Integrating Planning and Interpretable Goal Recognition for Autonomous Driving

Stefano V. Albrecht*†, Cillian Brewitt*†, John Wilhelm*†,
Francisco Eiras*, Mihai Dobre*, Subramanian Ramamoorthy*†
*FiveAI Ltd., UK, {firstname.lastname}@five.ai
†School of Informatics, University of Edinburgh, UK

Abstract—The ability to predict the intentions and driving trajectories of other vehicles is a key problem for autonomous driving. We propose an integrated planning and prediction system which leverages the computational benefit of using a finite space of maneuvers, and extend the approach to planning and prediction of *sequences* (plans) of maneuvers via rational inverse planning to recognise the *goals* of other vehicles. Goal recognition informs a Monte Carlo Tree Search (MCTS) algorithm to plan optimal maneuvers for the ego vehicle. Inverse planning and MCTS utilise a shared set of defined maneuvers to construct plans which are explainable by means of *rationality*, i.e. plans are optimal in given metrics. Evaluation in simulations of four urban driving scenarios demonstrate the system's ability to robustly recognise the goals of other vehicles while generating near-optimal plans. In each scenario we extract intuitive explanations for the recognised goals and maneuver predictions which justify the system's decisions.

I. INTRODUCTION

The ability to predict the intentions and driving trajectories of other vehicles is a key problem for autonomous driving [22]. This problem is significantly complicated by the requirement to make fast and accurate predictions based on limited observation data which originate from a dynamically evolving environment with coupled multi-agent interactions.

A common approach to make prediction tractable in such conditions is to assume that agents use one of a finite number of distinct behaviours [1, 2]. This has become a standard approach in autonomous driving [8, 9, 14, 15, 24, 31], in which the behaviours typically represent high-level maneuvers such as lane-follow, lane-change, turn, stop, etc. A classifier of some type is used to detect a vehicle's current executed maneuver based on its observed driving trajectory, which is then used to predict future trajectories to inform the planning of the ego vehicle under control. The limitation in such methods is that they only detect the *current* maneuver of other vehicles, hence planners using such predictions are essentially limited to the timescales of the detected maneuvers.

An alternative approach is to specify a finite set of possible *goals* for each other vehicle (such as the various road exit points) and to plan a full trajectory to each goal from the vehicle's observed local state [5, 11]. While this approach can
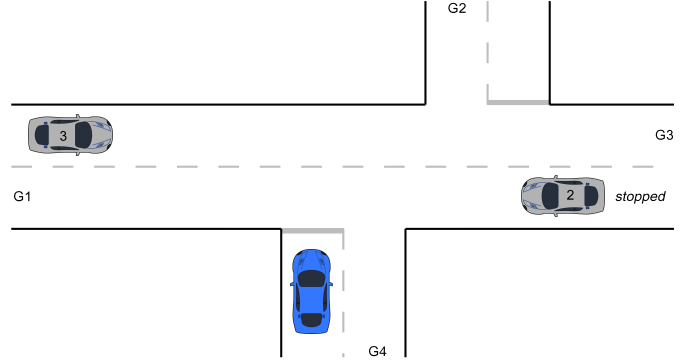
Fig. 1: Ego vehicle shown in blue, its goal is to reach G1. Car 3 is driving east at constant speed. Car 2 slowed down until reaching its shown stopping location. Ego vehicle may infer that Car 2 is stopping because it intends to reach G2 but must wait until Car 3 has passed, then turn toward G2. This provides an opportunity for the ego vehicle to turn onto the road while Car 2 is waiting.

generate longer-term predictions, it is limited by the fact that the generated trajectories must be relatively closely followed by a vehicle to yield high-confidence predictions.

Recent methods based on deep learning have shown promising results for trajectory prediction in autonomous driving [7, 18, 21, 29, 30]. Prediction models are trained on large datasets that are becoming available through data gathering campaigns involving sensorised vehicles traversing city roads. The resulting sensor information (typically various forms of visual and LIDAR feeds, sometimes including also RADAR) are used to produce forecasts through models ranging from video prediction to more contextualised forecasts taking into account environmental cues specifically. Reliable prediction over several second horizons remains a hard problem, in part due to the difficulties in capturing the coupled evolution of traffic. Recent work such as [21] begin to address this although much remains to be done towards addressing scenarios such as the ones we consider in this paper. Another important limitation in some of these methods is the difficulty of incorporating hard constraints in the optimisation objective which must be differentiable (e.g. to avoid nonsensical predictions for a given situation). Finally, in our view one of the most significant limitations of this class of methods is the difficulty in extracting interpretable predictions in a form that is amenable to efficient integration with planning methods that effectively represent

multi-dimensional and hierarchical task objectives.

Our starting point is that in order to predict the future maneuvers of a vehicle, we must reason about *why* – that is, to what end – the vehicle performed its current and past maneuvers, which will yield clues as to its intended goal. Knowledge of the goals of other vehicles enables prediction of their future maneuvers and trajectories in relation to their goals. Such long-term predictions facilitate planning over extended timescales to realise opportunities which might not otherwise present themselves, as illustrated in example in Figure 1. To the extent that our predictions are structured around the interpretation of observed trajectories in terms of high-level maneuvers, the goal recognition process lends itself to *intuitive interpretation* for the purposes of debugging, at a level of detail suggested in Figure 1. Ultimately, as we develop towards making our autonomous systems more trustworthy, these notions of interpretation and the ability to justify (explain) the system's decisions is key. We map this in our specific work to goal-based predictions and how these predictions influence decisions.

To this end, we propose an integrated planning and prediction system which leverages the computational advantages of using a finite space of maneuvers (which is expressive enough for the purposes of *interpreting* observed behaviours), but extends the approach to planning and prediction of *sequences* (i.e., plans) of maneuvers. We achieve this via a novel integration of rational inverse planning [4, 20] to recognise the goals of other vehicles, with Monte Carlo Tree Search (MCTS) [6] to plan optimal maneuvers for the ego vehicle. Inverse planning and MCTS utilise a shared set of defined maneuvers to construct plans which are explainable by means of *rationality* principles, i.e. that plans are optimal with respect to given metrics. Rather than matching plans directly as in prior work [5, 11], our approach instead evaluates the extent to which an observed trajectory is rational for a given goal, providing robustness with respect to variability in trajectories. Similarly to MPDM [9], our MCTS algorithm performs closed-loop forward-simulations of vehicle dynamics and their coupled interactions, but by separating control from maneuvers we are able to optimise the velocity profile across maneuvers and can leverage simplified open-loop control to increase efficiency of inverse planning.

