a data-driven posterior distribution over future trajectories. The strength of this update is controlled by an estimate of the *evidence* that the training data provides for a given future being associated with the scene context. In this way, in OOD scenarios, the posterior will remain close to the traffic-law compliant prior, but in for ID scenes with high representation in the training dataset, the powerful learning-based model is trusted to better capture the nuances of interactive driving. Importantly, as in PosteriorNet [34], RuleFuser only requires access to nominal driving logs and does not require exposure to OOD scenarios.

Statement of Contributions: The main contributions of this work are threefold: (i) We develop a novel method for injecting traffic rules in IL-based motion planners by leveraging evidential deep learning. This method permits altering the integration level between the IL and the rule-based planner without necessitating any re-training, thereby facilitating greater design flexibility. (ii) We introduce RuleFuser, a transformer-based neural framework that uses dynamic input anchor splines in a joint encoder with a posterior-net style normalizing flow decoder to estimate both epistemic and aleatoric uncertainty. We adopt the rule-based planner in [177] and use its output as a prior in a novel fusion strategy to provide robustness against OOD scenarios. (iii) We demonstrate the ability of RuleFuser as a planner and predictor to adapt to OOD scenarios on the real-world nuPlan [30] autonomous driving dataset and deliver safety levels exceeding those of learning- and rule-based frameworks alone.

7.2 Related Work

Our work builds on a rich literature on both rule-based trajectory prediction and planning, as well as uncertainty-aware learning-based trajectory forecasting and motion

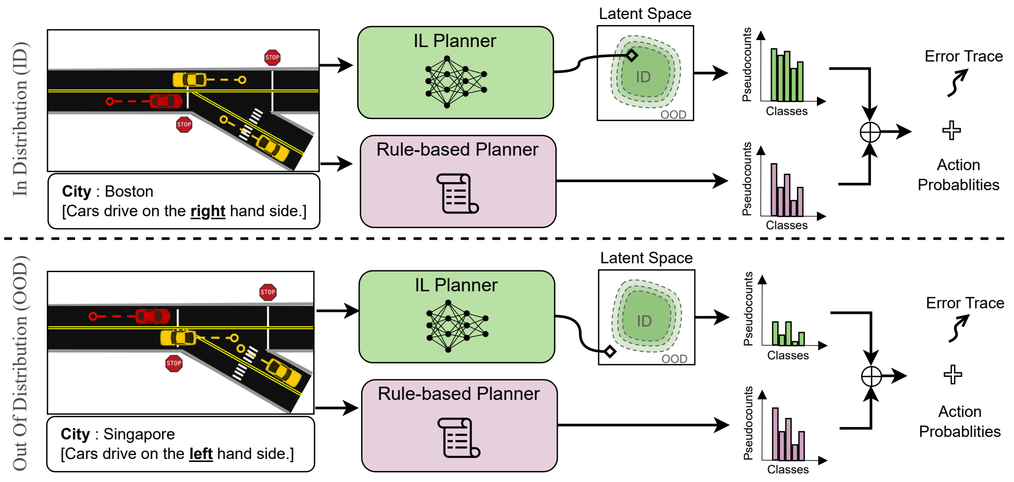


Figure 7.1: RuleFuser adopts a parallel setup with two planners: a learned uncertainty-aware IL planner and a rule-based planner. *(Top)* For in-distribution driving data, in this illustration Boston, the IL planner learns to map the input to higher likelihood areas in latent space, indicating higher evidence. This gives the IL planner more pseudo-counts to contribute towards the Bayesian posterior contribution, surpressing the impact of the rule-based prior. *(Bottom)* For out-of-distribution scenarios, in this case Singapore, the input is mapped to lower evidence, so the posterior is largely controlled by the rule-based prior.

planning models.

Integrating Traffic Rules in Learned Motion Planning. The behavior of road users is largely constrained by local traffic rules and customs. As such, it is appealing to leverage traffic law in trajectory prediction and planning. To do so requires both (i) *representing* traffic law, comprised of complex temporal rules as well as a variety exceptions depending on circumstances, as well as (ii) *utilizing* a representation of these rules in the motion planners. Various representations of the traffic law have been explored in the literature, such as natural language [94, 109], formal logic formulae [105, 49, 142], and hierarchical rules [175, 32, 172]. In this work, we lean on the hierarchical representation [175] due to its effectiveness in encoding a broader scope of traffic law while being flexible enough to permit traffic rule relaxations in the event of an exception, e.g., allowing speed limit violation to avoid collision. Despite the interpretability of purely rule-based methods for trajectory prediction and planning, these methods can struggle in corner cases that fail to align with modeling assumptions. Conversely, while learning-based methods sidestep these modeling assumptions, their output can be erratic, and sensitive to distribution shift. Recent work proposes several strategies attempting to achieve the best of both worlds by integrating traffic rule

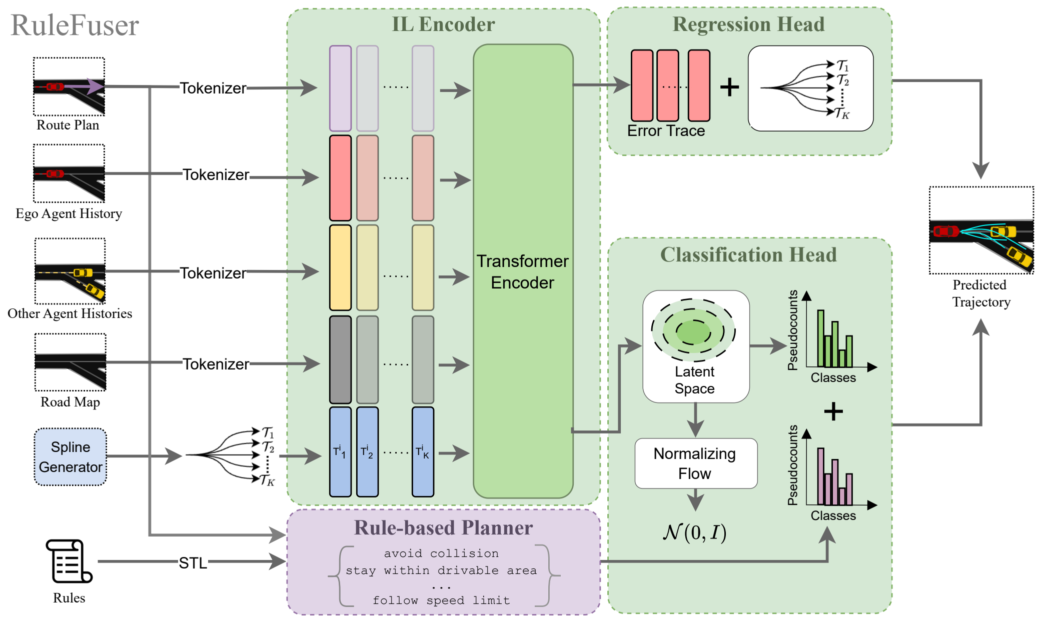


Figure 7.2: Overview of RuleFuser. A spline generator produces a set of dynamically generated candidate future trajectories. Each one of these trajectories is augmented with a copy of scene context inputs (ego and agent histories, map and route information), and are processed individually by a transformer-based encoder. The regression head outputs an error trace for each candidate, while the classification head estimate the the likelihood of each candidate under the training data distribution. The results are fused with prior psuedo-counts from a rule-based planner to yield the final prediction.

compliance into learned motion planning. One strategy involves integrating rules into the training process, by adding a reinforcement learning (RL) objective capturing a subset of rules like collision avoidance [103]. Similarly, [97] adds rule-supervision at train-time by training a network to predict rule-compliance of trajectory candidates in addition to predicting their likelihood, and using these predictions to re-weight candidates at inference time to favor candidates which satisfy rules over those that violate them; similarly themed approaches that explicitly use a traffic rule loss during training include [195, 194, 140, 100]. Other approaches incorporate rules only at inference-time. The *safety filter* strategy, where rule-based filter directly modifies the output of a learned planner to ensure constraint satisfaction, has seen application across learning-based planning and control (see [72] for a survey). Recently, similar strategies have been developed specifically for probabilistic planning architectures, where rule-compliance checks are used to steer the sampling process at inference time to prefer plans that meet safety constraints [8, 182, 183]. Overall, while these strategies are

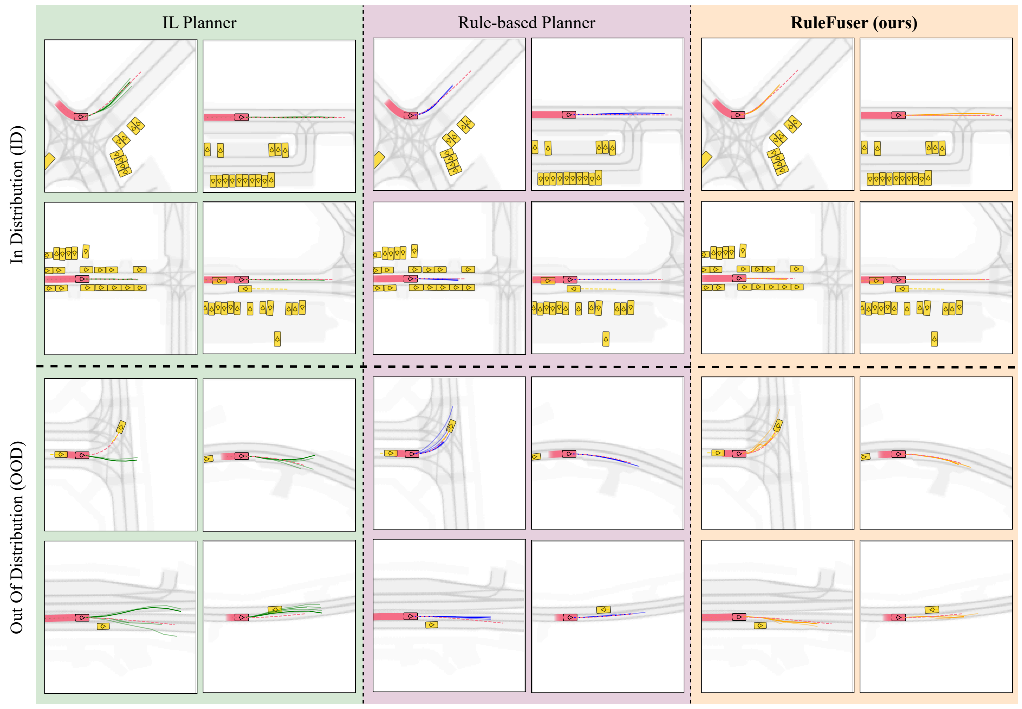


Figure 7.3: Figure shows the qualitative results for the three methods. While the predicted trajectories using Neural Predictor showcase good performance in Boston (ID), the performance deteriorates in the Singapore (OOD). Rule-aware Predictor has consistent performance in both ID and OOD but fails to capture nuances in speed and turning radius. RuleFuser shows consistent performance in both ID and OOD scenarios preferring higher performance in ID and falling back to safety in OOD.

effective in balancing the utility of rule-based and learned motion planning, the degree to which rule-compliance shapes the output plan at test time is static. In contrast, we propose a strategy which *dynamically* adapts the influence of rule-compliance by estimating the epistemic uncertainty of the learned model at test time.

Epistemic Uncertainty Quantification in Neural Networks. Our goal is to impose rule-based structure to learned motion planning only in those situations where we expect the learned model to perform poorly. Quantifying our confidence in the model is a question of estimating *epistemic uncertainty*, the uncertainty in predictions stemming from limited data. There is a rich literature on quantifying epistemic uncertainty in neural network models:

Bayesian methods, or approximations such as Monte-Carlo Dropout [60], Deep Ensembles [87], or variational inference based approaches [162, 150, 24, 104] aim to quantify epistemic uncertainty by modeling the distribution of models that are consistent with training data, but require additional computation beyond a single forward pass of the model. Other approaches aim to directly quantify epistemic uncertainty in the forward pass, either by reasoning about deviations from training data, either in the input or latent space [88, 199, 122], directly training the model to provide an estimate [174, 107, 108], or a combination [34]. Related to our proposed approach, [31] introduces PIETRA, which integrates physics priors into the theory of evidential neural networks. Also related is [77], which applies evidential learning to the problem of trajectory forecasting to create a model which falls back to an uninformed prediction over future agent behavior in out-of-distribution settings. This strategy does not directly extend to motion planning, as ultimately, the vehicle must pick a single plan to execute in any given setting. In this work, we address this limitation by leveraging a rules-based planner as an informative prior which grounds predictions on out-of-distribution scenarios.

7.3 Problem Setup and Preliminaries

Let $\boldsymbol{x} \in \mathcal{X}$ be the state of the ego agent, $\boldsymbol{y} \in \mathcal{Y}$ be the joint state for all other traffic agents in the scene, $\boldsymbol{s} \in \mathcal{S}$ be the map features (lane centerlines and road boundaries), and $\boldsymbol{r} \in \mathcal{R}$ be the desired route-plan (the lanes the ego desires to track in the future). Our goal is ego motion planning: Specifically, given the past behavior of the ego $\boldsymbol{x}_{t-H:t}$, other agents $\boldsymbol{y}_{t-H:t}$ and the map features and route plan $\boldsymbol{s}, \boldsymbol{r}$, our goal is to predict the future ego behavior $\boldsymbol{x}_{t:t+F}$, where H is the length of the past context available, and F is the horizon length for prediction. In sum, we would like a model to predict $p(\boldsymbol{x}_{t:t+F} \mid \mathcal{H})$, where we use $\mathcal{H} := (\boldsymbol{x}_{t-H:t}, \boldsymbol{y}_{t-H:t}, \boldsymbol{s}, \boldsymbol{r})$ as shorthand for all historical context available to the planner.

We structure the model's predictions by first generating a set of K anchor trajectories encoding possible futures $\{\mathcal{T}_k := \hat{\boldsymbol{x}}^k_{t:t+F}\}_{k=1}^K$ by exploiting the differential flatness of the bicycle dynamics model and fitting splines connecting the current ego state to different potential future states [161]. Choosing these splines reduces a high dimensional regression problem to a classification problem: now, our predictions take the form of a categorical distribution $\boldsymbol{q} \in \Delta^K$ over these anchor trajectories, where Δ^K represents the K-dimensional simplex.

We take a Bayesian perspective, and assume that we have a prior $\mathbb{P}(\boldsymbol{q} \mid \mathcal{H})$ over Δ^K conditioned on the scene context $\mathcal{H}$. How should we use our training data to update this prior? Suppose we treat each scene context independently, and that for a particular $\boldsymbol{y}_{t-H:t}$ and $\boldsymbol{s}$, we have N relevant examples in our training dataset. Let n_k represent the number of examples where anchor trajectory $\mathcal{T}_k$ corresponded to the true ego future, and let $\boldsymbol{n}$ be the vector of these counts. By Bayes rule,

$$\mathbb{P}(\boldsymbol{q} \mid \mathcal{H}, \boldsymbol{n}) \propto \mathbb{P}(\boldsymbol{q} \mid \mathcal{H}) \cdot \mathbb{P}(\boldsymbol{n} \mid \boldsymbol{q}, \mathcal{H}). \tag{7.1}$$

In this classification setting, $\mathbb{P}(\boldsymbol{n} \mid \boldsymbol{q}, \mathcal{H})$ is a multinomial distribution. The conjugate prior for a multinomial distribution is a Dirichlet distribution. This means that if we parameterize the prior as $\mathrm{Dir}(\boldsymbol{\beta}_{\text{prior}})$, then the posterior according to (7.1) will also be a Dirichlet distribution, $\mathrm{Dir}(\boldsymbol{\beta}_{\text{post}})$ where $\boldsymbol{\beta}_{\text{post}} = \boldsymbol{\beta}_{\text{prior}} + \boldsymbol{n}$. We can see that the number of matching examples $N = \mathbf{1}^T \boldsymbol{n}$, also known as the total *evidence*, controls the influence of the prior on the posterior prediction. Furthermore, this expression highlights why the parameter $\boldsymbol{\beta}_{\text{prior}}$ can be interpreted as *pseudo-counts* for each class. In reality, as $\boldsymbol{y}_{t-H:t}$ and $\boldsymbol{s}$ are continuous, we will not have any exact matching scenarios in our training dataset. However, scenarios are not independent from one another: training examples from similar scenes should influence our predictions on new scenes. Thus, while the Bayesian update logic above may not directly apply, we can nevertheless achieve a similar behavior by training a neural network model to estimate the evidence $\boldsymbol{n}$. Indeed, as in [34], we leverage a latent-space normalizing flow model to estimate the evidence n_k for each anchor trajectory, thereby ensuring that in order to assign higher evidence to the regions in the latent space covered by the training data, we must take evidence away from other regions of the latent space. In this way, in-distribution $(\mathcal{H}, \mathcal{T}_k)$ pairs receive higher evidence, while out-of-distribution pairs are assigned lower evidence.

