IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

SoRTS: Learned Tree Search for Long Horizon Social Robot Navigation

Ingrid Navarro[1*], Jay Patrikar[1*], Joao P. A. Dantas[1], Rohan Baijal[1], Ian Higgins[1], Sebastian Scherer[1] and Jean Oh[1]

***Abstract*—The fast-growing demand for fully autonomous robots in shared spaces calls for developing trustworthy agents that can safely and seamlessly navigate crowded environments. Recent models for motion prediction show promise in characterizing social interactions in such environments. However, using them for downstream navigation can lead to unsafe behavior due to their myopic decision-making. Prompted by this, we propose *Social Robot Tree Search* (SoRTS), an algorithm for safe robot navigation in social domains. SoRTS aims to augment existing socially aware motion prediction models for long-horizon navigation using Monte Carlo Tree Search.**

We use social navigation in *general aviation* as a case study to evaluate our approach and further the research in full-scale aerial autonomy. In doing so, we introduce X-PlaneROS, a high-fidelity aerial simulator that enables human-robot interaction. We use X-PlaneROS to conduct a first-of-its-kind user study where 26 FAA-certified pilots interact with a human pilot, our algorithm, and its ablation. Our results, supported by statistical evidence, show that SoRTS exhibits comparable performance to competent human pilots, significantly outperforming its ablation. Finally, we complement these results with a broad set of self-play experiments to showcase our algorithm's performance in scenarios with increasing complexity. [Code | Simulator | Video]

***Index Terms*—Human-aware Motion Planning, Safety in HRI, Aerial Systems: Perception and Autonomy.**

I. Introduction

A social robot strives to synthesize decision policies that enable it to interact with humans, ensuring social compliance while attaining its desired goal. While marked progress has been made in the fields of social navigation [1]–[3] and *socially-aware* motion prediction [4], achieving seamless navigation among humans while balancing social and self-interested objectives remains challenging.

Classical model-based approaches for social navigation in pedestrian settings have been proposed and remain prominent baselines [5]–[7]. Yet, their extension to *other domains* is often nontrivial owing to higher environmental and social complexities intrinsic to each domain [8]. Deep Reinforcement Learning methods have also been vastly explored within the field. Under this formulation, common strategies include approximations to the policy's reward function via hand-crafted design [9], [10], or self-play [11], [12]. Such techniques are generally promising in settings where data is sparse or where the robot is easily distinguishable from humans. However, tuning reward parameters for homogeneous navigation among humans is challenging [13]. Furthermore, RL-based policies are a function of the underlying simulator, often yielding undesirable behavior in real scenarios due to a lack of compatibility between the simulated environment and the real world. More recently, data-driven approaches have been extensively explored largely within the field of socially-aware motion prediction. These methods aim at characterizing human behavior and interactions observed in the data, alleviating the need for reward shaping and simulation. These models have achieved promising performance [4]. However, using them for downstream navigation can lead to unsafe behavior due to myopic decision-making, prompting the need for robustifying models deployed in the real world.

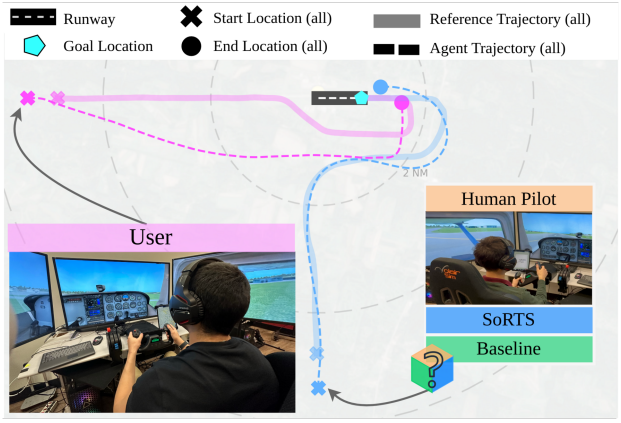

Fig. 1: Our flight simulator setup and a user study experiment. In each, the User interacts with a Human pilot, our proposed algorithm, SoRTS, and our Ablation. The figure also shows a resulting interaction between a User and SoRTS.

Motivated by these limitations, we introduce *Social Robot Tree Search (SoRTS)*, an algorithm that seeks to robustify offline-learned socially aware motion prediction models for their deployment in online settings. We build SoRTS upon the insight that, in social navigation settings, the actions of one agent influence those of another, and vice-versa [9], [14]. This type of temporally recursive decision-making has been used for modeling human-like gameplay via search-based policies [15], [16]. We, thus, propose using recursive search-based policies as a means to *augment* the aforesaid prediction methods. Focusing on aleatoric uncertainty, we choose to evaluate and robustify the in-domain performance of prediction models for navigation tasks. Specifically: we use Monte Carlo Tree Search (MCTS) [17] as our search policy that provides long-horizon simulations, collision checking, and goal conditioning.

Manuscript received: September, 15, 2023; Revised December, 19, 2023; Accepted February, 5, 2024.

This paper was recommended for publication by Editor Gentiane Venture upon evaluation of the Associate Editor and Reviewers' comments.

[1]Authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. {ingridn, jaypat, jdantas, ihiggins, basti, jeanoh}@cs.cmu.edu

*Equal contribution.

