Dynamically Switching Human Prediction Models for Efficient Planning

Arjun Sripathy*, Andreea Bobu*, Daniel S. Brown, and Anca D. Dragan

***Abstract*— As environments involving both robots and humans become increasingly common, so does the need to account for people during planning. To plan effectively, robots must be able to respond to and sometimes influence what humans do. This requires a *human model* which predicts future human actions. A simple model may assume the human will continue what they did previously; a more complex one might predict that the human will act optimally, disregarding the robot; whereas an even more complex one might capture the robot's ability to influence the human. These models make different trade-offs between computational time and performance of the resulting robot plan. Using only one model of the human either wastes computational resources or is unable to handle critical situations. In this work, we give the robot access to a suite of human models and enable it to assess the performance-computation trade-off online. By estimating how an alternate model could improve human prediction and how that may translate to performance gain, the robot can *dynamically switch human models* whenever the additional computation is justified. Our experiments in a driving simulator showcase how the robot can achieve performance comparable to always using the best human model, but with greatly reduced computation.**

I. Introduction

When robots operate in close proximity to humans, it is crucial that they anticipate what people will do to respond appropriately. Such prediction often involves equipping the robot with a model of human behavior [1]. This model could be physics-based [2]–[4], pattern-based [5]–[8], approximate rationality with respect to an objective function [9]–[17], or even two player games [18]–[21].

What human model a robot should be equipped with depends on the trade-offs it needs to make: some models, like the physics-based ones, are cheap to compute but don't capture human intentions and may be less accurate; others, like two player games, model interactions between the person and the robot, but at a high computational cost. For systems interacting with people in real time, like autonomous vehicles or mobile robots, compromising on either accuracy or computation is undesirable. For instance, in the driving scenario in Fig. 1, using the cheap model might pose a safety hazard, while picking the more accurate one may limit planning frequency or strain computational resources needed elsewhere (sensor suite, perception system, routing, etc).

We advocate that the robot should not be stuck with a single model, but instead have the ability to dynamically change which model it is using as the interaction progresses. In Fig. 1, the robot starts off with the cheap model. Anticipating a potential collision, it switches to a more complex model, reverting back once the critical maneuver is complete.

*Indicates equal contribution. Authors are with EECS at UC Berkeley. Research supported by the Air Force Office of Scientific Research (AFOSR), NSF grant IIS1734633 (SCHooL), and NSF grant IIS1652083 (CAREER).

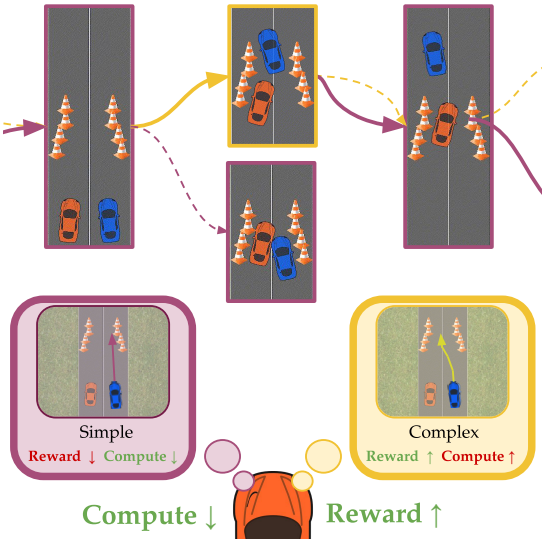

Fig. 1: The robot (orange car) can plan with either a complex model (yellow bubble) of the human (blue car) that is more accurate but more expensive or with a simple model (purple bubble) that is not as accurate but cheaper. Our algorithm uses the complex model when a collision is imminent (solid yellow line), but saves computation by switching to the simple one afterwards (solid purple line).

The idea of using multiple predictive models is not new; most works, like interactive multiple model filtering [22]–[25] or other ways of combining models [25] are focused on improving accuracy by leveraging complementary strengths of different models. In contrast, we are focused on the setting where we have complex but accurate models (which could be mixtures themselves), and cheap but less accurate ones. In such settings, if it weren't for computational costs, we would use the complex models all the time. The question becomes: when is the performance gain worth the extra computation?

We could plan with the complex model and measure the performance gain, but doing so defeats the purpose of saving computation. To avoid this, prior work proposed training a model to predict which agents "influence" or "affect" the planner, and prioritizing computation for them [26]. This approach does not estimate the performance-computation trade off, but heuristically assumes that agents who influence the planner are conducive to high gain. However, there is a fundamental difference between needing to consider an agent at all, and estimating the performance gain between predicting their behavior using a cheap versus a complex model. For example, in Fig. 1, a cheap model is sufficient in

the leftmost and rightmost frames where a critical maneuver is not necessary, despite the human influencing the planner.

Instead of heuristically allocating computation, we employ efficient, online estimation for how an alternate human model could change robot performance. This enables switching to the model which best trades off reward and computation in real time. In Fig. 1, a car using our switching methodology is able to achieve a behavior similar to the one produced with the most accurate model, but at a computational cost closer to that of the cheaper model.

