Supplementary Information for

Social Behavior for Autonomous Vehicles

Wilko Schwarting, Alyssa Pierson, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus

Wilko Schwarting.
E-mail: wilkos@mit.edu

This PDF file includes:

Supplementary text
Figs. S1 to S21
Tables S1 to S3
Caption for Movie S1
References for SI reference citations

Other supplementary materials for this manuscript include the following:

Movie S1

Supporting Information Text

S1. Detailed Step-by-Step Walk-through of Complete Approach

In this section we detail the use of our methodology in the context of a driving example. Applications of this methodologies to other fields are also possible. We hope that this example will make the "recipe" more concrete. Specifically, we provide the step-by-step execution of Algorithm 1 in the context of the example of autonomous merging with SVO shown in Figure S1. Our description is self-contained and simultaneously a detailed walk-through and summary of our approach.

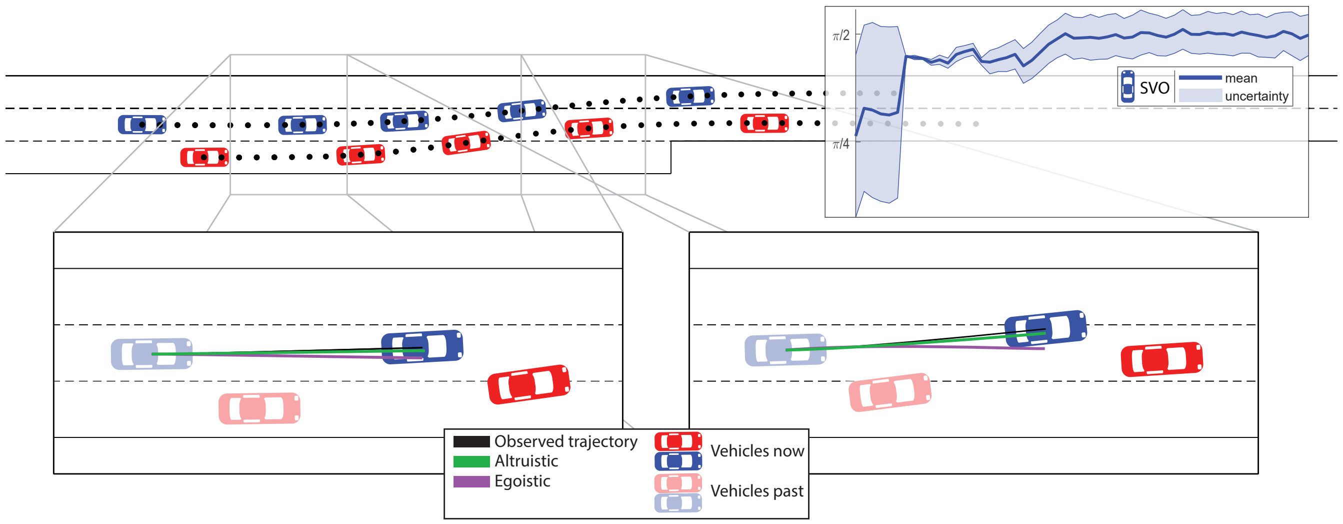


Fig. S1. Example of a lane merge on a highway on-ramp. The red car merges onto the highway while the blue car moves to the adjacent lane to free space for the red car. The SVO is estimated accordingly, shown in the top right inset, and allows the red autonomous vehicle to predict that a merge onto the adjacent lane is feasible. The bottom cutouts show predictions for altruistic and egoistic SVOs compared to the actually observed trajectory at two different timesteps.

We assume the agent has some prior system knowledge before executing the algorithm. We gather this knowledge from several sources and preliminary steps:

- Collect a dataset consisting of human trajectories of interacting agents over time. This paper focuses on the NGSIM dataset containing highway data, but other sources containing naturally interacting agents and their trajectories are also suitable.

- Employ Inverse Reinforcement Learning (IRL) to learn human reward functions and calibrate them to human-like decision-making based on the dataset. The reward functions are general function approximators with parameters learned from IRL. These reward functions essentially describe how much human drivers value certain factors such as comfort, collision-avoidance, or getting to their goal location quickly.

- If no dataset is available, reward functions can be hand-tuned by experts to represent plausible human-like reward functions that result in expected behavior.

