2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636282

Interpretable Run-Time Prediction and Planning in Co-Robotic Environments

Rahul Peddi and Nicola Bezzo

***Abstract*— Mobile robots are traditionally developed to be reactive and avoid collisions with surrounding humans, often moving in unnatural ways without following social protocols, forcing people to behave very differently from human-human interaction rules. Humans, on the other hand, are seamlessly able to understand why they may *interfere* with surrounding humans and change their behavior based on their reasoning, resulting in smooth, intuitive avoiding behaviors. In this paper, we propose an approach for a mobile robot to avoid interfering with the desired paths of surrounding humans. We leverage a library of previously observed trajectories to design a decision-tree based interpretable monitor that: i) predicts whether the robot is interfering with surrounding humans, ii) explains what behaviors are causing either prediction, and iii) plans corrective behaviors if interference is predicted. We also propose a validation scheme to improve the predictive model at run-time. The proposed approach is validated with simulations and experiments involving an unmanned ground vehicle (UGV) performing go-to-goal operations in the presence of humans, demonstrating non-interfering behaviors and run-time learning.**

I. Introduction

Autonomous mobile robots are rapidly finding their way into our society in an increasing number of applications, including delivering packages in urban environments or deployed in warehouses assisting human workers. As these robots share space with humans, we must consider how they affect the human experience; that is, how surrounding humans behave in the presence of a moving robot. In many cases, these robots treat surrounding humans as stationary obstacles, often moving in unnatural ways, leaving the human solely responsible for learning to work around the robots.

More recently, advanced machine learning techniques like Long Short Term Memory (LSTM) networks and Deep Reinforcement Learning (DRL) have been used to generate more natural robot behaviors around humans [1]–[3]. While good robot behaviors can be produced, the approaches often contain black-box models [4], and are unable to provide explanations or reasoning for decisions. In addition, these approaches typically are not adaptable at run-time and require dedicated training phases. Humans, on the other hand, accommodate others in very intuitive and easily interpretable ways. We are generally aware of our actions, and we can assess and explain if attributes of our behavior (e.g., how fast we are moving) will lead to some type of *interference* with other people, causing them to change their path [5]. We not only are able to explain whether we are interfering, but also intuitively use this explanation to change our motion–without exactly predicting where others will go. If robots

Rahul Peddi and Nicola Bezzo are with the Department of Systems and Information Engineering and the Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904, USA. Email: {rp3cy, nb6be}@virginia.edu

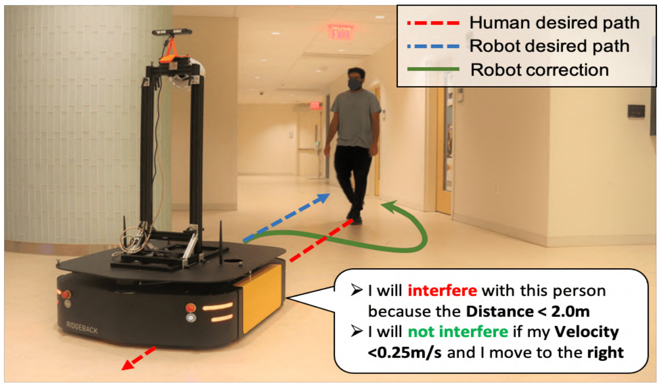

Human desired path
Robot desired path
Robot correction
I will interfere with this person because the Distance < 2.0m
I will not interfere if my Velocity <0.25m/s and I move to the right

Fig. 1. In our proposed approach, a robot predicts, explains and finds a corrective action to avoid interfering with an oncoming human.

could reason about their behavior and plan corrective actions in a similar way, their motion would be easy to understand and interpret for surrounding humans. In Fig. 1, we show a motivating example for this work in which a robot is able to predict, explain, and find a correction to avoid interfering with a person's intended path.

To achieve this behavior, we propose a novel method that leverages decision tree theory [6], [7] to predict, explain, and plan corrective actions at run-time in situations in which a robot will interfere with the motion of surrounding humans. Different from other learning-based approaches, besides the explainability aspect, another contribution of our work is that previously unobserved and misclassified data are considered at run-time through validation criteria to improve and refine predictions and corrective actions in future operations.

The rest of this paper is organized as follows: in Section II, we discuss related literature. In Section III, we formally define the problem and in Section IV, we describe our decision-tree based explainable monitoring and planning. In Sections V and VI, we present simulation and experimental results and finally we draw conclusions and discuss future work in Section VII.

II. Related Work

With recent advancements in mobile robots, there has been growing interest in enabling robots to navigate human environments in an intuitive and socially acceptable manner. Recent works in the field of machine learning have made substantial progress in enabling such behaviors, including [1], [8], where the authors use recurrent neural networks (RNNs) or long short term memory networks (LSTMs) to predict motion of humans, which is used to generate safe robot motion. While these methods are effective for predicting human trajectories, they contain complex network architecture, and it is difficult to understand the mapping from input

Authorized licensed use limited to: University of Virginia Libraries. Downloaded on August 24,2022 at 14:47:34 UTC from IEEE Xplore. Restrictions apply.

states to prediction. Authors in [9] use deep reinforcement learning (DRL) to attain socially acceptable behaviors, and in [3], the authors use DNNs to achieve similar results. While these approaches demonstrate good robot behaviors, none can provide explanations for why the resulting behavior was appropriate. In this paper, we complement the aforementioned works by interpreting and explaining predictions to generate non-interfering robot behaviors.