Digital Object Identifier (DOI): see top of this page.

We then bias its tree search using a *reference* module, which guides the planner to favorable regions, and a *social* module—consisting of the offline-trained motion prediction model—to provide short-horizon agent-to-agent context cues.

We note that SoRTS is intended to be *domain-independent*. Thus, we maintain the formulation details abstract to allow for flexibility in the representation. In this work, however, we use *General Aviation (GA)* as a case study and describe our motivation next. Unmanned Aerial Vehicle (UAV) operations have seen a significant increase in recent years. As a result, there is a growing demand for the development of automated technologies that can be reliably integrated into the National Airspace System [18], [19], enabling the concurrent use of airspace between human and autonomous aircraft. The next generation of UAVs is expected to operate in low-altitude terminal airspace, where most close-proximity interactions occur [20]. Thus, it is crucial to develop technologies that ensure the safe and seamless interaction of these autonomous systems with humans.

Toward this end, recent works have studied the domain of GA in non-towered airports in the context of socially aware motion prediction [20], [21], where the goal is to characterize joint interactions in such environments. In this domain, there is no central air traffic control authority to regulate aircraft operations, making pilots solely responsible for coordinating with each other and for following guidelines and traffic patterns established by the FAA to ensure proper operations. Our work builds upon [20], [21], exploring this novel domain in the context of social robot navigation. GA is a safety-critical domain demanding the development of competent and trustworthy robots that can properly follow navigation norms but also understand social cues to guarantee *seamless* and *safe* interactions. We separate these two notions into two axes: *navigation efficiency* and *safety*. We center the design of our algorithm, as well as our evaluations around these axes. In doing so, we conducted a user study with 26 experienced pilots using our custom simulator framework, X-PlaneROS. In our study, we investigate how pilots interact with our model in a realistic flight setting. We also study their perceptions of our model's performance along the preceding axes when compared to competent human pilots. Finally, as a complementary analysis, we provide evaluations via self-play experiments in more complex scenarios. *Statement of Contributions:*

1) We introduce SoRTS, an MCTS-based algorithm that aims at robustifying offline-learned socially-aware motion prediction policies for long-horizon navigation.
2) We introduce X-PlaneROS, a high-fidelity simulation environment for navigation in shared aerial space.
3) We showcase the efficacy of SoRTS in the General Aviation domain. Through a first-of-its-kind user study with 26 FAA-certified pilots and self-play experiments, we show that SoRTS is perceived comparably to a competent human pilot in terms of *navigation efficiency* and *safety* while outperforming its ablation algorithm.

II. Related Work

A. Social Navigation Methods

Social navigation has a rich body of work primarily focused on human crowds and autonomous driving [1], [4]. Several prominent model-based approaches [5]–[7] have been proposed for pedestrian settings, but their extension to settings with increasing complexities is difficult. RL-based methods [12], [22], [23] have produced promising results in these settings by leveraging safety-based handcrafted reward functions. However, shortcomings in simulator design [24], and domain-specific reward functions limit real-world performance [25]. Achieving scalability and robustness is challenging, often requiring expensive retraining for non-significant test distribution shifts.

Data-driven approaches focus on exploiting natural behavior by learning policies from datasets that record interactions between agents [1], [25]. These models do not need explicit reward construction and, therefore, can capture the rich, joint dynamics of social interactions. However, these methods are challenging to deploy owing to noisy demonstrations, and covariate shifts [26], [27]. To alleviate this, [9] used the gradients of a Q-value function for Model Predictive Control, and [14] 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 [28], but the convergence properties were not examined. Our method differs from the literature in that it does not require specific reward function design or simulator training. Our work is also more direct and intuitive in its use of sequential models, where calculating gradients or Q-values is not required. Instead, we transform the model's outputs into action distributions for the downstream planning task.

B. Monte Carlo Tree Search for Robot Navigation

Recent works have leveraged MCTS robot navigation applications. To perform adaptive planning, [29] biases MCTS using a CNN trained to learn informative data gathering actions for fast online re-planning. Their work, however, was not extended for use in dynamic *social* settings. In contrast, [30] proposes a local planner for socially aware settings that leverages MCTS and RNNs to simulate future states. Their predictive model, however, does not account for agent-to-agent interactions, making it unable to characterize rich social dynamics and use them to bias the tree search. Similar to our approach, [11] and [31] introduce an RL method that leverages MCTS to train and deploy policies using pre-defined reward functions and simulator training.

While these methods rely on pre-defined reward functions and simulator training, our work extends them to use offline expert-based policies. Finally, although we believe that the core insights of our work apply to other domains, we provide a domain-specific treatment for social navigation in shared airspace to further the research in this domain.