We evaluate our system in simulations of four urban driving scenarios, including the scenario in Figure 1, roundabout entry, and dense lane merging, showing that the system robustly recognises the goals of other vehicles and generates near-optimal plans under diverse scenario initialisation. We extract intuitive explanations for the recognised goals and maneuver predictions in each scenario which justify the system's decisions.

II. PRELIMINARIES AND PROBLEM DEFINITION

Let $\mathcal{I}$ be the set of vehicles interacting with the ego vehicle in a local neighbourhood (including the ego vehicle). At time t, each vehicle $i \in \mathcal{I}$ is in a local state $s_t^i \in \mathcal{S}^i$, receives a local observation $o_t^i \in \mathcal{O}^i$, and can choose an action $a_t^i \in \mathcal{A}^i$. We write $s_t \in \mathcal{S} = \times_i \mathcal{S}^i$ for the joint state and $s_{a:b}$ for the tuple $(s_a, ..., s_b)$, and similarly for $o_t \in \mathcal{O}, a_t \in \mathcal{A}$. Observations depend on the joint state via $p(o_t^i | s_t)$, and actions depend on the

observations via $p(a_t^i | o_{1:t}^i)$. In our system, a local state contains a vehicle's pose, velocity, and acceleration (throughout the paper we use velocity and speed interchangeably.); an observation contains the poses and velocities of nearby vehicles; and an action controls the vehicle's steering and acceleration.

The probability of a sequence of joint states $s_{1:n}$, $n \geq 1$, is given by

$$p(s_{1:n}) = \prod_{t=1}^{n-1} \int_{\mathcal{O}} \int_{\mathcal{A}} p(o_t|s_t)p(a_t|o_{1:t})p(s_{t+1}|s_t, a_t) \, do_t \, da_t$$

(1)

where $p(s_{t+1}|s_t, a_t)$ defines the joint vehicle dynamics, and we assume independent local observations and actions, $p(o_t|s_t) = \prod_i p(o_t^i|s_t)$ and $p(a_t|o_{1:t}) = \prod_i p(a_t^i|o_{1:t}^i)$. Vehicles react to other vehicles via their local observations $o_{1:n}^i$.

We define the planning problem as finding an optimal policy π^* which selects the actions for the ego vehicle, ε, to achieve a specified goal, g^ε, while optimising the driving trajectory via a defined reward function. Here, a policy is a function $\pi : (\mathcal{O}^\varepsilon)^* \mapsto \mathcal{A}^\varepsilon$ which maps an observation sequence $o_{1:n}^\varepsilon$ to an action a_t^ε. A goal can be any (partial) state description $g^\varepsilon \subset \mathcal{S}^\varepsilon$, but in this paper we focus on goals that specify target locations. Formally, define

$$\Omega_n = \left\{ s_{1:n} \,\middle|\, s_n^\varepsilon \subseteq g^\varepsilon \wedge \nexists m < n : s_m^\varepsilon \subseteq g^\varepsilon \right\} \quad (2)$$

where $s_n^\varepsilon \subseteq g^\varepsilon$ means that s_n^ε satisfies g^ε. The second condition in (2) ensures that $\sum_{n=1}^{\infty} \int_{\Omega_n} p(s_{1:n})ds_{1:n} \leq 1$ for any policy π, which is needed for soundness of the sum in (3). The problem is to find π^* such that

$$\pi^* \in \arg \max_\pi \sum_{n=1}^{\infty} \int_{\Omega_n} p(s_{1:n})R^\varepsilon(s_{1:n}) \, ds_{1:n} \quad (3)$$

where $R^i(s_{1:n})$ is the reward of $s_{1:n}$ for vehicle i (see Sec. III-E). Intuitively, maximising (3) entails optimising the probability of achieving the goal and the rewards of the generated trajectories. In practice, we approximate (3) using a finite planning horizon (detailed in Sec.III-G).

While our experiments in Section IV use scenarios with fixed goals, the above problem formulation allows our method to drive through full routes with the route planner module continually updating the goal location g^ε of the ego vehicle depending on its current location on the route.

III. METHOD

A. System Overview

Our general approach relies on two assumptions: (1) each vehicle seeks to reach some (unknown) goal location from a set of possible goals, and (2) each vehicle follows a plan generated from a finite library of defined maneuvers.

Our proposed system approximates the optimal policy π^* as follows: For each other vehicle, enumerate its possible goals and inversely plan for that vehicle to each goal, giving the probabilities and predicted trajectories to the goals. The resulting goal probabilities and trajectories inform the simulation process of a Monte Carlo Tree Search (MCTS) algorithm to generate

Macro action:	Additional applicability condition:	Maneuver sequence (maneuver parameters in brackets):
Continue		*lane-follow* (end of visible lane)
Continue to next exit	Must be in roundabout and not in outer-lane	*lane-follow* (next exit point)
Change left/right	There is a lane to the left/right	*lane-follow* (until target lane clear), *lane-change-left/right*
Exit left/right	Exit point on same lane ahead of car and in correct direction	*lane-follow* (exit point), *give-way* (relevant lanes), *turn-left/right*
Stop	There is a stopping goal ahead of the car on the current lane	*lane-follow* (close to stopping point), *stop*

TABLE I: Macro actions used in our system. Each macro action concatenates one or more maneuvers and sets their parameters (see Sec.III-C).
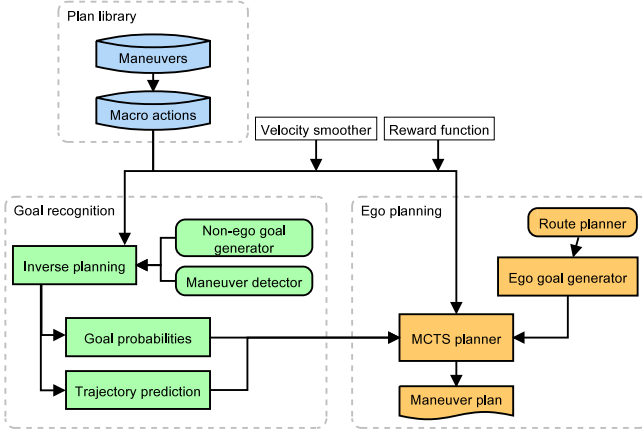

Fig. 2: System overview.

an optimal maneuver plan for the ego vehicle. In order to keep the required search depth shallow and hence efficient, both inverse planning and MCTS plan over macro actions which flexibly concatenate the maneuvers using context information.

Figure 2 provides an overview of the components in the proposed system. The following sections will detail each of the components.