In our more recent work [10], we exploited Hidden Markov Model theory to obtain probabilistic predictions of temporal reachable states and developed a virtual physics-based planner, similar to the "social-force" developed in [11] to operate the robot. In this paper, we bypass this requirement of explicit predictions and rely only on a more computationally efficient binary classification to achieve socially acceptable motion in co-robotic environments.

Towards explaining machine learning models, authors in [12], [13] provide techniques to explain the predictions of a classifier. Instead of using global explanations, they find local reasoning as to why a data point was assigned to a certain class in simple classification learners, such as decision trees (DT). We take inspiration from these works and our previous work [7], where we have shown that DTs can be used to control a robot under bounded disturbance. We extend this work to handle a more dynamic environment considering multiple actors.

III. Problem Formulation

Consider a mobile robot tasked to navigate an environment while avoiding other actors, in particular, humans. Without prior knowledge about the intended goal of the surrounding humans, this robot would not be able to predict their path. We note however that humans tend to move in certain way, typically in the direction of the desired goal. Let us define the path followed by a human as $\boldsymbol{q}_i^*$, with $i = 1, \ldots, N_h$ where N_h is the number of humans in sensing range with the robot. With such premises we would like to design a framework for a robot to directly predict *interference* with all the humans in its sensing range and plan its motion accordingly to minimize the deviation of human paths due to its presence along the way. Formally, the problem can be cast as:

Problem 1: **Non-Interfering Motion Planning and Control.** Design a policy to predict and explain future interfering interactions between a robot and surrounding humans and to plan corrective actions, $\boldsymbol{u}$, that do not cause human paths to deviate more than a distance δ from the intended trajectory:

$$||\boldsymbol{q}_h(t) - \boldsymbol{q}_h^*(t)|| \leq \delta, \forall h = 1, \ldots, N_h(t), t \geq 0 \qquad (1)$$

where $\boldsymbol{q}_h(t)$ and $\boldsymbol{q}_h^*(t)$ are the observed path and intended path of the i^{th} actor at time t respectively.

A correlated problem that we propose to investigate in this work is to improve robot behavior over time in response to previous experience. To this end, we create a strategy to validate and update predictions and planning at run-time when undesirable behaviors are observed or when run-time observations are unmodeled in the training data.

IV. Approach

Our proposed interpretable monitoring framework follows the architecture in Fig. 2.

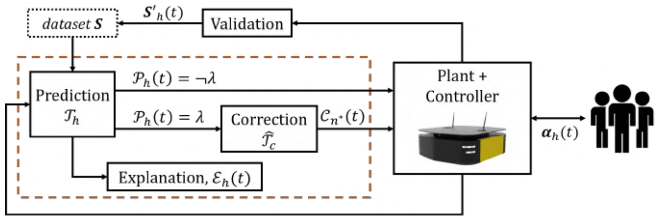

dataset S
$S'_h(t)$
Validation
Prediction $\mathcal{T}_h$
$\mathcal{P}_h(t) = \neg\lambda$
$\mathcal{P}_h(t) = \lambda$
Correction $\hat{\mathcal{T}}_c$
$\mathcal{C}_{n^*}(t)$
Explanation, $\mathcal{E}_h(t)$
Plant + Controller
$\boldsymbol{\alpha}_h(t)$

Fig. 2. Block diagram of the presented approach.

At the core of our framework we leverage decision tree (DT) theory to predict interferences and correct robot behaviors. With observations of surrounding humans $\boldsymbol{\alpha}_h(t)$, a local DT, $\mathcal{T}_h$ is constructed with a dataset of human-robot trajectories to compute a prediction $\mathcal{P}_h(t)$ and explanation $\mathcal{E}_h(t)$ as to whether the robot will *interfere* (λ) or *not interfere* ($\neg\lambda$) with paths of surrounding humans. Then, when interference is predicted, a cascaded (secondary) tree $\hat{\mathcal{T}}_c$ is used to generate corrective behaviors that reduces robot interference with human paths. Finally, a validation scheme is proposed to update the dataset online, improving future DT operations. In the next sections, we describe in detail each component of our framework.

A. Decision Tree Formulation and Training

Decision trees (DTs) are a form of supervised learning that consist of white-box models which make predictions easy to interpret [6]. In this work, DTs are constructed as binary classification models that are made up of a network of nodes; the outermost nodes, known as leaves, correspond to labels given in the training (decisions). Internal nodes define the split criteria for leaves based on the input variables (attributes). In this work, we grow DTs using the Gini Index as the splitting criterion (see [7] for more details).

For training, we generate a dataset of trajectories in both simulation and in real experiments that consist of human motion from multiple initial to final positions with varying velocity in the presence of a moving robot. In the training, the robot does not react to the humans, so that the prediction model can learn when an interference occurs. In simulation, humans are controlled by a virtual physics-based method [10], which triggers a reaction (interference) if a distance threshold, δ_{th}, is violated. By training in this way, the desired effect is that the robot plans actions that keep a minimum distance of δ_{th} from all surrounding humans.

Attributes should be meaningful to the application, and typically more attributes improve the precision of the prediction. However, too many attributes can lead to redundancy and poor classification [6]. For the human-robot interaction case in this work, attributes are derived from the joint state between the robot and surrounding human, based on explicit sensor data available and implicit data we can compute (e.g., velocity from range sensor readings over time). Specifically, we define the attributes as

$$\boldsymbol{\alpha} = [\ d_x \quad d_y \quad \theta \quad d \quad d' \quad v_h \quad v_r \quad \ell\ ] \qquad (2)$$

where d_x, d_y, and θ, are relative x-y positions and heading, respectively, d and d' are the Euclidean distance and distance derivative (i.e., the rate of change of the Euclidean distance) between human and robot, and v_h and v_r are the human and

Authorized licensed use limited to: University of Virginia Libraries. Downloaded on August 24,2022 at 14:47:34 UTC from IEEE Xplore. Restrictions apply.

robot velocities, respectively. The robot's operating "lane," ℓ is a discretization of the robot's y-position, assuming that the robot is typically moving forward, i.e., along the x-direction.