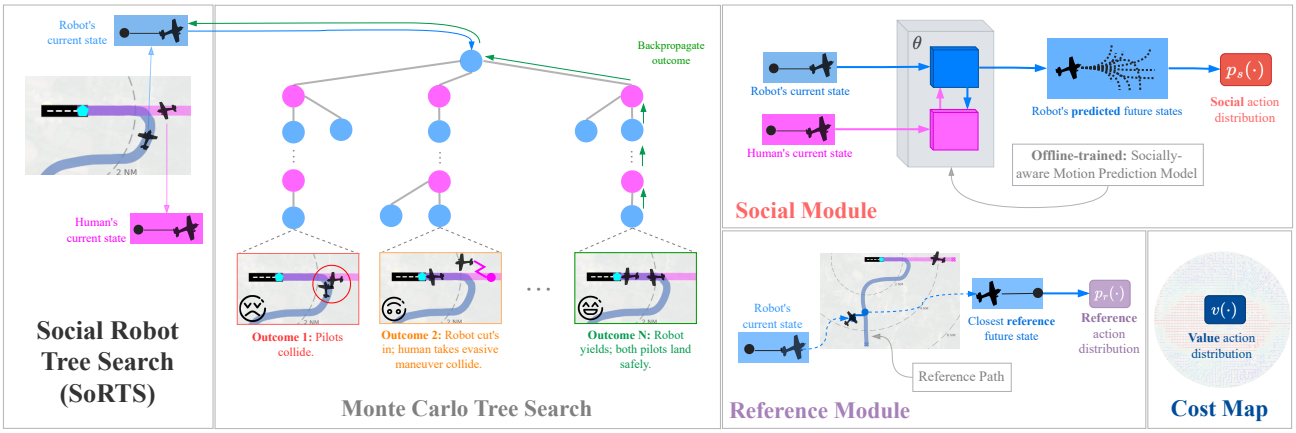

Fig. 2: An overview of SoRTS, a Monte Carlo Tree Search (MCTS)-based planner for social robot navigation which provides long-horizon simulations, collision checking and goal conditioning. Its tree search is biased by three components; a **Social Module**, a **Reference Module** and a **Cost Map**. The social module uses a socially-aware motion prediction model to predict a set of possible future states given the *social dynamics* of the scene. The reference module provides the agent with a global path that embodies navigation guidelines the agent must follow. The cost map encodes a global visitation to encourage the agent to move toward more desirable regions.

III. Problem Formulation

A. Notation Details

We use $\mathbf{s}_t^i \in \mathcal{S}$ to represent the state of an agent i at time-step t, and the agent's start and goal locations as $\mathbf{s}_0^i, \mathbf{s}_g^i$, respectively. Let $\mathbf{s}_{t_m:t_n}^i$ represent a trajectory of states from time-step t_m to time-step t_n. Following this, we refer to the observed trajectory of an agent as $\mathbf{s}_{t-H:t}^i$, where t is the current time-step and H is the length of the agent's observed history. We omit superscripts to represent the joint state for multiple agents, *e.g.*, $\mathbf{s}_t = [\mathbf{s}_t^1, \ldots, \mathbf{s}_t^N]$. We follow a similar notation for the action, $\mathbf{a}_t^i \in \mathcal{A}$, and a joint action $\mathbf{a}_t$. We use $\tau_R^i = [\mathbf{s}_0^i, \ldots, \mathbf{s}_g^i] \in \mathcal{D}$ to represent an agent's reference path. In our work, reference action probabilities are obtained from a dataset, $\mathcal{D}$, comprised of real-world interaction data recorded offline. Note that the mathematical details are abstract to allow flexibility in the problem formulation and domain. For our specific representation, refer to Section V-C.

B. Problem Statement

Let us consider a robot (ego-agent) e in a crowded scene. The robot follows a dynamic model, s.t. $\mathbf{s}_{t+1}^e = f(\mathbf{s}_t^e, \mathbf{a}_t^e)$. We assume that the robot can observe the current states of all other agents in the scene via some observation function, s.t. $\mathbf{s}_t^j = o(\cdot), \forall j \neq e$. The goal for the robot is to find a sequence of control inputs $\pi^e = [\mathbf{a}_0^e, \ldots, \mathbf{a}_g^e]$, s.t. it follows a socially-compliant path toward its goal, $\tau_S^e = \{\mathbf{s}_0^e, \ldots, \mathbf{s}_g^e\}$. In our work, a socially-compliant path must ensure *safety* and *navigation efficiency*. To satisfy the safety requirement with all agents in the scene, the robot must ensure $||\mathbf{s}_t^e - \mathbf{s}_t^j|| < d, \forall j \neq e$, where d is the minimum separation distance to satisfy the safety objective with all agents in the scene. To satisfy the efficiency component, it needs to stay close to the reference trajectory $\min ||\tau_S^e - \tau_R^e||$.

IV. Approach

A. An Overview of SoRTS

SoRTS is an MCTS-based planner whose tree search is guided by the three modules shown in Figure 2: (1) a *Social Module* which handles the short-horizon dynamics in the scene, characterizing social cues and interactions; (2) a *Reference Module* which provides the agent with a reference path representing a standardized navigation guideline; and (3) a *Cost Map* which encodes the value function representing the desirability of each state in the state-space. MCTS further provides collision checking and long-horizon simulations.

