

# Learned Tree Search for Long-Horizon Social Robot Navigation in Shared Airspace

Ingrid Navarro<sup>1\*</sup>, Jay Patrikar<sup>1\*</sup>, Joao P. A. Dantas<sup>2</sup>, Rohan Baijal<sup>3</sup>, Ian Higgins<sup>1</sup>,  
Sebastian Scherer<sup>1</sup> and Jean Oh<sup>1</sup>

**Abstract**—The fast-growing demand for fully autonomous aerial operations in shared spaces necessitates developing trustworthy agents that can safely and seamlessly navigate in crowded, dynamic spaces. In this work, we propose Social Robot Tree Search (SoRTS), an algorithm for the safe navigation of mobile robots in social domains. SoRTS aims to augment existing socially-aware trajectory prediction policies with a Monte Carlo Tree Search planner for improved downstream navigation of mobile robots. To evaluate the performance of our method, we choose the use case of social navigation for general aviation. To aid this evaluation, within this work, we also introduce X-PlaneROS, a high-fidelity aerial simulator, to enable more research in full-scale aerial autonomy. By conducting a user study based on the assessments of 26 FAA-certified pilots, we show that SoRTS performs comparably to a competent human pilot, significantly outperforming our baseline algorithm. We further complement these results with self-play experiments, showcasing our algorithm’s behavior in scenarios with increasing complexity. [Code]<sup>1</sup> | [Video]<sup>2</sup>

## I. 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 [1]–[3], 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) [4], [5]. Deep Reinforcement Learning (DRL) methods [6] 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 [7]. 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.

<sup>1</sup>Authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. {ingridn, jaypat, ihiggins, basti, jeanooh}@cs.cmu.edu

<sup>2</sup>Author is with the Institute for Advanced Studies, Sao Jose dos Campos, SP, Brazil. Work done as a visiting researcher at Carnegie Mellon University. dantasjp@fab.mil.br

<sup>3</sup>Author is with the Indian Institute of Technology Kanpur, Kanpur, Uttar Pradesh, India. Work done as a visiting researcher at Carnegie Mellon University. rbaijal@iitk.ac.in

\*Equal contribution.

<sup>1</sup>Codebase: <https://github.com/cmubig/sorts>

<sup>2</sup>Video: <https://youtu.be/PBE3O4cW2rI>

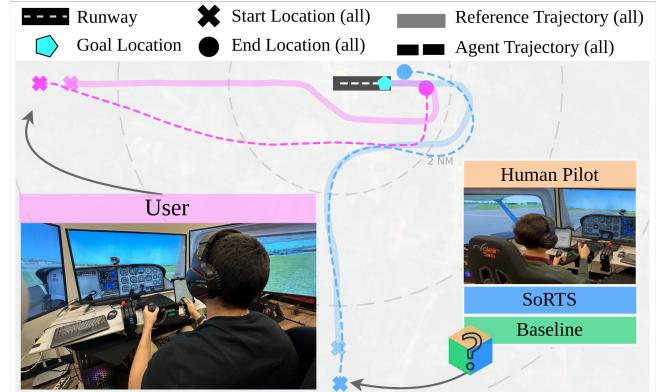


Fig. 1: The flight simulator setup used in our user study where, in randomly selected order, each **User** interacted with our **Human** pilot, **SoRTS** and the **Baseline**. The figure also shows a resulting interaction between a **User** and **SoRTS**.

On the other hand, data-driven approaches are common in social trajectory prediction. They aim to directly characterize human interactions observed in the data [2], eliminating the need for reward shaping and accurate simulations. Recent sequence-to-sequence models, for instance, have achieved promising results in intent prediction [8]–[10]. However, using these models for downstream navigation is difficult as they often suffer from prediction failures [11] 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 [4], [12]. This temporally recursive decision-making intuition has been used for modeling human-like gameplay [13], [14]. 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) [15] 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 [16], [17]. We, therefore, select the domain of general aviation (GA) to showcase our approach. GA was recently framed within the paradigm of social navigation [9], [10], 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. Through a user study conducted on a custom simulator framework, X-PlaneROS, with 26 experienced pilots, we investigate how pilots interact with our model in a realistic flight setting. We further analyze how they gauge our model along these axes as compared to a competent human pilot. We also complement our evaluations with self-play experiments in scenarios with increasing complexity.

#### *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;
- 3) Through a user study with 26 FAA-certified pilots and through self-play simulations in more complex scenarios, we show that SoRTS is perceived comparably to a competent human pilot in terms of *navigation performance* and *safety* and significantly outperforms its baseline algorithm.