To validate the attributes, which were chosen via experimental sensitivity analysis, we analyzed the average predictor importance [6] of the attributes over 100 random local trees taken from subsets of the training data. A non-zero importance indicates that the attribute is valuable to decision tree predictions, and it is clear in Fig. 3 that while some attributes may be more important than others, all attributes have an effect on the prediction.

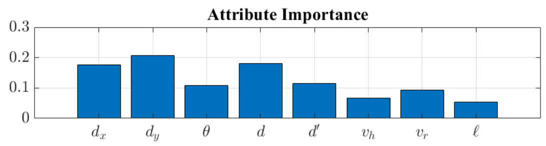


Fig. 3. Attribute importance as a proportion of total attribute importance.

Through the training, we obtain a global dataset $\boldsymbol{\mathcal{S}} = \langle \boldsymbol{\alpha_s}, \boldsymbol{\lambda_s} \rangle$ that includes both the attributes and corresponding classes of all training instances.

B. Prediction and Explanation

Let us now consider first the case of one human approaching the robot. To predict interference, we construct local DTs at run-time with the observed attributes related to a surrounding human, $\boldsymbol{\alpha}_h(t)$, since a global DT for the entire training set can often return inaccurate and imprecise predictions and explanations due to the presence of irrelevant data. A local tree, $\mathcal{T}_h$, is trained by collecting a subset of points $\boldsymbol{\mathcal{S}_h(t)} \subset \boldsymbol{\mathcal{S}}$ from the global dataset that are within a neighborhood Δ of the attributes of $\boldsymbol{\alpha}_h(t)$:

$$\boldsymbol{\mathcal{S}_h(t)} \subset \boldsymbol{\mathcal{S}} \mid ||\boldsymbol{\alpha}_h(t) - \boldsymbol{\alpha_s}|| \leq \Delta \;\; \forall \boldsymbol{\alpha_s} \in \boldsymbol{\mathcal{S}} \qquad (3)$$

The distance Δ is a measure of how close the local training data should be to the observed data. The exact value of Δ is selected based on the quality of the available training dataset. With a very rich dataset, a small Δ may result in very accurate predictions and explanations. For a sparse dataset, Δ should be large enough to ensure the decision tree has enough context to generate accurate predictions and explanations. After constructing the local tree, the prediction $\mathcal{P}_h(t) \in [\lambda, \neg\lambda]$ is obtained by evaluating the run-time observation $\boldsymbol{\alpha}_h(t)$ in the tree: $\mathcal{P}_h(t) = \mathcal{T}_h(\boldsymbol{\alpha}_h(t))$.

Given a prediction, we compute an explanation $\mathcal{E}_h(t)$ by traversing the path through $\mathcal{T}_h$. A prediction directly corresponds to a leaf, $\mathcal{V}_p$, within the tree. If $\mathcal{V}_0$ is the root of $\mathcal{T}_h$, an explanation is computed by traversing a path, Γ from $\mathcal{V}_0$ to $\mathcal{V}_p$, taking into account the split criterion, c, for the N_i internal nodes along the path. The conjunction of split criterion along Γ is the explanation of the prediction:

$$\mathcal{E}_h(t) = \bigwedge_{k=1}^{N_i} c_k \quad \text{with} \quad \Gamma \mid \mathcal{P}_h(t) \qquad (4)$$

Traversing all other paths, $\Gamma_j \in \boldsymbol{q}$ with $j = 1, \ldots, N_{\boldsymbol{q}}$ that lead to the opposite decision, $\neg\mathcal{P}_h(t)$, in a similar way, provides a set of counterfactual rules, $\boldsymbol{\mathcal{C}}_h(t)$, to the previously obtained prediction:

$$\boldsymbol{\mathcal{C}}_h(t) = \bigvee_{j=1}^{N_{\boldsymbol{q}}} \bigwedge_{k=1}^{N_j} c_k \quad \text{with} \quad \Gamma_j \mid \neg\mathcal{P}_h(t) \qquad (5)$$

where each path Γ_j contains N_j nodes to the leaf. These counterfactuals denote which attributes, if changed, would reverse the decision.

Shown in Fig. 4 is an example of a local DT used for prediction and explanation based on the following attributes,

$$\boldsymbol{\alpha}_h(t) = \begin{bmatrix} d_x & d_y & \theta & d & d' & v_h & v_r & \ell \\ 1.43 & -4.71 & -81 & 4.93 & -0.43 & 1.0 & 0.6 & 0 \end{bmatrix}$$

The output of the DT is $\mathcal{P}_h(t) = \lambda$, shown by the dark red path and leaf node. Through (4) we compute an explanation:

$\mathcal{E}_h(t) =$ ***Interfering*** because: $\{d' < 0.24, d_x < 1.76\}$

Through (5), we compute the following counterfactuals:

$\boldsymbol{\mathcal{C}}_h(t) =$ ***Not Interfering*** when:

$\{d' > 0.24, d_x < 1.15\} \vee$
$\{\theta > -56, d' > 0.24, d_x < 1.15\} \vee$
$\{-0.45 \leq d' < 0.24, 1.76 \leq d_x < 1.83\} \vee$
$\{-0.58 \leq d' < 0.24, d_x > 1.83, d < 4.65\}$

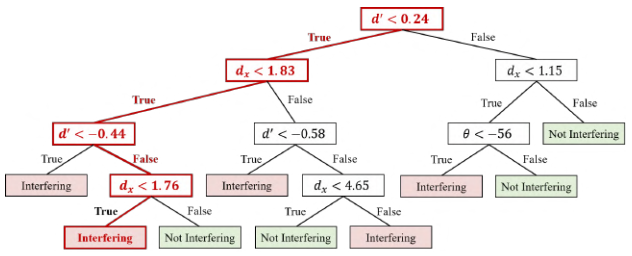


Fig. 4. Example prediction DT. The internal nodes (white squares) of the tree are binary tests on one of the attributes and the leaf nodes (colored squares) are the class decisions. The bold path shows the current decision.