This paper makes three key contributions: (1) a formalism for the robot's decision process that optimally trades off between computation and accuracy across multiple predictive models, (2) an approximate solution that solves this decision process online, and (3) a comparative analysis of our model switching algorithm in a human-autonomous car system. Together, these contributions give robots the autonomy to decide in real-time what predictive human models are most appropriate to use in different interactive scenarios. Code and videos are made available at arjunsripathy.github.io/model_switching

II. Method

We focus on a system consisting of a robot R interacting in an environment with other human agents H. The robot's goal is to plan around the humans in a manner that is most effective while minimizing computational time. We present our theory for a general single human, single robot setting where the other agents' behavior is known, although our method can easily be extended by running it separately for each human. We use the running example of an autonomous car sharing the road with a human driver to illustrate the proposed approach and demonstrate the utility of our method.

A. Problem Statement

We model the system as a fully observable dynamical system in which one agent's controls potentially impact the other's. As such, let the state $x \in \mathcal{X}$ include the positions and velocities of both agents, where the human action $u_H \in \mathcal{U}_H$ and robot action $u_R \in \mathcal{U}_R$ each can affect the next state in a combined dynamics model: $x^{t+1} = f(x^t, u_R^t, u_H^t)$.

Let $\mathbf{x} = [x^1, \dots, x^N]$ be a finite horizon N state sequence, $\mathbf{u}_R = [u_R^1, \dots, u_R^N]$ the robot's continuous control inputs, and $\mathbf{u}_H = [u_H^1, \dots, u_H^N]$ the human's. The robot optimizes its controls $\mathbf{u}_R$ according to a reward function that depends on the joint sequence of states and controls: $R_R(x^0, \mathbf{u}_R, \mathbf{u}_H) = \sum_{\tau=1}^{N} r_R(x^\tau, u_R^\tau, u_H^\tau)$, where x^0 is the starting state and each state thereafter is obtained via the dynamics model from the previous robot and human controls. The person chooses their action at time t, u_H^t, according to an internal policy $\pi_H(x^t, u_R^t)$, which, when applied at every state x^t, results in $\mathbf{u}_H$. We let $\mathbf{u}_R$ and $\mathbf{u}_H$ represent the true executed robot and human controls, respectively.

Our system is a Markov Decision Process (MDP) with states $\mathcal{X}$, actions $\mathcal{U}_R$, transition function $f(x, u_R, u_H)$, and reward $r_R(x, u_R, u_H)$. Since the robot does not know π_H (that would require access to the human's brain), it uses a *human model* $M : \mathcal{X} \times \mathcal{U}_R^N \to \mathcal{U}_H^N$ to make a *prediction* of the human controls $\bar{\mathbf{u}}_H$. The robot then seeks a *plan* $\bar{\mathbf{u}}_R^0(M) \coloneqq \bar{\mathbf{u}}_R(x^0, M)$ that maximizes the MDP reward R_R based on the predicted $\bar{\mathbf{u}}_H^0(\bar{\mathbf{u}}_R, M) \coloneqq \bar{\mathbf{u}}_H(x^0, \bar{\mathbf{u}}_R, M)$.

Unfortunately, this type of *offline planning* doesn't consider modeling errors introduced by imperfect models M. Hence, it is more common for the robot to perform *online planning* at every time step t to obtain more accurate plans $\bar{\mathbf{u}}_R^t(M)$. For computational efficiency, we follow [18] and use Model Predictive Control (MPC) [27] with a finite horizon $K < N$, where at each time step t the robot optimizes its controls to maximize the cumulative reward:

$$\bar{\mathbf{u}}_R^t(M) = \arg\max_{\mathbf{u}_R} R_R(x^t, \mathbf{u}_R, \bar{\mathbf{u}}_H^t(\mathbf{u}_R, M)) \ , \quad (1)$$

where R_R is evaluated only over the first K states starting at x^t. The robot executes the first action from its plan and then replans at the next time step $t+1$. To simplify notation, we will denote the first action of a plan as $\bar{u}_R^t(M) \coloneqq \bar{\mathbf{u}}_R^t(M)[0]$.

The crucial question is what human model M should the robot use for planning with Eq. (1)? Restricting ourselves to any single model either hurts performance or computational efficiency. We propose an algorithm which enables efficient switching between models of varying accuracy and complexity, allowing for both high performance and low computation.

B. Model Switching Formalism

We assume the robot has access to a ladder of human models $\mathcal{M} = \{M_0, \dots, M_n\}$, where as we climb the ladder we sacrifice computational time for greater expected reward: $\mathbb{E}_x[R_R(x, \bar{\mathbf{u}}_R(M_j), \mathbf{u}_H)] \geq \mathbb{E}_x[R_R(x, \bar{\mathbf{u}}_R(M_i), \mathbf{u}_H)]$ and $T(M_j) > T(M_i), \forall i < j$, where $T(M)$ is the time to solve Eq. (1) under M.

At every time step t, the robot needs to choose the human model $M^t \in \mathcal{M}$ to use for planning $\bar{\mathbf{u}}_R^t$. To determine which model the robot should use at time t, we construct a *meta MDP* on top of the previous MDP, with states x^t, actions $M^t \in \mathcal{M}$ representing the choice of human model, transition function $f(x^t, \bar{u}_R^t(M^t), u_H^t)$, and meta-reward $r_{meta}^t = r_R(x^t, \bar{u}_R^t(M^t), u_H^t) - \lambda * T(M^t)$, with λ trading off actual reward gained by planning under M^t and computational time spent on the plan. Lower values for λ favor more complex models, whereas higher values result in more usage of less expensive ones.