## II. RELATED WORK

### A. Social Navigation Algorithms

Social navigation has a rich body of work focused on pedestrian and autonomous driving domains [1]. In pedestrian settings, classical model-based approaches [18], [19] have been proposed and remain prominent baselines. Yet, their extension to other domains is non-trivial. Recent DRL methods [6], [20], [21] that use handcrafted safety-focused reward functions [22] have produced promising results in these domains. However, shortcomings in simulator design [23], and domain-specific reward functions limit real-world performance [22]. Achieving scalability and robustness is challenging, often requiring expensive retraining. Similar to our approach, [5] 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 [1], [22].

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 [24], [25]. To alleviate this, [4] used the gradients of a Q-value function for Model Predictive Control, and [12] 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 [26], 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.

### B. Social Navigation Evaluation

Different metrics have been considered for the evaluation of social robot navigation [1], [2], [27]. Some of the main axes of analysis for evaluation include behavior naturalness based on a reference trajectory or irregularity of the executed path [28], [29], performance and efficiency [30]–[32], and notions of physical personal space or discomfort [33], [34]. User studies are often conducted to evaluate more subjective aspects such as the perceived discomfort and trust that a robot induces during an interaction [35]. 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.

## III. 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 [4], [5], [36]. Following [9], [10], 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  $\tau_R$ . 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  $\pi = \{\mathbf{a}_0, \dots, \mathbf{a}_g\}$  such that the agents follow collision-free trajectories  $\tau = \{\mathbf{s}, \dots, \mathbf{s}_g\}$ . The generated trajectories need to ensure  $\|\mathbf{s}_t^i - \mathbf{s}_t^j\| \geq d \forall i, j \in \{1, \dots, M\} \forall t$  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.

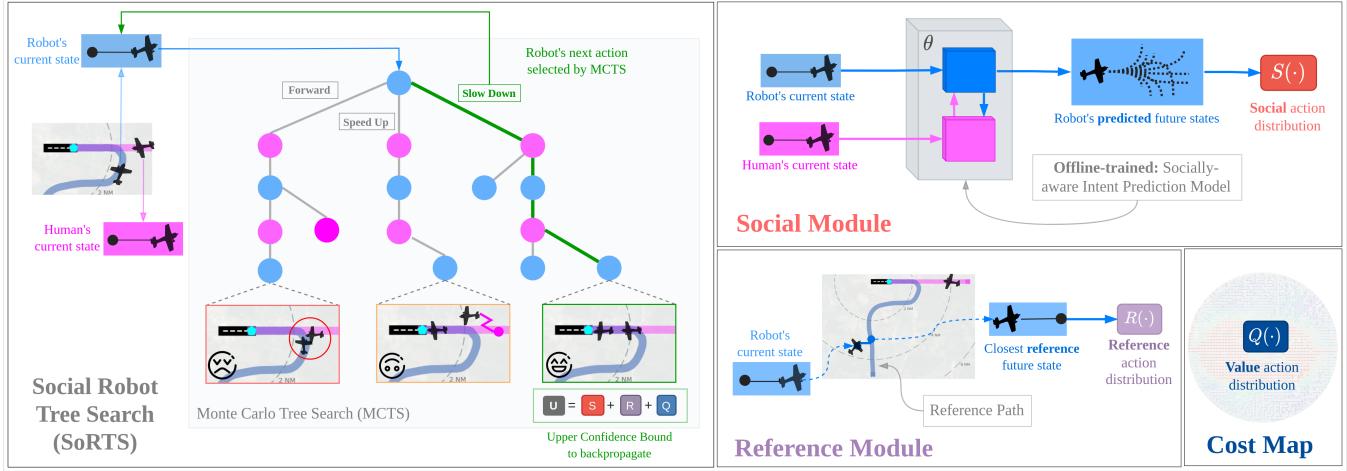


Fig. 2: An overview of SoRTS, a Monte Carlo Tree Search (MCTS)-based planner for social robot navigation whose tree search is guided by three components; a **Social Module**, a **Reference Module** and a **Cost Map**. The Social Module leverages an offline-trained intent prediction model to characterize 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.

#### IV. 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. 2 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.

##### A. 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 1 and 2.

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

$$p_\theta(\mathbf{s}_t^i, \mathbf{a}_t^i) \sim P_\theta(\mathbf{a}_t^i | \mathbf{s}_{t-t_{obs}:t}, \mathbf{s}_g^i) \quad (1)$$

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 [10], an algorithm which predicts multi-future trajectories from learned interactions that exploit motion pattern information in the data.

