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).
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 ability to make real-time, in an environment with other human agents. We use the running example of an autonomous car sharing the road with a human driver to illustrate the capability of our theory for a general single human, single robot setting.

A. Problem Statement

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 $\pi_H$ (that would require access to the human’s brain), it uses a human model $M : \mathcal{X} \times \mathcal{U}_R^N \rightarrow \mathcal{U}_H^N$ to make a prediction of the human controls $u_H$. The robot then seeks a plan $\bar{u}_R^0(M) := \bar{u}_R(x^0, M)$ that maximizes the MDP reward $R_R$ based on the predicted $\bar{u}_H^0(M, u_R, M) := \bar{u}_H(x^0, u_R, M)$.

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 $\bar{u}_R^t(M)$. 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:

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

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 $\bar{u}_R^0(M)[0]$.

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 $M = \{M_0,\ldots, M_n\}$, where as we climb the ladder we sacrifice computational time for greater expected reward: $E_x[R_R(x, \bar{u}_R(M_1), u_H)] \geq E_x[R_R(x, \bar{u}_R(M_i), u_H)]$ and $T(M_i) > T(M_j), \forall i < j$, 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 $\bar{u}_R^t$. 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 $r^t_{meta} = r_R(x^t, \bar{u}_R^t(M^t), u_H^t) - \lambda \cdot T(M^t)$, with $\lambda$ trading off actual reward gained by planning under $M^t$ and computational time spent on the plan. Lower values for $\lambda$ 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 $u_H$ 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 $r^t_{meta}$ 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
the first control to that plan. Given myopic rewards \( M \), we 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 \( \mathbf{u}_h^t(M_2) \) by approximating it as \( \mathbf{u}_h^t(M_2) = \mathbf{u}_h^t(M_1) + \Delta \mathbf{u}_R \). Here, we want to choose \( \Delta \mathbf{u}_R \) such that it maximizes the robot reward \( R_R \) under the complex model \( M_2 \). However, optimizing \( \Delta \mathbf{u}_R \) using Eq. (1) is equivalent to planning. To get an efficient estimate, we use a quadratic Taylor series approximation of \( R_R \), denoted \( \hat{R}_R \), evaluated around \((x^t, \mathbf{u}_R^t(M_1), \mathbf{u}_H^t(\mathbf{u}_R^t(M_1), M_1))\), our current plan and human prediction:

\[
\Delta \mathbf{u}_R = \arg \max_{\Delta \mathbf{u}_R} \hat{R}_R(x^t, \mathbf{u}_R^t(M_2), \mathbf{u}_H^t(\mathbf{u}_R^t(M_2), M_2)) .
\]  

(2)

#### Estimate Change in Human Prediction

Eq. (2) requires approximating the \( M_2 \) human’s response \( \mathbf{u}_h^t(M_2) \) to the robot’s plan \( \mathbf{u}_R^t(M_2) \), which can be broken down into \( M_2 \)’s response to the current plan \( \mathbf{u}_R^t(M_1) \) plus the change coming from \( \Delta \mathbf{u}_R \): \( \mathbf{u}_h^t(\mathbf{u}_R^t(M_2), M_2) = \mathbf{u}_h^t(\mathbf{u}_R^t(M_1) + \Delta \mathbf{u}_R, M_2) \). We can linearize around this:

\[
\mathbf{u}_H^t(\mathbf{u}_R^t(M_2), M_2) = \mathbf{u}_H^t(\mathbf{u}_R^t(M_1), M_2) + \frac{d \mathbf{u}_H}{d \mathbf{u}_R} \cdot \Delta \mathbf{u}_R ,
\]  

(3)

where \( \frac{d \mathbf{u}_H}{d \mathbf{u}_R} = \frac{d \mathbf{u}_H(\mathbf{u}_R, M)}{d \mathbf{u}_R} \bigg|_{\mathbf{u}_R = \mathbf{u}_R(M_1)} \). 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 \( \mathbf{u}_R^t(M_2) \) and the changed human prediction from Eq. (3) into our reward expression we maximize in Eq. (2):

\[
\hat{R}_R(x^t, \mathbf{u}_R^t(M_1) + \Delta \mathbf{u}_R, \mathbf{u}_H^t(\mathbf{u}_R^t(M_1), M_2) + \frac{d \mathbf{u}_H}{d \mathbf{u}_R} \cdot \Delta \mathbf{u}_R) .
\]  

(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 \( r_{\text{meta}}^t \) under \( M_2 \), \( r_R(x^t, \mathbf{u}_R^t(M_2), \mathbf{u}_H^t(\mathbf{u}_R^t(M_2), M_2)) \), where \( \mathbf{u}_H^t(\mathbf{u}_R^t(M_2), M_2) \) would be the human’s response under the true model \( M_H \) to the robot’s plan with \( M_2 \), and \( \mathbf{u}_H^t(\mathbf{u}_R^t(M_2), M_2) \) is the first control of that response.