Solving this MDP exactly is impossible, since it requires access to the true human controls $\mathbf{u}_H$ ahead of time. Moreover, even if we approximate the true human controls with those predicted by our most accurate model M_n, the robot needs to find the model sequence that maximizes the cumulative meta reward across the episode—a procedure exponential in the episode horizon and, thus, intractable. We could alleviate this computational burden by assuming the robot myopically decides when to switch models: instead of considering the cumulative meta reward, only look at r_{meta}^t to decide whether to switch at time $t+1$. On the plus side, this simplification is more tractable and only needs the current human control. Unfortunately, even this relaxation requires planning with every model and picking the one with the best

r^t_{meta}, which is worse than simply using M_n from the get-go. Thus, we now propose an approximate solution that avoids computing every plan while still switching models as needed.

C. Approximate Solution: Switching between Two Models

For ease of exposition, we first discuss how the robot can decide whether to switch from a simple model M_1, used for planning at timestep t, to a complex model M_2, which we are considering for timestep $t+1$. We are interested in estimating the reward the robot would gain from using M_2.

Estimate Change in Robot Plan

Leveraging our plan computed at timestep t using M_1, we get around explicitly generating $\bar{\mathbf{u}}^t_R(M_2)$ by approximating it as $\hat{\mathbf{u}}^t_R(M_2) = \bar{\mathbf{u}}^t_R(M_1) + \Delta\hat{\mathbf{u}}_R$. Here, we want to choose $\Delta\hat{\mathbf{u}}_R$ such that it maximizes the robot reward R_R under the complex model M_2. However, optimizing $\Delta\hat{\mathbf{u}}_R$ using Eq. (1) is equivalent to planning. To get an efficient estimate, we use a quadratic Taylor series approximation of R_R, denoted $\tilde{R}_R$, evaluated around $(x^t, \bar{\mathbf{u}}^t_R(M_1), \bar{\mathbf{u}}^t_H(\bar{\mathbf{u}}^t_R(M_1), M_1))$, our current plan and human prediction:

$$\Delta\hat{\mathbf{u}}_R = \arg\max_{\Delta\mathbf{u}_R} \tilde{R}_R(x^t, \hat{\mathbf{u}}^t_R(M_2), \bar{\mathbf{u}}^t_H(\hat{\mathbf{u}}^t_R(M_2), M_2)) \ . \tag{2}$$

Estimate Change in Human Prediction

Eq. (2) requires approximating the M_2 human's response $\bar{\mathbf{u}}^t_H$ to the robot's plan $\hat{\mathbf{u}}^t_R(M_2)$, which can be broken down into M_2's response to the current plan $\bar{\mathbf{u}}^t_R(M_1)$ plus the change coming from $\Delta\hat{\mathbf{u}}_R$: $\hat{\mathbf{u}}^t_H(\hat{\mathbf{u}}^t_R(M_2), M_2) = \hat{\mathbf{u}}^t_H(\bar{\mathbf{u}}^t_R(M_1) + \Delta\hat{\mathbf{u}}_R, M_2)$. We can linearly approximate this:

$$\hat{\mathbf{u}}^t_H(\hat{\mathbf{u}}^t_R(M_2), M_2) = \bar{\mathbf{u}}^t_H(\bar{\mathbf{u}}^t_R(M_1), M_2) + \frac{d\hat{\mathbf{u}}_H}{d\hat{\mathbf{u}}_R} \cdot \Delta\hat{\mathbf{u}}_R \ , \tag{3}$$

where $\frac{d\hat{\mathbf{u}}_H}{d\hat{\mathbf{u}}_R} = \frac{d\hat{\mathbf{u}}_H(\hat{\mathbf{u}}_R, M)}{d\hat{\mathbf{u}}_R}\Big|_{\substack{\hat{\mathbf{u}}_R = \bar{\mathbf{u}}^t_R(M_1) \\ M = M_2}}$. This requires evaluating the complex model M_2, which might be expensive, though not nearly as expensive as planning with it.

We may now substitute the changed robot plan $\hat{\mathbf{u}}^t_R(M_2)$ and the changed human prediction from Eq. (3) into our reward expression we maximize in Eq. (2)

$$\tilde{R}_R(x^t, \bar{\mathbf{u}}^t_R(M_1) + \Delta\hat{\mathbf{u}}_R, \bar{\mathbf{u}}^t_H(\bar{\mathbf{u}}^t_R(M_1), M_2) + \frac{d\hat{\mathbf{u}}_H}{d\hat{\mathbf{u}}_R} \cdot \Delta\hat{\mathbf{u}}_R) \ . \tag{4}$$

Performance Gain from the Change in Robot Plan

Ultimately, to assess the robot's performance gain by using M_2, we want to estimate the reward component of r^t_{meta} under M_2, $r_R(x^t, \hat{u}^t_R(M_2), \bar{u}^t_H(\hat{\mathbf{u}}^t_R(M_2), M_H))$, where $\bar{\mathbf{u}}^t_H(\hat{\mathbf{u}}^t_R(M_2), M_H)$ would be the human's response under the true model M_H to the robot's plan with M_2, and $\bar{u}^t_H(\hat{\mathbf{u}}^t_R(M_2), M_H)$ is the first control of that response. To simplify notation, we will denote this reward $r^t_R(M_2)$.