2) *Reference Path Module*: Given the start and goal state of agent  $i$  the algorithm samples a suitable reference trajectory  $\tau_r^i \in \mathcal{T}_R$ , line 1 in Algorithm 1. Similar to section IV-A.1, this trajectory is used to compute a *reference* action distribution  $p_r(\cdot)$ ,

$$p_r(\mathbf{s}_t^i, \mathbf{a}_t^i) \sim P_r(\mathbf{a}_t^i | \mathbf{s}_t^i, \tau_r^i) \quad (2)$$

proportional to the L2 norm between  $\mathbf{s}$  and  $\tau_r$  at time  $t$ . In Algorithm 2, the reference action is obtained in line 5.  $\tau_R$  can be drawn from expert distributions  $\mathcal{D}$ , global path planning algorithms like A-Star, logic specifications [37] or can also be handcrafted.

3) *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 2 uses the cost map in line 3. This value function can be either learned, *e.g.*, via self-play [5] or pre-computed from a prior distribution and captures the value of the joint state distribution.

##### B. 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 [15], 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,

$$U(\mathbf{s}, \mathbf{a}) = Q(\mathbf{s}, \mathbf{a}) + c_1 \cdot S(\mathbf{s}, \mathbf{a}) + c_2 \cdot R(\mathbf{s}, \mathbf{a}) \quad (3)$$

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 line 14 of algorithm 2, these values follow the update:

$$Q(\mathbf{s}, \mathbf{a}) = \frac{N(\mathbf{s}, \mathbf{a}) \cdot Q(\mathbf{s}, \mathbf{a}) + v(\mathbf{s})}{N(\mathbf{s}, \mathbf{a}) + 1} \quad (4)$$

$$R(\mathbf{s}, \mathbf{a}) = \frac{N(\mathbf{s}, \mathbf{a}) \cdot R(\mathbf{s}, \mathbf{a}) + p_r(\mathbf{s}, \mathbf{a})}{N(\mathbf{s}, \mathbf{a}) + 1} \quad (5)$$

$$S(\mathbf{s}, \mathbf{a}) = \frac{\sqrt{N(\mathbf{s})}}{N(\mathbf{s}, \mathbf{a}) + 1} \cdot p_\theta(\mathbf{s}, \mathbf{a}) \quad (6)$$

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.<sup>3</sup> At the end of  $planHorizon$ , the ego agent takes the first action that maximizes the visitation count in line 4 of Algorithm 1. The tree is reset and the process continues till goal is reached.

#### Algorithm 1 SoRTS( $\mathbf{s}_0, \mathbf{s}_g, \theta, \tau_R, v$ )

```

1:  $\tau_r \leftarrow \text{GetReferencePaths}(\mathbf{s}_0, \mathbf{s}_g, \tau_R)$ 
2: while  $\mathbf{s}_t^0 \neq \mathbf{s}_g^0$  or  $timeElapsed \leq planHorizon$  do
3:    $N(\cdot) \leftarrow \text{SocialMCTS}(\mathbf{s}_t, \mathbf{s}_G, \tau_r, \theta, v, 0)$ 
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
```

## V. 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.

### A. User Study

We recruited 26 FAA-certified pilots (14 private, 8 commercial, 3 student pilots and 1 airline transport pilot), who on average have 986 flight hours. Using the flight deck setup and simulator shown in fig. 1, each pilot was asked to land an aircraft on a specified runway. Simultaneously, a second pilot was solving the same task, thus, requiring the pilot's