The point at which this prediction is made is highlighted in Fig. 5, taken from our MATLAB simulations.

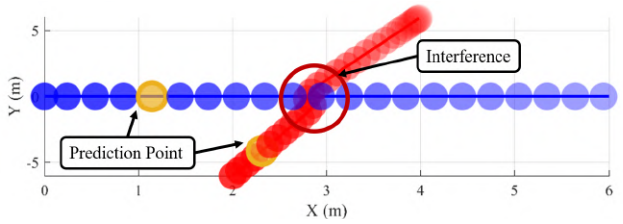


Fig. 5. Human (red) and robot (blue) trajectories, showing the point (in yellow) at which a prediction is made. The markers fade as time increases and the actors reach their goals.

As seen in the example, DTs used for prediction provide a logical and interpretable explanation, but the counterfactuals cannot be controlled by the robot's actions alone, as they pertain to human-dependent attributes.

C. Corrective Counterfactual Analysis

In case of interference, the counterfactuals provide a set of configurations in which the robot would not have interfered with the path of the human. However, it is not practical to

Authorized licensed use limited to: University of Virginia Libraries. Downloaded on August 24,2022 at 14:47:34 UTC from IEEE Xplore. Restrictions apply.

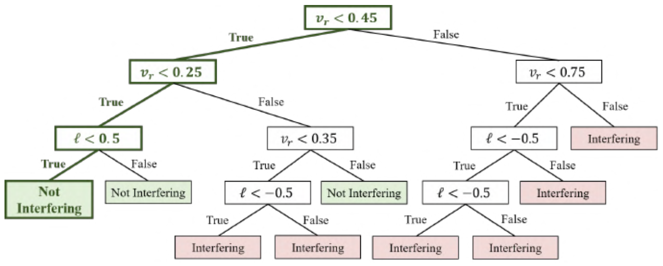


Fig. 6. Example correction decision tree. Nodes and split criteria only pertain to $[v_r,\ \ell]$. The bold path shows the optimal counterfactual $\mathcal{C}_{n^*}(t)$.

manipulate attributes like distance derivative d' or relative heading θ, since they depend on human motion.

The only controllable attributes in our case are the velocity and lane to track by the robot, $\boldsymbol{\alpha}_r = [v_r,\ \ell]$. To generate actionable counterfactuals, we build a secondary tree, $\mathcal{T}_c$ in which we first fix the human dependent attributes $\boldsymbol{\alpha}_c = \boldsymbol{\alpha}_h \setminus \boldsymbol{\alpha}_r$ and search in the training set for similar attributes as done in (3) creating a new set $\boldsymbol{\mathcal{S}}_c \subset \boldsymbol{\mathcal{S}}$ (note that $\boldsymbol{\mathcal{S}}_h \subset \boldsymbol{\mathcal{S}}_c$). In this way, the new DT remains local in the human-related attributes but includes different lanes and velocity pairs, enabling the system to find suitable corrections. The new DT output in this way will be decisions and counterfactuals that only include v_r and ℓ.

In Fig. 6, we show the correction tree associated with the example in Section IV-B and Fig. 4 in which the robot was initially running with $v_r = 0.6$ and $\ell = 0$. After running the procedure in this section, with (5), we obtain the following set of actionable counterfactuals:

$\boldsymbol{\mathcal{C}}_h(t) =$ ***Not Interfering*** when:

$\boldsymbol{\{v_r < 0.25, \ell < 0.5\}} \vee$
$\{v_r < 0.25, \ell \geq 0.5\} \vee$
$\{0.35 \leq v_r < 0.45\}$

To decide which counterfactual to select, we consider two measures that describe the quality of the nodes of the tree, *node error*, e_n, and *node risk*, r_n, with $n = 1, \dots, N_c(t)$, where $N_c(t)$ is the number of counterfactuals. Node error is the fraction of differently classified training points at a specific leaf. For a leaf that predicts $\neg\lambda$, the node error is:

$$e_n = 1 - p(\lambda) \tag{6}$$

Node risk is a weighted measure of impurity (Gini Index in our work):

$$r_n = 1 - \sum_{i=1}^{2} {\rho_i}^2 \tag{7}$$

where ρ_i is the fraction of elements labeled with class $i = [\lambda, \neg\lambda]$. A lower node risk indicates that there will be less of chance of an incorrect decision. In our approach, we combine both measures by taking the product $e_n r_n$, since our goal is to identify the best node to use. The use of the product is viable here because both node error and node risk represent different probabilities that rely on information about the training points for the local tree, enabling the use of the general multiplication rule of probabilities for identifying the best node [14]. Then, the optimal counterfactual is computed as follows:

$$n^* = \arg\min_n (e_n r_n) \tag{8}$$

The selected counterfactual rule, $\mathcal{C}_{n^*}(t)$ consists of an optimal velocity and lane $\boldsymbol{\alpha^*} = [v_r^*, \ell^*]$. In the example shown in Fig 6, $\boldsymbol{\alpha^*} = \boldsymbol{\{v_r < 0.25, \ell < 0.5\}}$.