Since our approximation of the meta MDP considers myopic rewards r_R, we really are only interested in the first control $\hat{u}^t_R(M_2)$ of the changed plan $\hat{\mathbf{u}}^t_R(M_2)$, and the first control $\bar{u}^t_H(\hat{\mathbf{u}}^t_R(M_2), M_H)$ of the true human's response to that plan. Given $\Delta\hat{\mathbf{u}}_R$, we only need the change for the first control $\Delta\hat{u}_R$ combined with the approximation of $\bar{u}^t_H(\hat{\mathbf{u}}^t_R(M_2), M_H)$ as in Eq. (3) to estimate $\hat{r}^t_R(M_2)$ as:

$$r_R(x^t, \bar{u}^t_R(M_1) + \Delta\hat{u}_R, \bar{u}^t_H(\bar{\mathbf{u}}^t_R(M_1), M_H) + \frac{d\hat{u}_H}{d\hat{u}_R} \cdot \Delta\hat{u}_R) \ , \tag{5}$$

where $\frac{d\hat{u}_H}{d\hat{u}_R} = \frac{d\hat{u}_H(\hat{\mathbf{u}}_R, M)}{d\hat{u}_R}\Big|_{\substack{\hat{\mathbf{u}}_R = \bar{\mathbf{u}}^t_R(M_1) \\ M = M_H}}$. Here, we don't know M_H, but we can approximate it with the complex model M_2.

One Time Step Simplification

All of this requires evaluating the complex model M_2 at the current plan $\bar{\mathbf{u}}^t_R(M_1)$, which, although cheaper than fully planning with M_2, is still expensive. Since the meta reward only evaluates the first control of the plans, we can do better by simplifying our formulation further to planning and predicting a single control. That is, we consider the changed control $\hat{u}^t_R(M_2) = \bar{u}^t_R(M_1) + \Delta\hat{u}_R$, with its corresponding changed human control prediction from Eq. (3):

$$\hat{u}^t_H(\hat{\mathbf{u}}^t_R(M_2), M_2) = \bar{u}^t_H(\bar{\mathbf{u}}^t_R(M_1), M_2) + \frac{d\hat{u}_H}{d\hat{u}_R} \cdot \Delta\hat{u}_R \ , \tag{6}$$

where $\frac{d\hat{u}_H}{d\hat{u}_R} = \frac{d\hat{u}_H(\hat{\mathbf{u}}_R, M)}{d\hat{u}_R}\Big|_{\substack{\hat{\mathbf{u}}_R = \bar{\mathbf{u}}^t_R(M_1) \\ M = M_2}}$.

We obtain $\Delta\hat{u}_R$ by optimizing the following objective:

$$\arg\max_{\Delta u_R} \tilde{r}_R(x^t, \hat{u}^t_R(M_2), \bar{u}^t_H(\bar{\mathbf{u}}^t_R(M_1), M_2) + \frac{d\hat{u}_H}{d\hat{u}_R} \cdot \Delta u_R) \ . \tag{7}$$

Our simplified derivation still requires the gradient $\frac{d\hat{u}_H}{d\hat{u}_R}$, but this is cheaper to compute than $\frac{d\hat{\mathbf{u}}_H}{d\hat{\mathbf{u}}_R}$.

Armed with an estimated $\hat{r}^t_R(M_2)$ from Eq. (5), we can now determine whether the performance gain from M_2 is worth the additional compute by considering

$$\Delta r^t_{meta} = \hat{r}^t_R(M_2) - \lambda * T(M_2) - (r^t_R(M_1) - \lambda * T(M_1)) \ , \tag{8}$$

where we already know $r^t_R(M_1)$ from the previous time step, and $T(M_1)$ and $T(M_2)$ are known a priori from their model specifications. If Δr^t_{meta} is positive, the robot should switch to M_2; otherwise, the robot should continue using M_1.

Intuitive Interpretation

In Eq. (8), since T's are constants, the robot's decision of whether to switch relies on comparing $r^t_R(M_1) = r_R(x^t, \bar{u}^t_R(M_1), \bar{u}^t_H(\bar{\mathbf{u}}^t_R(M_1), M_1))$ to $\hat{r}^t_R(M_2) = r_R(x^t, \bar{u}^t_R(M_1) + \Delta\hat{u}_R, \bar{u}^t_H(\bar{\mathbf{u}}^t_R(M_1), M_2) + \frac{d\hat{u}_H}{d\hat{u}_R} \cdot \Delta\hat{u}_R)$. Looking at these rewards, a few key distinctions stand out.

First, the robot is interested in knowing whether M_2 would give a different prediction $\bar{u}^t_H(\bar{\mathbf{u}}^t_R(M_1), M_2)$ than M_1's $\bar{u}^t_H(\bar{\mathbf{u}}^t_R(M_1), M_1)$ on the current plan. For example, in Fig. 1 we realize a more complex model foresees the human turning into the bottleneck compared to naive constant velocity. Here this foresight prevents a collision.