In this work, we propose using a rule-based planner to compute the prior pseudo-counts $\boldsymbol{\beta}_{\text{prior}}$. As a result, if the scene is OOD, then the total evidence $\|\boldsymbol{n}\|_1$ is small, and $\boldsymbol{\beta}_{\text{prior}}$ dominates the final prediction, resulting in a smooth fallback to a rule-based planner. Conversely, if the scene is ID, then $\|\boldsymbol{n}\|_1$ is larger, and the final posterior relies more on the learnt network. Fig. 7.1 illustrates the framework for both ID and OOD scenarios.

7.4 RuleFuser Framework

In this section we introduce RuleFuser, a concrete implementation of the ideas outlined above; see Fig. 2.1 for an architectural overview of RuleFuser. For further details on implementation we would like to point the readers to the ArXiv [138] version.

7.4.1 Rule-Hierarchy (RH) Planner

The Rule-Hierarchy planner, henceforth referred to as RH planner, is tasked with defining a prior distribution over trajectories in accordance with their compliance with a prescribed set of traffic rules. RH planner builds on the implementation in [177], and we detail its components below:

Route Planner. The route planner generates a reference polyline by determining a sequence of lanes for the ego vehicle to follow in order to reach a desired goal location. It then combines the centerlines of these lanes into a single reference polyline. The route planner is unaware of obstacles and other agents as well as their dynamics, it is only aware of the ego's intended position a few seconds ahead into the future and the lane graph which is made available from an HD map or an online mapping unit. The route planner chooses the nearest lane to the ego's future goal position and performs a depth-first backward tree search to the ego's current position.

Rules. We express the traffic rules in the form of a rule hierarchy [175] expressed in signal-temporal logic (STL) [106] using STLCG [91]. Rule hierarchies permit violation of the less important rules in favor of the more important ones if all rules cannot be simultaneously met. This flexibility allows for more human-like behaviors [70]. Our rule hierarchy consists of seven rules in decreasing order of importance: (i) `avoid collision`; (ii) `stay within drivable area`; (iii) `follow traffic lights`; (iv) `follow speed limit`; (v) `forward progression`; (vi) `stay near the route plan`; and (vii) `stay aligned with the route plan`. This 7-rule hierarchy expands on the 4-rule hierarchy in [177]. Note that for evaluating the `avoid collision` rule, similar to [177], we use a constant-velocity predictor to predict the future of non-ego agents in the scene.

Trajectory Evaluation. The rule hierarchy comes equipped with a scalar reward function R which measures how well a trajectory adheres to the specifications provided in the hierarchy. The exact specifics and construction of the reward function is beyond the scope of this paper; see [175] for more details.

Prior Computation. Let $\{R_1, \cdots, R_K\}$ be the reward for the K anchor trajectories. We transform these rewards into a Boltzmann distribution by treating the rewards as the negative of the Boltzmann energy. For each trajectory i,

$$[\boldsymbol{p}]_k = \frac{\exp(R_k/\zeta)}{\sum_{k'=1}^{K}\exp(R_{k'}/\zeta)}, \tag{7.2}$$

where ζ is the Boltzmann temperature. Now, we use this distribution to define a prior by assigning pseudocounts in proportion to the probability assigned to each anchor trajectory. Let N_{prior} be a hyperparameter representing a budget on the counts. Then, we choose, $\boldsymbol{\beta}_{\text{prior}} := N_{\text{prior}} \cdot \boldsymbol{p}$ which is then used to define the prior as $\mathbb{P}(\boldsymbol{q} \mid \mathcal{H}) := \text{Dir}(\boldsymbol{\beta}_{\text{prior}})$. A large N_{prior} will require more evidence to diverge away from the prior, relying more on the RH planner, and vice-versa. Effectively, we can control how much we want to trust the rule-based prior by controlling N_{prior}; we will study the effect of varying this parameter in Sec. 7.5.

7.4.2 Evidential Neural Planner

To complement the rule-based planner, we build an evidential learned model utilizing a Transformer-based architecture which has demonstrated strong performance in learned trajectory prediction and planning [163, 128, 127]. The network takes in the scene context as well as the same K anchor trajectories as the RH planner, and is tasked with estimating the *evidence* the training dataset assigns to each candidate future given the scene context. The network consists of an encoder which maps the scene context and candidate future trajectory into a latent space, as well as a decoder, which uses a normalizing flow model to assign evidence to each trajectory candidate and additionally estimate an error trace to fine-tune the raw spline-based anchor trajectories. See Fig. 2.1 for an overview.

Encoder. The encoder takes as input the position history of the ego $\boldsymbol{x}_{t-H:t}$, as well as that of nearby agents in the scene $\boldsymbol{y}_{t-H:t}$. In addition, the network has access to the scene geometry $\boldsymbol{s}$ in the form of lane centerlines, road boundaries, and route information. In addition, we also provide the network with the same K anchor trajectories, $\{\mathcal{T}_k\}_{k=1}^{K}$, as the rule-based planner to serve as choices of ego future behavior. First, all inputs are lifted into a d-dimensional space, using a learned linear map on agent trajectories, and applying PointNet [146] on map elements which correspond to a sequence of points (e.g., polylines). With all inputs now taking the form of sets of d-dimensional vectors, we apply a sequence of Transformer blocks with factorized attention to encode these inputs. Specifically, we first encode the scenario history by stacking the ego history with the neighbor history, and applying attention layers which sequentially perform self-attention across time, self-attention across agents, and finally, cross-attention with the map features. As we are only interested in ego future prediction, we keep only the features corresponding to the ego history. We apply encoding operation on the ego future candidates, and concatenate the resulting future features with the encoded ego history feature in the time dimension, yielding an embedding of shape $[K, H+F, d]$, where $H+F$ is the total number of timesteps, K is the number of candidates, and d is the latent feature dimension. We apply the same sequence of factorized attention on these features to allow history features to interact with future candidate features. Note, there is no attention across different future candidates; each candidate is encoded independently. The result is a set of features, $\mathcal{Z}$, of shape $[K, H+F, d]$

Decoder. The decoder is responsible for predicting the evidence, $\boldsymbol{n}$, to assign to each candidate as well as to predict an error trace for each anchor trajectory. The regression head operates directly from the temporal features, applying an MLP independently for each time-step and anchor trajectory, mapping the d-dimensional feature to the 2-dimensional error vector representing the perturbation to the original spline-based trajectory. To compute the evidence, we first average-pool the features along the temporal dimension, resulting in a d-dimensional feature per future candidate. Then, an MLP projects this feature to a lower, d_{flow}-dimensional space, $\boldsymbol{z}_k = \text{MLP}(\text{MeanPool}(\mathcal{Z}_k))$. The evidence assigned to each candidate is computed using a normalizing flow with parameters ψ applied over this latent space, $[\boldsymbol{n}]_k = N \cdot p_\psi(\boldsymbol{z}_k)$, where N is an evidence budget which we set to the size of the training dataset, following [34].

Training Loss To train this model, we follow prior work on training mixture models for trajectory prediction and train for classification and regression independently. Specifically, we impose a regression loss using a masked mean-square-error (MSE) loss function, applied only to the error trace of the mode closest to the ground truth:

$$\mathcal{L}_{MSE} = \sum_{k=1}^{K} \mathbf{1}_{k=k^*} \cdot ||(\boldsymbol{e}^k_{t:t+F} + \mathcal{T}_k) - \boldsymbol{x}_{t:t+F}||_2^2.$$

where $k^* = \arg\min_k ||\mathcal{T}_k - \boldsymbol{x}_{t:t+F}||_2^2$.

To train the classification head, we first convert the predicted evidence $\boldsymbol{n}$ into a marginal distribution over classes, $\boldsymbol{q} = \boldsymbol{n}/\mathbf{1}^T\boldsymbol{n}$. Then, following [34], we construct a training objective by combining a classification loss on the marginal prediction $\boldsymbol{q}$ with a penalty on evidence assigned to incorrect modes. Specifically, we use a binary cross entropy loss (independently over the anchor trajectories), with an entropy reward to discourage assigning evidence to incorrect classes:

$$\mathcal{L}_{UCE} = \sum_{k=1}^{K} \mathrm{BCE}(\bar{q}_k, \mathbf{1}_{k=k^*}) - H(\boldsymbol{n})$$

where BCE is the binary cross entropy loss, and $H(\boldsymbol{n})$ is the entropy of the Dirichlet distribution. The parameters of the embedding networks, transformer encoder, decoder networks, and normalizing flow layers are all optimized through stochastic gradient descent to optimize a weighted sum of $\mathcal{L}_{MSE}$ and $\mathcal{L}_{UCE}$.

7.4.3 Bayesian Fusion Strategy

At inference time, our model fuses the outputs of the RH planner and the Evidential Neural Planner to produce an ultimate probabilistic prediction for the future agent behavior. First, we apply Bayes rule using our estimated evidence $\boldsymbol{n}$ to compute the posterior, here simplifying to $\mathbb{P}(\boldsymbol{q} \mid \mathcal{H}, \boldsymbol{n}) = \mathrm{Dir}(\boldsymbol{\beta}_{\text{post}})$, where $\boldsymbol{\beta}_{\text{post}} = \boldsymbol{\beta}_{\text{prior}} + \boldsymbol{n}$. Incorporating the regression outputs, the posterior over the ego future behavior ultimately takes the form of a Dirichlet-Normal distribution:

$$\boldsymbol{q} \sim \mathrm{Dir}(\boldsymbol{\beta}_{\text{post}}), \qquad k \mid \boldsymbol{q} \sim \mathrm{Cat}(\boldsymbol{q}), \qquad \boldsymbol{x}_{t:t+F} \mid k \sim \mathcal{N}(\mathcal{T}_k + \boldsymbol{e}^k_{t:t+F}, \boldsymbol{I}),$$

where Cat indicates the categorical distribution and $\mathcal{N}(\mu, \Sigma)$ indicates the normal distribution with mean μ and covariance Σ. Marginalizing over $\boldsymbol{q}$, the final predictive distribution of our model can be viewed as a Mixture of Gaussians:

$$\bar{\boldsymbol{q}} = \mathbb{E}[\boldsymbol{q}] = \boldsymbol{\beta}_{\text{post}} / (\mathbf{1}^T \boldsymbol{\beta}_{\text{post}})$$
$$p(\boldsymbol{x}_{t:t+F} \mid \mathcal{H}) = \sum_{k=1}^{K} \bar{p}_k \cdot \mathcal{N}(\boldsymbol{x}_{t:t+F}; \mathcal{T}_k + \boldsymbol{e}^k_{t:t+F}, \boldsymbol{I})$$

While probabilistic interpretation can be useful for trajectory prediction, when applying this model as a planner we simply return the mode with the highest posterior probability: $\mathcal{T}_{k^*} + \boldsymbol{e}^{k^*}_{t:t+F}$ where $k^* = \arg\max \bar{\boldsymbol{q}}$.

7.5 Experiments

In this section, we demonstrate the ability of RuleFuser to plan motions that balance safety and imitation in both ID and OOD data.

7.5.1 Datasets and Implementation Details

We evaluate RuleFuser on the NuPlan [30] dataset, which provides labeled driving data from multiple cities around the world. We train models on data from individual cities, and subsequently test on both held-out data from the same city (ID test data), and data from another city (OOD test data). This set-up allows us to specifically test a realistic distribution shift that may arise as AVs are deployed beyond the geographical region from which training data was collected. We consider two cities for our tests, Boston and Singapore, which differ in road geometry, traffic density, and driving convention (right-side vs left-side).

For each city, RuleFuser is trained end-to-end on the chosen train split of the NuPlan dataset. The model is trained on 8xA100 GPUs. Final network weights are chosen by their performance on the validation set from the same city.

7.5.2 Planners

We present results for three categories of planners: IL planner, RH planner, and RuleFuser. IL planner mirrors the architecture of RuleFuser differing only in the choice of the prior on the anchor trajectories which is always set to be uniform. Effectively, the neural model of the IL planner is “blind" to the traffic rules and relies solely on the training data. The RH planner is the rules-based planner described in Sec. 7.4.1 that operates exclusively according to user-defined traffic rules without utilizing data. RuleFuser integrates both approaches, as described in Sec. 7.4, leveraging both the training data and user-defined rules.

7.5.3 Metrics

We use multiple metrics to compare the performance of the three planners. We separate metrics into two categories: imitation and safety.

Imitation Metrics

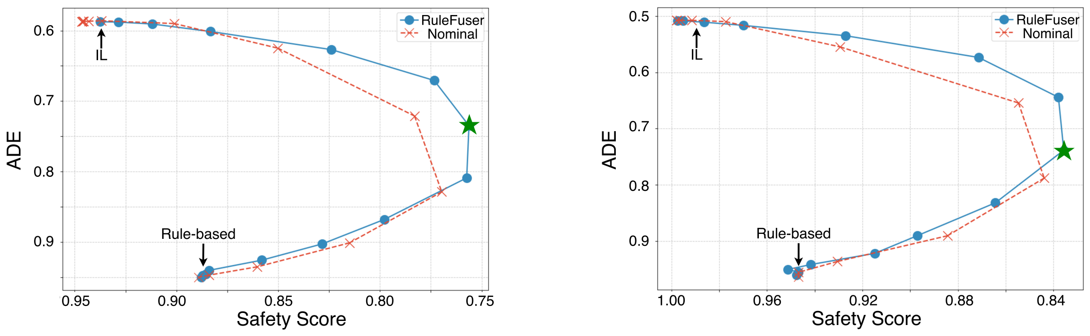
- **ADE** (Average Displacement Error) is the average Euclidean distance between the top predicted plan and the ground truth trajectory.
- **FDE** (Final Displacement Error) is the smallest Euclidean distance between the terminal position of the top predicted plan and the terminal position of the ground truth trajectory.
- **pADE/pFDE** (Probability Weighted Average/Final Displacement Error) is the sum of the ADE/FDE over all K plans, each weighted by the final categorical distribution over anchor trajectories $\bar{\boldsymbol{q}}$.
- **Acc.** (Accuracy) is the fraction of predicted class labels that match the ground truth class label.