D. Corrective Planning and Control

Once a counterfactual rule is selected, the robot moves to implement the correction, which is represented as a "ghost" moving target, and the robot switches into a pure-pursuit based mode of operation [15] until it reaches the ghost vehicle. This is necessary because the corrective action represents what the robot should have been doing at the instance t at which the correction was found, meaning that the robot would only satisfy non-interfering conditions if $v_r(t) = v_r^*$ and $\ell(t) = \ell^*$.

During the pure-pursuit corrective operation, the robot uses the ghost vehicle's state to make predictions until the tracking error between robot and ghost $e(t) = \boldsymbol{p}_g(t) - \boldsymbol{p}_r(t) \approx 0$, after which it reverts to performing predictions based on the actual state of the robot. This is needed because as the robot performs corrective behaviors, we observe transition states that are unmodeled in the training data, since the robot does not react to the humans in the training.

E. Multiple Decision Trees

In this section, we discuss how our approach extends to scenarios with multiple actors. Predictions and explanations can be computed as discussed previously, as the system needs to understand if it's interfering with each actor individually. Finding corrections, however, is more challenging, as the corrective action must not only remove interference with one actor, but also should not cause interference with others.

To consider different counterfactual rules of multiple DTs at once, we use *decision tree ensemble models* (DTEM). The main principle behind DTEM (Fig. 7) is that a group of weak learners come together to form a strong learner. Some

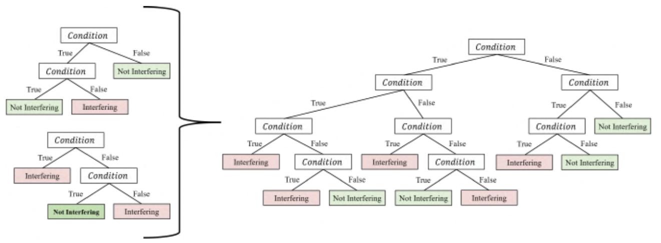


Fig. 7. General example of a Decision Tree Ensemble Model. Two weak learners come together to form a stronger learner.

popular methods for generating DTEMs include bagging, boosting, or using random forests, but these consist of random sampling methods [16], which are not viable for our case, since we need to capture all relevant data.

We instead take principles of majority voting DTEMs [17], which typically compare predictions from each DT, and select the majority output. We extend this concept by considering a single combined corrective tree, $\hat{\mathcal{T}}_c$, that incorporates the local training data of all individual trees, rather than just the prediction.

Before combining, the local data for each person are normalized such that $|\boldsymbol{\mathcal{S}}_i(t)| \approx |\boldsymbol{\mathcal{S}}_j(t)|$ with $i, j = 1, \dots, N_h$, where $|\cdot|$ gives the size of the enclosed dataset. In this

Authorized licensed use limited to: University of Virginia Libraries. Downloaded on August 24,2022 at 14:47:34 UTC from IEEE Xplore. Restrictions apply.

way, corrections will not be incorrectly biased towards those with more local data. We combine to obtain the training dataset: $\hat{\boldsymbol{\mathcal{S}}}(t) = [\boldsymbol{\mathcal{S}}_1(t), \ldots, \boldsymbol{\mathcal{S}}_{N_h}(t)]$, with which $\hat{\mathcal{T}}_c$ is constructed and counterfactuals are analyzed with the procedure discussed in Sec. IV-C, to obtain the optimal target, $\boldsymbol{\alpha}^*$, and the robot is controlled as described in Sec. IV-D.

Building $\hat{\mathcal{T}}_c$ in this way, however, only enables the system to find the best action if one exists within the data, meaning that there can be cases where all considered corrective actions are interfering. This can happen if: 1) the considered corrective data are sparse, meaning that the training set is not rich enough or 2) all possible actions are not feasible, for example, if the robot is surrounded by a crowd. For the former case, we propose a randomized approach to choose a velocity-lane pair that is not included in the local data: $\boldsymbol{\alpha}_s \setminus \hat{\boldsymbol{\alpha}(t)} \in \hat{\boldsymbol{\mathcal{S}}}(t)$. By doing so, the vehicle explores new options until it finds a solution. If no solution is found, the robot switches into a fail-safe mode of operation, which consists of a very low velocity and a reactive obstacle avoidance behavior for safety.

F. Online Validation and Updating

Due to the dynamic and dense nature of the environment, new and unmodeled human behaviors can be observed and corrective actions may not always eliminate all interference. Since our DTs are constructed at run-time, it is possible to introduce new data to the training set $\boldsymbol{\mathcal{S}}$ as observations are made. To avoid an exploding dataset, we introduce two test cases for adding new data: 1) decision validation, and 2) checking for unbounded observations.

Case 1 Decision validation is necessary when corrective actions from $\hat{\mathcal{T}}_c$ still result in an interference. This can occur when observations contain attribute values that are close to splitting conditions in the DTs leading to misclassification, or when a correction cannot be found within the DTEM. If the robot observes that the distance threshold δ_{th} is violated at run-time, the recorded attributes are included in $\boldsymbol{\mathcal{S}}$.