Meanwhile, the derivative $\frac{d\hat{u}_H}{d\hat{u}_R}$ captures the *influence* that the robot's controls have on the human's. Simple models may ignore this, but more complex models, like the two player game, capture this dynamic enabling certain critical maneuvers. For instance, humans will yield should the robot merge to create proper spacing. Only if aware of this influence will the robot feel confident merging into tight windows.

So how much do these two terms matter? Intuitively, by computing $\Delta\hat{u}_R$ via Eq. (7), we see how much a model

Algorithm 1: Dynamic Model Switching

Input: Ladder $\mathcal{M} = \{M_0, \ldots, M_n\}$, episode time N.
Start with time $t = 0$, current model index i.
while $t \leq N$ **do**
  Compute $\bar{\mathbf{u}}_R^t(M_i)$ given x^t and execute $\bar{u}_R^t(M_i)$.
  if $i < n$ **then**
    Substitute $(M_1, M_2) \leftarrow (M_i, M_n)$ in Eq. (5), (7), (8), with $\bar{u}_H^t(\bar{\mathbf{u}}_R^t(M_i), M_n) \leftarrow u_H^t$.
    Compute Δr_{meta}^t using Eq. (5), (7), (8).
    if $\Delta r_{meta}^t > 0$ **then**
      $i \leftarrow n$ (Switch up), continue
  if $i > 0$ *and cooldown complete* **then**
    Substitute $(M_1, M_2) \leftarrow (M_i, M_{i-1})$ in Eq. (5), (7), (8).
    Compute Δr_{meta}^t using Eq. (5), (7), (8).
    if $\Delta r_{meta}^t > 0$ **then**
      $i \leftarrow i - 1$ (Switch Down).

that either captures influence or makes different predictions affects the robot's plan. If neither bears much weight, then $\Delta\hat{u}_R$ will likely be negligible and we will not switch. If however $\Delta\hat{u}_R$ is significant, we further evaluate if that translates to significant performance gain. Only when that's the case will we ultimately switch.

D. Approximate Solution: Switching Between the Ladder

Although we presented our method in the context of switching up, the framework holds when M_1 is the current model and M_2 is any alternative: higher or lower. When the alternate model is lower the question becomes: is the computational saving worth the loss in reward? Generalizing our derivation to a ladder of models $\mathcal{M} = \{M_0, \ldots, M_n\}$, suppose that at time t the robot used model M_i to plan $\bar{\mathbf{u}}_R^t(M_i)$. We would like to decide on a model for time $t+1$.

First, we evaluate if it's worthwhile to switch to a higher model M_j, $j > i$. The robot could consider M_{i+1}, the model immediately above, and successively switch up. However, in urgent, safety-critical situations we may need to switch higher than that immediately to avoid an accident. Thus, we exclusively consider the best model and upper bound $\hat{r}_R^t(M_j)$ with $\hat{r}_R^t(M_n)$, the estimated reward from using the best model available. To avoid having to evaluate M_n's expensive predictions we substitute a perfect prediction $\bar{u}_H^t(\bar{\mathbf{u}}_R^t(M_i), M_N) \approx u_H^t$. This effectively upper bounds M_n with the true human model M_H.

If the robot should have switched down to M_j, $j < i$, we observe that $r_R^t(M_j) \leq r_R^t(M_{i-1}) \leq r_R^t(M_i)$. Thus, for efficient switching, we only consider the model directly below, M_{i-1}. Since $T(M_{i-1}) < T(M_i)$, it is reasonable to compute true model predictions in our approximation.

Finally, if Δr_{meta}^t is positive for M_H, the robot switches up to M_n. Otherwise, if Δr_{meta}^t is positive for M_{i-1}, the robot switches down to M_{i-1}. Otherwise we stay as summarized in Algorithm 1.

In practice evaluating M_{i-1}'s predictions can still be significant, but unlike switching up, safety and performance concerns do not force us to check every timestep. One may wait K timesteps after failing to switch down before trying again. This cooldown hyperparameter, K, should be set based on how often one actually switches.

III. Experiments

We now demonstrate the efficacy of our model switching algorithm in three simulated autonomous driving scenarios.

A. Driving Simulator

We model the dynamics of the vehicles as a 4D bicycle model. Let the state of the system be $\mathbf{x} = [x\ y\ \theta\ v]^T$, where x, y are the coordinates of the vehicle, θ is the heading, and v the speed. The actions are $u = [\omega\ a]^T$, where ω is the steering input and a is the linear acceleration. We use α as the friction coefficient, and the vehicle's dynamics model is:

$$[\dot{x}\ \dot{y}\ \dot{\theta}\ \dot{v}] = [v \cdot \cos(\theta) \quad v \cdot \sin(\theta) \quad v \cdot \omega \quad a - \alpha \cdot v] \ . \quad (9)$$

B. Human Predictive Models

For our experiments, we use the following 3 models of varying computational complexity and accuracy:

1) Constant Velocity: This model, deemed *Naive*, predicts that the person will provide zero acceleration control, maintaining their current heading and speed.