Safety Metrics

- **Percentage Collision Rate** is the percentage of times the plan resulted in a collision with the ground truth future trajectory of another traffic agent.
- **Percentage Off-Road Rate** is percentage of times the plan exited the road boundaries.
- **Safety Score** is an aggregate safety score based on the rank of the planned trajectory under a 2-rule rule hierarchy [175, Definition 1] with collision avoidance and offroad avoidance as the two rules, listed in decreasing order of importance. The best rank a trajectory can get is 1 and the worst is 4. The safety score, which lies between 0 and 100, is computed by transforming the ranks as follows: $(100 \times (\text{rank} - 1))/3$ where the lower safety score indicates a better safety outcome.

7.5.4 Results

We present results for two different experimental setups: In one, Boston serves as the ID dataset and Singapore as the OOD dataset, while in the other, Singapore serves as the ID dataset while Boston as the OOD dataset. For both these setups, we train the IL planner on the train split of the ID dataset and evaluate the IL planner, RH planner, and 15 instantiations of RuleFuser with different N_{prior} on the test splits



(a) Trained in Boston (b) Trained in Singapore

Figure 7.4: RuleFuser can generate an entire imitation-safety Pareto frontier, shown in this plot on combined ID and OOD test sets. The horizontal axis (safety score) decreases to the right, and the vertical axis (ADE) decreases upwards, so points further up and to the right are better. The bulge to the right indicates that mixing of the planners via RuleFuser results in better safety. RuleFuser also outperforms the nominal mixing model, indicating the importance of the evidential mixing strategy.

of both the ID and OOD datasets. In Fig. 7.4 we plot the ADE and safety score of RuleFuser on the combined test splits of ID and OOD datasets for each N_{prior}. In Table 7.1 we report detailed metrics for the instantiations of RuleFuser that exhibited the best safety score on the validation split of the respective ID datasets for both the setups; these instantiations are denoted by a star in Fig. 7.4.

7.5.5 Discussion

Our experiments provide two key findings: (i) RuleFuser provides an effective way to balance imitation and safety; (ii) RuleFuser exhibits safer motion plans than the IL and RH planners standalone in both ID and OOD datasets. In the rest of this section we will elaborate on these findings.

RuleFuser balances between imitation and safety.

RuleFuser provides a convenient way to balance imitation and safety, allowing practitioners to control the trade-off between the two through the N_{prior} hyperparameter. Fig. 7.4 plot an imitation-safety Pareto frontier by sweeping N_{prior} across a range of values. Points on the far left of the frontier are proximal to the IL planner and exhibit good imitation (lower ADE) while points on the far right are proximal to the RH

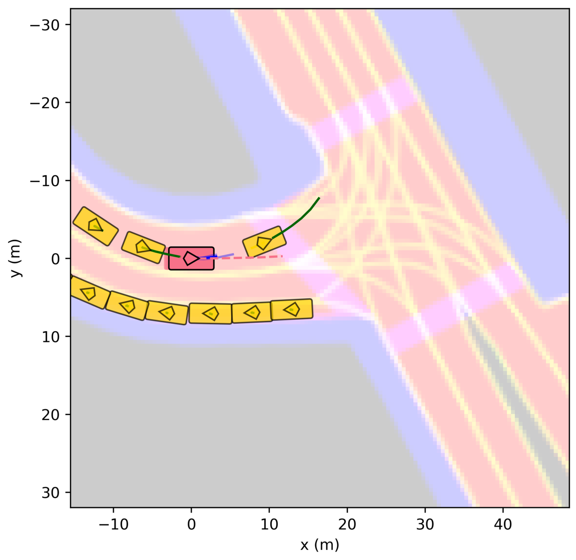


Figure 7.5: Rear-end collision due to constant-velocity prediction. RH Planner plans the blue trajectories for the ego vehicle (pink box). Ground truth trajectory of non-ego vehicles (yellow box) is in green.

planner and exhibit good safety. Notably, both the Pareto plots significantly bulge to the right indicating that mixing of the two approaches can improve overall safety beyond what a standalone planner can achieve without significantly hurting imitation. Furthermore, unlike methods which incorporate rules directly into the training objective [103], RuleFuser does not require retraining to explore different points on the Pareto frontier. This makes our approach more computationally efficient for rule injection in IL planners.

To study the impact of our evidential mixing strategy for fusing IL and RH planners, we compare our approach with a nominal mixing strategy which, unlike RuleFuser, does not consider similarity of a scene to the training distribution. The nominal strategy performs a simple convex combination of the categorical distribution over candidates output by both planners, $\boldsymbol{q} := \lambda \boldsymbol{q}_{\mathrm{IL}} + (1-\lambda)\boldsymbol{q}_{\mathrm{RH}}$, $\lambda \in [0, 1]$. The Pareto frontiers for the nominal mixing strategy are depicted with dashed lines in Fig. 7.4. The Pareto frontier for RuleFuser is further to the right of the nominal mixing model, validating the utility of the OOD-aware mixing strategy of RuleFuser.

Impact of RuleFuser in ID and OOD datasets.

As expected from an imitation-learned policy, the IL planner excels on imitation metrics in the ID datasets. Initially we anticipated the RH planner to outperform the IL planner on safety metrics. While the RH planner did indeed outperform the IL planner on offroad rate, it surprisingly underperformed on collision metrics; see Table 7.1. Upon closer inspection we discovered that the underlying cause for RH planner's high collision rate was its reliance on the constant-velocity predictor which fails to account for non-ego agents' acceleration/deceleration; see Fig. 7.5 for an example where ego does not plan to move forward sufficiently fast to avoid a rear-end collision due to the constant-velocity prediction for the agent behind it. An interesting insight that arises from this study is that when operating within distribution, IL approaches tend to outperform rules-based approaches when dealing with phenomenon that are challenging to model via first-principle rules, such as predicting the future trajectories of other agents. Offroad rate, on the other hand, is devoid of such modeling complexities and therefore, sees better performance from the RH Planner.

For both ID and OOD datasets, RuleFuser always lies between the IL planner and RH planner on imitation metrics; however, RuleFuser consistently achieves the *best* safety metrics[2]. In particular, for the OOD scenarios in Tables 7.1(a) and (b), RuleFuser achieves a 58.18% and 18.67% improvement over the IL planner, respectively (i.e., an average of 38.43% improvement). This suggests that, in OOD scenarios, RuleFuser effectively combines the two models to produce an overall improvement over the standalone models. This is the outcome of RuleFuser's "intelligent" mixing of the models which favors the model that is more suitable for a given scenario. Indeed, not every scenario in the OOD dataset will be outside the domain of competency of the IL planner; for example, a vehicle going straight and following a lane will behave similarly in both Boston and Singapore. Consequently, there are many scenarios in the OOD dataset where it is still preferable to trust the IL planner. Indeed, as shown in Table 7.1, RuleFuser does not assign zero evidence in the OOD datasets. Remarkably, RuleFuser in the Singapore (OOD) dataset in Table 7.1(a) achieves a safety score of 0.23 outperforming the IL planner trained *specifically* on the Singapore (ID) dataset in Table 7.1(b) with a safety score of 0.31, providing further evidence for the ability of RuleFuser to outperform the standalone planners on safety.

[2]Interestingly, in the Singapore dataset the collision rate for all planners is lower than the Boston dataset. This is the result of the much lower traffic density in Singapore with an average of 3.23 neighboring vehicles per scene as compared to Boston with an average of 11.26 neighboring vehicles per scene.

(a) Boston (ID), Singapore (OOD)

Metrics	Imitation Metrics			Safety Metrics			OOD
	ADE/FDE ↓	pADE/pFDE ↓	Acc. ↑	% Coll. Rate ↓	% Off. Rate ↓	Saf. Score ↓	Evi. $(1^T n)$
Boston (ID):							
IL Planner	**0.41/0.93**	**0.46/1.02**	**0.75**	0.76	0.10	0.55	–
RH Planner	1.04/2.25	1.26/2.8	0.09	1.12	**0.03**	0.76	–
RuleFuser	0.70/1.55	0.98/2.22	0.56	**0.74**	0.05	**0.51**	10.4e+6
Singapore (OOD):							
IL Planner	**0.76/1.72**	**0.82/1.83**	**0.37**	0.59	0.45	0.55	–
RH Planner	1.08/2.32	1.34/2.96	0.16	0.36	0.01	0.25	–
RuleFuser	0.96/2.09	1.18/2.64	0.30	**0.34**	**0.01**	**0.23**	3.5e+6

(b) Singapore (ID), Boston (OOD)

Metrics	Imitation Metrics			Safety Metrics			OOD
	ADE/FDE ↓	pADE/pFDE ↓	Acc. ↑	% Coll. Rate ↓	% Off. Rate ↓	Saf. Score ↓	Evi. $(1^T n)$
Singapore (ID):							
IL Planner	**0.44/0.97**	**0.48/1.08**	**0.67**	0.39	0.14	0.31	–
RH Planner	1.08/2.32	1.34/2.96	0.16	0.36	0.01	0.25	–
RuleFuser	0.80/1.75	1.09/2.47	0.44	**0.29**	**0.01**	**0.20**	10.7e+5
Boston (OOD):							
IL Planner	**0.57/1.30**	**0.62/1.43**	**0.45**	1.05	0.15	0.75	–
RH Planner	1.04/2.25	1.26/2.8	0.09	1.12	**0.03**	0.76	–
RuleFuser	0.85/1.87	1.07/2.42	0.33	**0.90**	**0.03**	**0.61**	8.5e+5

Table 7.1: Quantitative performance metrics for the IL planner, RH planner, and RuleFuser on ID and OOD datasets. The planned trajectories have a horizon of 3 seconds. RuleFuser's performance on IL metrics always lies between the IL planner and the RH planner, but it always outperforms IL and RH planners on the safety score. Effectively, RuleFuser provides greater safety without significant detriment to imitation.

7.6 Conclusion and Future Work

This work introduced RuleFuser, a novel motion planning framework for balancing imitation and safety by integrating an IL planner with a rule-based (RH) planner using an evidential Bayes approach. Both planners used a common set of motion plan candidates generated on-the-fly. The RH planner provided a prior distribution over these candidates, which RuleFuser combined with the IL planner's predictions. The fusion, guided by an OOD measure using normalizing flows, produced a posterior distribution on the motion plan candidates; the more OOD the scene, the more the posterior aligns with the prior. On real-world AV datasets, we demonstrated RuleFuser's ability to leverage the data-driven IL planner in ID scenes and the "safer" rule-based planner in OOD scenes.

This work opens up several exciting future directions: (i) enhancing the fusion of the IL and RH planners by incorporating additional evidence beyond just the OOD measure, such as historical performance and situation criticality; (ii) conducting thorough closed-loop evaluation of RuleFuser in nuPlan; and (iii) investigating the impact of rule injection via RuleFuser in end-to-end planning networks like ParaDrive [186] and UniAD [74].

Part IV

[Unknown-Knowns] Mitigating Latent Failure Modes

Chapter 8

Distilling Symbolic Rules via Inductive Logic Programming from Text and Trajectory Data

8.1 Introduction

Neuro-symbolic artificial intelligence integrates data-driven models with the structured reasoning of symbolic systems. This fusion aims to leverage the strengths of both approaches: the adaptability and learning efficiency of machine learning models, and the interpretability and logical rigor of symbolic reasoning. By combining these methodologies, neuro-symbolic AI aspires to build systems capable of both learning from data and reasoning about it in a human-understandable manner [112, 181, 25].

A significant advantage of neuro-symbolic AI is its potential to mitigate issues like hallucinations—instances where AI systems generate plausible but incorrect information. While data-driven models can be prone to such errors due to their statistical nature, incorporating symbolic reasoning provides a layer of verification and consistency. For example, integrating knowledge graphs allows AI systems to cross-verify outputs against structured data, reducing the likelihood of generating factually incorrect information [79]. This combination enhances the reliability and trustworthiness of AI systems, especially in applications requiring high levels of accuracy and explainability. A critical application of neuro-symbolic AI is in rule distillation, which involves extracting interpretable rules from complex models or data [131]. This process is essential for domains like aviation [125] and self-driving [137] where understanding the decision-making process is as important as the decisions themselves. Rule distillation enables the translation of intricate patterns learned by neural networks into human-readable rules, enhancing transparency and trust in AI systems .

Within the realm of symbolic AI, Inductive Logic Programming (ILP) serves as a method for learning logical rules from observed data [41]. Unlike deductive reasoning, which applies general rules to specific cases, inductive reasoning involves generalizing from specific instances to broader rules. This approach is inherently more complex due to the vast space of possible generalizations and the need to infer rules that accurately capture underlying patterns. ILP systems like Popper [42] address this challenge by employing a generate-test-constrain loop, systematically exploring hypotheses and refining them based on their performance against training data.

In the context of aviation, ensuring safety on airport surfaces is paramount. Despite technological advancements, incidents such as runway incursions and near-misses continue to occur. Systems like the Airport Surface Detection Equipment, Model X (ASDE-X) [57], provide controllers with real-time surveillance data to monitor aircraft movements. However, these systems have limitations, including coverage restricted to certain airports and a lack of direct alerts to pilots . As air traffic increases, the risk of surface incidents may rise, highlighting the need for more comprehensive and interpretable safety monitoring solutions. By applying neuro-symbolic AI and ILP, we can develop systems that not only detect potential hazards but also provide understandable explanations, enhancing decision-making and safety in aviation operations.

In this work, we introduce a neuro-symbolic pipeline that integrates large language models (LLMs) with inductive logic programming (ILP) system to distill interpretable first-order logic (FOL) rules for monitoring airport surface operations. Our approach leverages both historical incident narratives and routine operational data to iteratively learn and refine safety rules, ensuring consistency and explainability in high-stakes aviation environments. We present results on real-world scenarios that demonstrate the ability of the proposed system to identify rule breaks in real-world interaction data.