B. Maneuvers

We assume that at any time, each vehicle is executing one of a finite number of maneuvers. Our system uses the following maneuvers: *lane-follow, lane-change-left/right, turn-left/right, give-way, stop.*

Each maneuver ω specifies applicability and termination conditions. A maneuver is available in a given state if and only if the state satisfies the maneuver's applicability condition. For example, lane-change-left is only applicable if there is a lane in same driving direction on the left of the vehicle (one could also check for space constraints). The maneuver terminates if the state satisfies the termination condition.

If applicable, a maneuver specifies a local trajectory $\hat{s}^i_{1:n}$ to be followed by the vehicle, which includes a reference path in the global coordinate frame and target velocities along the path. For convenience in exposition, we assume that $\hat{s}^i$ uses the same representation and indexing as s^i, but in general this does not have to be the case (for example, $\hat{s}$ may be indexed by longitudinal position rather than time, which can be interpolated to time indices). In our system, the reference path is generated via a Bezier spline function fitted to a set of points extracted from the road topology, and target velocities are set using domain heuristics similar to [10]. As a general principle, we assume that vehicles will attempt to drive at local speed limit

when possible. This target is reduced to the velocity of the slower vehicle in front on same lane (if any) or as a function of local curvature on the driving path.

The *give-way* maneuver slows the vehicle down to some specified velocity while driving toward a given location, usually a crossing or turning point where oncoming traffic has priority. At the location, the maneuver terminates if the specified lanes are clear (allowing the vehicle to proceed with next maneuver without fully stopping), otherwise it fully stops the vehicle and then terminates once the specified lanes are clear. When used as part of our system, we also allow *give-way* to terminate early if our system predicts safe entry (cf. Sec. IV).

The *lane-follow* and *give-way* maneuvers have open parameters in their termination conditions: for *lane-follow*, the parameter specifies the driving distance; for *give-way*, the parameter specifies lanes which must be monitored for oncoming traffic. Such open parameters are automatically set by macro actions, which we define in next section.

In contrast to methods such as MPDM [9], our maneuvers are separated from control. This separation serves two purposes: First, when constructing sequences of maneuvers, each maneuver must optimise its velocity profile in anticipation of subsequent maneuvers. This is not possible if control is integrated into independent policies, as in MPDM. Having access to trajectories allows us to perform a velocity smoothing operation across the concatenation of the trajectories, which we describe in Section III-D. Second, maneuvers can be simulated under different control models (open-loop, closed-loop) in inverse planning and MCTS, which offer different tradeoffs between prediction accuracy and simulation efficiency.

C. Macro Actions

Macro actions concatenate one or more maneuvers in a flexible way. Using macro actions relieves the planner in two important ways: they specify common sequences of maneuvers, and they automatically set the free parameters in maneuvers based on context information (usually road layout). Table I defines the macro actions used in our system.

The applicability condition of a macro action is given by the applicability condition of the first maneuver in the macro action as well as optional additional conditions (see Table I). The termination condition of a macro action is given by the termination condition of the last maneuver in the macro action.

Note that macro actions as used in this work do not define a hierarchy of decomposable actions; they simply define sequences of actions [13].

Both inverse planning (Sec. III-F) and MCTS (Sec. III-G) search over macro actions rather than maneuvers, which

significantly increases their efficiency by reducing search depth.

D. Velocity Smoothing

To accommodate natural variations in driving behaviours and to obtain a feasible trajectory across maneuvers for a vehicle i, we define a velocity smoothing operation which optimises the target velocities in a given trajectory $\hat{s}_{1:n}^i$. Let $\hat{x}_t$ be the longitudinal position on the reference path at $\hat{s}_t^i$ and $\hat{v}_t$ its corresponding target velocity, for $1 \leq t \leq n$. We define $\kappa : x \rightarrow v$ as the piecewise linear interpolation of target velocities between points $\hat{x}_t$. Given the time elapsed between two time steps, Δt; the maximum velocity and acceleration, v_{max}/a_{max}; and setting $x_1 = \hat{x}_1, v_1 = \hat{v}_1$, we define the smoothing problem as

$$
\min_{x_{2:n}, v_{2:n}} \quad \sum_{t=1}^{n} ||v_t - \kappa(x_t)||_2 + \lambda \sum_{t=1}^{n-1} ||v_{t+1} - v_t||_2
$$
$$
x_{t+1} = x_t + v_t \Delta t
$$
$$
0 < v_t < v_{\max} \tag{4}
$$
$$
|v_{t+1} - v_t| < a_{\max} \Delta t
$$
$$
v_t \leq \kappa(x_t)
$$

where $\lambda > 0$ is the weight given to the acceleration part of the optimisation objective. The last constraint to treat target velocities as upper bounds results in more realistic braking behaviours in our system.

Equation (4) is a nonlinear non-convex optimisation problem which can be solved, e.g., using a primal-dual interior point method (we use IPOPT [27]). From the solution of the problem, $(x_{2:n}, v_{2:n})$, we can interpolate to obtain the achievable velocities at the original points $\hat{x}_t$. If $\hat{x}_t \leq x_n$ for all t, then we can interpolate from this solution for $\hat{x}_{1:n}$. Otherwise, we can solve a similar problem starting from x_n, and repeat the procedure until all $\hat{x}_{1:t}$ are within the bound.

Velocity smoothing should respect zero-velocities in the input trajectory, which indicate full stops. A simple way of achieving this is to split a trajectory into segments separated by stopping events, and to apply the smoothing function to each segment.

E. Reward Function

We define the reward of a trajectory $s_{1:n}$ for vehicle i as a weighted sum of K reward components,

$$
R^i(s_{1:n}) = \sum_{k=1}^{K} w_k R_k^i(s_{1:n}) \tag{5}
$$

with weights $w_k > 0$ and $R_k^i(s_{1:n}) > 0$. Our system includes reward components for execution time, longitudinal jerk, lateral jerk, path curvature, and safety distance to leading vehicle.