The core insight of SoRTS is to use the aforesaid modules to bias MCTS to search through multiple decision modalities, favoring those that prioritize socially compliant and safe behaviors. As a *toy* example, Figure 2 depicts a situation where the intended paths of two aircraft merge onto a single one, akin to a highway merger. To resolve this situation, the search algorithm will run forward simulations by combining actions. Through them, it will not only prune branches that lead to future collisions (*e.g.*, outcome 1 in the figure) but it will also leverage the social module to choose between socially undesirable (*e.g.*, cutting in, outcome 2) and socially desirable (*e.g.*, yielding, outcome 3) modalities, thereby producing socially-compliant and safe behavior.

B. Modules

We provide an overview of the three components used to bias the search algorithm. For specific implementation details on each of these modules, we refer the reader to Section V-C.

1) Social Module: To account for the short-term agent-to-agent interactions in a scene, we leverage an offline-learned socially-aware motion prediction model. For an agent i in the scene, the model produces a joint distribution of future actions and states conditioned on the motion histories of all agents in a scene $\mathbf{s}_{t-H:t}$, demonstration data $\mathcal{D}$, and the agent's goal $\mathbf{s}_g^i$,

$$p_s(\mathbf{s}_t^i, \mathbf{a}_t^i) \sim P_\theta(\mathbf{s}_t^i, \mathbf{a}_t^i \mid \mathbf{s}_{t-H:t}, \mathbf{s}_g^i, \mathcal{D}) \quad (1)$$

where P_θ is the motion prediction model and θ its learned parameters. While the output of the trajectory prediction model is typically in continuous state space, we convert it into a distribution in discrete motion primitive action space.

2) Reference Module: To encourage the search to adhere to navigation norms, this module calculates the *reference* joint action and state distribution $p_r(\cdot)$ conditioned on reference trajectory $\tau_R^i \in \mathcal{D}$.

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

More generally, reference action probabilities can be obtained through other methods, *e.g.*, agent-agnostic global path planners, STL specifications [32], *etc*.

3) Cost Map: To bias the search toward more desirable regions, our algorithm uses a cost map of the environment. The cost map captures the underlying *value* $v(\cdot)$ of the joint state distribution, thus providing a score at a given joint position. The higher the score, the more likely the agents are navigating in favorable regions. These evaluations are used to update MCTS' action value Q during the back-propagation process. In general, $v(\cdot)$ can be learned via self-play [11], or manually computed from a prior, as said in Section V-C.

C. Social Monte Carlo Tree Search

MCTS is a search-based algorithm frequently used to solve problems requiring sequential decision-making. It expands its tree search toward highly rewarding trajectories by searching the state space and building statistical evidence for the most available decision modalities at a given state [33]. MCTS is divided into four stages: selection, expansion, simulation, and back-propagation. During the *selection* process, it uses a tree policy to search the regions of the tree that have been already explored. Most commonly, said policy builds upon the Upper Confidence Bounds applied to Trees (UCT) algorithm [17] which proposes a formula that aims to balance the degree of state exploration and exploitation.

$$U(\mathbf{s}, \mathbf{a}) = Q(\mathbf{s}, \mathbf{a}) + c\frac{\sqrt{N(\mathbf{s})}}{N(\mathbf{s}, \mathbf{a})} \tag{3}$$

where $Q(\mathbf{s}, \mathbf{a})$ is the empirical average of playing an action $\mathbf{a}$ from state $\mathbf{s}$, $N(\mathbf{s})$ is the number of times $\mathbf{s}$ has been visited in previous iterations, $N(\mathbf{s}, \mathbf{a})$ is the number of times $\mathbf{a}$ has been sampled at $\mathbf{s}$, and c is for controlling the exploration and exploitation. Note that we drop the corresponding superscripts and subscripts for ease of notation. In our work, we modify the UCT formula in Equation (3) to include the influences from the components presented in Section IV-B,

$$U(\mathbf{s}, \mathbf{a}) = Q(\mathbf{s}, \mathbf{a}) + c_1 P_S(\mathbf{s}, \mathbf{a}) + c_2 P_R(\mathbf{s}, \mathbf{a}) \tag{4}$$

where, $P_S(\mathbf{s}, \mathbf{a})$, is the normalized visitation component for choosing $\mathbf{a}$ from $\mathbf{s}$ according to the socially-aware prediction network; $P_R(\mathbf{s}, \mathbf{a})$ is the expected value according to the reference path, and c_1 and c_2 are the exploration hyper-parameters. These terms are updated according to Equation (5)-7. These updates are performed iteratively within a time budget, or until a leaf state is found. The algorithm further *expands* the tree by adding new leaf states. Then, at each time-step, a new forward *simulation* tree is iteratively constructed by alternately expanding the agents' future states in a round-robin fashion In practice, for $N > 2$, we only use the robot and the closest agent to the robot for tree expansion. While the tree is explicitly constructed only for two agents, p_s 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.. Therein, branches that lead to a collision state are pruned. Upon finalizing this process, the outcomes from the simulations are *back-propagated* to all nodes along the path from the leaf node and the root. Finally, the ego agent uses the updated statistics to select the action that maximizes the normalized visitation count as in [15].