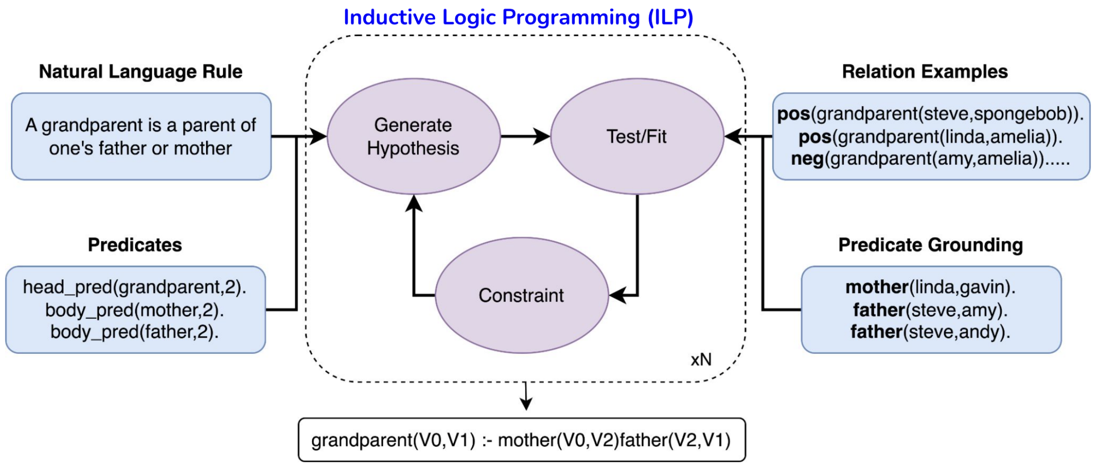

Inductive Logic Programming (ILP)
Natural Language Rule
A grandparent is a parent of one's father or mother
Predicates
head_pred(grandparent,2). body_pred(mother,2). body_pred(father,2).
Generate Hypothesis
Test/Fit
Constraint
xN
Relation Examples
pos(grandparent(steve,spongebob)). pos(grandparent(linda,amelia)). neg(grandparent(amy,amelia)).....
Predicate Grounding
mother(linda,gavin). father(steve,amy). father(steve,andy).
grandparent(V0,V1) :- mother(V0,V2)father(V2,V1)

Figure 8.1: Overview of Inductive Logic Programming System

8.2 Approach

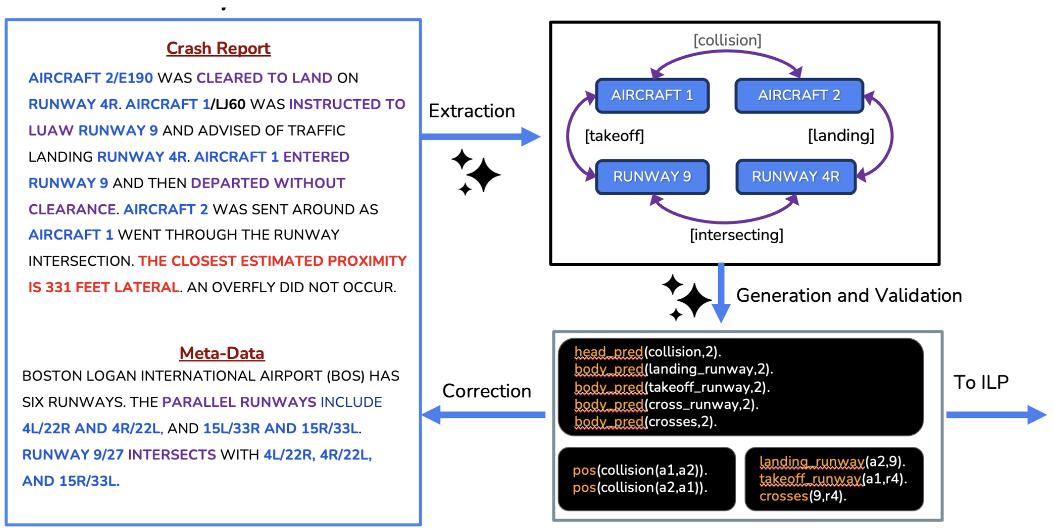
The approach has three main components: the ILP Symbolic System, the crash report based negative data mining and the trajectory based positive mining.

8.2.1 Inductive Logic Programming System

Irrespective of choice of the underlying solver, most ILP systems need three core components for reasoning:

1. Bias: Defines the hypothesis space by specifying constraints such as allowed predicates, clause structures, and recursion depth.
2. Background Knowledge (BK): A set of known facts and rules that provide context for learning.
3. Examples: Labeled data comprising positive and negative instances that the system uses to induce general rules.

These components are fundamental to ILP systems, enabling them to learn logical rules that generalize from the provided examples within the context of the background knowledge and under the constraints defined by the bias. Figure 8.1 shows how these interact in the Popper loop.


Crash Report

AIRCRAFT 2/E190 WAS CLEARED TO LAND ON RUNWAY 4R. AIRCRAFT 1/LJ60 WAS INSTRUCTED TO LUAW RUNWAY 9 AND ADVISED OF TRAFFIC LANDING RUNWAY 4R. AIRCRAFT 1 ENTERED RUNWAY 9 AND THEN DEPARTED WITHOUT CLEARANCE. AIRCRAFT 2 WAS SENT AROUND AS AIRCRAFT 1 WENT THROUGH THE RUNWAY INTERSECTION. THE CLOSEST ESTIMATED PROXIMITY IS 331 FEET LATERAL. AN OVERFLY DID NOT OCCUR.

Meta-Data

BOSTON LOGAN INTERNATIONAL AIRPORT (BOS) HAS SIX RUNWAYS. THE PARALLEL RUNWAYS INCLUDE 4L/22R AND 4R/22L, AND 15L/33R AND 15R/33L. RUNWAY 9/27 INTERSECTS WITH 4L/22R, 4R/22L, AND 15R/33L.

```
head_pred(collision,2).
body_pred(landing_runway,2).
body_pred(takeoff_runway,2).
body_pred(cross_runway,2).
body_pred(crosses,2).
```

```
pos(collision(a1,a2)).
pos(collision(a2,a1)).
```

```
landing_runway(a2,9).
takeoff_runway(a1,r4).
crosses(9,r4).
```

Figure 8.2: Figure shows the various steps in extracting the required ILP files from the carsh data.

In order to aid the grounding, we use predefined predicates that cover the airport example. The list of predicates and their meanings for the domains under consideration are part of the LLM extraction prompt.

8.2.2 Crash Report based Negative Data

Negative examples are derived from unstructured aviation safety reports, such as the NASA Aviation Safety Reporting System (ASRS) or Aviation Safety Information Analysis and Sharing (ASIAS). These narratives are processed using LLMs to extract relevant entities and events, which are then transformed into grounded predicates based on a predefined vocabulary of airport operations. As shown in Fig. 8.2 , the extraction takes place in multiple steps.

Addition of Meta-Data

In order to supplement the information included in the crash reports, we also provide additional information like runway namings, geometries, orientations and airport configuration. This is appended to the crash report as meta data.

Example metadata for John Wayne Airport (SNA) in Orange County, California is

```
John Wayne Airport (SNA) in Orange County, California, features two
closely spaced parallel runways: the main runway, 2L/20R, measuring
5,700 feet (1,737 meters) in length and 150 feet (46 meters) in width,
accommodates both commercial and general aviation aircraft; the shorter
runway, 2R/20L, at 2,887 feet (880 meters) long and 75 feet (23 meters)
wide, serves general aviation and smaller aircraft.  Both runways are
oriented in a north-south direction, with 2L/20R equipped with an
Instrument Landing System (ILS) to assist in low-visibility conditions.
Due to their parallel configuration and staggered thresholds, pilots
must exercise caution during operations, particularly when taxiing
across runways.
```

Figure 8.3: Example metadata for John Wayne Airport (SNA) in Orange County, California.

shown in Fig. 8.3.

Entity and Relation Extraction

For each crash report, we first have the LLM extract the entities involved and provide relations between them. The entities are labelled by types (eg Runway, Vehicle, Aircraft etc) and their associated relation with other entities if that exists. This step also serves as the intermediate Chain-Of-Thought step before the actual logic grounding and extraction.

Context Extraction

Once the entity extraction is completed, the LLM is prompted to generate the bias files, the example, and the background files. The LLM is also prompted to not generate any new predicates but to stick to the master list. In addition to the prompt we also provide multiple in-context examples to aid the process. To keep the LLM query costs down, we do steps 1-3 in the same LLM API call. The associated prompts are as follows: Base Prompt [Fig. 8.4],

```
Given a text document, you are a very faithful format converter that
translate natural language aircraft crash scenario descriptions to a
fix-form format prolog files to appropriately describe the scenario.
This will be used later to derive collision rules so only focus on final
configuration where the situation is dangerous.
   Convert these descriptions into 4 files.
   - file_1: This file identifies all the relevant entities from the text
   - file_2: This file contains the list of head and body predicates that
   are relevant to the text, including their input types and some atomic
   definitions.
   - file_3: This file contains the predicate grounding for the entities
   in the text.
   - file_4: This file contains the positive and negative examples for
   the head predicate.

Given the following information, generate the Prolog files for the
scenario described in the crash report.
```

Figure 8.4: System Prompt for crash report prolog extraction.

Consistency and Validity Checking

To ensure the accuracy and consistency of these extractions, we perform cross-validation against established domain knowledge, flagging and correcting any inconsistencies or hallucinations. If issues are found, the prompt is executed again with additional notes to the LLM.

8.2.3 Trajectory based Positive Data

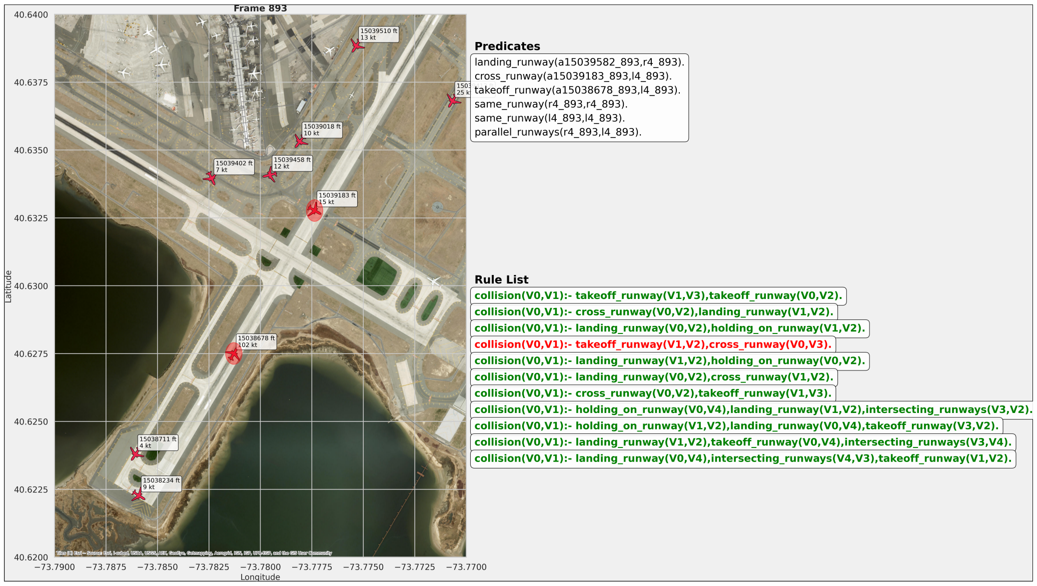
For the positive samples, we use the rountine data collected as part of the Amelia-48 setup [125]. While the system is agnostic to the choice of grounding algorithms, for the current setup we use a geofenced grounding with user-defined limits for each of the predicates. For all the agents within each frame, we select the relevant predicates and ground the agents within them. In order to enable generalizability, future work might involve exploring state-of-the-art grounding and scene graph methods for predicate grounding.

```
% Ensure that the generated files adhere to the following rules:
%    Only use the given predicates. Do not create new predicates.
Definitions of predicates:

% - collision(Agent1, Agent2).     % Agent1 is in collision or near miss
with Agent2
% - pos(collision(Agent1, Agent2)).    % Positive example of collision
between Agent1 and Agent2
% - neg(collision(Agent1, Agent2)).    % Negative example of collision
between Agent1 and Agent2
% - landing_runway(Agent1, Runway1).    % Agent1 is landing on Runway1
and it within 5 miles or closer, Agent 1 should be in motion
% - takeoff_runway(Agent1, Runway1).    % Agent1 is taking off from
Runway1, Agent 1 should be in motion
% - cross_runway(Agent1, Runway1).     % Agent1 is crossing Runway1
% - holding_short_runway(Agent1, Runway1).   % Agent1 is holding short of
Runway1, did not cross the hold line
% - on_extended_area_runway(Agent1, Runway1).   % Agent1 is on the
extended area of Runway1, this could be the extended center line or the
approach to the runway but not on the runway
% - holding_on_runway(Agent1, Runway1).    % Agent1 is holding on
Runway1, beyond the hold line (LUAW) or after landing (has not crossed
the hold line)
% - parallel_runways(Runway1, Runway2).    % Runway1 and Runway2 are
parallel runways
% - intersecting_runways(Runway1, Runway2).    % Runway1 and Runway2 are
intersecting runways
% - same_runway(Runway1, Runway2).     % Runway1 and Runway2 are the same
runway
```

Figure 8.5: Predicate Sets for crash report prolog extraction.

```
% Common Abbreviations:
% - ATC: Air Traffic Control
% - VEH: Vehicle
% - A/C: Aircraft
% - LUAW: Line Up and Wait
% - GC: Ground Control
% - LC: Local Control
% - ASDE-X: Airport Surface Detection Equipment - Model X
% - ASSC: Airport Surface Surveillance Capability
% - ILS: Instrument Landing System
% - TXY  : Taxiway
% - RWY  : Runway

% Prolog formatting rules:
% 1. **Always use lowercase for predicate names and constants.**
% 2. **All rules must end with a period (`.`).**
% 3. **Head predicates (`head_pred/2`) are always required.**
% 4. **Body predicates (`body_pred/2`) should only be included if used in
`bk`.**
% 5. **All predicates used in `bk` must also be defined in `bias`.**
% 6. **Only use entities that appear in `data_**.pl`. Do not introduce
undefined entities.**
% 7. **All entities must include the report number
[entity]_[report_number] (e.g., `a1_101`, `r4_101`).**
% 8. **Agent include aircraft, helicopters or vehicles**
% 9. **ALL ENTITY NAMES MUST ALWAYS START WITH A LETTER. Runway 4R in
report 101 is `r4_101` and not `4r_101`**
% 10. **Do not add any additional information that is not present in the
report.**
% 11. **Do not add comments or headers to the generated files.**
```

Figure 8.6: Additional info and instructions



Predicates

```
landing_runway(a15039582_893,r4_893).
cross_runway(a15039183_893,l4_893).
takeoff_runway(a15038678_893,l4_893).
same_runway(r4_893,r4_893).
same_runway(l4_893,l4_893).
parallel_runways(r4_893,l4_893).
```

Rule List

```
collision(V0,V1):- takeoff_runway(V1,V3),takeoff_runway(V0,V2).
collision(V0,V1):- cross_runway(V0,V2),landing_runway(V1,V2).
collision(V0,V1):- landing_runway(V0,V2),holding_on_runway(V1,V2).
collision(V0,V1):- takeoff_runway(V1,V2),cross_runway(V0,V3).
collision(V0,V1):- landing_runway(V1,V2),holding_on_runway(V0,V2).
collision(V0,V1):- landing_runway(V0,V2),cross_runway(V1,V2).
collision(V0,V1):- cross_runway(V0,V2),takeoff_runway(V1,V3).
collision(V0,V1):- holding_on_runway(V0,V4),landing_runway(V1,V2),intersecting_runways(V3,V2).
collision(V0,V1):- holding_on_runway(V1,V2),landing_runway(V0,V4),takeoff_runway(V3,V2).
collision(V0,V1):- landing_runway(V1,V2),takeoff_runway(V0,V4),intersecting_runways(V3,V4).
collision(V0,V1):- landing_runway(V0,V4),intersecting_runways(V4,V3),takeoff_runway(V1,V2).
```

Figure 8.7: Figure shows the system in-action for a real-world runway incursion at JFK. The predicates are shown on the top right. Botton right shows the distilled rules. The highlighted rule (in red) is the one that is broken.

8.3 Experiment

8.3.1 Implementation details

The system is written using LangChain and while the system is agnostic to the choice of LLM, we currently use OpenAIs GPT-4o as our base LLM. We use crash reports from the ASIAS database by filtering them for Runway Incursions. Care is taken to take reports from years not overalapping the Amelia-48 data collection. For the routine trajectory data, we use Amelia-48. Here, care is taken to make sure that the routine data contains no negative examples.

The figure 8.7 shows the system in action for a runway incursion that happned in JFK in Januray 2023. The results show the current predicates for the scene, the rule-set prodiced by the system and rule being broken in this scenario.

8.4 Conclusion

In conclusion, our neuro-symbolic pipeline demonstrates the effective integration of data-driven models with symbolic reasoning to enhance aviation safety. By applying this approach to real-world runway incursion data, we have successfully distilled interpretable rules that capture critical safety constraints and operational patterns. These results underscore the potential of combining machine learning with logical inference to proactively identify and mitigate risks in complex, safety-critical environments like airport surface operations.