Case 2 If an observation is outside the bounds of the training data, reliable predictions or corrections cannot be expected. It has been shown for learning components that testing data within the vertices of the smallest convex set around training data produces the most accurate predictions [18]. In this work, convex hulls are generated around local training data using the Quickhull algorithm [19] to form a boundary denoted $Conv(\hat{\boldsymbol{\mathcal{S}}})$. Then, we check if observations are within the outermost points of each dimension (attribute) of the convex hull using linear inequalities [20]:

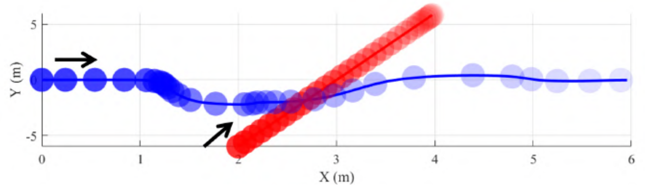
$$\min(Conv(\hat{\boldsymbol{\mathcal{S}}})) \leq \boldsymbol{\alpha}_h(t) \leq \max(Conv(\hat{\boldsymbol{\mathcal{S}}})) \qquad (9)$$

If any part of $\boldsymbol{\alpha}_h(t)$ is outside the convex hull, the data are labeled and included in $\boldsymbol{\mathcal{S}}$. The dataset updates at run-time through the presented test cases, resulting in more refined local trees, and therefore better decision making in the future.

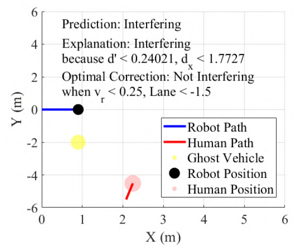
V. Simulations

We performed a series of simulations in MATLAB to test the effectiveness of our approach. Training included velocities $\boldsymbol{v_r} = [0.2, 0.4, 0.6, 0.8, 1.0]$m/s, and lanes $\boldsymbol{\ell} = [-2, -1, 0, 1, 2]$m simulating a classical non-holonomic UGV. The robot considers humans within a range of 5m and has a nominal velocity of 0.6m/s, and corrections are limited to any discrete velocity or lane seen in the training, that is from $\boldsymbol{v_r}$ and $\boldsymbol{\ell}$, respectively.

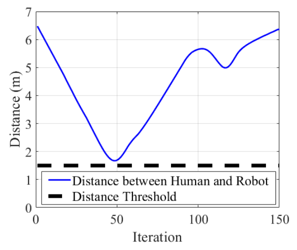
In the baseline simulation shown in Fig. 8, the robot (blue) is tasked to move from $(0, 0)$m to $(6, 0)$m while predicting, explaining, and correcting to avoid interfering with a person (red) moving along a trajectory previously unknown to the robot (i.e., different from the training set). The paths are



(a) Paths of human(red) and robot(blue).



(b) Decision tree operations.



(c) Distance to human.

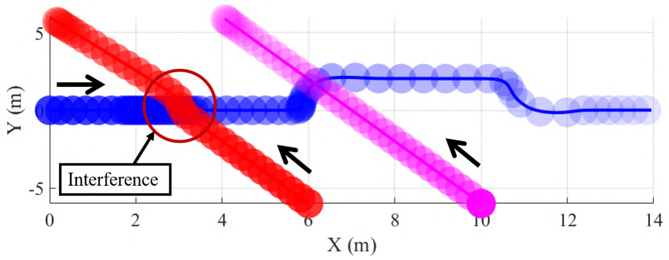
Fig. 8. Baseline simulation.

shown in Fig. 8(a), and prediction, explanation, optimal correction, and ghost vehicle are shown in Fig. 8(b). The robot predicts $\mathcal{P}_h(t) = \lambda$ and determines that it must apply the correction: $\boldsymbol{\alpha}^* = [v_r < 0.25, \ell < -1.5]$. The ghost vehicle (yellow marker) immediately applies these corrections. In Fig. 8(c), the distance between robot and human is shown to verify that the distance threshold $\delta_{th} = 1.5$m is not violated.

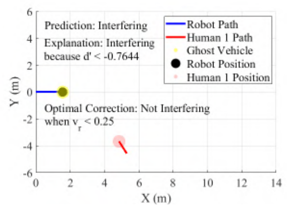
In Fig. 9, we show the effects of online validation and learning by performing a simulation with DTs trained on an incomplete training dataset. The robot's goal is $(14, 0)$m and the humans in this simulation take identical paths in succession to test whether the robot has improved its behavior. The robot makes an incorrect decision at first, applying the explanation and correction shown in Fig. 9(b), only slowing down to $v_r < 0.25$m/s. Both test cases (Sec. IV-F) are violated, shown by the deviation in the red path of Fig. 9(a) and the observation (red point) outside the partial convex hull in Fig. 9(d). Note that partial 3-dimensional convex hulls are shown for visualization purposes, due to the high dimensionality of our attributes, $\boldsymbol{\alpha} \in \mathbb{R}^8$. The magenta path in Fig. 9(a) shows no interference, as a different correction was selected (Fig. 9(c)), since the observations are now included in the local data (Fig. 9(e)).

We also extensively test our approach in handling multiple people at a time, shown in Fig. 10. The robot navigates through 10 people, changing its lane (ℓ) and velocity (v_r) to avoid interfering. This test was run for 50 trials with random human trajectories. All decision tree operations in these simulations took between $30 - 90$ms. Our approach successfully eliminated interferences 72% of the time, with an average minimum distance of $\delta_{\min} = 1.58$m. Where interference occurred, we observed that the robot was in

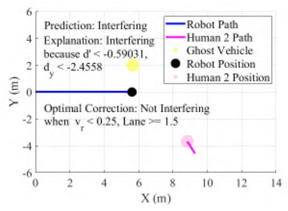
Authorized licensed use limited to: University of Virginia Libraries. Downloaded on August 24,2022 at 14:47:34 UTC from IEEE Xplore. Restrictions apply.