2) Human Plans First: This model assumes that the person acts according to a reward function parameterized as a linear combination of features ϕ: $R_H(x^0, \mathbf{u}_R, \mathbf{u}_H) = \sum_{\tau=1}^N r_H(x^\tau, u_R^\tau, u_H^\tau) = \sum_{\tau=1}^N \theta^T \phi(x^\tau, u_R^\tau, u_H^\tau)$. Additionally, under this model the human believes that the robot will maintain a constant velocity, and optimizes their reward w.r.t the imagined robot plan $\tilde{\mathbf{u}}_R$ to obtain a plan $\bar{\mathbf{u}}_H$:

$$\bar{\mathbf{u}}_H(x^0, \tilde{\mathbf{u}}_R) = \arg\max_{\mathbf{u}_H} R_H(x^0, \tilde{\mathbf{u}}_R, \mathbf{u}_H) \ . \quad (10)$$

We refer to this model as *Turn* because the person takes the first turn in choosing their controls.

3) Cognizant of Effects on Human Action: Our last model is based on [18] and models the human as an agent which will optimally respond to the robot's plan. This results in a nested optimization as the robot accounts for how its plan affects the human's. As we rationalize the human's thought process we refer to this model as "Theory of Mind" [28], or *ToM* for short.

Planning with the first two approaches involves optimizing the robot's control against the fixed human prediction. Meanwhile, ToM's nested optimization returns both a human and robot control with no further optimization required. From our experiments, planning using Turn and ToM takes roughly twice and four times as long as using Naive, respectively. However, their expected accuracy has the reverse ordering.

Both Turn and ToM rely on knowing a reward parameter θ and a set of features ϕ for the human reward. To learn a good θ for every scenario, we collected demonstrations of a single human driver in an environment with multiple autonomous cars following precomputed routes, and performed inverse reinforcement learning [29]. The base features we used include higher speed moving forward (against a speed

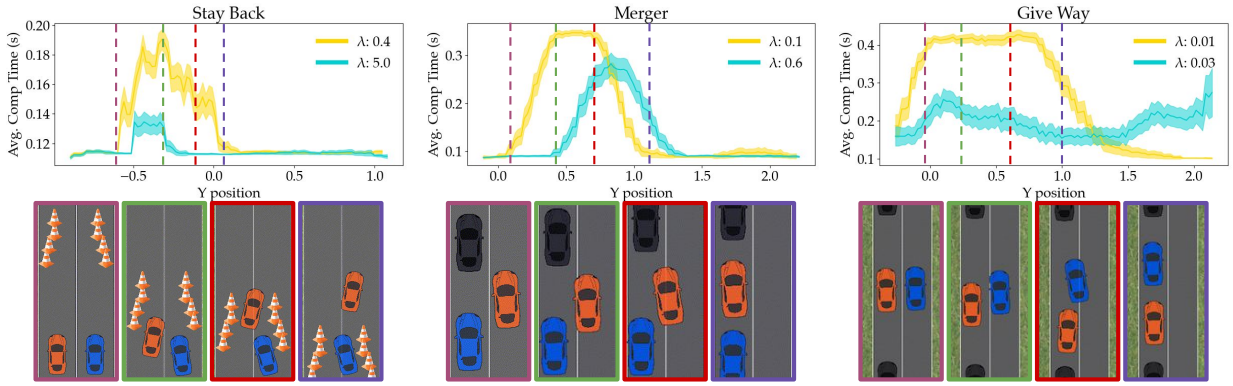

Fig. 2: (Top) Average single step computation time over the course of Stay Back (left), Merger (center), and Give Way (right), for a conservative, lesser λ, switcher (yellow) and an aggressive, larger λ, one (light blue). (Bottom) Example conservative MS robot (orange) behavior around the target human (blue) and other cars (black).

limit), lateral and directional alignment in the lane, collision avoidance, and distance to the boundaries of the road.

Across our experiments, we compare each of these three models against our model switcher (MS) that dynamically chooses between all of them. We additionally wanted to analyze different performances for designers that might be more or less conservative about the reward vs. computation tradeoff, so we show results for different values of λ.

C. Miscellaneous Experimental Details

We conduct experiments using TensorFlow [30] 2.1, running on a 2015 MacBook Pro, for gradient calculation and optimization. All planners optimize for a horizon $T = 5$ using 20 vanilla gradient descent steps. For switching down, we used a cooldown of $K = 3$. In Eq. (2), the quadratic approximation may be ill conditioned, so we restricted Δu_R to exist within some reasonable bounds.

D. Evaluation Strategy

For every scenario, we run every method against the same simulated human driver. For diversity, we vary the starting positions of the human and robot cars across 30 different seeds per scenario. For every time step, we keep track of the reward and computation time.

In Fig. 2 we visualize average combined planning and decision time for the MS as it progresses through each scenario. Further, we juxtapose a conservative designer (yellow), who values reward relatively more, with an aggressive one (blue), who prefers computational savings. Snapshots below provide context for key points of the experiment denoted by dashed lines above, taken from a single conservative MS run.

In Fig. 3, we showcase the average reward and computational time per episode, averaged across seeds. In each plot, the left bars represent computational time, and the right ones reward, in units given by the left and right axes respectively. Within the total computational time, we separate planning and time deciding a human model.

We hypothesized that our MS algorithm would maintain a reward similar to that of the top model, while being significantly cheaper to compute. Additionally, we expected to see that conservative switchers obtain better rewards than aggressive switchers, but at higher computational complexity.