F. Goal Recognition

By assuming that each vehicle $i \in \mathcal{I}$ seeks to reach one of a finite number of possible goal locations $g^i \in \mathcal{G}^i$, using plans constructed from our defined macro actions, we can use the framework of rational inverse planning [4, 20] to compute a

---

Algorithm 1 Goal recognition algorithm

Input: vehicle i, current maneuver ω^i, observations $s_{1:t}$
Returns: goal probabilities $p(g^i|s_{1:t}, \omega^i)$

1: Generate possible goals $g^i \in \mathcal{G}^i$ from state s_t^i
2: Set prior probabilities $p(g^i)$ (e.g. uniform)
3: **for all** $g^i \in \mathcal{G}^i$ **do**
4: $\quad \hat{s}_{1:n}^i \leftarrow$ A*SEARCH(ω^i) from $\hat{s}_1^i = s_1^i$ to g^i
5: $\quad$ Apply velocity smoothing to $\hat{s}_{1:n}^i$
6: $\quad \hat{r} \leftarrow$ reward $R^i(\hat{s}_{1:n}^i)$
7: $\quad \bar{s}_{1:m}^i \leftarrow$ A*SEARCH(ω^i) from $\bar{s}_t^i$ to g^i, with $\bar{s}_{1:t}^i = s_{1:t}^i$
8: $\quad$ Apply velocity smoothing to $\bar{s}_{t+1:m}^i$
9: $\quad \bar{r} \leftarrow$ reward $R^i(\bar{s}_{1:m}^i)$
10: $\quad L(s_{1:t}|g^i) \leftarrow \exp(-\beta(\bar{r} - \hat{r}))$
11: Return $p(g^i|s_{1:t}) \propto L(s_{1:t}|g^i)\,p(g^i)$

Bayesian posterior distribution over vehicle i's goals at time t,

$$
p(g^i|s_{1:t}) \propto L(s_{1:t}|g^i)p(g^i) \tag{6}
$$

where $L(s_{1:t}|g^i)$ is the likelihood of i's observed trajectory given goal g^i, and $p(g^i)$ specifies the prior probability of g^i.

The likelihood is a function of the reward difference between two plans: the reward $\hat{r}$ of the optimal trajectory from i's initial observed state s_1^i to goal g^i after velocity smoothing, and the reward $\bar{r}$ of the trajectory which follows the observed trajectory until time t and then continues optimally to goal g^i, with smoothing applied only to the trajectory after t. Then, the likelihood is defined as

$$
L(s_{1:t}|g^i) = \exp(-\beta(\bar{r} - \hat{r})) \tag{7}
$$

where β is a scaling parameter (we use $\beta = 1$). This definition assumes that vehicles behave *rationally* by driving optimally to achieve goals, but allows for a degree of deviation. If a goal cannot be achieved, we set its probability to zero.

It is important to note that velocity smoothing cannot be applied to the part of the trajectory that has already been observed, i.e. $s_{1:t}$. Otherwise, the effect of velocity smoothing could be to wash out evidence that would hint at certain goals.

Algorithm 1 shows the pseudo code for our goal recognition algorithm, with further details in below subsections. The algorithm allows for efficient parallel processing by using parallel threads for each vehicle i, goal g^i, and rewards $\hat{r}, \bar{r}$.

1) Goal Generation: A heuristic function is used to generate the set of possible goals $\mathcal{G}^i$ for vehicle i based on its location and context information such as road layout and traffic rules. In our system, we include one goal for the end of the vehicle's current road and goals for end of each reachable connecting road, bounded by the ego vehicle's view region (as shown in Figure 1). We do not include infeasible goals, such as locations behind the vehicle.

In addition to static goals which depend only on the vehicle's location and road layout/rules, we may also add dynamic goals which depend on current traffic. For example, in the

dense merging scenario used in Section IV, stopping goals are dynamically added to model a vehicle's intention to allow the ego vehicle to merge in front of the vehicle.

2) **Maneuver Detection:** Maneuver detection is used to detect the current executed maneuver of a vehicle (at time t), allowing inverse planning to complete the maneuver before planning onward. We assume a module which computes probabilities over current maneuvers, $p(\omega^i)$, for each vehicle i. One option is Bayesian changepoint detection algorithms such as CHAMP [19], as used in MPDM [9]. The details of maneuver detection are outside the scope of our paper and in our experiments we use a simulated detector.

As different current maneuvers may hint at different goals, we perform inverse planning for each possible current maneuver for which $p(\omega^i) > 0$. Thus, each current maneuver produces its own posterior probabilities over goals, denoted by $p(g^i \mid s_{1:t}, \omega^i)$. For efficiency, the inverse planning may be limited to some subset of maneuvers such as the most-likely maneuvers.

3) **Inverse Planning:** Inverse planning is done using A* search [12] over macro actions. A* starts after completing the current maneuver ω^i which produces the initial trajectory $\hat{s}_{1:\tau}$. Each search node q corresponds to a state $s \in \mathcal{S}$, with initial node at state $\hat{s}_\tau$, and macro actions are filtered by their applicability conditions applied to s. A* chooses the next macro action leading to a node q' which has lowest estimated total cost[1] to goal g^i, given by $f(q') = l(q') + h(q')$. The cost $l(q')$ to reach node q' is given by the driving time from i's location in the initial search node to its location in q', following the trajectories returned by the macro actions leading to q'. The cost heuristic $h(q')$ to estimate remaining cost from q' to goal g^i is given by the driving time from i's location in q' to goal via straight line at speed limit. This definition of $h(q')$ is admissible as per A* theory, which ensures that the search returns an optimal plan. After the optimal plan is found, we extract the complete trajectory $\hat{s}_{1:n}^i$ from the maneuvers in the plan and the initial segment $\hat{s}_{1:\tau}$ from current maneuver.

Several design choices were made to minimise the computational cost of inverse planning. First, macro actions are executed using open-loop control. This means that trajectories are simulated using an idealised driving model which executes the trajectory with linearly-interpolated target velocities. Second, we use the assumption that all other vehicles not planned for use a constant-velocity lane-following model after their observed trajectories. This assumption allows *give-way* maneuver and *Change left/right* macro actions to predict when traffic will be clear. Third, no smoothing is applied during search and a surrogate cost definition based only on approximate driving time is used. Finally, we do not check for collisions during inverse planning; due to the use of open-loop control and constant velocities of other vehicles, there may be situations where collisions happen inevitably. These simplifications result in a level of abstraction which has low computational cost and is sufficiently informative for the goal probabilities.

4) **Trajectory Prediction:** Our system predicts multiple plausible trajectories for a given vehicle and goal, rather than a single optimal trajectory. This is required because there are situations in which different trajectories may be (near)optimal but may lead to different predictions which could require different behaviour on the part of the ego vehicle.

To predict multiple trajectories and associated probabilities to a given goal, we run A* search for a fixed amount of time and let it compute a set of plans with associated rewards (up to some fixed number of plans). Any time A* search finds a node that reaches the goal, the corresponding plan is added to the set of plans. Given a set of computed trajectories $\{\hat{s}_{1:n}^{i,k} | \omega^i, g^i\}_{k=1..K}$ to goal g^i with initial maneuver ω^i and associated reward $r_k = R^i(\hat{s}_{1:n}^{i,k})$ after smoothing, we compute a distribution over the trajectories by using a Boltzmann distribution:

$$p(\hat{s}_{1:n}^{i,k}) = \eta \, \exp(-\gamma \, r_k) \tag{8}$$

where γ is a scaling factor (we use $\gamma = 1$) and η is a normaliser. This encodes the assumption that trajectories which are closer to optimal are more likely.