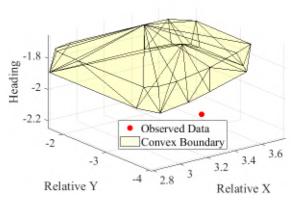
(a) Paths of humans (red, magenta) and robot (blue).

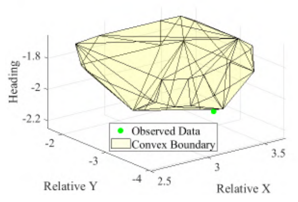


(b) Incorrect DT operations.



(c) Corrected DT operations.





(d) Partial convex hull prior to updates. (e) Partial convex hull after updates.

Fig. 9. Simulation of run-time validation and updating.

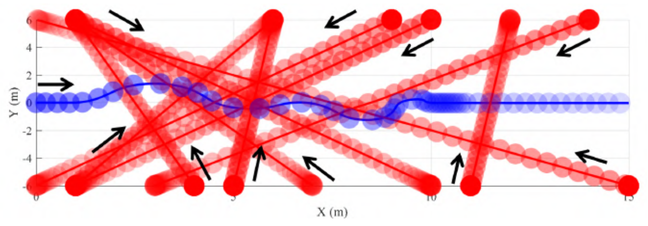


Fig. 10. Human (red) and robot (blue) paths in a multi-actor simulation

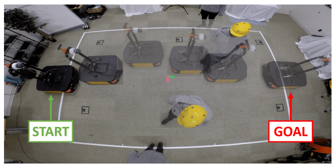
dense crowds, negotiating with on average $N_h(t) \geq 7$.

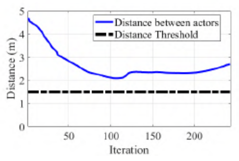
VI. EXPERIMENTS

The proposed approach was also validated experimentally on a Clearpath Robotics Ridgeback Omnidirectional Platform (see Fig. 1) in indoor environments. In the first experiments, a VICON motion capture (MOCAP) system was used to obtain robot and human states. In the second experiment, the robot uses an on-board ASUS Xtion RGB-D camera with the SPENCER people tracking package [21]. Below we present a few cases that capture the essence of the proposed framework. Videos of the presented experiments can be found in the submitted supplemental material.

1) MOCAP Experiments: In the first experiment shown in Fig. 11, the robot predicts, explains, and takes corrective actions proactively to avoid the human. The robot moves from $(-2.5, 0)$m to $(2.5, 0)$m at a nominal velocity of $v_r = 0.6$m/s, and the distance threshold $\delta_{th} = 1$m. The robot predicts $\mathcal{P}_h(t) = \lambda$ and explains:

$\mathcal{E}_h(t) =$ ***Interfering*** because: $\{d' < 0.21, d_x > 2.59\}$





(a) Snapshots of experiment. (b) Distance between actors.

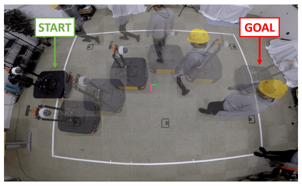
Fig. 11. Snapshots and distances of lab experiment.

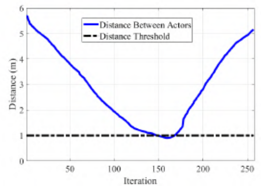
With the correction tree, the robot computes:

$\mathcal{C}_h(t) =$ ***Not Interfering*** when: $\{v_r \geq 0.5, \ell > 0.5\}$

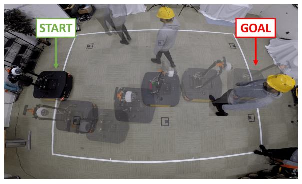
Thus, the corrective action is to maintain nominal velocity, and move to the lane in the positive y-direction. The minimum distance between actors is $\delta_{\min} = 2.09$m (Fig. 11(b)).

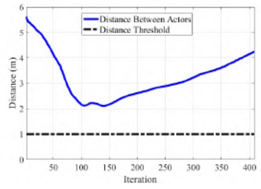
In the experiment shown in Fig. 12, we examine the effects of online validation and updating. As a proof of concept, we





(a) Experiment snapshots before updates. (b) Distance between actors before update.





(c) Experiment snapshots after updates. (d) Distance between actors after update.

Fig. 12. Experiment showing the effects run-time updates.

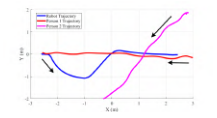
removed a large portion of our training data, and as shown in the first run in Fig. 12(a) we observed that the robot moved incorrectly toward the human. The distance threshold is violated, $\delta_{\min} = 0.91$m (Fig. 12(b)), and the person alters his path, reacting to the robot's interfering behavior. After the model is updated and reinforced at run-time, the robot makes the appropriate correction and no interference is observed, with $\delta_{\min} = 2.13$m (Fig. 12(c-d)).

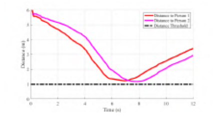
In Fig. 13, we show the effectiveness of our approach with two people in the lab environment. The robot has the goal to reach $(2.5, 0)$m and successfully avoids interference with both people at once, even in a small space, showing that our approach scales to accommodating multiple actors at the same time. The trajectories for all agents are shown in Fig. 13(e), and the minimum distance between any human and robot was 1.23m, which is above the threshold $\delta_{th} = 1$m.