$$Q(\mathbf{s}, \mathbf{a}) = \frac{N(\mathbf{s}, \mathbf{a})Q(\mathbf{s}, \mathbf{a}) + v(\mathbf{s})}{N(\mathbf{s}, \mathbf{a}) + 1} \tag{5}$$

$$P_R(\mathbf{s}, \mathbf{a}) = \frac{N(\mathbf{s}, \mathbf{a}) \cdot P_R(\mathbf{s}, \mathbf{a}) + p_r(\mathbf{s})}{N(\mathbf{s}, \mathbf{a}) + 1} \tag{6}$$

$$P_S(\mathbf{s}, \mathbf{a}) = \frac{\sqrt{N(\mathbf{s})}}{N(\mathbf{s}, \mathbf{a}) + 1} \cdot p_s(\mathbf{s}, \mathbf{a}) \tag{7}$$

V. Experimental Setup

A. Experiment Design

Our experimental setup involves two or more pilots attempting to land on the same runway at a non-towered airport. The pilots are spawned at the same distance and altitude from the runway. Since there's no central authority to manage the landing, social coordination between aircraft is essential for a safe landing. Moreover, landing at non-towered involves following explicit guidelines established by the FAA [34] as well as social norms that exist implicitly in operations at these airports. Both, our user study and self-play experiments follow the design outlined above. We provide more details in sections VI-A and VI-B, respectively.

B. Ablation

Our ablation serves to contrast the benefit of our long-horizon planning strategy against classical social navigation approaches that leverage short-term strategies [7]. We hypothesize that single-step planning leads to worse outcomes as the planner may choose what, in the short-term, seems like the best action to take which in the long-term may not necessarily lead to the best outcome. Thus, the ablation is designed to select the optimal action for each time step by weighting the efficiency and safety components in Section IV-B,

$$\mathbf{a}^* = \arg\max_{\mathbf{a} \in \mathcal{A}} \left[\lambda \cdot p_r(\mathbf{s}_t, \mathbf{a}) + (1 - \lambda) \cdot p_s(\mathbf{s}_t, \mathbf{a})\right]$$

This decision-making strategy loosely follows from [7], where an agent chooses its next action as a compromise between two objectives. Here $\lambda \in \mathbb{R} : [0, 1]$ is a hyperparameter controlling the importance of the action probabilities given by each objective.

C. Algorithm Implementation Details

We leverage TrajAir [20], a dataset consisting of 111 days of aircraft trajectory data collected in non-towered terminal airspace at the Pittsburgh-Butler Regional Airport, in which standard traffic patterns are followed [35]. Our social module was trained offline on TrajAir following the implementation details in [21]. We also used TrajAir to build a library of FAA-abiding global paths used by the reference module. Finally, similar to [32], we build our cost map based on the flight frequency in TrajAir; where we first split the data by discretizing based on the wind direction, and then build a 3D histogram discretizing by aircraft position.

Following [32], SoRTS' state space represents the continuous 3D locations of the aircraft. Its action space consists of 252 motion primitives discretizing the continuous action space in airspeed, vertical speed, and turn angle applied for a length of 20 seconds, following the dynamic model in [32]. Empirically, we set the number of tree expansions and maximum steps to 50 and 100, respectively. Similarly, the exploration parameters in the UCT equation are set to $c_1 = 2$ and $c_2 = 5$, and we chose $\lambda = 0.3$ for the ablation planner. For further details regarding the state representation, the aircraft dynamic model, and the cost map, we refer the reader to [20], [21], [32].

D. Simulator

To evaluate our algorithm in our user study, we introduce X-PlaneROS, a simulation environment that aims at enabling research for human-AI interaction in full-scale aerial autonomy applications. X-PlaneROS is a system that combines two main modules; X-Plane-11 and ROS-Plane autopilot [36]. X-Plane-11 [37] is a high-fidelity simulator that provides realistic aircraft models and visuals, as well as an open API that supports multi-agent gameplay. ROS-Plane is a widely accepted tool for research and teaching with reliable and autonomous flight control loops. Together, these two modules enable the use of high- and low-level control commands for GA aircraft in realistic world scenarios. X-PlaneROS interfaces with X-Plane-11 using NASA's X-PlaneConnect [38]. The state information from X-Plane is published over ROS topics. The ROS-Plane integration then uses this information to generate actuator commands based on higher-level input to the autopilot system. These actuator commands are then sent back to X-Plane through X-PlaneConnect.

VI. Experiments

A. User Study

We recruited 26 FAA-certified pilots[1] (14 private, 8 commercial, 3 student pilots, and 1 airline transport pilot), who on average have 986 flight hours. Each pilot had to complete a set of landing tasks on a specified runway at an airport using X-PlaneROS and a flight deck like those shown in Figure 1. In each task, a second pilot was simultaneously attempting to land on the same runway. Here, the second pilot was either a human, SoRTS or the ablation discussed in Section V-B. In the subsequent paragraphs, we use *second pilot* and *algorithm* interchangeably.

We followed a *within-subject* design where each user tested against each algorithm. We let users get familiarized with the simulator and controls before beginning the tasks. In each instance, we spawn the pilots within a 10 km radius. Their incoming direction was either north (N), south (S) or west (W), defining six possible scenarios for the pilot pair: $\{(N,S),(S,N),(N,W),(W,N),(S,W),(W,S)\}$. The algorithm order and scenario were randomly chosen. The scenario remained fixed throughout the three tests. We note that pilots do not see the reference trajectory, but they have access to a global map in the simulator and are instructed to follow the standard flight patterns.