Note that because we wanted to showcase regions where a conservative switcher would react differently from an aggressive one, the hyperparameter λ varies widely across our scenarios. This is a reflection of differing reward scales, and in general the designer would have to select an appropriate λ depending on the problem and desired reward-computation tradeoff. For greater stability and generalization, one may find a highly conservative λ to be effective, reducing switching sensitivity to situations where the human has little bearing on the reward.

E. Scenario 1: Stay Back

In our first scenario in Fig. 2 (left), the robot and human begin driving alongside each other at the same speed. Ahead a series of cones creates a bottleneck in the road. Either the robot or human must yield to avoid a collision.

For this scenario, we restrict ourselves to Naive and Turn for simplicity, while the other two will showcase the potential of a broader model ladder. As shown in Fig. 3 (left), the Naive model struggles to correctly anticipate whether the human will go first and act accordingly, but it is much cheaper than Turn which safely navigates the scenario. Meanwhile, both aggressive or conservative switchers manages to obtain rewards close to that of the Turn, but with computational time closer to Naive. Additionally, notice that the decision time doesn't add excessive overhead, which underlines the efficiency of our approximation to the meta MDP.

In Fig. 2 (left), we see that a conservative switcher ($\lambda = 0.4$) will generally use Turn before and during the bottleneck, whereas an aggressive one ($\lambda = 5.0$) uses Turn only to intervene when using Naive is headed towards a collision.

F. Scenario 2: Merger

In the next scenario shown in Fig. 2 (middle), the robot would like to merge into the left lane. The gap is too small for the robot, but if it gradually angles its hood the target human will yield allowing the robot to enter.

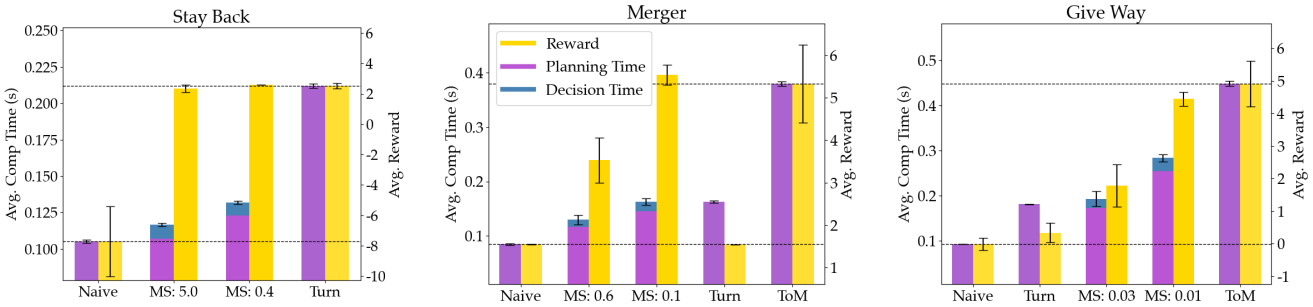

Fig. 3: Average computational time and reward for the 3 scenarios, with models ordered by computation. Model switchers achieve comparable performance to the best model with less computation. Note: the aggressive switchers with greater values for λ are to the left of the conservative ones.

As shown in Fig. 3 (middle), only ToM is able to anticipate that the human will yield should it begin entering the lane. However, before and after merging, ToM provides no further advantage, so the switcher can exploit that to reduce computational complexity.

In Fig. 3 (middle), we see that both the conservative and aggressive switchers obtain rewards closer to ToM, but with significantly less total computation. In Fig. 2 (middle), we see that the model switcher only uses ToM for merging, as it provides no comparative advantage afterwards. A conservative model switcher ($\lambda = 0.1$) switches up earlier merging faster, whereas an aggressive one ($\lambda = 0.6$) switches later but requires less computation. Delaying an inevitable switch hurts the aggressive MS, highlighting the importance of a conservative λ for safety-critical applications Turn is used sparingly during the transition as it provides little comparative advantage to Naive.

G. Scenario 3: Give Way

In our last scenario shown in Fig. 2 (right), the person is driving alongside the robot and would like to enter the robot's lane; however, other drivers around the robot do not allow enough space either way. The robot would like to help the human enter. Of course, the robot may create sufficient space by moving forward or backing up.

As shown in Fig. 3 (right), ToM is the only one capable of understanding the robot's ability to help the person to merge. Turn occasionally succeeds when it yields fearing a collision. Both switchers obtain higher rewards than those of the cheaper models, but we notice again a large gap between the two.

The earlier the robot makes space for the person, the better. A conservative model switcher ($\lambda = 0.01$) quickly switches up and allows the human to enter, albeit slower than pure ToM. A more aggressive switcher ($\lambda = 0.03$) delays the switch to ToM resulting in lesser reward, but keeps overall computation lower. The delay between yielding and the human entering presents a challenge for our myopic, one time step, reward simplification. A very low λ works here, but the issue of myopic gain potentially underestimating longer horizon gain remains.

IV. Discussion

Summary: In this paper, we formalized the robot's decision making process over which predictive human model to use as a meta MDP. We introduced an approximate solution that enables efficient switching to the most suitable available model within this MDP. The resulting decisions maintain rewards similar to those of the best model available, while dramatically reducing computational time.