2) On-Board Sensing Experiment: To demonstrate the applicability of our approach outside MOCAP settings, we deployed our technique on the same robot using only the on-board RGB-D and Lidar sensors to identify and track surrounding humans and localize itself. Fig. 14 shows the results for this experiment in which the robot is able to successfully avoid interference with surrounding people and

Authorized licensed use limited to: University of Virginia Libraries. Downloaded on August 24,2022 at 14:47:34 UTC from IEEE Xplore. Restrictions apply.

(a) Initial positions of actors. (b) Robot performs corrective behavior. (c) Intermediate positions of actors

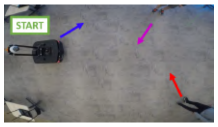


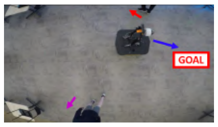


(d) Final positions of actors. (e) Trajectories of actors. (f) Distance between actors.

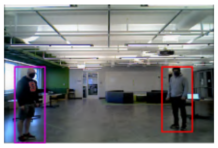
Fig. 13. Snapshots, trajectories and distances of 2-person lab experiment.

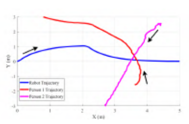
reaches its goal without violating the distance threshold despite noisy camera measurements and uncertain person detection and tracking.

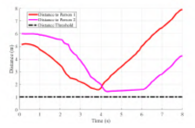




(a) Initial positions of actors. (b) Robot performs corrective behavior. (c) Actors arrive at final positions







(d) Robot first person view. (e) Trajectories of actors. (f) Distance between actors.

Fig. 14. Snapshots and trajectories of 2-person lab experiment.

VII. Conclusions & Future Work

In this work, we have presented a novel approach for interpretable prediction and planning of a robot in a co-robotic environment. We relax the requirement of explicitly predicting human paths, and instead directly predict, explain, and find counterfactual rules for interfering behaviors with binary decision trees and a library of pre-trained trajectories. Unique from other works in this field, we validate robot behaviors to update the predictive model at run-time, resulting in improved behaviors in future operations. While we focused on human-robot operations in this work, our framework works for any path planning operation with multiple actors. The results overall show desirable robot behaviors among humans, though we found that performance can decrease in very dense crowds.

In current and future work we are exploring the idea of training decision trees using probabilities measured, collected, and updated at run-time to assist decision making in traditional planners and controllers. In addition, we plan to study how learning online can be leveraged safely and efficiently without any reliance on a pre-trained dataset.

VIII. Acknowledgement

This material is based upon work supported by NSF under grant number #1816591 and the Air Force Research Laboratory and the Defense Advanced Research Projects Agency under Contract No. FA8750-18-C-0090.

References

[1] Z. Huang, A. Hasan, and K. Driggs-Campbell, "Intention-aware residual bidirectional lstm for long-term pedestrian trajectory prediction," 2020.

[2] 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.

[3] Y. Xu, Z. Piao, and S. Gao, "Encoding crowd interaction with deep neural network for pedestrian trajectory prediction," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 5275–5284.

[4] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[5] R. Blake and M. Shiffrar, "Perception of human motion," *Annual Review of Psychology*, vol. 58, no. 1, pp. 47–73, 2007, pMID: 16903802.

[6] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[7] C. D. Franco and N. Bezzo, "Interpretable run-time monitoring and replanning for safe autonomous systems operations," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2427–2434, 2020.

[8] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 4674–4683.

[9] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1343–1350, 2017.

[10] R. Peddi, C. Di Franco, S. Gao, and N. Bezzo, "A data-driven framework for proactive intention-aware motion planning of a robot in a human environment," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[11] Y. Gao, P. B. Luh, H. Zhang, and T. Chen, "A modified social force model considering relative velocity of pedestrians," in *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug 2013, pp. 747–751.

[12] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 1135–1144.

[13] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, "Local rule-based explanations of black box decision systems," *arXiv preprint arXiv:1805.10820*, 2018.

[14] F. Xia, W. Zhang, F. Li, and Y. Yang, "Ranking with decision tree," *Knowl. Inf. Syst.*, vol. 17, no. 3, p. 381–395, Dec. 2008.

[15] E. Horváth, C. Hajdu, and P. Kőrös, "Novel pure-pursuit trajectory following approaches and their practical applications," in *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, 2019, pp. 000 597–000 602.

[16] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A comparison of decision tree ensemble creation techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 173–180, 2007.

[17] Y. Akiba, S. Kaneda, and H. Almuallim, "Turning majority voting classifiers into a single decision tree," in *Proceedings Tenth IEEE International Conference on Tools with Artificial Intelligence (Cat. No.98CH36294)*, 1998, pp. 224–230.

[18] S. Ding, X. Nie, H. Qiao, and B. Zhang, "A fast algorithm of convex hull vertices selection for online classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 4, pp. 792–806, 2018.

[19] S. Srungarapu, D. P. Reddy, K. Kothapalli, and P. J. Narayanan, "Fast two dimensional convex hull on the gpu," in *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*, March 2011, pp. 7–12.

[20] G. Amato, F. Scozzari, and E. Zaffanella, "Efficient constraint/generator removal from double description of polyhedra," *Electronic Notes in Theoretical Computer Science*, vol. 307, pp. 3 – 15, 2014, fifth International Workshop on Numerical and Symbolic Abstract Domains (NSAD).

[21] T. Linder, S. Breuers, B. Leibe, and K. O. Arras, "On multi-modal people tracking from mobile platforms in very crowded and dynamic environments," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 5512–5519.

Authorized licensed use limited to: University of Virginia Libraries. Downloaded on August 24,2022 at 14:47:34 UTC from IEEE Xplore. Restrictions apply.