The following is a step-by-step breakdown of the algorithm routine executed by the AV:

(i) We initialize without any knowledge about the other drivers' SVOs, such that our initial estimated SVO of other vehicles is a uniform distribution over the SVO ring. This is visualized by a large spread of SVO estimate and particles in the SVO estimate plot in Figure S1.

(ii) The AV needs to perceive its own state, including its position on the road network, and the relative position of other cars, see Figure S2A.

(iii) The AV tracks other vehicles' trajectories over time, Figure S2B.

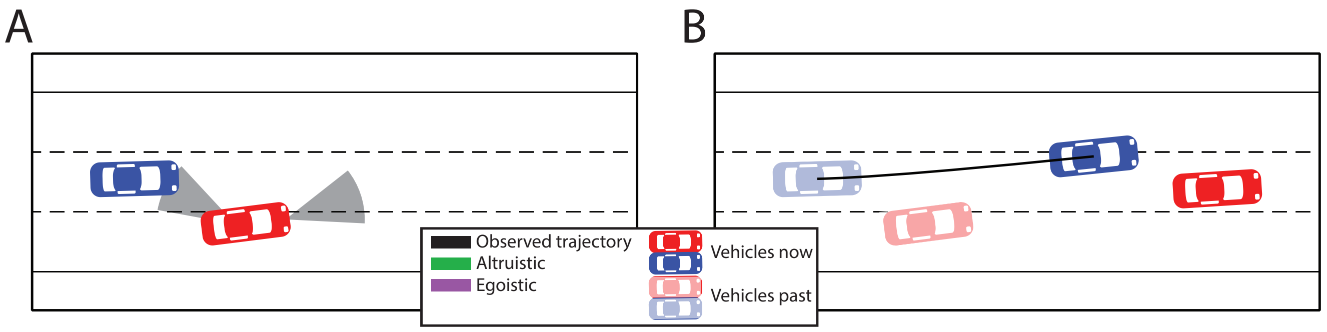


Fig. S2. (A) Red vehicle perceives the state of the blue vehicle and its own state in the road network. **(B)** The state of the blue vehicle is tracked over time and accumulated into a trajectory.

(iv) Given the state of another vehicle and the SVO of the other driver, the AV can predict their trajectory from the game-theoretic interaction model computing a Nash equilibrium (see Section S2). We use the following example as an illustration:

(a) In our example of a lane-merge on the highway the blue car, which is already on the highway, will incur negative rewards for changing its speed or changing to the adjacent lane because it would incur costs for decreased comfort and a time delay for reaching its goal location. We can associate costs to decisions made by the blue car. As an illustrative example, let's assume that the blue car only has two options to act: B1) Continuing as is without changing lanes or speed, B2) Changing lanes into the adjacent left lane. Option B1) Does not incur any costs if there is no collision, while option B2) incurs comfort costs.

(b) The other red car is on the highway on-ramp and needs to merge onto the highway. It incurs negative rewards for accelerating, braking, and steering abruptly because of decreased comfort. Most importantly, it needs to avoid crashing into the barrier at the end of the on-ramp. Again, let's evaluate the cost and limit the red car's action options to either R1) keeping its speed and changing lanes and R2) breaking and changing lanes behind the blue car.

(c) Note that limiting the options to two choices is a very strong simplification of our approach, and used for illustrative purposes. In the case that the blue car chooses option B1 and the red car chooses option R1: The blue car would continue on its lane, while the red car would merge into the blue vehicle's lane. This would result in a crash. Nonetheless, the situation does not occur, since the reward functions are dynamic over time. The cost of driving too close to another vehicle would continuously increase until it is cheaper for the red car to brake and merge behind the blue vehicle. Thus, we neglect this combination of options.

(d) Therefore, if the blue car is egoistic and takes only its own reward into account, it would always choose option B1 and continue driving without a change of lanes or speed forcing the red car to choose option R2, to brake and merge behind the blue car. In contrast, if the blue car were altruistic, it would choose option B2 and merge into the adjacent lane, enabling the red car to choose option R1, to comfortably merge onto the highway without braking. This results from the utility-maximizing behavior. In the case of altruistic behavior, the utility of the blue car will always be higher when assisting in achieving a higher reward for the red car, whereas in the case of egoistic behavior the blue car would never assist the red car and only maximize its own reward.