<sup>3</sup>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,  $p_\theta$  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 2 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:      $v_s \leftarrow \text{GetValue}(v, \mathbf{s}_t)$ 
4:      $S(\cdot) \leftarrow \text{GetSocialActionProbabilities}(\mathbf{s}_t, \theta)$ 
5:      $p_r(\cdot) \leftarrow \text{GetReferenceActionProbabilities}(\tau_r^p, \mathbf{s}_t^p)$ 
6:      $N(\mathbf{s}_t^p) \leftarrow 1$ 
7:     return  $v_s$ 
8:   end if
9:    $\mathcal{A}' \leftarrow \text{CollisionCheck}(\mathcal{A}, d, \mathbf{s}_t)$ 
10:   $\mathbf{a}^p \leftarrow \arg \max_{\mathbf{a}' \in \mathcal{A}'} U(\mathbf{s}_t^p, \mathbf{a}')$            ▷ See eq. 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:   $\text{SocialMCTS}(\mathbf{s}, \mathbf{s}_g, \tau_r, \theta, v, p)$ 
14:   $\text{Update}(Q, R, S, N)$                                 ▷ See eq. 4 to 6
15: end while
16: return  $N(\cdot)$ 
```

---

coordination to land safely. Here, the second pilot was either a human, SoRTS, or the baseline algorithm<sup>4</sup>.

Our experimental setup follows the design delineated next. We followed a *within-subject* design in which each user tests against each algorithm. We allowed the user to get familiarized with the simulator and controls prior to the tests. The pilots are spawned at a 10 km radius from the airport, where their incoming direction is either north (N), south (S), or west (W), defining six possible scenarios, *i.e.*,  $\{(N, S), (S, N), (N, W), (W, N), (S, W), (W, S)\}$ . The algorithm order and scenario were selected randomly at the beginning of the experiment. The scenario, the initial states and final goals remained fixed throughout the three tests.

After each test, the user completed a 5-scale Likert questionnaire evaluating the second pilot along various factors we deemed relevant for high navigation performance and safety. Since operation in safety-critical domains demands high competence, we also take an interest in understanding which aspects the users prioritize for deeming a pilot as trustworthy and competent. We, thus, ask users to also rate how *trustworthy* and *human-like* they perceived the second pilot, based on their interaction with them and their responses along the axes of navigation and safety. We summarize the components of our user study questionnaire in table I. Finally, we also collected the trajectory data from each experiment for further analysis using relevant metrics discussed in section V-C.

### B. Self-Play

We complement our user study with self-play simulations to assess SoRTS and our baseline on a wider variety of scenarios. The simulations follow a similar design as the user study, but now allow each agent's location to be anywhere around the 10km radius to increase the number of possible scenarios. We also consider multi-agent scenarios varying from 2 to 4 agents. We randomly generate 100 episodes for each setting, where in each episode an agent is deemed

<sup>4</sup>We use *second pilot* and *algorithm* interchangeably.

unsuccessful if it breaches a minimum separation distance, or if it reaches a maximum number of allowed steps.

### C. Metrics

To quantify the trajectory data collected in our user study and self-play experiments along our axes of analysis, we consider the metrics listed below;

- 1) *Reference Error (RE)*: The Euclidean distance between the agent’s reference trajectory and its executed path.
- 2) *Loss of Separation (LS)*: The duration that two agents break a minimum distance of each other. This metric is relevant for the aviation domain [38], but is akin to commonly used metrics for social robots in pedestrian settings, *e.g.*, personal space [1], [27].

### D. Baseline

To showcase the benefits of SoRTS, we introduce a baseline which naively chooses the next action by balancing the *reference* and *social* values over the state-action space following the equation below,

$$\mathbf{a} = \arg \max_{\mathbf{a}' \in \mathcal{A}} [\lambda \cdot p_r(\mathbf{s}, \mathbf{a}) + (1 - \lambda) \cdot p_\theta(\mathbf{s}, \mathbf{a})]$$

where  $\lambda \in \mathbb{R} : [0, 1]$  balances the importance we give to  $p_\theta(\cdot)$  and  $p_r(\cdot)$ . This baseline translates to replacing line 3 and line 4 for the above equation in algorithm 1.

### E. Simulator

To evaluate the proposed algorithm and enable future research in the domain of full-scale aerial autonomy, we introduce X-PlaneROS. The system combines two main components via a ROS bridge, X-Plane-11 and ROS-Plane autopilot [39], enabling the use of high and low-level commands to control a general aviation aircraft in realistic world scenarios. X-Plane-11 is a high-fidelity simulator which provides realistic aircraft models and visuals. ROS-Plane is a widely accepted tool which provides reliable autonomous flight control loops. Our simulator further adds support for following a select set of motion primitives, as well as visualization utilities that aid in tuning the control loops. The documentation and source code is open-sourced<sup>5</sup>.