G. Ego Vehicle Planning

To compute an optimal plan for the ego vehicle, we use the goal probabilities and trajectory predictions to inform a Monte Carlo Tree Search (MCTS) algorithm [6]. MCTS combines the statistical back-propagation operators used in temporal-difference reinforcement learning [25] with a dynamic tree expansion to focus the search on the current state. Algorithm 2 gives the pseudo code of our MCTS algorithm (our algorithm is a "rollout-based" version of MCTS [16]).

The algorithm performs a number of simulations $\hat{s}_{t:n}$, starting in the current state $\hat{s}_t = s_t$ down to some fixed search depth or until a goal state is reached. At the start of each simulation, for each other vehicle, we first sample a current maneuver, then goal, and then trajectory for the vehicle using the associated probabilities (cf. Section III-F).[2] The sampled trajectories will be used to simulate the motion of the other vehicles, in closed-loop or open-loop modes (detailed below). As in A* search, each node q in the search tree corresponds to a state $s \in \mathcal{S}$ and macro actions are filtered by their applicability conditions applied to s. After selecting a macro action μ using some exploration technique (we use UCB1 [3]), the state in current search node is forward-simulated based on the trajectory generated by the macro action and the sampled trajectories of other vehicles, resulting in a partial trajectory $\hat{s}_{\tau:\iota}$ and new search node q' with state $\hat{s}_\iota$. Collision checking is performed on $\hat{s}_{\tau:\iota}$ to check whether the ego vehicle collided, in which case we set the reward to $r \leftarrow r_{coll}$ which is back-propagated using (9), where r_{coll} is a method parameter. Otherwise, if the new state $\hat{s}_\iota$ achieves the ego goal g^ε, we compute the reward for back-propagation as $r = R^\varepsilon(\hat{s}_{t:n})$. If the search reached its maximum depth d_{max} without colliding or achieving the

---

[1] Here we use the term "cost" in keeping with standard A* terminology and to differentiate the simpler cost definition used by our A* search from the reward function defined in Sec. III-E.

[2] Sampling g^i from the mixed posterior $\sum_{\omega^i} p(g^i|s_{1:t}, \omega^i) p(\omega^i)$ is not a sound approach because examples can be constructed where this may lead to incompatible sampling of ω^i/g^i (i.e. g^i cannot be achieved after ω^i).

Algorithm 2 Monte Carlo Tree Search algorithm

Returns: optimal maneuver for ego vehicle ε in state s_t

Perform D simulations:

1: Search node $q.s \leftarrow s_t$ ($root$ node)
2: Search depth $d \leftarrow 0$
3: **for all** $i \in \mathcal{I} \setminus \{\varepsilon\}$ **do**
4:      Sample current maneuver $\omega^i \sim p(\omega^i)$
5:      Sample goal $g^i \sim p(g^i \mid s_{1:t}, \omega^i)$
6:      Sample trajectory $\hat{s}_{1:n}^i \in \{\hat{s}_{1:n}^{i,k} \mid \omega^i, g^i\}$ with $p(\hat{s}_{1:n}^{i,k})$
7: **while** $d < d_{max}$ **do**
8:      Select macro action μ for ε applicable in $q.s$
9:      $\hat{s}_{\tau,\iota} \leftarrow$ Simulate macro action until it terminates, with other vehicles following their sampled trajectories $\hat{s}_{1:n}^i$
10:      $r \leftarrow \emptyset$
11:      **if** ego vehicle collides during $\hat{s}_{\tau,\iota}$ **then**
12:          $r \leftarrow r_{coll}$
13:      **else if** $\hat{s}_{\iota}^{\varepsilon}$ achieves ego goal g^{ε} **then**
14:          $r \leftarrow R^{\varepsilon}(\hat{s}_{t:n})$
15:      **else if** $d = d_{max} - 1$ **then**
16:          $r \leftarrow r_{term}$
17:      **if** $r \neq \emptyset$ **then**
18:          Use (9) to backprop r along search branches (q, μ, q') that generated the simulation
19:          Start next simulation
20:      $q'.s = \hat{s}_{\iota}; q \leftarrow q'; d \leftarrow d + 1$
Return maneuver for ε in s_t, $\mu \in \arg\max_{\mu} Q(root, \mu)$

---

goal, we set $r \leftarrow r_{term}$ which can be a constant or based on heuristic reward estimates similar to A* search.

The reward r is back-propagated through search branches (q, ω, q') that generated the simulation, using a 1-step off-policy update function (similar to Q-learning [28]) defined by

$$Q(q, \mu) \leftarrow Q(q, \mu) + \begin{cases} \delta^{-1}[r - Q(q, \mu)] \text{ if } q \text{ leaf node, else} \\ \delta^{-1}[\max_{\mu'} Q(q', \mu') - Q(q, \mu)] \end{cases}$$
(9)

where δ is the number of times that macro action μ has been selected in q. After the simulations are completed, the algorithm selects the best macro action for execution in s_t from the root node, $\arg\max_{\mu} Q(root, \mu)$.

The MCTS algorithm is re-run at a given frequency to account for new observations since the last MCTS call (information from the past search tree is not reused).

We use two different control modes to simulate maneuvers and macro actions. The ego vehicle's motion always uses closed-loop mode, while other vehicles can be simulated in either closed-loop or open-loop mode.

1) Closed-Loop Simulation: Closed-loop simulation uses a combination of proportional control and adaptive cruise control (ACC). Two independent proportional controllers control the acceleration and steering of the vehicle. If there is another vehicle close ahead of the controlled vehicle, control is given

to ACC which keeps the vehicle at a safe distance to the leading vehicle (our ACC is based on IDM [26]). No velocity smoothing is applied since the combination of P/ACC control achieves approximately smooth control. Termination conditions in maneuvers are monitored in each time step based on the vehicle's observations.

2) Open-Loop Simulation: Open-loop simulation works in the same way as in A* search (see Sec. III-F3), by setting the vehicle's position and velocity directly as specified in trajectory. Hence, there is no automatic distance keeping in open-loop control. Velocity smoothing is applied to the trajectory to improve realism of the prediction. Termination conditions in maneuvers such as "wait until oncoming traffic is clear", e.g. as used in *give-way* maneuver, are realised by waiting until traffic is *predicted* to be clear assuming that non-controlled vehicles use a constant-velocity lane-following model.