(e) The takeaway of the previous simplified decision-making is that different SVOs result in different prediction outcomes. The resulting trajectories of the blue vehicle are shown in Figure S3. The red vehicle has started changing lanes and attempts to change lanes form the on-ramp to the adjacent lane. If the blue car has an egoistic SVO, it would follow option B1 and keep moving inside its lane, or at the time of the screenshot move back center of the lane. This would force the red car to abort the lane change, to brake, and to attempt another lane change behind the blue vehicle at a later time. In the altruistic case the blue car follows option B2 and moves to the left adjacent lane allowing the red vehicle to complete the merge.

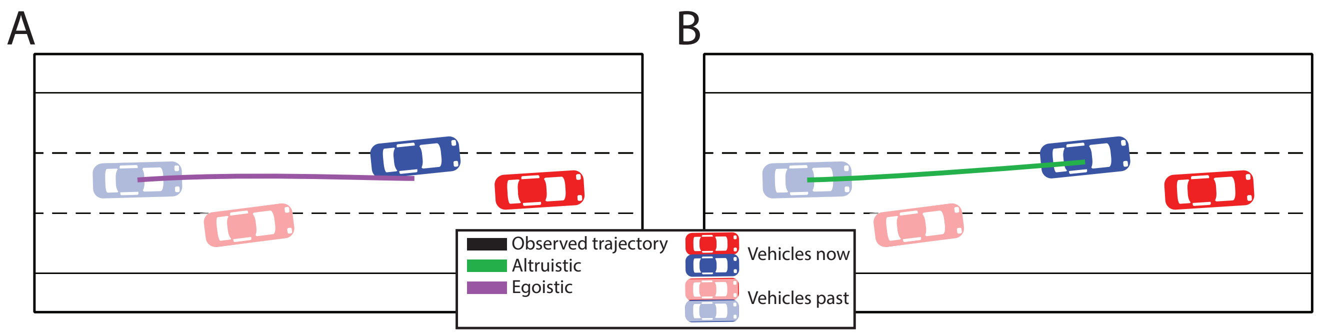


Fig. S3. Given the state and SVO of the blue car at a previous time, we can predict their future trajectory. The predicted trajectories vary for different SVOs. While an egoistic SVO desires to return to the lane's center, **(A)**, the altruistic SVO yields a merge to the next adjacent lane, **(B)**.

(v) We sample predicted trajectories for possible SVO values by computing maximum-utility trajectories. Comparing these to the actual observed trajectories allows us to score each of them with a likelihood function, as described in Section S3.E and shown in Figure S4. In this example, the trajectory predicted based on the altruistic SVO is closer to the observation than the trajectory predicted from the egoistic SVO. Followingly, the altruistic SVO is more likely than the egoistic SVO. While we have described the intuitive prediction based likelihood function comparing sampled and observed trajectories here, we have developed a similar maximum entropy likelihood function in Section S3.E.2.

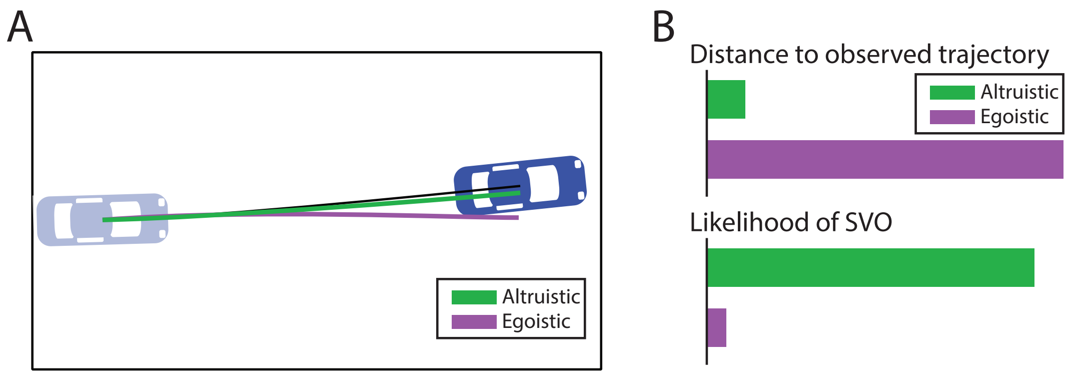


Fig. S4. **(A)** Comparison of predictions given altruistic and egoistic SVOs. The altruistic prediction is much closer to the observed trajectory than the egoistic prediction, shown in **(B)**, top. Therefore, the blue car is more likely to be altruistic rather than egoistic, shown in **(B)**, bottom.