### F. Implementation Details

The modules in section IV-A leverage TrajAir [9], a dataset consisting of aircraft trajectory data collected in non-towered terminal airspace; the *Social Module* uses TrajAir to train its intent prediction module offline, where we followed the training details in [10]. We also use TrajAir to build a library of FAA-abiding paths used by the *Reference Module*. Finally, to build our *cost map* we discretized TrajAir’s flight frequency based on aircraft poses and wind direction.

The action space,  $\mathcal{A}$ , of our planner is comprised of the set of 252 motion primitives for aerial navigation used in [37]. Empirically, we set *planHorizon*=10s in algorithm 2 and the exploration parameters in the UCT equation to  $c_1 = 2$  and  $c_2 = 5$ , and  $\lambda = 0.3$  for the baseline planner.

TABLE I: We asked users to rate the *second pilot* along these aspects of *navigation performance* and *safety*, and to rate it along *trustworthiness* and *humanness*.

Navigation Performance	Safety	Was the second pilot...
Follow FAA Guidelines	Collision Risk	trustworthy?
Overall Flying Skill	Comfort	a human?
Flight Smoothness	Cooperative	
	Abrupt	
	Predictable	

## VI. RESULTS

We now present our results and insights. For our analysis below, a *competent algorithm* refers to one which performs highly along the axes of navigation performance and safety. We first examine the relevance of the factors in table I in characterizing the *trustworthiness* of a competent algorithm. Leveraging the results from our *trustworthiness* analysis, we then perform a comparative analysis between SoRTS, the baseline and our human pilot. Finally, we briefly explore how pilots perceive human competence along the factors in our questionnaire.

### A. On competence and trustworthiness

As a means to determine how to better assess the competence of a robot, *e.g.*, which metrics to prioritize, we compare the factors in table I with the user’s perceived trust. We do so via a Pearson’s correlation analysis with repeated measures, whose results are summarized in table II.

We find strong correlations between trust and the components within navigation performance, with flight smoothness being the strongest one. Within safety, we see that *comfort*, followed by *predictability* showed the strongest correlations. Although one would expect for *abruptness* and *collision risk* to be as crucial for rating trust. We find this result is, in part, due to our experiments not being explicitly designed to exhibit adversarial behavior. We also believe these results showcase the strength of our model in unprecedented scenarios. As said earlier, because data-driven models can heavily misbehave when exposed to out-of-distribution states [11], SoRTS is a means to augment these models online. Since, the user study serves as a mechanism for testing our model in settings that weren’t observed in the training data, the lower correlation between the aforesaid components may indicate robustness toward unsafe behavior.

### B. On each algorithm’s competence

The previous section studied how each factor in table I tied to the perception of a trustworthy pilot. This section will now focus on providing a pairwise comparison between the *algorithms* for both the user study and self-play, leveraging the results obtained in the previous analysis.

1) *User study*: Leveraging the trustworthiness results in table II, we compute each user’s average score over the questions with highest correlations along each axis, *i.e.*, for navigation performance, we compute the mean score

<sup>5</sup>Code for X-PlaneROS: [https://github.com/castacks/xplane\\_ros](https://github.com/castacks/xplane_ros)

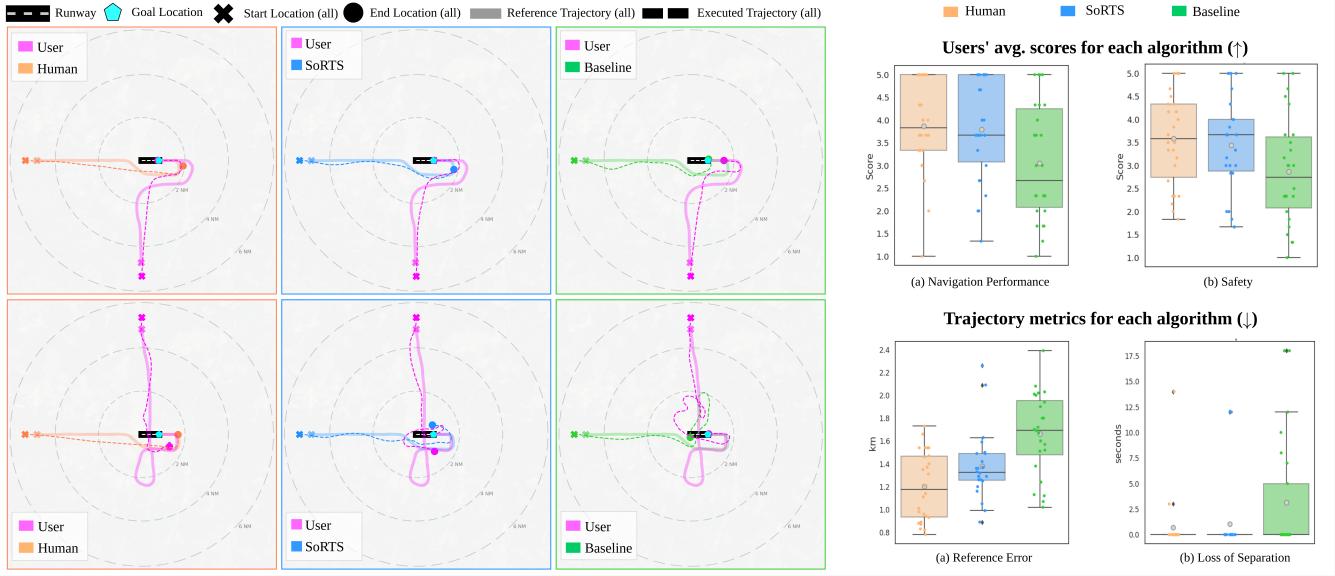


Fig. 3: User study results. **Left:** Examples of two experiments. Each row shows the resulting trajectories of a **User** interacting with our **Human** pilot, **SoRTS**, and the **Baseline**. The top row shows a user that successfully followed the expected guideline; we can see that the baseline did not fly as smoothly as SoRTS, and abruptly cut short close to the goal. The bottom row shows a user that did not follow the expected guideline; our algorithm still managed to navigate properly, whereas the baseline behaved erratically, unsafely crossing over the runway. **Right:** Box-plots showing the *per-algorithm* distribution of the results from the user study: the top ones show the average scores given by the user for navigation performance (a) and safety (b). The bottom ones show the average reference error (a) and loss-of-separation (b) metrics obtained from the trajectory data.