After each task, users completed a 5-point Likert questionnaire to evaluate the second pilot's performance along our two notions of interest: efficiency and safety. To assess efficiency, we asked users to rate the other pilot's (1) ability to adhere to FAA guidelines, (2) overall flying skill, and (3) flight smoothness. For safety, users rated the second pilot along the following components: (1) collision risk, (2) comfort, (3) abruptness, (4) cooperativeness, and (5) predictability. The users were also asked to rate the *trustworthiness* of each algorithm to gain insights into which components of efficiency and safety were deemed more relevant. Finally, we also collected the trajectory data from the experiments to further analyze efficiency and safety using the metrics in Section VI-C.

B. Self-Play

Recruiting certified pilots to conduct our evaluations is challenging. As such, our user study focused on evaluating a limited number of scenarios. To assess SoRTS' performance on a broader set of scenarios, we complement our study with self-play simulation experiments. Our self-play simulation experiments follow a similar design to that of the proposed user study; *i.e.*, agents are spawned 10km away from the airport, and their task is to land at the specified runway. In contrast with the user study, however, the initial location for each agent can be any location around the 10km radius to encourage higher scenario diversity. We consider multi-agent scenarios with 2 to 5 agents. We randomly generated 100 episodes for each setting, where, an agent is deemed unsuccessful if it breaches a minimum separation distance with another agent (Section VI-C), gets off-track, or reaches a maximum number of allowed steps.

C. Metrics

Following [1], [39], we also assess the performance of our user study and self-play experiments using the following objective metrics: (1) *Reference Error (RE):* the distance between a reference trajectory and the agent's executed path. (2) *Loss of Separation (LS):* the duration that two agents break a minimum distance from each other. Although LS is specific to the domain of aviation [40], it is akin to *personal space* metrics commonly used within pedestrian settings to assess the

[1]We received approval from Carnegie Mellon University's Institutional Review Board for this study (protocol no. *STUDY2022_00000195*, approved June 15th, 2022). The committee authorized all aspects, including data handling, and adherence to ethical guidelines. All participants gave written informed consent. We ensured participant anonymity.

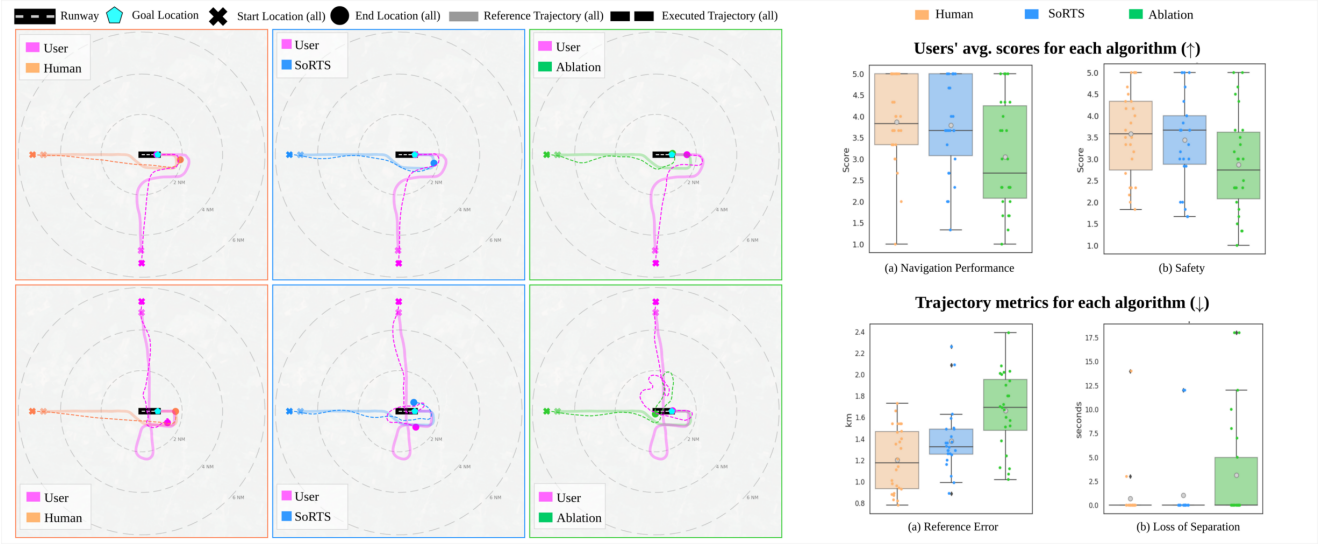

Fig. 3: User study results. **Left:** Each row shows the resulting trajectories of a User interacting with our Human pilot, SoRTS, and the Ablation; reference paths are shown as solid lines, executed ones as dashed lines. The top row shows a user that successfully followed the expected path. We can observe that the ablation did not follow the reference path as smoothly as SoRTS, and also cut short when approaching the goal. The bottom row shows a user that did not follow the expected path. Here, SoRTS still managed to navigate properly. In contrast, the ablation behaved erratically, unsafely crossing over the runway twice. **Right:** Box-plots showing per-algorithm results. The top ones show the avg. efficiency (a) and safety (b) scores given by the users. The bottom ones show the avg. RE (a) and LS (b) metrics obtained from the trajectory data.

level of discomfort incurred by the robot to the surrounding agents [1]. We use RE and LS as proxies for efficiency and safety.