IV. EVALUATION

We evaluate our system in simulations of four urban driving scenarios with diverse scenario initialisations. We show that:

- Our method correctly recognises the goals of other vehicles
- Goal recognition leads to improved driving behaviour
- We can extract intuitive explanations for the recognised goals and predictions, to justify the system's decisions

A. Scenarios

We use the following scenarios:

S1 T-junction (Figure 3): Ego vehicle approaches T-junction from west, its goal is to reach east end (blue goal). Vehicle V_1 is on same road as ego on adjacent lane, its goal is to exit the road and drive south (purple goal) for which it changes to ego lane in front of ego. Vehicle V_2 is approaches the T-junction from south, its goal is to drive east (blue goal). V_2 has to give way at the junction.

S2 X-junction (Figure 4): Ego vehicle approaches X-junction from south, its goal is to reach west end (blue goal). Vehicle V_1 approaches X-junction from west end, its goal is to reach the east end (yellow goal). Vehicle V_2 is approaches X-junction from east end, its goal is to reach north end (purple goal). V_2 has to wait for V_1 to pass.

S3 Roundabout (Figure 5): Ego vehicle approaches roundabout from west, its goal is to drive east (green goal). Vehicle V_1 is inside roundabout, its goal is to exit south (orange goal).

S4 Merge (Figure 6): Ego vehicle approaches main road from east, its goal is to reach north end (purple goal). Several other vehicles are on main road, queuing behind a red light. Vehicle V_1 leaves a gap for ego vehicle to merge in. Vehicle V_2 is driving south (green goal).

For each scenario, we generate 100 instances with randomly offset initial longitudinal positions (offset $\sim [-10, +10]$ metres) and initial speed sampled from range $[5, 10]$ m/s for each vehicle including ego vehicle.

B. Baselines & Parameters

We compare the following versions of our system. **Full:** Full system using goal recognition and MCTS. **MAP:** Like Full,

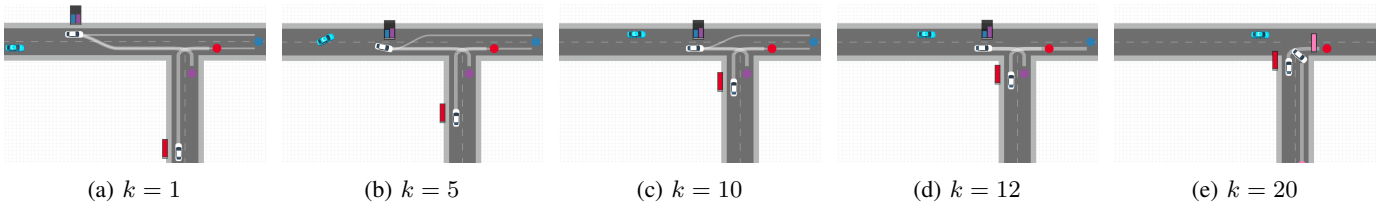(a) $k = 1$ (b) $k = 5$ (c) $k = 10$ (d) $k = 12$ (e) $k = 20$

Fig. 3: **S1**: With vehicle V_1 on the ego's road, and vehicle V_2 approaching from south. (a) Initially we attribute a uniform distribution to all goals for V_1 and V_2. (b) V_1 changes from left to right lane, biasing the ego prediction towards the belief that V_1 will exit, since a lane change would be irrational if V_1's goal was to go east. As exiting will require a significant slowdown, the ego decides to switch lanes to avoid being slowed down too. (c)/(d) The ego's belief that V_1 will exit increases as V_1 slows down when approaching the junction, encouraging the ego to continue in its lane. (e) V_1 takes the exit and the ego can safely continue as V_2 will merge to the right lane, its only possible goal.

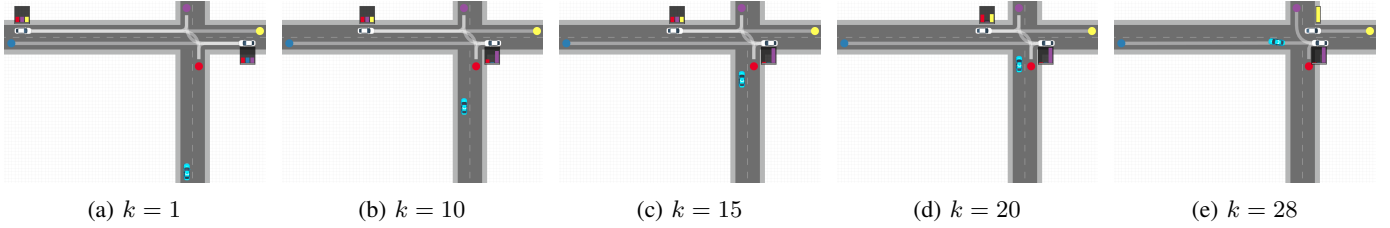(a) $k = 1$ (b) $k = 10$ (c) $k = 15$ (d) $k = 20$ (e) $k = 28$

Fig. 4: **S2**: With vehicle V_1 approaching the junction from west, and vehicle V_2 approaching it from east. (a) Initially we attribute a uniform distribution to all goals for V_1 and V_2. (b)/(c) As V_2 approaches the junction, slows down and waits to take a turn, the ego's belief that V_2 will turn right increases significantly, since it would be irrational to stop if the goal was to turn left or go straight. V_1's distribution over goals remains uniform due to the uninformative execution. (d) Due to V_1's constant speed, the ego rules out the north goal since a left turn at V_1's speed and location would require a sudden, costly (in terms of reward) braking from V_1. Meanwhile, the maintenance of V_2 at the junction reinforces ego's belief that V_2 intends to go north. (e) Since ego recognised V_2's goal is to go north, it predicts that V_2 will wait until V_1 has passed, giving the ego an opportunity to enter the road.