TABLE II: Using repeated measures, we show the Pearson correlations ( $R$ ,  $p$ -value=0.05) for each factor listed in table I *vs.* perceived trustworthiness and humanness.

Axis	Factor	Trustworthy?		Human?	
		R	p-value	R	p-value
Nav. Performance	Flight Smoothness	0.81	4.80e-19	0.08	0.47
	Follow FAA Guidelines	0.76	2.99e-15	0.10	0.37
	Overall Flying Skill	0.71	4.56e-13	0.15	0.20
Safety	Comfortable	0.92	3.55e-31	0.17	0.14
	Predictable Behavior	0.77	5.19e-16	0.21	0.07
	Cooperative	0.65	1.61e-10	0.06	0.62
	Collision Risk	-0.56	1.10e-07	-0.11	0.36
	Abrupt	-0.56	1.13e-07	-0.09	0.42

between *following FAA* and *flight smoothness*, and for safety we use *predictability* and *comfort*. The scores are shown in fig. 3, and the pairwise statistical differences using ANOVA with repeated measures in table III. Here, we did not find statistical evidence that the scores for the human pilot and SoRTS were different, suggesting that the users rated their competence similarly. Our results further show evidence that our baseline was generally rated lower on both axes of competence, while also exhibits more variance.

We then examine if the RE and LS metrics in section V-C—which here tie to navigation performance and safety, respectively—reflect the users’ assessments for each algorithm. Thus, we compute them on the collected trajectories and show their respective mean scores and statistical comparisons in fig. 3 and table III. As before, we show the mean scores for each algorithm in fig. 3 and the statistical results in

table III. For the RE metric, we find significantly different results between the algorithms; wherein SoRTS yields higher error compared the human pilot, but markedly lower than the baseline and less variance than the other two. Similarly, the LS metric computed at 0.5km, shows that generally neither the human pilot nor SoRTS breach this distance. In contrast, the baseline more frequently invades others’ personal space, creating more situations for potential collisions.

TABLE III: Statistical significance between algorithmic pairs for results in fig. 3 with  $t^* \geq 2.060$  and  $p \leq 0.05$ .

Algorithmic Pair	NP		Safety		RE		LS	
	t-val	p-val	t-val	p-val	t-val	p-val	t-val	p-val
Baseline-Human	3.121	0.009	3.062	0.016	5.782	0.000	1.321	0.199
Baseline-SoRTS	3.018	0.009	2.626	0.022	2.105	0.011	0.397	0.694
Human-SoRTS	0.415	0.682	0.322	0.322	2.944	0.022	1.211	0.237

NP: Navigation Performance, LS: Loss of Separation, RE: reference error.

Figure 3 shows examples of trajectories from our experiments. Each row represents one user *vs.* the three *algorithms*. We show a reference trajectory along with the actual executed trajectory<sup>6</sup>. The top row shows an example where the user performed successfully. In this experiment, we see that the ablation unexpectedly cut short while approaching the runway instead of following the traffic pattern, while SoRTS did so smoothly and safely. The bottom row shows a user that did not follow the standard traffic pattern. Despite

<sup>6</sup>Human pilots do not see the reference trajectory, but they see the map in the simulator, and are instructed to follow the appropriate traffic patterns.

this, SoRTS manages to successfully complete the task, while the baseline behaves erratically, not following the traffic pattern and unsafely traversing the runway twice.

As per the user study, we conclude that SoRTS performs comparable to a competent human pilot and significantly better than the ablation, both as perceived by the users and through our metrics. Our results further highlight the benefits of using the long-horizon socially-aware simulations via MCTS, as opposed to purely using a data-driven model on the wild with a simple weighting over the objectives of interest, leading to an erratic and unsafe behavior.

TABLE IV: Summary of the task performance of the baseline and SoRTS’ agents in self-play.

# Agents	Algorithm	Success (%)	Failure (%)		RE
			LS	Timeout	
2	Baseline	69.0	30.0	1.0	1.59
	SoRTS	96.5	2.0	1.5	2.01
3	Baseline	48.3	49.3	2.4	1.55
	SoRTS	89.7	8.3	2.0	2.01
4	Baseline	43.5	54.0	2.5	1.56
	SoRTS	71.0	21.0	8.0	1.96

LS: Loss of Separation at 0.5km, RE: reference error for successful agents.

2) *Self-play*: We now analyze the performance of our algorithm in more complex scenarios. The results summarized in Table IV show the percentage of agents that were able to land on the runway, *i.e.*, successful agents, the agents that were unsuccessful, due to either a loss of separation, or because they exceeded the maximum allowed time. Finally, we show the average reference error for successful agents.

Analyzing the table, we observe a decrease in task success as the number of agents increases. Nonetheless, we can see that SoRTS performs significantly better than the ablation algorithm improving the task success by 28%, 46% and 38% for the 2, 3 and 4 agent scenarios, respectively. We also find that, the average reference error for SoRTS is higher than the ablation. Though this error was computed on successful agents only, we hypothesize that it being higher is due to conflict resolution to avoid collisions with other agents.