Chapter 9

Safe Action Adjudication for Learning-based Systems

9.1 Introduction

Recent advances in AV navigation stacks rely more and more on the use of black-box machine-learning methods to provide real-time nuanced decision making. While these methods have improved the general driving performance of AVs, this progress has been achieved at the cost of reduced transparency in how these driving decisions are made. Within jurisprudence, a *precedent* refers to a principle or rule established in a legal case that becomes authoritative to a court when deciding subsequent cases with similar facts [139]. As autonomous vehicles (AVs) increasingly encounter complex real-world scenarios, they need to make decisions that lead to safe and legal outcomes. With growing calls for accountability in autonomous decision-making, the ability to use verifiable precedents to adjudicate critical driving decisions on-the-fly is crucial in improving trust and explainability of learning-based systems.

State-of-the-art AV navigation stacks rely on large-scale human driving datasets to train behaviors with the aim to align the AV's decisions with human-like decision making [186]. While these datasets cover a wide range of scenarios, they mostly contain *positive* incident-free interactions [28]. The access to *positive*-only driving data confines the AVs to only reason about scenarios that fall within the positive spectrum of the driving experience, hampering the ability to effectively reason about *negative* outcomes. Humans, on the other hand, use both positive and negative experiences to

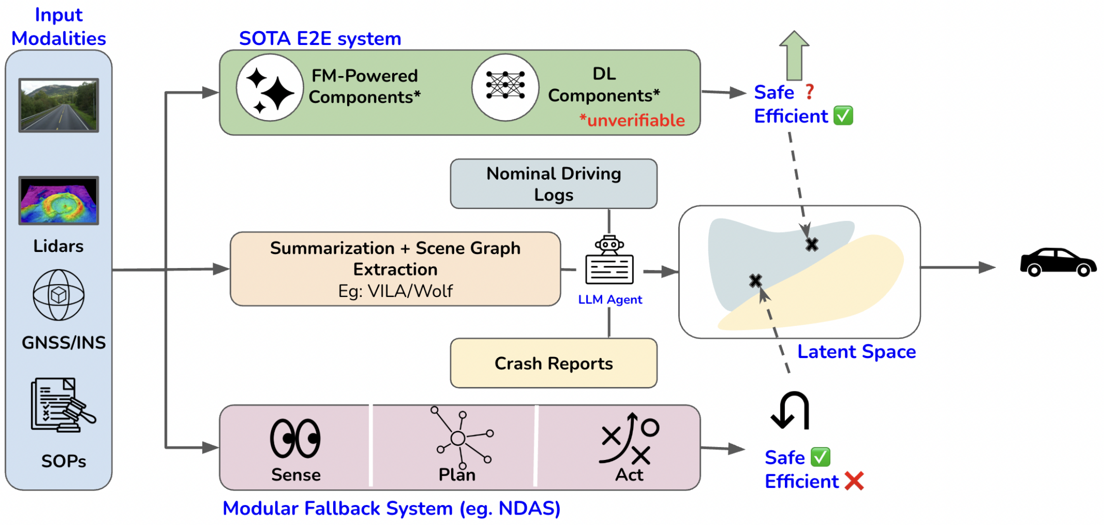


Figure 9.1: Block diagram for using crash data for action arbitration.

inform their decisions while driving in complex scenarios.

Automobile crash reports are a rich source of *negative* interactions. The National Highway Traffic Safety Administration (NHTSA) is an agency of the U.S. federal government that collects and publishes detailed crash data to help scientists and engineers analyze motor vehicle crashes and injuries. The crash reports, collected since the early 1970s, can be used to provide AVs with an understanding of negative outcomes for certain actions, providing the necessary grounding for counterfactually evaluating various driving decisions. However, unlike the positive data which is collected on vehicles instrumented with many sensors (e.g. multiple cameras with known intrinsics, IMU, GPS etc.), negative data is typically less precise, e.g., taking the form of a text description or dashcam video footage.

This work aims to provide a framework that enables uses both positive and negative examples of driving behaviour to provide a more holistic and transparent understanding of autonomous driving decisions. Given a particular scenario, an AV needs to reason about possible feasible actions and the impact of these actions on the other agents. We plan to use historical data to provide corroborating evidence for each of the possible actions. The strength and direction of this support informs the action choice for the AV.

9.2 Background

9.2.1 Explainability in robot decision-making

Autonomous agents that rely on learning-based end-to-end decision-making are being increasingly deployed in safety critical scenarios. The black-box nature of these methods has prompted research in providing justifications and reasoning to ensure safer and more trustworthy decision-making. Explainable AI (X-AI) is a sub-branch of AI research that focuses on the capability of AI systems to explain their behavior in human-understandable ways [157]. Within self-driving, a few X-AI approaches have gained more prominence [36]. Visual saliency maps [148] are an natural way to represent the attention of the model around the ego vehicle but their effectiveness in downstream decision-making is still challenging. In addition to saliency maps, recent end-to-end models [186] also provide access to intermediate latent representations for other tasks like image segmentation, depth estimation and trajectory prediction. The outputs of these tasks can be used to arbitrate between desirable and undesirable behaviours but the downstream impact of these intermediate representations in the final output is still unclear. Language provides another modality for conveying driving decision. Several methods [80, 20, 191, 187] that use both vision and language captions for providing decision explanations exist but only focus on positive scenarios.

9.2.2 Scenario Evaluation in Autonomous Driving

Critical Scenario Identification (CSI), especially as it pertains to autonomous driving behavior, is a key component in the larger verification and validation pipeline [196]. Classically, methods drawn from reachability theory [17] have been used to provide an understanding of a "criticality" of a particular interaction. Vanilla reachability uses the worst-case assumption on the behavior of other agents which makes it overly conservative limiting its use in real-world situations. While methods that assume a more reasonable behavior from the other agent have been proposed [121], the effectiveness of these methods in distribution shifts and high-dimensional data are not well studied. Uncertainty quantification [138, 77] is another paradigm that can be used to identify unsafe scenarios given a dataset of safe scenarios. While UQ methods are effective in identifying scenarios that are far from the safe distribution, these methods tend to be more conservative as all out-of-distribution scenarios are classified as unsafe and they fail to offer counterfactuals. Rules [169] and catalog-based [184] methods that rely on humans to craft hand-engineered rules have also been proposed, but these methods are painstaking to develop and can struggle with far edge-case scenarios that were overlooked during development.

9.3 Methodology

9.3.1 Overview

Given a particular scene, the ego agent is tasked with choosing an action that is not only safe but also performant. Given the position of the ego agent within a map as well as position and possible intents of other agents, the ego agent planner can generate multiple possible actions, some of which can be feasibly executed. The goal of a AV decision adjudicating module is therefore to reason among these actions ensuring the safety and progress towards that goal. This work aims to find evidence from historical data that either positively and negatively supports each of these decisions. If a particular decision has more support from the negative spectrum, it is down-voted and vice-versa for positive actions.

Given a historical dataset of positive and negative examples, the proposed work aims to semantically tie each of these decisions to either side of the spectrum by providing concrete examples from the data in support of these different decision modalities.

Figure 9.1, provides a high-level diagram for the methodology. The method has a data set creation step and an evaluation step. Both steps use a multimodal scene understanding VLM, a semantic scene graph generator and parser, a retrieval mechanism, and a generator decoder.

9.3.2 GraphRAG on Crash Data:

The scene-action graph processing first builds the retrieval databases for both positive and negative examples. Each graph represents one datapoint and encodes different elements of the scene like the map, the position and actions of other agents, ego's position and action, as well as the outcome of that action. The graph processing steps are distinct based on the modality, but the final outcome is consistent to ensure

```
V1 was traveling eastbound in the left lane of a two way roadway with two
lanes of travel in each direction. V2 was traveling westbound in the left
lane of a two way roadway with two lanes of travel in each direction. As V1
was making a left turn to go northbound at the above intersection, the
frontal plane of V1 made contact with the left side plane of V2. Both
vehicles came to final rest on the roadway.
```

Figure 9.2: Example Crash Report from the NHTSA Dataset

```
-Goal-
Given a text document, you are a very faithful format converter that
translate technical natural language crash scenario descriptions to a
more easy to understand scenario description. V1 is always the vehicle of
interest or ego vehicle. Do not use V1, V2 or any such notations in the
output. Use ego vehicle, second vehicle, other vehicle and such. Do not
use cardinal directions like east, west, northwest. Use directions
relative to ego.
```

Figure 9.3: Crash Report Style Transfer Prompt

that the retrievals are semantically relevant. This is followed by the retrieval mechanism which follows a similar structure in constructing the query graph. In the end, a decoder completes the final adjudication by using the graph-based retrievals.

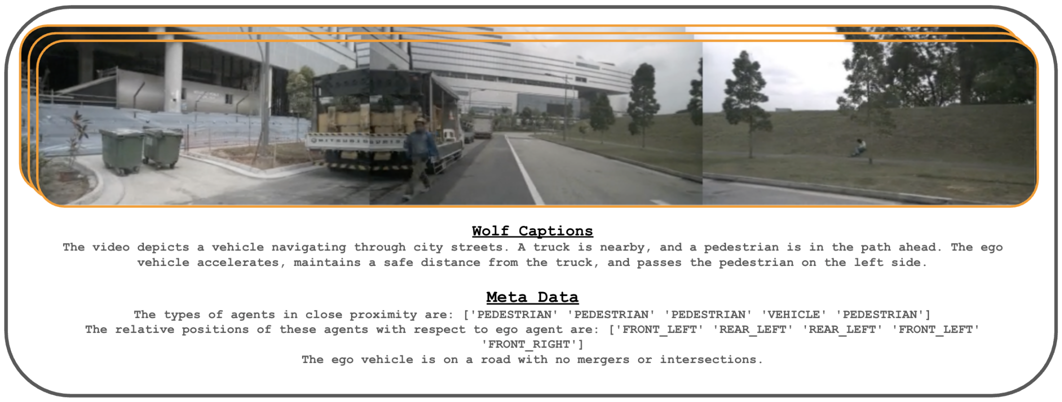
Negative Data

We use the NHTSA crash report dataset to extract crash report summaries that can serve as negative labels. An example report is shown in Fig. 9.2. Due to the specific style of the report, we first conduct a style transfer to bring the report to an ego-centric perspective. The prompt for the style transfer is shown in Fig. 9.3. We also provide in-context examples to aid the transfer process.

In order to generate the scene action graph, we follow steps similar to the GraphRAG [47] approach. We split the The full prompt is shown in Fig. 9.4. In-context examples are provided to aid the extraction process. Once a graph is generated, we provide language embeddings for each node and store that along with the final graph.

```
-Goal-
Given a text document, you are a very faithful format converter that
translate natural language crash scenario descriptions to a fix-form
format to appropriately describe the scenario. V1 is always the vehicle
of interest or ego vehicle. Do not use V1, V2 or any such notations in the
output. Use ego vehicle, second vehicle and such. Do not use cardinal
directions like east, west, northwest. Use directions relative to ego.
Find all the entities that are potentially relevant to this activity and
a list of entity types, identify all entities of those types from the text
and all relationships among the identified entities.
-Steps-
1. Identify all entities. For each identified entity, extract the
following information:
- entity_name: Name of the entity
- entity_type: One of the following possible types:
[VEHICLE,MAP,ACTION,OBSTACLES,EFFECT]
- entity_description: Comprehensive description of the entity's
attributes
Format each entity as
("entity"|<entity_name>|<entity_type>|<entity_description>)

2. Return output in English as a single list of all the entities
identified in step 1. Use **record_delimiter** as the list delimiter. No
prose in the output.

3. When finished, output completion_delimiter
```

Figure 9.4: Crash Report Graph Extraction Prompt



Wolf Captions

The video depicts a vehicle navigating through city streets. A truck is nearby, and a pedestrian is in the path ahead. The ego vehicle accelerates, maintains a safe distance from the truck, and passes the pedestrian on the left side.

Meta Data

The types of agents in close proximity are: ['PEDESTRIAN' 'PEDESTRIAN' 'PEDESTRIAN' 'VEHICLE' 'PEDESTRIAN']
The relative positions of these agents with respect to ego agent are: ['FRONT_LEFT' 'REAR_LEFT' 'REAR_LEFT' 'FRONT_LEFT' 'FRONT_RIGHT']
The ego vehicle is on a road with no mergers or intersections.

Figure 9.5: An example positive scene: the multi-camera FPV ego image, the Wolf caption as well as meta-data is shown.

Positive data

Given the contextual nature of the adjudication, we first use a fine-tuned Wolf [95] summarization framework to provide a concise caption for the current scenario. Wolf is an automated captioning framework that adopts a mixture-of-experts approach, leveraging complementary strengths of VLMs. By combining image and video models, Wolf framework captures different levels of information and summarizes them. For datasets with available meta-data, we also inject that into the framework to generate a text-only summary of the scene with all relevant agents. This is then followed by the same extraction method as for the crash reports. We use scenes samples from NuScenes [29] dataset as the routine positive dataset. An example scene, the Wolf caption as well as meta-data is shown in Fig. 9.5.

Retriever:

The retrieval sampler uses an average similarity score between all the intersecting nodes between a query graph and the database to retrieve the semantically k-closest examples to input and action pairs. This is done by first using Wolf to first construct the scene caption. This is followed by the query graph extractor that also adds and encodes the selected action to the query graph.

Decoder:

The decoder uses the input as well as the retrieved examples to produce final outputs, which can be any prediction that would benefit from experience of both positive and negative driving experiences relevant to the current scene. For example, the decoder could directly provide if recommended plan is good or in an offline case provide a score to be used to either improve the planner or evaluate the performance of the planner in particular scenarios.

The structure of the decoder can vary: in the case of estimating a safety score of a candidate plan, a simple design could be a simple K-nearest neighbor classifier which estimates whether the scene and plan are closer to more negative or positive examples. Alternatively, a higher-capacity decoder could perform more complex predictions informed by the retrieved examples. We use an LLM based decoder which can use the retrieved examples as context when reasoning about safety or ego planning in the scene. Our framework has the potential to improve a host of downstream tasks by reducing the impact of the safe-driving bias of most AV datasets by bringing in relevant negative examples from other data modalities.

9.4 Experiments

9.4.1 Experimental Setup

We evaluate our method on a held-out subset of the nuScenes dataset, focusing on scenarios where the ego vehicle is presented with multiple feasible action choices. Each scenario is annotated with a set of plausible actions, which vary in safety, legality, and social acceptability. A human annotator is asked to classify each action into one of three categories: **Safe**, **Reasonable**, or **Unsafe**. For example, stopping behind a double-parked vehicle may be safe but inefficient, while overtaking by crossing the double yellow line may be reasonable under low-traffic conditions but risky in others.

Our framework is then applied to the same scenarios. Given the ego scene and a set of candidate actions, the system uses positive and negative data retrievals to reason about the consequences of each action and provide a label. We compare these system-generated labels with the human-annotated ground truth.

We report classification performance in terms of accuracy, precision, recall, and

F1 score across the three label classes. We also compare our full GraphRAG-based pipeline to a baseline model that does not use retrieval-augmented generation and instead relies only on local scene information and an LLM for adjudication.

9.4.2 Results and Discussion

Our method achieves a **37.5%** improvement in macro-averaged F1 score as compared to the baseline that does not use RAG.

Several key benefits emerge from our approach:

1. **Avoidance of Common-Mode Failures:** By incorporating negative historical data, the system can better recognize unsafe decisions even in situations that are underrepresented in standard training datasets. This leads to improved generalization under distribution shift.
2. **Nuanced Evaluation of Ambiguous Cases:** Many actions in real-world driving are not strictly safe or unsafe. Our retrieval-based adjudication supports a more nuanced interpretation of these scenarios, aligning better with human reasoning.
3. **Potential for Runtime Use:** With recent advances in efficient RAG and LLM distillation, our approach holds promise for real-time deployment. The modularity of our architecture enables lightweight retrieval and evaluation, making it amenable to integration into AV runtime stacks.
4. **Enhanced Interpretability and Control:** Because decisions are grounded in retrieved precedent examples, the system is more transparent and debuggable. This also enables easier policy refinement by inspecting and modifying the retrieval corpus rather than retraining the entire model.