(vi) The result is a likelihood distribution over SVOs computed from the observed trajectory. We integrate the likelihood function into a recursive filtering framework, a particle filter, updating the current posterior distribution over SVOs at every timestep once a new observation is available and a new likelihood distribution over SVOs can be computed. The particle filter's SVO posterior over time is shown in Figure S1.

(vii) The ego AV iteratively uses Algorithm 1 to sample and maintain its predictions about other vehicles during interactions. Given that we maintain an estimate over the SVOs of other agents in the environment, we can better predict their trajectories and behavior, and thus take better actions.

S2. Game Theoretic Formulation for Multi-Agent Decision Making

This section provides further detail on our multi-agent game theoretic decision-making formulation. First, we provide an optimization primer defining key terms and concepts. Next, we present the two-agent Stackelberg game formulation, followed by the generalized multi-agent Stackelberg game. We then present a solution method to the constrained multi-agent Nash Equilibrium.

S2.A. Optimization Primer. Our approach utilizes a game theoretic formulation to model the interactions between agents in the system. In this section, we introduce several key concepts and terminology used in describing our approach. For agents $i = \{1, ..., m\}$, the agent's state at time k is denoted $\mathbf{x}_i^k$ and the control policy is denoted $\mathbf{u}_i^k$. We assume that each agent is governed by constrained dynamics, such that $\dot{\mathbf{x}}_i = f_i(\mathbf{x}_i, \mathbf{u}_i)$, where the function $f_i(\cdot)$ is subject to constraints $c_i(\cdot) \leq 0$. For brevity, we write the state and control policy of all agents as $\mathbf{x} = [\mathbf{x}_1^\top, ..., \mathbf{x}_m^\top]^\top$ and $\mathbf{u} = [\mathbf{u}_1^\top, ..., \mathbf{u}_m^\top]^\top$. The states evolve according to system dynamics in Eq. (2):

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = [f_1(\mathbf{x}_1, \mathbf{u}_1)^\top, \ldots, f_m(\mathbf{x}_m, \mathbf{u}_m)^\top]^\top.$$

We denote the discrete time transition function as $\mathbf{x}^{k+1} = \mathbf{x}^k + \int_k^{k+\Delta t} f(\mathbf{x},\mathbf{u})\,\mathrm{d}t = \mathcal{F}(\mathbf{x}^k,\mathbf{u}^k)$, with individual functions denoted $\mathcal{F}_i(\mathbf{x}^k,\mathbf{u}^k)$. The agents calculate their individual control policies $\mathbf{u}_i$ by solving a general discrete-time constrained optimization over a given time horizon. At each time step in the horizon, the agent computes a utility function $g_i(\cdot)$ based on their state, controls, and SVO.

Table S1. Main symbols.

$\mathbf{x} = [\mathbf{x}_1^T, ..., \mathbf{x}_m^T]^T$	State trajectory of all agents
$\mathbf{u} = [\mathbf{u}_1^T, ..., \mathbf{u}_m^T]^T$	Control trajectory of all agents
$\dot{\mathbf{x}}_i = f_i(\mathbf{x}_i, \mathbf{u}_i)$	Continuous dynamics of each agent i
$\mathbf{x}_i^{k+1} = \mathcal{F}_i(\mathbf{x}_i^k, \mathbf{u}_i^k)$	Discrete dynamics of each agent i
$c_i(\cdot) \leq 0$	Inequality constraints
$r_i(\cdot)$	Instantaneous reward of agent i
$g_i(\cdot)$	Instantaneous utility of agent i
$G_i(\cdot)$	Cumulative utility over time horizon of agent i

We define the utility function by weighting the reward functions of the agents by their SVO functions. Recall an agent's SVO is denoted as the angle φ_i. For a two-agent game, the utility function of agent $i = 1$, written in Eq. (1), is

$$g_i(\cdot) = \cos(\varphi_1) r_1(\cdot) + sin(\varphi_1) r_2(\cdot),$$

where $r_i(\cdot)$ is an agent's reward function. The reward function for each agent encodes their "payout." In the context of autonomous driving, this may be travel time, comfort, distance between other cars, braking and acceleration efforts, progress towards a goal, and other priorities to the driver. Here, we learn the reward function $r_i(\cdot)$ through Inverse Reinforcement Learning (IRL), discussed further in Section S3. We generalize the utility function from Eq. (1) to m agents as

$$g_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_{\neg i}, \varphi_i) = \frac{1}{m-1} \sum_{j \in \neg i} \left[\cos(\varphi_i) r_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_j) + \sin(\varphi_i) r_j(\mathbf{x}, \mathbf{u}_j, \mathbf{u}_i)\right], \quad \text{[s1]}$$

where $\mathbf{x}$ is the state of all agents, and $r_i(\cdot)$ and $r_j(\cdot)$ are the reward functions of agent i and agent j respectively. The generalized utility function essentially weights own rewards r_i with $\cos(\varphi_i)$ against the sum of all other agents rewards $r_{\neg i}$ scaled by $1/(m-1)\sin(\varphi_i)$. In simpler terms this describes how the agent i weighs own rewards against rewards of the other agents.