### C. On competence and human performance

Often arising debate, is the question of whether an algorithm can pass as a human. However, in domains such as aviation, where high competence is central, one would expect for humans to be perceived as such. If so was the case, one would strive for an algorithm to exhibit a performance similar to that of a human. We study the aforesaid by asking users to gauge whether the second pilot was a human based on their assessments along table I.

Surprisingly, we find a marked disagreement within the users’ responses to this question, with almost a 50-50% split between the responses for the *Human Pilot* (14: No, 12: Yes) and the *Baseline* (14: No, 12: Yes), whereas for *SoRTS* (8: No, 18: Yes) more users perceived its performance as human-like. Further, table II shows that the user’s assessments for

navigational performance and safety correlate weakly with the *humanness* prediction. To explain this contrasting result, we isolated the responses given to SoRTS from the other two, and found that *predictability* had the highest correlation value among all factors, with  $R=0.53$  and  $p=0.01$ . In contrast, the values for the Human pilot and the Baseline were  $R=0.22$ ,  $p=0.03$  and  $R=-0.06$ ,  $p=0.06$ , respectively. We believe this result may suggest that because users generally perceived the behavior of SoRTS as more predictable, they rated its performance as human-like more frequently.

We believe this contrasting result does not affect our analyses from the previous sections, as we find that the user’s responses for *trustworthiness* also correlate weakly with their assessment of *humanness*, with  $R=0.26$ ,  $p=0.02$ . As such, we conclude that users value more their perception of trust than the nature of type of agent they interact with.

## VII. 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. We find that users perceive SoRTS comparable to a competent human pilot and significantly better than our baseline. In self-play, we show that SoRTS outperforms the baseline by 28-46% as the complexity of scenarios increases. 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.

We identify various avenues for future work. First, we assumed task homogeneity, *i.e.*, agents landing on the same runway, whereas in a real scenario pilots with different objectives may require to interact. Thus, future work includes studying interactions with agents with heterogeneous tasks. We also assumed perfect intent and state estimation. Accordingly, robustifying prediction models with uncertainty- and adversarial-awareness is another promising direction. Finally, improvements on the scalability of the model, as well as exploring other domains are also potential avenues.

## ACKNOWLEDGMENT

This work was supported by the Army Futures Command Artificial Intelligence Integration Center (AI2C), the Ministry of Trade, Industry and Energy (MOTIE), the Korea Institute of Advancement of Technology (KIAT), and the Brazilian Air Force. The views expressed in this article do not necessarily represent those of the aforementioned entities. We thank Jasmine Jerry Aloor for her support. We also thank the pilots from Condor Aero Club (KPJC) and ABC Flying Club (KAGC) for their participation in our user study.

## REFERENCES

- [1] C. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld, and J. Oh, "Core challenges of social robot navigation: A survey," *arXiv preprint arXiv:2103.05668*, 2021.
- [2] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.
- [3] R. Tian, L. Sun, A. Bajcsy, M. Tomizuka, and A. D. Dragan, "Safety assurances for human-robot interaction via confidence-aware game-theoretic human models," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 11 229–11 235.
- [4] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. 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*, vol. 42, no. 7, pp. 1405–1426, 2018.
- [5] B. Riviere, W. Hönig, M. Anderson, and S.-J. Chung, "Neural tree expansion for multi-robot planning in non-cooperative environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6868–6875, 2021.
- [6] S. Matsuzaki and Y. Hasegawa, "Learning crowd-aware robot navigation from challenging environments via distributed deep reinforcement learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4730–4736.
- [7] C. I. Mavrogiannis, V. Blukis, and R. 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)*. IEEE, 2017, pp. 6817–6824.
- [8] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *European Conference on Computer Vision*. Springer, 2020, pp. 683–700.
- [9] J. Patrikar, B. Moon, J. Oh, and S. 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)*. IEEE, 2022, pp. 2525–2531.