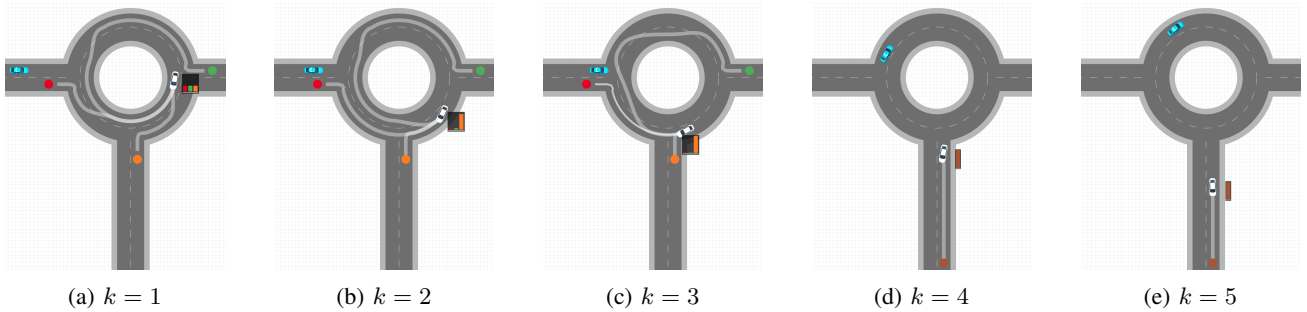(a) $k = 1$ (b) $k = 2$ (c) $k = 3$ (d) $k = 4$ (e) $k = 5$

Fig. 5: **S3**: With vehicle V_1 inside the roundabout. (a) Initially we attribute a uniform distribution to all goals for V_1. (b) As V_1 changes from the inside to the outside lane of the roundabout and decreases its speed, it significantly biases the ego prediction towards the belief that V_1 will leave in the next exit since that is the rational course of action for that goal. (c) The ego's belief that V_1 will leave increases further as V_1 approaches the exit and continues to slow down, encouraging the ego to enter the roundabout while V_1 is still in roundabout. (d)/(e) V_1 takes the exit, at which point the ego has already entered the roundabout.

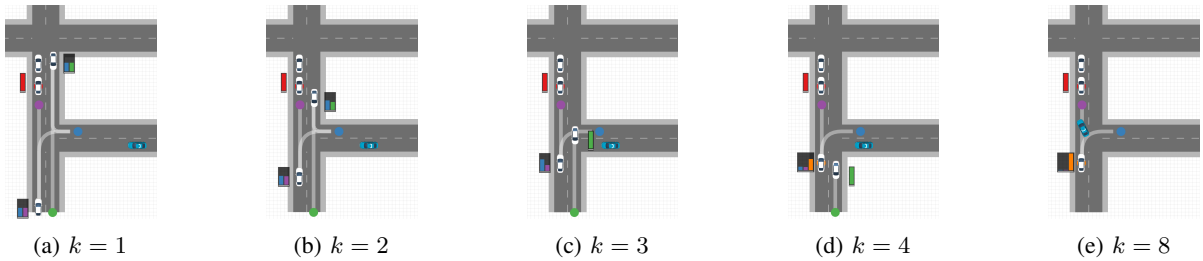(a) $k = 1$ (b) $k = 2$ (c) $k = 3$ (d) $k = 4$ (e) $k = 8$

Fig. 6: **S4**: With two vehicles stopped at the junction at a traffic light, vehicle V_1 approaching them from behind, and vehicle V_2 crossing in the opposite direction. (a) Initially we attribute a uniform distribution to all goals for V_1 and V_2. (b) Goal probabilities for both vehicles remain near-uniform due to very minor changes in speeds. (c) V_1 has begun to decelerate at a point where it is more indicative of exiting the road, since the north goal would not require slowing down quite as early according to reward function. Hence ego waits. (d) V_1's zero velocity reveals a stopping goal in its current position, shifting the distribution towards it, since stopping is not rational for north/east goals. The interpretation is that V_1 wants the ego to merge in. (e) Given the recognised goal, the ego merges onto the road in front of V_1.

but MCTS uses only the most probable goal and trajectory for each vehicle. **CVel:** MCTS without goal recognition, replaced by constant-velocity lane-following prediction. **Cons:** Like CVel, but using a conservative *give-way* maneuver which waits until all oncoming vehicles on priority lanes have passed. All of these baselines use closed-loop simulation in MCTS. We also evaluate the Full baseline with closed-loop (**Full-CL**) and open-loop (**Full-OL**) simulation.

Maneuvers, macro actions, and goal generation heuristic are as defined earlier in paper. For each other vehicle and generated goal, we generate up to 3 predicted trajectories. We simulate noisy detection of current maneuvers for each other vehicle by giving 0.95 probability to correct current maneuver and the rest uniformly to other maneuvers. MCTS is run at a frequency of 1 Hz, performs $D = 30$ simulations, with a maximum search depth of $d_{max} = 5$. Rewards for collision and maximum search depth are set to $r_{coll} = r_{term} = -1$. Prior probabilities for achievable goals are uniform.

C. Results

Figures 3–6 show snapshots for scenario instances at different planning stages of our Full system. The bar plots give the goal probabilities for each other vehicle associated with their most probable current maneuver. For each goal, we show the most probable trajectory prediction from vehicle to goal, with thickness proportional to its probability. We extract intuitive explanations for the goal recognition and maneuver predictions in each scenario, which are given in the figure captions.

Figure 8 shows the evolution of probability assigned to correct goal over time, in four scenario instances. As can be seen, the probability approaches the correct goal as other goals are being ruled out by means of rationality principles. We observed this behaviour in all scenario instances.

Figure 7 shows the average times (in seconds) and standard deviations required by each baseline to complete a scenario instance. **(S1)** All baselines switch lanes in response to V_1 switching lanes. Full and MAP anticipate V_1's slowdown earlier than other baselines due to inverse planning, allowing them to switch lines slightly earlier. CVel and Cons only switch lanes once V_1 already started to slow down, and are unable to explain V_1's behaviour. **(S2)** Cons requires substantially more time to complete the scenario since it waits for V_2 to clear the lane, which in turn must wait for V_1 to pass. Full and MAP anticipate this behaviour, allowing them to safely enter the road earlier. CVel produces the same result due to zero-velocity of V_2, but cannot fully justify (explain) its decisions since it is unable to explain V_2's waiting behaviour. **(S3)** Both CVel and Cons require more time to complete the scenario. Here, the constant-velocity prediction in CVel, and waiting for actual clearance as in Cons, amount to approximately equal time of entry for ego vehicle. Full and MAP are able to enter earlier as they recognise V_1's goal, which is to exit the roundabout. MAP enters earlier than Full since it fully commits to the most probable goal for V_1, while Full exhibits more cautious behaviour due to residual uncertainty about V_1's goal which could hypothetically lead to crashes. **(S4)** Cons must wait until
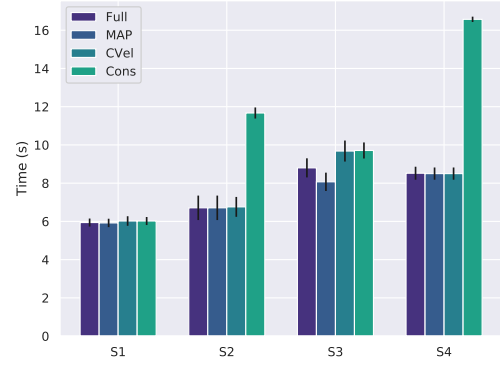