Recall we use the notation $\mathbf{u}_{\neg i}$ to denote all other agents' control policies excluding the $i-$th agent. The ego agent computes their utility over a given time horizon, denoted $G_i(\cdot)$. The time horizon τ is discretized over N steps, such that $\tau = \sum_{k=0}^{N} \Delta t$. The utility at any given time for agent i is given by $g_i(\mathbf{x}^k, \mathbf{u}^k, \varphi_i)$ and the terminal utility $g_i^N\left(\mathbf{x}^N, \varphi_i\right)$, and written over the horizon in Eq. (3),

$$G_i(\mathbf{x}^0, \mathbf{u}, \varphi) = \sum_{k=0}^{N-1} g_i\left(\mathbf{x}^k, \mathbf{u}^k, \varphi_i\right) + g_i^N\left(\mathbf{x}^N, \varphi_i\right).$$

The sum results from integrating the instantaneous utilities encountered by agent i at each point in time over the time horizon. The terminal utility captures the utility at the end of the time horizon. To solve for an agent's control policy, we construct a utility-maximizing optimization problem. Thus, each agent will find the optimal control policy that maximizes their utility over the horizon. In general, this is written

$$\mathbf{u}_i^* = \arg\max_{\mathbf{u}_i} G_i(\mathbf{x}^0, \mathbf{u}, \varphi_1). \quad \text{[s2]}$$

In the following section, we detail our game theoretic formulation of this problem, as well as present our method for solving this constrained optimization.

S2.B. Constrained Stackelberg Game Formulation. The traditional two-agent Stackelberg game (1) consists of a leader ($i = 1$) and follower ($i = 2$). The leader chooses its control policy $\mathbf{u}_1$ based on the assumption that the follower maximizes their control given the leader policy, or $\mathbf{u}_2^*(\mathbf{u}_1)$, yielding an under-actuated system

$$\mathbf{x}^{k+1} = \begin{bmatrix} \mathbf{x}_1^{k+1} \\ \mathbf{x}_2^{k+1} \end{bmatrix} = \begin{bmatrix} \mathcal{F}_1(\mathbf{x}_1^k, \mathbf{u}_1^k) \\ \mathcal{F}_2\left(\mathbf{x}_2^k, \mathbf{u}_2^{k,*}(\mathbf{u}_1^k)\right) \end{bmatrix}. \quad \text{[s3]}$$

This formulation captures that agent 1 can influence agent 2's actions $\mathbf{u}_2^*(\mathbf{u}_1)$ by changing the own controls $\mathbf{u}_1$. Agent 1 therefore has indirect control over what agent 2 will do. Taking this form of interaction into account agent 1 can now actively reason about how to influence agent 2's actions to help themselves in the best way. Under this formulation, one can predict realistic behavior and interactions of other agents given its actions (2). The constrained Stackelberg game can be formulated as:

$$\begin{aligned} \mathbf{u}_1^* &= \arg\max_{\mathbf{u}_1} \; G_1(\mathbf{x}^0, \mathbf{u}_1, \mathbf{u}_2^*(\mathbf{u}_1), \varphi_1), \\ \text{s.t. } & \mathbf{x}_1^{k+1} = \mathcal{F}_1(\mathbf{x}_1^k, \mathbf{u}_1^k), \\ & c_1(\mathbf{x}, \mathbf{u}_1, \mathbf{u}_2^*(\mathbf{u}_1)) \leq 0, \\ & \mathbf{u}_2^*(\mathbf{u}_1) = \arg\max_{\mathbf{u}_2} \; G_2(\mathbf{x}^0, \mathbf{u}_1, \mathbf{u}_2, \varphi_2) \\ & \text{s.t. } \mathbf{x}_2^{k+1} = \mathcal{F}_2(\mathbf{x}_2^k, \mathbf{u}_2^k), \\ & \quad c_2(\mathbf{x}, \mathbf{u}) \leq 0. \end{aligned} \tag{s4}$$

The Stackelberg game entails a bilevel optimization: An optimization on the higher level which contains a lower level optimization. For every optimization step on the higher level an optimization problem on the lower level needs to be solved. A simple interpretation is that agent 1 optimizes its own actions given that agent 2 optimizes their own actions depending on the actions of agent 1, and thus resulting in the underactuated system formulation given above in Eq. (s3).