Since our approximation of the meta MDP considers myopic rewards \( r_R \), we are only interested in the first control \( \mathbf{u}_R(M_2) \) of the changed plan \( \mathbf{u}_R^t(M_2) \), and the first control \( \mathbf{u}_H^t(\mathbf{u}_R^t(M_2), M_2) \) of the true human’s response to that plan. Given \( \Delta \mathbf{u}_R \), we only need the change for the first control \( \Delta \mathbf{u}_R \) combined with the approximation of \( \mathbf{u}_H^t(\mathbf{u}_R^t(M_2), M_2) \) as in Eq. (5) to estimate \( \hat{r}_R^t(M_2) \) as:

\[
r_R(x^t, \mathbf{u}_R(M_1) + \Delta \mathbf{u}_R, \mathbf{u}_H^t(\mathbf{u}_R^t(M_1), M_2) + \frac{d \mathbf{u}_H}{d \mathbf{u}_R} \cdot \Delta \mathbf{u}_R ,
\]  

(5)

where \( \frac{d \mathbf{u}_H}{d \mathbf{u}_R} = \frac{d \mathbf{u}_H(\mathbf{u}_R, M)}{d \mathbf{u}_R} \bigg|_{\mathbf{u}_R = \mathbf{u}_R(M_1)} \). 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 \( \mathbf{u}_R^t(M_1) \), 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 \( \mathbf{u}_R^t(M_2) = \mathbf{u}_R(M_1) + \Delta \mathbf{u}_R \), with its corresponding changed human control prediction from Eq. (3):

\[
\hat{u}_H^t(\mathbf{u}_R^t(M_2), M_2) = \hat{u}_H^t(\mathbf{u}_R(M_1), M_2) + \frac{d \mathbf{u}_H}{d \mathbf{u}_R} \cdot \Delta \mathbf{u}_R ,
\]  

(6)

We obtain \( \Delta \mathbf{u}_R \) by optimizing the following objective:

\[
\arg \max_{\Delta \mathbf{u}_R} \hat{R}_R(x^t, \mathbf{u}_R^t(M_2), \hat{u}_H^t(\mathbf{u}_R(M_1), M_2) + \frac{d \mathbf{u}_H}{d \mathbf{u}_R} \cdot \Delta \mathbf{u}_R) .
\]  

(7)

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

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

\[
\Delta r_{\text{meta}}^t = \hat{r}_R^t(M_2) - \lambda \cdot T(M_1) - (r_R^t(M_1) - \lambda \cdot T(M_1)) ,
\]  

(8)

where we already know \( r_R^t(M_1) \) from the previous time step, and \( T(M_1) \) and \( T(M_2) \) are known a priori from their model specifications. If \( \Delta r_{\text{meta}}^t \) 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 \( r_R^t(M_1) \) to \( \hat{r}_R^t(M_2) \) to \( r_R^t(M_2) \) to \( \hat{r}_R^t(M_2) \). 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 \( \hat{u}_H^t(\mathbf{u}_R^t(M_2), M_2) \) than \( M_1 \)'s \( \hat{u}_H^t(\mathbf{u}_R^t(M_1), M_1) \) 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 \( \frac{d \mathbf{u}_H}{d \mathbf{u}_R} \) 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 \( \Delta \mathbf{u}_R \) 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 \( \tilde{u}_R^t(M_i) \) given \( x^t \) and execute \( \tilde{u}_R^t(M_i) \).

if \( i < n \) then

Substitute \((M_1, M_2) \leftarrow (M_i, M_n)\) in Eq. (5), (7), (8), with \( \tilde{u}_R^t(M_1, M_n) \leftarrow \tilde{u}_R^t \).

Compute \( \Delta r_{\text{meta}}^t \) using Eq. (5), (7), (8).

if \( \Delta r_{\text{meta}}^t > 0 \) then

\( i \leftarrow i + 1 \) (Switch Up).

else

\( i \leftarrow i - 1 \) (Switch Down).

end if

end if

end while

end Algorithm 1:

that either captures influence or makes different predictions affects the robot’s plan. If neither bears much weight, then \( \Delta \bar{u}_R \) will likely be negligible and we will not switch. If however \( \Delta \bar{u}_R \) 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 \( M_j \) is the current model and \( M_j \) 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 \( \tilde{u}^t_R(M_i) \). 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 \( r^i_H(M_j) \) with \( r^i_H(M_n) \), the estimated reward from using the best model available. To avoid having to evaluate \( M_n \)’s expensive predictions we substitute the exact prediction \( \tilde{u}^t_H \). 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 \( r^i_H(M_j) \leq r^i_H(M_{i-1}) \leq r^i_H(M_i) \). 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 \( \Delta r_{\text{meta}}^t \) is positive for \( M_H \), the robot switches up to \( M_j \). Otherwise, if \( \Delta r_{\text{meta}}^t \) 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 \( x = [x, y, \theta, v] \), where \( x, y \) are the coordinates of the vehicle, \( \theta \) is the heading, and \( v \) is the speed. The actions are \( u = [\omega, a]^T \), where \( \omega \) is the steering input and \( a \) is the linear acceleration. We use \( \alpha \) as the fricion coefficient, and the vehicle’s dynamics model is:

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta} \\
\dot{v}
\end{bmatrix} =
\begin{bmatrix}
\cos(\theta) & \sin(\theta) & 0 & 0 \\
\sin(\theta) & \cos(\theta) & 0 & 0 \\
\omega & 0 & 0 & 1 \\
a & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
\theta \\
v
\end{bmatrix}
+ \begin{bmatrix}
0 \\
0 \\
\alpha \cdot v - a & 1
\end{bmatrix}
.
\]

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 \( \phi: R_H(x^0, u_R, u_H) = \sum_{\tau=1}^{N} \theta^T \phi(x^\tau, u^\tau_R, u_H^\tau) \). 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 \( \tilde{u}_R \) to obtain a plan \( \tilde{u}_H \):

\[
u_H(x^0, \tilde{u}_R) = \arg \max_{u_H} R_H(x^0, \tilde{u}_R, u_H) .\]

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 \( \theta \) and a set of features \( \phi \) for the human reward. To learn a good \( \theta \) 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 $\lambda$, switcher (yellow) and an aggressive, larger $\lambda$, 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 $\lambda$.

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 $\Delta u_R$ 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. 2 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.

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.
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 $\lambda$ 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 $\lambda$ 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.