Future Work: Because the robot cannot see the human's true future controls, we were limited to basing switching decisions on what the person did in the past. We could approximate the true human trajectory using the top model's prediction, but that would relinquish most computational savings. For example, Naive planning and Turn prediction takes as long as Turn planning. Additionally, because the decision to switch relies on a single time step simplification, our scenarios needed consistent reward signal. Future work must address adapting our algorithm to sparse reward settings where one step reward gradients are not meaningful. Learning a value function to replace reward in our formulation would be an interesting direction.

Moreover, all this work happened in a simple driving simulator, albeit with what we think are complex scenarios. To put this on the road, we will need more emphasis on safety, as well as longer decision horizons. Lastly, our algorithm focuses on single human decisions. We could run our method separately for every nearby human, evaluating the differential benefit of switching each, but we are yet to conduct experiments in that setting. Alternatively, we can imagine adapting it to multiple humans by either adding more complex multi-player game theoretic models, or combining it with the prioritization schema presented by [26].

Conclusion: Despite these limitations, we are encouraged to see robots have more autonomy over what human models to use when planning online, without hand-coded heuristics. We look forward to applications of our model switching ideas beyond autonomous driving: to mobile robots, quadcopters, or any human-robot interactive scenarios where planning with multiple predictive human models might be beneficial.

References

[1] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: a survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020. [Online]. Available: https://doi.org/10.1177/0278364920917446

[2] A. Barth and U. Franke, "Where will the oncoming vehicle be the next second?" in *2008 IEEE Intelligent Vehicles Symposium*, 2008, pp. 1068–1073.

[3] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," in *2008 11th International Conference on Information Fusion*, 2008, pp. 1–6.

[4] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, pp. 4282–4286, May 1995. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.51.4282

[5] S. Lefevre, A. Carvalho, Y. Gao, E. Tseng, and F. Borrelli, "Driver models for personalized driving assistance," *Vehicle System Dynamics*, vol. 53, 07 2015.

[6] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 961–971.

[7] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3399–3406.

[8] M. Pfeiffer, U. Schwesinger, H. Sommer, E. Galceran, and R. Siegwart, "Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2096–2101, 2016.

[9] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, ser. IROS'09. IEEE Press, 2009, p. 3931–3936.

[10] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016. [Online]. Available: https://doi.org/10.1177/0278364915619772

[11] H. Kretzschmar, M. Kuderer, and W. Burgard, "Learning to predict trajectories of cooperatively navigating agents," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4015–4020.

[12] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," in *Robotics: Science and Systems*, 2012.

[13] A. Bobu, D. R. R. Scobee, J. F. Fisac, S. S. Sastry, and A. D. Dragan, "Less is more: Rethinking probabilistic models of human behavior," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 429–437. [Online]. Available: https://doi.org/10.1145/3319502.3374811

[14] C. Liu, J. B. Hamrick, J. F. Fisac, A. D. Dragan, J. K. Hedrick, S. S. Sastry, and T. L. Griffiths, "Goal inference improves objective and perceived performance in human-robot collaboration," *CoRR*, vol. abs/1802.01780, 2018. [Online]. Available: http://arxiv.org/abs/1802.01780

[15] J. Mainprice, R. Hayne, and D. Berenson, "Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 885–892, 2015.

[16] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 201–214.

[17] Z. Sunberg, C. J. Ho, and M. J. Kochenderfer, "The value of inferring the internal state of traffic participants for autonomous freeway driving," *2017 American Control Conference (ACC)*, pp. 3004–3010, 2017.

[18] D. Sadigh, S. Sastry, S. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Robotics: Science and Systems*, 2016.

[19] J. F. Fisac, M. A. Gates, J. B. Hamrick, C. Liu, D. Hadfield-Menell, M. Palaniappan, D. Malik, S. Sastry, T. Griffiths, and A. D. Dragan, "Pragmatic-pedagogic value alignment," in *ISRR*, 2017.

[20] D. S. Brown and S. Niekum, "Machine teaching for inverse reinforcement learning: Algorithms and applications," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 7749–7758.

[21] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. Tomlin, "Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1475–1481, 2020.

[22] X. Rong Li and V. P. Jilkov, "Survey of maneuvering target tracking. part v. multiple-model methods," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1255–1321, 2005.

[23] E. A. I. Pool, J. F. P. Kooij, and D. Gavrila, "Using road topology to improve cyclist path prediction," *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 289–296, 2017.

[24] A. Pentland and A. Liu, "Modeling and prediction of human behavior," *Neural Comput.*, vol. 11, no. 1, p. 229–242, Jan. 1999. [Online]. Available: https://doi.org/10.1162/089976699300016890

[25] P. A. Lasota and J. A. Shah, "A multiple-predictor approach to human motion prediction," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2300–2307.

[26] K. S. Refaat, K. Ding, N. Ponomareva, and S. Ross, "Agent prioritization for autonomous navigation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3-8, 2019*. IEEE, 2019, pp. 2060–2067. [Online]. Available: https://doi.org/10.1109/IROS40897.2019.8967743

[27] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335 – 348, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109889900022

[28] A. Gopnik and H. M. Wellman, *The theory theory*. Cambridge University Press, 1994, p. 257–293.

[29] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, ser. AAAI'08. AAAI Press, 2008, pp. 1433–1438. [Online]. Available: http://dl.acm.org/citation.cfm?id=1620270.1620297

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/