Qualitative examples show that the system correctly downvotes aggressive left turns in busy intersections based on prior crash cases, while supporting assertive merges in low-speed traffic when backed by safe precedent cases. This evidence-grounded adjudication improves both trust and safety in AV decision-making.

9.5 Conclusion

In this chapter, we introduced a framework for safe action adjudication in autonomous vehicles (AVs) by integrating both positive and negative driving experiences. By leveraging historical crash data alongside successful driving logs, our approach enables AVs to evaluate potential actions with a more comprehensive understanding of safety implications. The incorporation of a multimodal encoder and a latent space sampler facilitates the retrieval of semantically similar scenarios, providing context-aware decision-making capabilities. This methodology addresses the limitations of traditional AV systems that predominantly rely on positive data, enhancing the explainability and robustness of AV decision-making processes. Our framework lays the groundwork for more transparent and accountable autonomous driving systems capable of reasoning about the consequences of their actions in complex, real-world scenarios.

Chapter 10

Conclusion

10.1 Summary

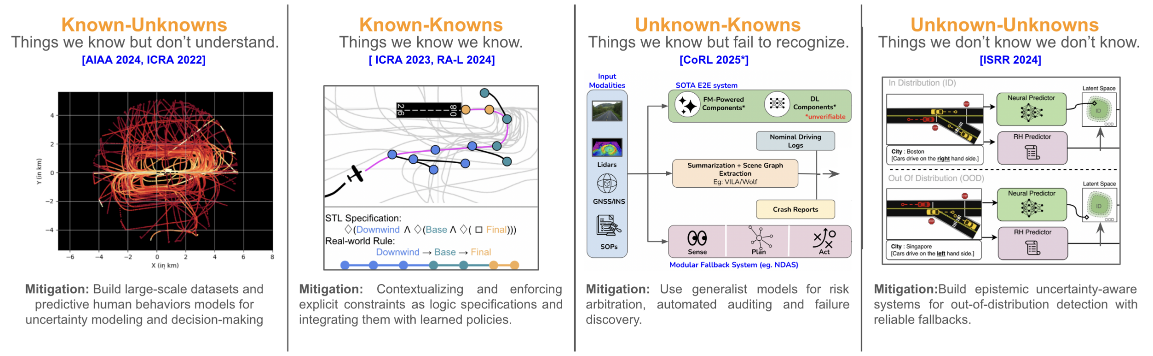

Known-Unknowns
Things we know but don't understand.
[AIAA 2024, ICRA 2022]
Mitigation: Build large-scale datasets and predictive human behaviors models for uncertainty modeling and decision-making
Known-Knowns
Things we know we know.
[ICRA 2023, RA-L 2024]
STL Specification:
◇(Downwind ∧ ◇(Base ∧ ◇(□ Final)))
Real-world Rule:
Downwind → Base → Final
Mitigation: Contextualizing and enforcing explicit constraints as logic specifications and integrating them with learned policies.
Unknown-Knowns
Things we know but fail to recognize.
[CoRL 2025*]
Input Modalities
SOTA E2E system
FM-Powered Components*
DL Components*
*unverifiable
Nominal Driving Logs
Lidars
Summarization + Scene Graph Extraction
Eg: VILA/Wolf
GNSS/INS
Crash Reports
SOPs
Sense
Plan
Act
Modular Fallback System (eg. NDAS)
Mitigation: Use generalist models for risk arbitration, automated auditing and failure discovery.
Unknown-Unknowns
Things we don't know we don't know.
[ISRR 2024]
In Distribution (ID)
Neural Predictor
RH Predictor
Latent Space
City : Boston
[Cars drive on the right hand side.]
Out Of Distribution (OOD)
City : Singapore
[Cars drive on the left hand side.]
Mitigation:Build epistemic uncertainty-aware systems for out-of-distribution detection with reliable fallbacks.

Figure 10.1: Summary figure shows the mitigation strategies across the four quadrants of knowledge while showcasing some of my approaches for risk mitigation by systematically managing both known and unknown uncertainties.

This thesis rethinks the safety case for deploying autonomous systems in social, safety-critical settings by framing safety as risk-aware, reasonable behavior. Rather than avoiding risk altogether, it proposes managing different kinds of uncertainty—categorized as known-unknowns, known-knowns, unknown-unknowns, and unknown-knowns—through distinct mitigation strategies. This framework combines the adaptability of data-driven models with the robustness of rule-based reasoning to support safe and seamless human-robot interaction.

To handle behavioral uncertainty, the thesis introduces large-scale aviation datasets

(TrajAir, TartanAviation, Amelia-48) and a socially-aware trajectory predictor (TrajAirNet), enabling agents to anticipate human behavior in complex environments. For rule adherence, it presents a neurosymbolic planning framework that fuses logic specifications with learned policies and a novel LLM+ILP pipeline to mine safety rules from crash reports. To manage unforeseen scenarios, RuleFuser combines imitation learning with rule-based priors, dynamically adjusting behavior based on out-of-distribution detection. Finally, to address latent failure modes, a multi-planner adjudication system uses LLMs and human-generated crash data for context-aware arbitration.

Altogether, this thesis provides a structured approach to building trustworthy, socially competent autonomous agents capable of operating safely in the unpredictable, human-centered world.

10.1.1 Part I: Known-Unknowns

This thesis addresses the challenge of modeling uncertainty in human behavior—categorized as "known-unknowns"—by developing and releasing a suite of pioneering datasets and predictive models tailored for the aviation domain. The introduction of TrajAir , the first open-source 3D trajectory prediction dataset for general aviation, provides a foundational resource for understanding aircraft movements in non-towered airspace. Building upon this, TartanAviation offers a comprehensive multi-modal dataset encompassing vision, speech, and trajectory data, capturing the complexity of terminal-area operations. Further advancing the field, Amelia-48 stands as the largest public dataset of its kind, encompassing extensive surface movement data across 48 U.S. airports. Complementing these datasets, the development of TrajAirNet, a socially-aware trajectory prediction algorithm, demonstrates the efficacy of modeling agent-agent, agent-environment, and agent-context interactions. Collectively, these contributions lay the groundwork for enhancing the predictive capabilities of autonomous systems, enabling them to better anticipate and adapt to the nuances of human behavior in complex, dynamic environments.

10.1.2 Part II: Known-Knowns

This part tackles the challenge of ensuring rule adherence in safety-critical environments—categorized as "known-knowns"—by integrating data-driven learning with symbolic reasoning. We present a neurosymbolic approach that fuses Signal Temporal

Logic (STL) specifications with learning-from-demonstration (LfD) policies using Monte Carlo Tree Search (MCTS), enabling dynamic enforcement of rules during planning. This method improves online robustness and achieves a 60% gain over standard LfD baselines in complex aviation scenarios. Extending this, we introduce SoRTS, a social navigation algorithm for general aviation that combines intent prediction with global guidance, evaluated in X-PlaneROS, a high-fidelity simulator with real pilot feedback.

10.1.3 Part III: Unknown-Unknowns

This part addresses the "unknown-unknowns" quadrant—risks arising from unforeseen, out-of-distribution (OOD) scenarios that challenge the generalization capabilities of data-driven systems. To manage such conditions, we propose RuleFuser, a motion planning framework that fuses imitation learning (IL) with rule-based reasoning through a principled evidential Bayesian approach. By replacing the uniform prior in evidential learning with a rule-informed Dirichlet prior, RuleFuser ensures fallback to safer rule-compliant behavior when encountering unfamiliar scenarios. The system uses a shared candidate set, where a rule-based planner provides a safety-prior over trajectories, and a neural IL model contributes data-driven likelihoods. Guided by an OOD score computed via normalizing flows, the fusion dynamically shifts control from the IL planner to the rule-based planner in high-risk conditions. Evaluated on real-world autonomous driving datasets, RuleFuser demonstrates strong performance in both in-distribution and OOD conditions, balancing nuanced behavior with conservative safety guarantees. This work shows the potential of structured fusion as a safety mechanism and opens avenues for broader integration of rule-aware priors into modern planning stacks and end-to-end autonomous driving frameworks.

10.1.4 Part IV: Unknown-Knowns

This part explores the "unknown-knowns" quadrant—risks that stem from latent failure modes embedded within embodied AI systems, often undetected until deployment. To mitigate such risks, we propose a multi-engine safety framework inspired by the "Swiss-Cheese" model, where multiple diverse planners—learned and rule-based—are run in parallel to provide redundancy against common-mode failures. To resolve conflicts among these planners, we introduce an adjudication mechanism powered by Large Language Models (LLMs), which leverages context from human crash reports using a modified GraphRAG pipeline. These reports serve as a source of grounded, counterfactual safety reasoning, enabling nuanced arbitration over candidate actions. By combining both positive and negative experiences from real-world driving logs, and embedding them in a semantically meaningful latent space, our framework enhances the explainability, accountability, and runtime robustness of autonomous decision-making. This work contributes a pathway toward AV systems that can reason more like humans—drawing on past failures to proactively avoid future ones—and represents a step toward building embodied intelligence that is transparent, self-aware, and resilient to latent failure. To reduce reliance on handcrafted rule sets, we propose a hybrid LLM + Inductive Logic Programming (ILP) pipeline that mines interpretable rules from crash reports and enforces them as runtime monitors. Together, these contributions demonstrate how integrating symbolic constraints with learned models supports safer, more reliable decision-making in structured environments like airport operations.

10.2 Future Work

Building upon the foundations laid in this thesis, several promising directions emerge to advance the algorithms underpinning risk-aware, socially intelligent autonomous systems.

10.2.1 Risk Mitigation in End-to-End Large-Scale Learned Policies

The emergence of Vision-Language-Action (VLA) models, such as RT-2 [26] and OpenVLA [84], has significantly advanced the field of robotics by enabling systems to interpret complex instructions and perform sophisticated tasks through the integration of visual perception, language understanding, and action planning. These models leverage large-scale pretraining on diverse datasets to generalize across various tasks and environments. However, as with large language models (LLMs), VLAs are susceptible to safety concerns, including the potential for jailbreaking, [153] where models are manipulated to bypass safety protocols. In LLMs, this issue has been addressed through techniques like mechanistic interpretability [21], which aims to elucidate the internal workings of neural networks, and the implementation of guardrails to prevent undesirable outputs. Applying similar strategies to VLAs is challenging due to the embodied nature of robotics, where actions have real-world consequences, and safety is highly contextual and dependent on dynamic, stochastic environments.

To mitigate these challenges, research has explored the integration of context-aware safety mechanisms into VLA systems. Additionally, the development of hybrid control architectures like RuleFuser that combine data-driven policies with rule-based systems offers a pathway to provide fallback mechanisms in uncertain situations. These approaches underscore the necessity of developing safety strategies tailored to the unique demands of embodied AI, where the unpredictability of the physical world requires systems to adaptively assess risks and make contextually appropriate decisions.

10.2.2 Neuro-Symbolic Reasoning for Interpretable Safety

While end-to-end robotic systems offer streamlined pipelines, they often lack interpretability and modularity, making fault diagnosis and recovery challenging. Modular architectures, on the other hand, compartmentalize functionalities such as perception, planning, and control, allowing for targeted improvements and easier debugging. This structure facilitates the integration of foundation models into specific modules, enhancing performance without compromising the system's overall transparency. However, modular systems can be susceptible to cascading failures if inter-module dependencies are not carefully managed.

Neurosymbolic AI presents a promising solution by combining the pattern recognition capabilities of neural networks with the logical reasoning of symbolic systems. This hybrid approach enhances the interpretability and safety of robotic systems, enabling them to reason about their actions and adapt to new situations effectively. By embedding symbolic rules within neural architectures, robots can better understand and explain their decision-making processes, leading to increased trust and reliability in autonomous operations.

10.2.3 Advancing Standards for Data-Driven Robotics

As robotics systems increasingly incorporate data-driven technologies like machine learning and artificial intelligence, the limitations of existing safety standards become more apparent. Standards such as ISO 10218 and ISO/TS 15066 primarily address mechanical and control system safety, offering limited guidance on the dynamic and adaptive behaviors introduced by AI components. This gap underscores the urgent need for comprehensive safety frameworks that specifically address the unique challenges posed by data-driven robotics.

To foster public trust and ensure the ethical deployment of advanced robotic systems, it is imperative to develop and implement new safety standards that encompass data integrity, transparency, real-time monitoring, and robust testing protocols. Collaboration among industry leaders, policymakers, researchers, and the public is essential to create standards that align with societal values and expectations. By proactively addressing these challenges, we can guide the evolution of robotics in a manner that is both socially beneficial and aligned with human values.

Bibliography

[1] Software considerations in airborne systems and equipment certification. RTCA Standard, 2011. DO-178C, published by RTCA, Inc. and EUROCAE in 2011.

[2] Road vehicles – functional safety. International Standard, 2018. ISO 26262:2018.

[3] Federal Aviation Administration. Faa aerospace forecast—fiscal years 2023-2043, 2023.

[4] AICrowd. Airborne object tracking challenge, 2022.

[5] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[6] Busyairah Syd Ali. *A safety assessment framework for automatic dependent surveillance broadcast (ads-b) and its potential impact on aviation safety*. PhD thesis, Centre for Transport studies, Department of Civil and Environmental . . . , 2013.

[7] Richard Alligier and David Gianazza. Learning aircraft operational factors to improve aircraft climb prediction: A large scale multi-airport study. *Transportation research part C: emerging technologies*, 96:72–95, 2018.

[8] Jasmine Jerry Aloor, Jay Patrikar, Parv Kapoor, Jean Oh, and Sebastian Scherer. Follow the rules: Online signal temporal logic tree search for guided imitation learning in stochastic domains. In *Proc. IEEE Conf. on Robotics and Automation*, pages 1320–1326. IEEE, 2023.

[9] Heather M Arneson, Pallavi Hegde, Michael E La Scola, Antony D Evans, Richard M Keller, and John E Schade. Sherlock data warehouse. Technical report, 2019.

[10] International Air Transport Association. Air passenger market analysis - december 2022. Technical report, IATA, 2022.

[11] Jean-Philippe Aurambout, Konstantinos Gkoumas, and Biagio Ciuffo. Last mile delivery by drones: An estimation of viable market potential and access to citizens across european cities. *European Transport Research Review*, 11(1):1–21, 2019.

[12] Arwa S. Aweiss, Brandon D. Owens, Joseph Rios, Jeffrey R. Homola, and Christoph P. Mohlenbrink. *Unmanned Aircraft Systems (UAS) Traffic Management (UTM) National Campaign II*. Aerospace Research Central, 2018.

[13] Samet Ayhan and Hanan Samet. Aircraft trajectory prediction made easy with predictive analytics. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 21–30, 2016.

[14] J Andrew Bagnell. An invitation to imitation. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, 2015.

[15] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[16] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[17] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[18] Shane T Barratt, Mykel J Kochenderfer, and Stephen P Boyd. Learning probabilistic trajectory models of aircraft in terminal airspace from position data. *IEEE Transactions on Intelligent Transportation Systems*, 20(9):3536–3545, 2018.

[19] Mohammad Ali Bashiri, Brian Ziebart, and Xinhua Zhang. Distributionally robust imitation learning. *Advances in Neural Information Processing Systems*, 34:24404–24417, 2021.