VII. Results

In Section VII-A, we first analyze how human pilots perceived each algorithm w.r.t. *trustworthiness* based on the performance components in Section VI-A. We then use those results to compare the algorithmic pairs in Section VII-B.

A. On performance and trust

We first compare the factors from Section VI-A with the user's perceived trust for each of the *challenger* pilots. To do so, we perform a Pearson's correlation with repeated measures analysis and summarize the results in Table I. The table shows that the users' perceived trust strongly correlates to all of the factors within the axis of navigation efficiency, hinting that users notice and prioritize aspects relating to flight smoothness and the ability to follow navigation guidelines. For the safety aspect, we observe that the users' assessments for trust were more strongly correlated to how comfortable the users felt during the interaction and how erratic or unpredictable was the behavior of the pilot.

B. On the performance of each algorithm

We now provide a comparison between the *algorithms* in the user study and self-play experiments.

1) User study: For each algorithm, we first obtain average scores for the efficiency and safety leveraging the trustworthiness results in Table I. For the efficiency component we compute the mean score between *following FAA guidelines* and *flight smoothness*, which correspond to the highest correlating factors for this axis. For safety, we use *predictability* and *comfort*. The scores are shown in Figure 3 (top-right).

TABLE I: Pearson correlations (R, p-value=0.05) for each component in Section VI-A *vs.* perceived trustworthiness.

Axis	Factor	Trustworthiness	
		R	p-value
Nav. Efficiency	Flight Smoothness	0.81	4.80e-19
	Follow FAA Guidelines	0.76	2.99e-15
	Overall Flying Skill	0.71	4.56e-13
Safety	Comfortable	0.92	3.55e-31
	Predictable Behavior	0.77	5.19e-16
	Cooperative	0.65	1.61e-10
	Collision Risk	-0.56	1.10e-07
	Abrupt	-0.56	1.13e-07

We then use ANOVA with repeated measures to compute the pairwise statistical differences between the algorithms shown in Table II. Our analysis suggests that there is no statistical evidence that the scores for Human and SoRTS were different. This hints that the users rated their performances similarly. We also find that the ablation was generally rated lower on both of these axes while also displaying higher variance, compared to the other algorithms. We compute the RE and LS metrics on the resulting trajectories to examine how they tie to the users' assessments for efficiency and safety. Their corresponding average values and statistical analysis are also

shown in Figure 3 (bottom-right) and Table II. For the RE metric, we find marked difference between the algorithms; wherein SoRTS yields higher error compared to the human pilot, but lower than the ablation. SoRTS also exhibits less variance than the other two. For the safety metric we observe that, in general, neither the human pilot nor SoRTS breach the safe distance here set to 0.3 km. In contrast, we see that the ablation does it more frequently, creating more situations for potential collisions.

We also show trajectory visualizations of our experiments in Figure 3. Each row represents one user *vs.* the three *algorithms*. We show a reference path along with the executed path. The top row shows an example of a successful user. Here, we observed that the ablation unexpectedly cuts short while approaching the runway instead of following the reference path. In contrast, SoRTS yields to the user and smoothly follows the expected pattern. The bottom row shows a user that did not follow the standard navigation pattern. Despite this, SoRTS manages to successfully complete the task, while the ablation behaves erratically, not following the pattern and unsafely traversing the runway twice.

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

Algorithmic Pair	Nav. Efficiency		Safety		RE		LS	
	t-val	p-val	t-val	p-val	t-val	p-val	t-val	p-val
Ablation-Human	3.121	0.009	3.062	0.016	5.782	0.000	1.321	0.199
Ablation-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

We conclude that SoRTS performs comparable to a competent human pilot and significantly better than the ablation, both, as perceived by the users and according to our metrics.

2) Self-play: We summarize the results of the self-play experiments in Table III. The table shows the percentage of *successful* and *unsuccessful* agents and their average RE. Here, task success was higher for SoRTS than for the ablation for all experiments. Although we observe a decrease in task success as the number of agents increases for both algorithms, we see a more significant drop in performance for the ablation ($\sim$21%) than SoRTS ($\sim$11%) as the number of agents goes from 2 to 5. Finally, for all failure conditions, the failure percentage was significantly lower for SoRTS. Finally, Figure 4 shows qualitative results of the ablation (top) and SoRTS (bottom) experiments. We show situations in which the ablation agents, unable to resolve social conflicts, fail to complete their task due to a loss of separation. Then, under the same initial conditions, SoRTS agents are able to foresee and avoid the potential conflict and successfully complete their task.

VIII. Conclusion

We present SoRTS, an MCTS-based planner that aims to augment offline-trained socially aware motion prediction models for their deployment in online social navigation settings. Our work uses the domain of general aviation as a use case. In doing that, 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 up to 17.9% on the more crowded and complex scenarios.