Fig. 7: Average driving time (seconds) required to complete scenario instances, with standard deviation.
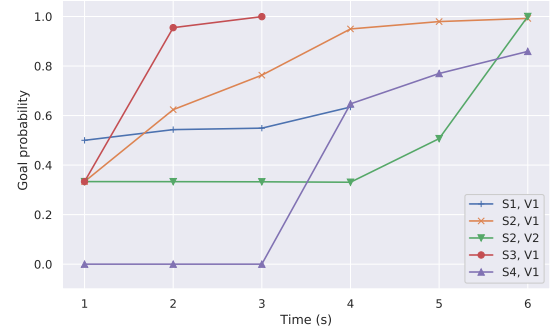

Fig. 8: Evolution of probability given to correct goal for selected vehicles in four scenario instances. Note: lines for S1/S3 are shorter than indicated in Fig. 7 since vehicle goals change after exit points (cf. Figs. 3&5) and we only show lines for initial goals.

V_1 decides to close the gap, after which the ego can enter the road, hence requiring more time. Full and MAP recognise V_1's goal and can enter safely. CVel again produces the same behaviour based on constant velocity of V_1, but cannot explain the waiting behaviour of V_1.

Full-CL and Full-OL achieved the same completion rate of scenario instances (100%) and required the same amount of driving time. We found that OL was sufficient, in our specific scenarios, to simulate the vehicle interactions. However, it is likely that more densely populated roads will benefit from CL simulation. As expected, OL required significantly less compute time than CL, in some cases up to 50% less.

V. Conclusion

We proposed an autonomous driving system which integrates planning and prediction over extended horizons, by leveraging the computational benefit of utilising a finite maneuver library. Prediction over extended horizons is made possible by recognising the goals of other vehicles via a process of rational inverse planning. Our evaluation showed that the system robustly recognises the goals of other vehicles in diverse urban driving scenarios, resulting in improved decision making while allowing for intuitive interpretations of the predictions to justify (explain) the system's decisions. We note that our system is general in that it uses relatively standard planning techniques which could be replaced with other related techniques, e.g.,

POMDP-based approximate planners [23]. Furthermore, while this work focused on prediction of other vehicles, the principles underlying our system are general and could be extended to include prediction of other traffic participants such as cyclists, or applied to other domains in which mobile robots interact with other robots/humans. An important future direction is to account for human irrational biases (e.g. [17]).

REFERENCES

[1] S. Albrecht and P. Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

[2] S. Albrecht, J. Crandall, and S. Ramamoorthy. Belief and truth in hypothesised behaviours. *Artificial Intelligence*, 235:63–94, 2016.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[4] C. Baker, R. Saxe, and J. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[5] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus. Intention-aware motion planning. In *Algorithmic Foundations of Robotics X*, pages 475–491. Springer, 2013.

[6] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[7] Y. Chai, B. Sappm, M. Bansal, and D. Anguelov. Multi-Path: multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Proceedings of the 3rd Conference on Robot Learning*, 2019.

[8] C. Dong, J. M. Dolan, and B. Litkouhi. Smooth behavioral estimation for ramp merging control in autonomous driving. In *IEEE Intelligent Vehicles Symposium*, pages 1692–1697. IEEE, 2018.

[9] E. Galceran, A. Cunningham, R. Eustice, and E. Olson. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment. *Autonomous Robots*, 41(6):1367–1382, 2017.

[10] C. Gámez Serna and Y. Ruichek. Dynamic speed adaptation for path tracking based on curvature information and speed limits. *Sensors*, 17(1383), 2017.

[11] J. Hardy and M. Campbell. Contingency planning over probabilistic obstacle predictions for autonomous road vehicles. *IEEE Transactions on Robotics*, 29(4):913–929, 2013.

[12] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. In *IEEE Transactions on Systems Science and Cybernetics*, volume 4, pages 100–107, July 1968.

[13] R. He, E. Brunskill, and N. Roy. Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research*, 40:523–570, 2011.

[14] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller. Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1671–1678. IEEE, 2017.

[15] C. Hubmann, J. Schulz, M. Becker, D. Althoff, and C. Stiller. Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction. *IEEE Transactions on Intelligent Vehicles*, 3(1): 5–17, 2018.

[16] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, pages 282–293. Springer, 2006.

[17] M. Kwon, E. Biyik, A. Talati, K. Bhasin, D. P. Losey, and D. Sadigh. When humans aren't optimal: Robots that collaborate with risk-aware humans. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, 2020.

[18] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker. DESIRE: distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017.

[19] S. Niekum, S. Osentoski, C. Atkeson, and A. Barto. Online Bayesian changepoint detection for articulated motion models. In *IEEE International Conference on Robotics and Automation*. IEEE, 2015.

[20] M. Ramírez and H. Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *24th AAAI Conference on Artificial Intelligence*, pages 1121–1126, 2010.

[21] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. PRECOG: prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2830, 2019.

[22] W. Schwarting, J. Alonso-Mora, and D. Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, 2018.

[23] A. Somani, N. Ye, D. Hsu, and W. S. Lee. DESPOT: online POMDP planning with regularization. In *Advances in Neural Information Processing Systems*, pages 1772–1780, 2013.

[24] W. Song, G. Xiong, and H. Chen. Intention-aware autonomous driving decision-making in an uncontrolled intersection. *Mathematical Problems in Engineering*, 2016, 2016.

[25] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[26] M. Treiber, A. Hennecke, and D. Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805, 2000.

[27] A. Wächter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106 (1):25–57, 2006.

[28] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[29] M. Wulfmeier, D. Z. Wang, and I. Posner. Watch this: Scalable cost-function learning for path planning in urban environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2089–2095. IEEE, 2016.

[30] Y. Xu, T. Zhao, C. Baker, Y. Zhao, and Y. N. Wu. Learning trajectory prediction with continuous inverse optimal control via Langevin sampling of energy-based models. *arXiv preprint arXiv:1904.05453*, 2019.

[31] B. Zhou, W. Schwarting, D. Rus, and J. Alonso-Mora. Joint multi-policy behavior estimation and receding-horizon trajectory planning for automated urban driving. In *IEEE International Conference on Robotics and Automation*, pages 2388–2394. IEEE, 2018.