[20] Hédi Ben-Younes, Éloi Zablocki, Patrick Pérez, and Matthieu Cord. Driving behavior explanation with multi-level fusion. *Pattern Recognition*, 123:108421, 2022.

[21] Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

[22] Jur van den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[23] Abhijat Biswas, Allan Wang, Gustavo Silvera, Aaron Steinfeld, and Henny Admoni. Socnavbench: A grounded simulation testing framework for evaluating social navigation. *ACM Transactions on Human-Robot Interaction (THRI)*, 11(3):1–24, 2022.

[24] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. *Int. Conf. on Machine Learning*, 2015.

[25] Djallel Bouneffouf and Charu C. Aggarwal. Survey on applications of neurosymbolic artificial intelligence. *arXiv preprint arXiv:2209.12618*, 2022.

[26] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[27] JohnTravis Butler and Arvin Agah. Psychological effects of behavior patterns of a mobile personal robot. *Autonomous Robots*, 10:185–202, 03 2001.

[28] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[29] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020.

[30] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuPlan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition ADP3 workshop*, 2021.

[31] Xiaoyi Cai, James Queeney, Tong Xu, Aniket Datar, Chenhui Pan, Max Miller, Ashton Flather, Philip R Osteen, Nicholas Roy, Xuesu Xiao, et al. Pietra: Physics-informed evidential learning for traversing out-of-distribution terrain. *IEEE Robotics and Automation Letters*, 2025.

[32] Andrea Censi, Konstantin Slutsky, Tichakorn Wongpiromsarn, Dmitry Yershov, Scott Pendleton, James Fu, and Emilio Frazzoli. Liability, ethics, and culture-aware behavior specification using rulebooks. In *Proc. IEEE Conf. on Robotics and Automation*, pages 8536–8542, 2019.

[33] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

[34] Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts. *Advances in neural information processing systems*, 33:1356–1367, 2020.

[35] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6015–6022. IEEE, 2019.

[36] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[37] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017.

[38] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017.

[39] Kyunghoon Cho and Songhwai Oh. Learning-based model predictive control under signal temporal logic specifications. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7322–7329. IEEE, 2018.

[40] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338, 2019.

[41] Andrew Cropper and Sebastijan Dumančić. Inductive logic programming at 30: A new introduction. *Journal of Artificial Intelligence Research*, 74:765–850, 2022.

[42] Andrew Cropper and Rolf Morel. Learning programs by learning from failures. *Machine Learning*, 110(4):801–856, 2021.

[43] Nachiket Deo and Mohan M Trivedi. Trajectory forecasts in unknown environments conditioned on grid-based plans. *arXiv preprint arXiv:2001.00735*, 2020.

[44] Ryan Dewsbury. Stratux — An Open Source ADS-B Receiver, 01 2019.

[45] Yiming Ding, Carlos Florensa, Mariano Phielipp, and Pieter Abbeel. Goal-conditioned imitation learning. *arXiv preprint arXiv:1906.05838*, 2019.

[46] Alexandre Donzé, Thomas Ferrere, and Oded Maler. Efficient robust monitoring for stl. In *International Conference on Computer Aided Verification*, pages 264–279. Springer, 2013.

[47] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[48] Rosemary Emery-Montemerlo, Geoffrey J. Gordon, Jeff G. Schneider, and Sebastian Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), 19-23 August 2004, New York, NY, USA*, pages 136–143. IEEE Computer Society, 2004.

[49] Klemens Esterle, Luis Gressenbuch, and Alois Knoll. Formalizing traffic rules for machine interpretability. In *Proc. of IEEE Connected and Automated Vehicles Symposium (CAVS)*, pages 1–7. IEEE, 2020.

[50] Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3052–3059. IEEE, 2018.

[51] Federal Aviation Administration (FAA). *Airplane Flying Handbook: FAA-H-8083-3C.* Aviation Supplies & Academics, Incorporated, 2021.

[52] Mohammad Farhadmanesh, Abbas Rashidi, and Nikola Marković. General aviation aircraft identification at non-towered airports using a two-step computer vision-based approach. *IEEE Access*, 10:48778–48791, 2022.

[53] Alec Farid, Sushant Veer, Boris Ivanovic, Karen Leung, and Marco Pavone. Task-relevant failure detection for trajectory predictors in autonomous vehicles. In *Conference on Robot Learning*, pages 1959–1969. PMLR, 2023.

[54] Francesca Favaro, Laura Fraade-Blanar, Scott Schnelle, Trent Victor, Mauricio Peña, Johan Engstrom, John Scanlon, Kris Kusano, and Dan Smith. Building a credible case for safety: Waymo's approach for the determination of absence of unreasonable risk. *arXiv preprint arXiv:2306.01917*, 2023.

[55] Federal Aviation Administration. *Chapter 7: Airport Traffic Patterns*, pages 7–1. Federal Aviation Administration, 2018.

[56] Federal Aviation Administration. General Aviation Safety, 07 2018.

[57] Federal Aviation Administration. Airport Surface Detection Equipment, Model X (ASDE-X). `https://www.faa.gov/air_traffic/technology/asde-x`, 2023. `https://www.faa.gov/air_traffic/technology/asde-x`.

[58] Antonio Franco, Damián Rivas, and Alfonso Valenzuela. Probabilistic aircraft trajectory prediction in cruise flight considering ensemble wind forecasts. *Aerospace science and technology*, 82:350–362, 2018.

[59] N. R. French and J. C. Steinberg. Factors governing the intelligibility of speech sounds, January 1947.

[60] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*, 2015.

[61] Yuxiang Gao and Chien-Ming Huang. Evaluation of socially-aware robot navigation. *Frontiers in Robotics and AI*, page 420, 2021.

[62] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[63] Sourish Ghosh, Jay Patrikar, Brady Moon, Milad Moghassem Hamidi, and Sebastian Scherer. Airtrack: Onboard deep learning framework for long-range aircraft detection and tracking, 2022.

[64] Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. Transformer networks for trajectory forecasting. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 10335–10342. IEEE, 2021.

[65] John J Godfrey. Air traffic control complete. *Linguistic Data Consortium, Philadelphia, USA http://www. ldc. upenn. edu/Catalog/CatalogEntry. jsp*, 1994.

[66] Matt Grote, Aliaksei Pilko, James Scanlan, Tom Cherrett, Janet Dickinson, Angela Smith, Andrew Oakey, and Greg Marsden. Sharing airspace with uncrewed aerial vehicles (uavs): Views of the general aviation (ga) community. *Journal of Air Transport Management*, 102:102218, 2022.

[67] Dongyue Guo, Yi Lin, Xuehang You, Zhongping Yang, Jizhe Zhou, Bo Yang, Jianwei Zhang, Han Shi, Shasha Hu, and Zheng Zhang. M2ats: A real-world multimodal air traffic situation benchmark dataset and beyond. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 213–221, 2023.

[68] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*, 2024.

[69] Wataru Hashimoto, Kazumune Hashimoto, and Shigemasa Takai. Stl2vec: Signal temporal logic embeddings for control synthesis with recurrent neural networks. *arXiv preprint arXiv:2109.04636*, 2021.

[70] Bassam Helou, Aditya Dusi, Anne Collin, Noushin Mehdipour, Zhiliang Chen, Cristhian Lizarazo, Calin Belta, Tichakorn Wongpiromsarn, Radboud Duintjer Tebbens, and Oscar Beijbom. The reasonable crowd: Towards evidence-based and interpretable models of driving behavior. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 6708–6715, 2021.

[71] Daryl Herzmann, Ray Arritt, and Dennis Todey. Iowa environmental mesonet. *Available at mesonet. agron. iastate. edu/request/coop/fe. phtml (verified 27 Sept. 2005). Iowa State Univ., Dep. of Agron., Ames, IA*, 2004.

[72] Kai-Chieh Hsu, Haimin Hu, and Jaime F Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 7, 2023.

[73] Haimin Hu and Jaime F Fisac. Active uncertainty reduction for human-robot interaction: An implicit dual control approach. *arXiv preprint arXiv:2202.07720*, 2022.

[74] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17853–17862, 2023.

[75] Maximilian Igl, Daewoo Kim, Alex Kuefler, Paul Mougin, Punit Shah, Kyriacos Shiarlis, Dragomir Anguelov, Mark Palatucci, Brandyn White, and Shimon Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2445–2451, 2022.

[76] Craig Innes and Subramanian Ramamoorthy. Elaborating on learned demonstrations with temporal logic specifications. *arXiv preprint arXiv:2002.00784*, 2020.

[77] Masha Itkina and Mykel Kochenderfer. Interpretable self-aware neural networks for robust trajectory prediction. In *Proc. Conf. on Robot Learning*, pages 606–617. PMLR, 2023.

[78] Athul Paul Jacob, David J. Wu, Gabriele Farina, Adam Lerer, Hengyuan Hu, Anton Bakhtin, Jacob Andreas, and Noam Brown. Modeling strong and human-like gameplay with kl-regularized search, 2021.

[79] Ziwei Ji, Zihan Liu, Nayeon Lee, Tiezheng Yu, Bryan Wilie, Min Zeng, and Pascale Fung. Rho (ρ): Reducing hallucination in open-domain dialogues with knowledge grounding. *arXiv preprint arXiv:2212.01588*, 2023.

[80] Bu Jin, Xinyu Liu, Yupeng Zheng, Pengfei Li, Hao Zhao, Tong Zhang, Yuhang Zheng, Guyue Zhou, and Jingjing Liu. Adapt: Action-aware driving caption transformer. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7554–7561. IEEE, 2023.

[81] Mary E Johnson and Yue Gu. Estimating airport operations at general aviation airports using the faa npias airport categories. *International Journal of Aviation, Aeronautics, and Aerospace*, 4(1):3, 2017.

[82] Krishna C Kalagarla, Rahul Jain, and Pierluigi Nuzzo. Model-free reinforcement learning for optimal control of markovdecision processes under signal temporal logic specifications. *arXiv preprint arXiv:2109.13377*, 2021.

[83] Parv Kapoor, Anand Balakrishnan, and Jyotirmoy V Deshmukh. Model-based reinforcement learning from signal temporal logic specifications. *arXiv preprint arXiv:2011.04950*, 2020.

[84] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[85] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[86] Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[87] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Conf. on Neural Information Processing Systems*, 2017.

[88] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Conf. on Neural Information Processing Systems*, 2018.

[89] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017.

[90] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer graphics forum*, number 3, pages 655–664. Wiley Online Library, 2007.

[91] Karen Leung, Nikos Aréchiga, and Marco Pavone. Back-propagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 432–449. Springer, 2020.

[92] Karen Leung, Nikos Aréchiga, and Marco Pavone. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *The International Journal of Robotics Research*, 42(6):356–370, 2023.

[93] Karen Leung and Marco Pavone. Semi-supervised trajectory-feedback controller synthesis for signal temporal logic specifications. *arXiv preprint arXiv:2202.01997*, 2022.

[94] Boyi Li, Yue Wang, Jiageng Mao, Boris Ivanovic, Sushant Veer, Karen Leung, and Marco Pavone. Driving everywhere with large language model policy adaptation. *arXiv preprint arXiv:2402.05932*, 2024.

[95] Boyi Li, Ligeng Zhu, Ran Tian, Shuhan Tan, Yuxiao Chen, Yao Lu, Yin Cui, Sushant Veer, Max Ehrlich, Jonah Philion, et al. Wolf: Accurate video captioning with a world summarization framework.

[96] Xiao Li, Jonathan DeCastro, Cristian Ioan Vasile, Sertac Karaman, and Daniela Rus. Learning a risk-aware trajectory planner from demonstrations using logic monitor. In *Conference on Robot Learning*, pages 1326–1335. PMLR, 2022.

[97] Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, et al. Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv preprint arXiv:2406.06978*, 2024.

[98] Jing Liang, Utsav Patel, Adarsh Jagan Sathyamoorthy, and Dinesh Manocha. Realtime collision avoidance for mobile robots in dense crowds using implicit multi-sensor fusion and deep reinforcement learning. *arXiv preprint arXiv:2004.03089*, 2020.

[99] Yi Lin, Qingyang Wang, Xincheng Yu, Zichen Zhang, Dongyue Guo, and Jizhe Zhou. Towards recognition for radio-echo speech in air traffic control: Dataset and a contrastive learning approach. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[100] Jiaxin Liu, Wenhui Zhou, Hong Wang, Zhong Cao, Wenhao Yu, Chengxiang Zhao, Ding Zhao, Diange Yang, and Jun Li. Road traffic law adaptive decision-making for self-driving vehicles. In *Proc. IEEE Int. Conf. on Intelligent Transportation Systems*, pages 2034–2041. IEEE, 2022.

[101] Shuijing Liu, Peixin Chang, Weihang Liang, Neeloy Chakraborty, and Katherine Driggs-Campbell. Decentralized structural-rnn for robot crowd navigation with

deep reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3517–3524. IEEE, 2021.

[102] Wenliang Liu, Noushin Mehdipour, and Calin Belta. Recurrent neural network controllers for signal temporal logic specifications subject to safety constraints. *IEEE Control Systems Letters*, 6:91–96, 2021.

[103] Yiren Lu, Justin Fu, George Tucker, Xinlei Pan, Eli Bronstein, Rebecca Roelofs, Benjamin Sapp, Brandyn White, Aleksandra Faust, Shimon Whiteson, et al. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7553–7560. IEEE, 2023.

[104] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3), 1992.

[105] Sebastian Maierhofer, Anna-Katharina Rettinger, Eva Charlotte Mayer, and Matthias Althoff. Formalization of interstate traffic rules in temporal logic. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 752–759. IEEE, 2020.

[106] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[107] Andrey Malinin, Sergey Chervontsev, Ivan Provilkov, and Mark Gales. Regression prior networks. *arXiv preprint arXiv:2006.11590*, 2020.

[108] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. *Conf. on Neural Information Processing Systems*, 31, 2018.

[109] Kumar Manas, Stefan Zwicklbauer, and Adrian Paschke. Robust traffic rules and knowledge representation for conflict resolution in autonomous driving. In *Proc. of the 16th International Rule Challenge and 6th Doctoral Consortium @ RuleML+RR*, 2022.

[110] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *European Conference on Computer Vision*, pages 759–776. Springer, 2020.

[111] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. Mantra: Memory augmented networks for multiple trajectory prediction.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7143–7152, 2020.

[112] Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, and Luc De Raedt. From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence*, 328:104062, 2024.

[113] Jasmin Martin, Jenna Riseley, and Jason J Ford. A dataset of stationary, fixed-wing aircraft on a collision course for vision-based sense and avoid. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 144–149. IEEE, 2022.

[114] Sango Matsuzaki and Yuji Hasegawa. Learning crowd-aware robot navigation from challenging environments via distributed deep reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4730–4736. IEEE, 2022.

[115] Christoforos Mavrogiannis, Patrícia Alves-Oliveira, Wil Thomason, and Ross A Knepper. Social momentum: Design and evaluation of a framework for socially competent robot navigation. *ACM Transactions on Human-Robot Interaction (THRI)*, 11(2):1–37, 2022.

[116] Christoforos Mavrogiannis, Francesca Baldini, Allan Wang, Dapeng Zhao, Pete Trautman, Aaron Steinfeld, and Jean Oh. Core challenges of social robot navigation: A survey. *arXiv preprint arXiv:2103.05668*, 2020.

[117] Christoforos Mavrogiannis, Alena M Hutchinson, John Macdonald, Patrícia Alves-Oliveira, and Ross A Knepper. Effects of distinct robot navigation strategies on human behavior in a crowded environment. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 421–430. IEEE, 2019.