- [10] I. Navarro and J. Oh, "Social-patternn: Socially-aware trajectory prediction guided by motion patterns," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 9859–9864.
- [11] A. Farid, S. Veer, B. Ivanovic, K. Leung, and M. Pavone, "Task-relevant failure detection for trajectory predictors in autonomous vehicles," in *Conference on Robot Learning*. PMLR, 2023, pp. 1959–1969.
- [12] H. Hu and J. F. Fisac, "Active uncertainty reduction for human-robot interaction: An implicit dual control approach," *arXiv preprint arXiv:2202.07720*, 2022.
- [13] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.
- [14] A. P. Jacob, D. J. Wu, G. Farina, A. Lerer, H. Hu, A. Bakhtin, J. Andreas, and N. Brown, "Modeling strong and human-like gameplay with kl-regularized search," 2021.
- [15] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Machine Learning: ECML 17th European Conference on Machine Learning, Berlin, Germany, September 18–22, 2006, Proceedings*, ser. Lecture Notes in Computer Science, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds., vol. 4212. Springer, 2006, pp. 282–293.
- [16] J.-P. Aurambout, K. Gkoumas, and B. Ciuffo, "Last mile delivery by drones: An estimation of viable market potential and access to citizens across european cities," *European Transport Research Review*, vol. 11, no. 1, pp. 1–21, 2019.
- [17] M. Grote, A. Pilko, J. Scanlan, T. Cherrett, J. Dickinson, A. Smith, A. Oakey, and G. Marsden, "Sharing airspace with uncrewed aerial vehicles (uavs): Views of the general aviation (ga) community," *Journal of Air Transport Management*, vol. 102, p. 102218, 2022.
- [18] J. v. d. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.
- [19] C. Mavrogiannis, P. Alves-Oliveira, W. Thomason, and R. A. Knepper, "Social momentum: Design and evaluation of a framework for socially competent robot navigation," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 11, no. 2, pp. 1–37, 2022.
- [20] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 285–292.
- [21] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6015–6022.
- [22] C.-E. Tsai and J. Oh, "A generative approach for socially compliant navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2160–2166.
- [23] A. Biswas, A. Wang, G. Silvera, A. Steinfeld, and H. Admoni, "Socnavbench: A grounded simulation testing framework for evaluating social navigation," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 11, no. 3, pp. 1–24, 2022.
- [24] M. A. Bashiri, B. Ziebart, and X. Zhang, "Distributionally robust imitation learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 404–24 417, 2021.
- [25] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9329–9338.
- [26] S. Schaefer, K. Leung, B. Ivanovic, and M. Pavone, "Leveraging neural network gradients within trajectory optimization for proactive human-robot interactions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9673–9679.
- [27] Y. Gao and C.-M. Huang, "Evaluation of socially-aware robot navigation," *Frontiers in Robotics and AI*, p. 420, 2021.
- [28] C. Mavrogiannis, A. M. Hutchinson, J. Macdonald, P. Alves-Oliveira, and R. 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)*. IEEE, 2019, pp. 421–430.
- [29] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, "Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 11 345–11 352.
- [30] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell, "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3517–3524.
- [31] M. Everett, Y. F. Chen, and J. 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)*. IEEE, 2018, pp. 3052–3059.
- [32] J. Liang, U. Patel, A. J. Sathyamoorthy, and D. 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.
- [33] E. Torta, R. H. Cuijpers, and J. F. Juola, "Design of a parametric model of personal space for robotic social navigation," *International Journal of Social Robotics*, vol. 5, no. 3, pp. 357–365, 2013.
- [34] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.
- [35] J. Butler and A. Agah, "Psychological effects of behavior patterns of a mobile personal robot," *Autonomous Robots*, vol. 10, pp. 185–202, 03 2001.
- [36] R. Emery-Montemerlo, G. J. Gordon, J. G. Schneider, and S. 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*. IEEE Computer Society, 2004, pp. 136–143.
- [37] J. J. Aloor, J. Patrikar, P. Kapoor, J. Oh, and S. Scherer, "Follow the rules: Online signal temporal logic tree search for guided imitation learning in stochastic domains," *arXiv preprint arXiv:2209.13737*, 2022.
- [38] T. Glozman, A. Narkawicz, I. Kamon, F. Callari, and A. Navot, "A vision-based solution to estimating time to closest point of approach for sense and avoid," in *AIAA Scitech 2021 Forum*, 2021, p. 0450.
- [39] G. Ellingson and T. McLain, "Rosplane: Fixed-wing autopilot for education and research," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 1503–1507.