We identify two main avenues for future work. Firstly, our work assumes *perfect* intent and state estimation. The scope of the paper is restricted to evaluating and robustifying the in-domain performance of prediction models for navigation tasks, thus focusing on aleatoric uncertainty. This challenge is in contrast to robustifying against out-of-distribution scenarios which is a different line of research as detailed in the future work section. Accordingly, robustifying prediction models with uncertainty and adversarial awareness is a promising direction [41], [42]. Finally, we believe the core insights of our approach and the modularity of its design make it suitable for its application to other domains.

TABLE III: Task performance summary for self-play agents.

# Agents	Algorithm	Success (↑%)	Failure (↓%)			RE
			LS	Timeout	Offtrack	
2	Ablation	88.0	12.0	0.00	0.00	2.05
	SoRTS	92.5	4.0	1.00	2.50	2.15
3	Ablation	74.7	18.0	5.30	2.00	2.02
	SoRTS	86.3	12.7	0.30	0.70	2.03
4	Ablation	74.0	18.5	5.30	3.20	2.09
	SoRTS	86.0	13.0	0.75	0.25	2.07
5	Ablation	69.4	21.2	6.20	3.20	2.04
	SoRTS	81.8	16.4	1.60	0.20	2.06

LS: Loss of separation at 0.2km, **RE**: reference error in km for successful agents.

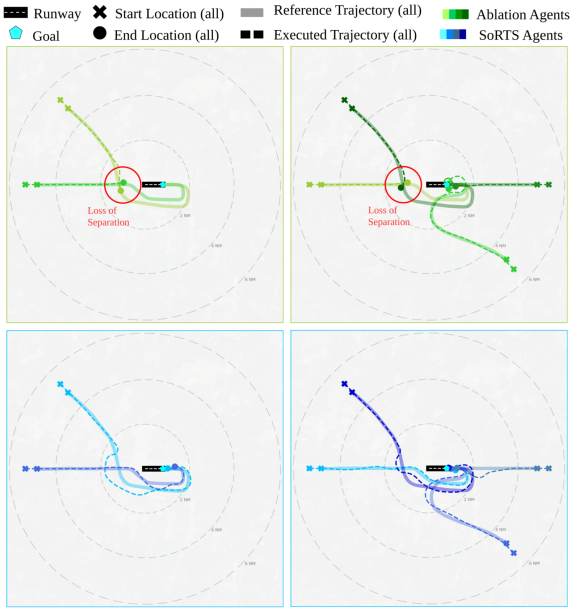

Fig. 4: Self-play results. The top row shows examples where the Ablation agents fail to resolve social conflicts and end in a loss of separation situation. With the same initial conditions, the bottom row shows that SoRTS agents are able to adjust their paths to properly resolve these situations.

ACKNOWLEDGMENT

This work was supported by the Army Futures Command Artificial Intelligence Integration Center, the Korean Ministry of Trade, Industry and Energy, the Korea Institute of Advancement of Technology (P0026022 and P0019782), and the Brazilian Air Force. We thank Condor Aero Club and ABC Flying Club pilots for participating 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," *ACM Transactions on Human-Robot Interaction*, vol. 12, no. 3, 2023.

[2] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "From proxemics theory to socially-aware navigation: A survey," *International Journal of Social Robotics*, vol. 7, no. 2, pp. 137–153, 2015.

[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] 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, 2020.

[5] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[6] 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.

[7] 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.

[8] W. Wang, L. Wang, C. Zhang, C. Liu, L. Sun, *et al.*, "Social interactions for autonomous driving: A review and perspectives," *Foundations and Trends® in Robotics*, vol. 10, no. 3-4, pp. 198–376, 2022.

[9] 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, 2018.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] H. Hu and J. F. Fisac, "Active uncertainty reduction for human-robot interaction: An implicit dual control approach," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2022, pp. 385–401.

[15] 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.

[16] 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," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9695–9728.

[17] 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.

[18] 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*, 2019.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] J. Rückin, L. Jin, and M. Popović, "Adaptive informative path planning using deep reinforcement learning for uav-based active sensing," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4473–4479.

[30] S. Eiffert, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, "Path planning in dynamic environments using generative rnns and monte carlo tree search," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 263–10 269.

[31] J. Oh, J. Heo, J. Lee, G. Lee, M. Kang, J. Park, and S. Oh, "Scan: Socially-aware navigation using monte carlo tree search," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7576–7582.

[32] 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," 2022.

[33] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk, "Monte carlo tree search: A review of recent modifications and applications," *Artificial Intelligence Review*, vol. 56, no. 3, pp. 2497–2562, 2023.

[34] F. A. Administration, "Aeronautical information manual," 2023.

[35] ——, *Airplane flying handbook (FAA-H-8083-3A)*. Skyhorse Publishing Inc., 2011.

[36] 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.

[37] "X-plane 11 flight simulator," https://www.x-plane.com.

[38] NASA, "X-plane connect," github.com/nasa/XPlaneConnect, 2022.

[39] Y. Gao and C.-M. Huang, "Evaluation of socially-aware robot navigation," *Frontiers in Robotics and AI*, p. 420, 2021.

[40] 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.

[41] 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.

[42] Q. Zhang, S. Hu, J. Sun, Q. A. Chen, and Z. M. Mao, "On adversarial robustness of trajectory prediction for autonomous vehicles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 159–15 168.