[118] Christoforos I Mavrogiannis, Valts Blukis, and Ross A Knepper. Socially competent navigation planning by deep learning of multi-agent path topologies. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6817–6824. IEEE, 2017.

[119] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14424–14432, 2020.

[120] Eric R Mueller, Parmial H Kopardekar, and Kenneth H Goodrich. Enabling airspace integration for high-density on-demand mobility operations. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 3086, 2017.

[121] Kensuke Nakamura and Somil Bansal. Online update of safety assurances using confidence-based predictions. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12765–12771. IEEE, 2023.

[122] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *Int. Conf. on Learning Representations*, 2019.

[123] National Transportation Safety Board. NTSB's Summary of US Civil Aviation Accidents for Calendar Year (CY) 2015, 2015.

[124] Ingrid Navarro and Jean Oh. Social-patternn: Socially-aware trajectory prediction guided by motion patterns. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9859–9864. IEEE, 2022.

[125] Ingrid Navarro, Pablo Ortega, Jay Patrikar, Haichuan Wang, Zelin Ye, Jong Hoon Park, Jean Oh, and Sebastian Scherer. Ameliatf: A large model and dataset for airport surface movement forecasting. In *AIAA AVIATION FORUM AND ASCEND 2024*, page 4251, 2024.

[126] Ingrid Navarro, Jay Patrikar, Joao PA Dantas, Rohan Baijal, Ian Higgins, Sebastian Scherer, and Jean Oh. Sorts: Learned tree search for long horizon social robot navigation. *IEEE Robotics and Automation Letters*, 2024.

[127] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. In *Proc. IEEE Conf. on Robotics and Automation*, pages 2980–2987. IEEE, 2023.

[128] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified architecture for predicting multiple agent trajectories. *arXiv preprint arXiv:2106.08417*, 2021.

[129] Dejan Ničković and Tomoya Yamaguchi. Rtamt: Online robustness monitors from stl. In *International Symposium on Automated Technology for Verification and Analysis*, pages 564–571. Springer, 2020.

[130] Nishant Nikhil and Brendan Tran Morris. Convolutional neural network for trajectory prediction. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[131] Parth Padalkar, Jaeseong Lee, Shiyi Wei, and Gopal Gupta. Improving interpretability and accuracy in neuro-symbolic rule extraction using class-specific sparse filters. *arXiv preprint arXiv:2501.16677*, 2025.

[132] Yutian Pang and Yongming Liu. Conditional generative adversarial networks (cgan) for aircraft trajectory prediction considering weather effects. In *AIAA Scitech 2020 Forum*, page 1853, 2020.

[133] Yutian Pang, Houpu Yao, Jueming Hu, and Yongming Liu. A recurrent neural network approach for aircraft trajectory prediction with weather features from sherlock. In *AIAA Aviation 2019 Forum*, page 3413, 2019.

[134] Jay Patrikar, Joao Dantas, Brady Moon, Milad Hamidi, Sourish Ghosh, Nikhil Keetha, Ian Higgins, Atharva Chandak, Takashi Yoneyama, and Sebastian Scherer. castacks/tartanaviation: 1.0, 2025.

[135] Jay Patrikar, Brady Moon, Sourish Ghosh, Jean Oh, and Sebastian Scherer. Trajair: A general aviation trajectory dataset, Jun 2021.

[136] Jay Patrikar, Brady Moon, Jean Oh, and Sebastian Scherer. Predicting like a pilot: Dataset and method to predict socially-aware aircraft trajectories in non-towered terminal airspace. *arXiv preprint arXiv:2109.15158*, 2021.

[137] Jay Patrikar, Brady Moon, Jean Oh, and Sebastian Scherer. Predicting like a pilot: Dataset and method to predict socially-aware aircraft trajectories in non-towered terminal airspace. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2525–2531. IEEE, 2022.

[138] Jay Patrikar, Sushant Veer, Apoorva Sharma, Marco Pavone, and Sebastian Scherer. Rulefuser: An evidential bayes approach for rule injection in imitation learned planners and predictors for robustness under distribution shifts. *arXiv preprint arXiv:2405.11139*, 2024.

[139] Shaun D Pattinson. The human rights act and the doctrine of precedent. *Legal studies*, 35(1):142–164, 2015.

[140] Chris Paxton, Vasumathi Raman, Gregory D Hager, and Marin Kobilarov. Combining neural networks and tree search for task and motion planning in challenging

environments. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 6059–6066. IEEE, 2017.

[141] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *European conference on computer vision*, pages 452–465. Springer, 2010.

[142] Federico Pigozzi, Eric Medvet, and Laura Nenzi. Mining road traffic rules with signal temporal logic and grammar-based genetic programming. *Applied Sciences*, 11(22):10573, 2021.

[143] Thomas Prevot, Joseph Rios, Parimal Kopardekar, John E Robinson III, Marcus Johnson, and Jaewoo Jung. Uas traffic management (utm) concept of operations to safely enable low altitude flight operations. In *16th AIAA Aviation Technology, Integration, and Operations Conference*, page 3292, 2016.

[144] Aniruddh G Puranic, Jyotirmoy V Deshmukh, and Stefanos Nikolaidis. Learning from demonstrations using signal temporal logic. *arXiv preprint arXiv:2102.07730*, 2021.

[145] Aniruddh G Puranic, Jyotirmoy V Deshmukh, and Stefanos Nikolaidis. Learning from demonstrations using signal temporal logic in stochastic and continuous domains. *IEEE Robotics and Automation Letters*, 6(4):6250–6257, 2021.

[146] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[147] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 81–87. IEEE, 2014.

[148] Katrin Renz, Kashyap Chitta, Otniel-Bogdan Mercea, A Koepke, Zeynep Akata, and Andreas Geiger. Plant: Explainable planning transformers via object-level representations. *arXiv preprint arXiv:2210.14222*, 2022.

[149] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2821–2830, 2019.

[150] Hippolyt Ritter, Aleksandar Botev, and D. Barber. A scalable laplace approximation for neural networks. *Int. Conf. on Learning Representations*, 2018.

[151] Benjamin Riviere, Wolfgang Hönig, Matthew Anderson, and Soon-Jo Chung. Neural tree expansion for multi-robot planning in non-cooperative environments. *IEEE Robotics and Automation Letters*, 6(4):6868–6875, 2021.

[152] Jim Robb. System wide information management (swim): program overview and status update. In *2014 Integrated Communications, Navigation and Surveillance Conference (ICNS) Conference Proceedings*, pages 1–15. IEEE, 2014.

[153] Alexander Robey, Zachary Ravichandran, Vijay Kumar, Hamed Hassani, and George J Pappas. Jailbreaking llm-controlled robots. *arXiv preprint arXiv:2410.13691*, 2024.

[154] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pages 549–565. Springer, 2016.

[155] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[156] Dorsa Sadigh, Nick Landolfi, Shankar S Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state. *Autonomous Robots*, 42(7):1405–1426, 2018.

[157] Fatai Sado, Chu Kiong Loo, Wei Shiung Liew, Matthias Kerzel, and Stefan Wermter. Explainable goal-driven agents and robots-a comprehensive review. *ACM Computing Surveys*, 55(10):1–41, 2023.

[158] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 683–700. Springer, 2020.

[159] Adarsh Jagan Sathyamoorthy, Jing Liang, Utsav Patel, Tianrui Guan, Rohan Chandra, and Dinesh Manocha. Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11345–11352. IEEE, 2020.

[160] Simon Schaefer, Karen Leung, Boris Ivanovic, and Marco Pavone. Leveraging neural network gradients within trajectory optimization for proactive human-robot interactions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9673–9679. IEEE, 2021.

[161] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *Proc. IEEE Conf. on Robotics and Automation*, pages 3399–3406, 2018.

[162] Apoorva Sharma, Navid Azizan, and Marco Pavone. Sketching curvature for efficient out-of-distribution detection for deep neural networks. *arXiv preprint arXiv:2102.12567*, 2021.

[163] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 35:6531–6543, 2022.

[164] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[165] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[166] L. J. Sivian and S. D. White. On minimum audible sound fields, April 1933.

[167] Elysia Q Smyers, Sydney M Katz, Anthony L Corso, and Mykel J Kochenderfer. Avoidds: Aircraft vision-based intruder detection dataset and simulator. *arXiv preprint arXiv:2306.11203*, 2023.

[168] Ajay Srinivasamurthy, Petr Motlicek, Ivan Himawan, Gyorgy Szaszak, Youssef Oualil, and Hartmut Helmke. Semi-supervised learning with semantic knowledge extraction for improved speech recognition in air traffic control. Technical report, 2017.

[169] Benjamin Stoler, Ingrid Navarro, Meghdeep Jana, Soonmin Hwang, Jonathan Francis, and Jean Oh. Safeshift: Safety-informed distribution shifts for robust trajectory prediction in autonomous driving. In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pages 1179–1186. IEEE, 2024.

[170] Ran Tian, Liting Sun, Andrea Bajcsy, Masayoshi Tomizuka, and Anca D Dragan. Safety assurances for human-robot interaction via confidence-aware game-theoretic human models. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 11229–11235. IEEE, 2022.

[171] Elena Torta, Raymond H Cuijpers, and James F Juola. Design of a parametric model of personal space for robotic social navigation. *International Journal of Social Robotics*, 5(3):357–365, 2013.

[172] Jana Tůmová, Luis I Reyes Castro, Sertac Karaman, Emilio Frazzoli, and Daniela Rus. Minimum-violation ltl planning with conflicting specifications. In *Proc. American Control Conference*, pages 200–205, 2013.

[173] Chieh-En Tsai and Jean Oh. A generative approach for socially compliant navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2160–2166. IEEE, 2020.

[174] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *Int. Conf. on Machine Learning*, volume 119, pages 9690–9700. PMLR, 13–18 Jul 2020.

[175] Sushant Veer, Karen Leung, Ryan Cosner, Yuxiao Chen, Peter Karkus, and Marco Pavone. Receding horizon planning with rule hierarchies for autonomous vehicles. *arXiv preprint arXiv:2212.03323*, 2022.

[176] Sushant Veer, Karen Leung, Ryan K Cosner, Yuxiao Chen, Peter Karkus, and Marco Pavone. Receding horizon planning with rule hierarchies for autonomous vehicles. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1507–1513. IEEE, 2023.

[177] Sushant Veer, Apoorva Sharma, and Marco Pavone. Multi-predictor fusion: Combining learning-based and rule-based trajectory predictors. *arXiv preprint arXiv:2307.01408*, 2023.

[178] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[179] Anirudh Vemula, Kathrina Mülling, and Jean Oh. Social attention: Modeling attention in human crowds. In *Proc. of IEEE Conference on Robotics and Automation (ICRA)*, 2018.

[180] Hao Wang, Haoyuan He, Weiwei Shang, and Zhen Kan. Temporal logic guided motion primitives for complex manipulation tasks with user preferences. *arXiv preprint arXiv:2202.04375*, 2022.

[181] Wenguan Wang, Yi Yang, and Fei Wu. Towards data-and knowledge-driven artificial intelligence: A survey on neuro-symbolic computing. *arXiv preprint arXiv:2210.15889*, 2022.

[182] Xiao Wang, Christoph Pillmayer, and Matthias Althoff. Learning to obey traffic rules using constrained policy optimization. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2415–2421. IEEE, 2022.

[183] Yunkai Wang, Quyu Kong, He Zhu, Dongkun Zhang, Longzhong Lin, Hao Sha, Xunlong Xia, Qiao Liang, Bing Deng, Ken Chen, Rong Xiong, Yue Wang, and Jieping Ye. Enhancing closed-loop performance in learning-based vehicle motion planning by integrating rule-based insights. *IEEE Robotics and Automation Letters*, 9(9):7915–7922, 2024.

[184] Hendrik Weber, Julian Bock, Jens Klimke, Christian Roesener, Johannes Hiller, Robert Krajewski, Adrian Zlocki, and Lutz Eckstein. A framework for definition of logical scenarios for safety assurance of automated driving. *Traffic injury prevention*, 20(sup1):S65–S70, 2019.

[185] Roland Weibel and R John Hansman. Safety considerations for operation of different classes of uavs in the nas. In *AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum*, page 6244, 2004.

[186] Xinshuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. Para-drive: Parallelized architecture for real-time autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15449–15458, 2024.

[187] Zhenhua Xu, Yujia Zhang, Enze Xie, Zhen Zhao, Yong Guo, Kwan-Yee K Wong, Zhenguo Li, and Hengshuang Zhao. Drivegpt4: Interpretable end-to-end autonomous driving via large language model. *IEEE Robotics and Automation Letters*, 2024.

[188] Dabin Xue, Li-Ta Hsu, Cheng-Lung Wu, Ching-Hung Lee, and Kam KH Ng. Cooperative surveillance systems and digital-technology enabler for a real-time standard terminal arrival schedule displacement. *Advanced Engineering Informatics*, 50:101402, 2021.

[189] Dabin Xue, Jian Yang, Zhizhao Liu, and Shiwei Yu. Examining the economic costs of the 2003 halloween storm effects on the north hemisphere aviation using flight data in 2019. *Space Weather*, 21(3):e2022SW003381, 2023.

[190] Shakiba Yaghoubi and Georgios Fainekos. Worst-case satisfaction of stl specifications using feedforward neural network controllers: a lagrange multipliers approach. In *2020 Information Theory and Applications Workshop (ITA)*, pages 1–20. IEEE, 2020.

[191] Jianhao Yuan, Shuyang Sun, Daniel Omeiza, Bo Zhao, Paul Newman, Lars Kunze, and Matthew Gadd. Rag-driver: Generalisable driving explanations with retrieval-augmented in-context learning in multi-modal large language model. *arXiv preprint arXiv:2402.10828*, 2024.

[192] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. *arXiv preprint arXiv:2103.14023*, 2021.

[193] Weili Zeng, Zhibin Quan, Ziyu Zhao, Chao Xie, and Xiaobo Lu. A deep learning approach for aircraft trajectory prediction in terminal airspace. *IEEE Access*, 8:151250–151266, 2020.

[194] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019.

[195] Wenyuan Zeng, Shenlong Wang, Renjie Liao, Yun Chen, Bin Yang, and Raquel Urtasun. Dsdnet: Deep structured self-driving network. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 156–172. Springer, 2020.

[196] Xinhai Zhang, Jianbo Tao, Kaige Tan, Martin Törngren, José Manuel Gaspar Sánchez, Muhammad Rusyadi Ramli, Xin Tao, Magnus Gyllenhammar, Franz Wotawa, Naveen Mohan, et al. Finding critical scenarios for automated driving systems: A systematic literature review. *arXiv preprint arXiv:2110.08664*, 2021.

[197] Dapeng Zhao and Jean Oh. Noticing motion patterns: A temporal cnn with a novel convolution operator for human trajectory prediction. *IEEE Robotics and Automation Letters*, 6(2):628–634, 2020.

[198] Ziyu Zhao, Weili Zeng, Zhibin Quan, Mengfei Chen, and Zhao Yang. Aircraft trajectory prediction using deep long short-term memory networks. In *CICTP 2019*, pages 124–135. 2019.

[199] Ev Zisselman and Aviv Tamar. Deep residual flow for out of distribution detection. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.

[200] Juan Zuluaga-Gomez, Karel Veselỳ, Igor Szöke, Alexander Blatt, Petr Motlicek, Martin Kocour, Mickael Rigault, Khalid Choukri, Amrutha Prasad, Seyyed Saeed Sarfjoo, et al. Atco2 corpus: A large-scale dataset for research on automatic speech recognition and natural language understanding of air traffic control communications. *arXiv preprint arXiv:2211.04054*, 2022.