This constrained bilevel optimization is hard to solve in practice, due to the fact that for every step of the upper-level optimization algorithm, the lower-level problem needs to be solved. In (2), direct gradients based on the stationarity condition of $G_1(\mathbf{x}^0, \mathbf{u}_1, \mathbf{u}_2, \varphi_1)$ with respect to $\mathbf{u}_2$ are defined and the linearized solution $\mathbf{u}_2^*(\mathbf{u}_1))$ is substituted back into $G_1(\mathbf{x}^0, \mathbf{u}_1, \mathbf{u}_2^*(\mathbf{u}_1), \varphi_1)$. However, the method loses the ability to propagate constraints, which may contain critical safety features and does not generalize to more than two agents.

Instead, an alternative approach is to reformulate the bilevel optimization problem as a local single-level optimization problem using Karush-Kuhn-Tucker (KKT) stationarity conditions (3). The locally-equivalent formulation of Eq. (s4) is

$$\mathbf{u}_1^* = \arg\max_{\mathbf{u}_1, \mathbf{u}_2, \mathbf{k}} \; G_1(\mathbf{x}^0, \mathbf{u}_1, \mathbf{u}_2, \varphi_1) + G_2(\mathbf{x}^0, \mathbf{u}_1, \mathbf{u}_2, \varphi_2), \tag{s5}$$

$$\text{s.t. } \mathbf{x}^{k+1} = \mathcal{F}(\mathbf{x}^k, \mathbf{u}^k),$$

$$c_1(\mathbf{x}, \mathbf{u}) \leq 0,$$

$$c_2(\mathbf{x}, \mathbf{u}) \leq 0, \tag{s6}$$

$$\nabla_{\mathbf{u}_2} G_2(\mathbf{x}^0, \mathbf{u}_1, \mathbf{u}_2, \varphi_2) + \mathbf{k}^\top \nabla_{\mathbf{u}_2} c_2(\mathbf{x}, \mathbf{u}) = 0, \tag{s7}$$

$$\mathbf{k}^\top c_2(\mathbf{x}, \mathbf{u}) = 0 \;, \tag{s8}$$

$$\mathbf{k} \geq 0, \tag{s9}$$

which includes constraints and can be solved by state-of-the-art nonlinear optimizers. Here, Eq. (s7) describes the stationarity condition, Eq. (s8) the complementary slackness, Eq. (s9) the dual and Eq. (s6) the primal feasibility constraint of the lower level optimization, and $\mathbf{k}$ the vector of dual variables. The sum of G_1 and G_2 is contained in the objective function to ensure solving for a maximum of the lower level. We therefore avoid to add an additional constraint to enforce negative definiteness of the Hessians to ensure the solver yields maxima.

A simplified explanation is that instead of solving the lower level optimization directly, we formulate a stationarity constraint which enforces optimality of the lower level optimization. In the standard Stackelberg formulation for every step at the higher-level optimization, a lower level optimization has to be solved. In our reformulation for every step in the upper level optimization only the stationarity constraint of the lower level optimization needs to be enforced. This can be handled easier than resolving an optimization on the lower level.

Note that in solving Eq. (s5), all safety constraints of the autonomous agents can be preserved in the optimization. The inclusion of constraints within the optimization is critical to guaranteeing safe operation and performance of any autonomous system.

S2.C. Multi-Agent Stackelberg Game Formulation. In the multi-agent case the Stackelberg game consists of a chain of leader ($i = 1$), follower ($i = 2$), subfollower ($i = 3$), subsubfollower ($i = 4$), until agent ($i = m$). We write the multi-agent case as

$$\begin{aligned} \mathbf{u}_1^* &= \arg\max_{\mathbf{u}_1} \; G_1(\mathbf{x}^0, \mathbf{u}_1, \mathbf{u}_{\neg 1}^*(\mathbf{u}_1), \varphi_1), \\ \text{s.t. } \mathbf{u}_2^*(\mathbf{u}_1) &= \arg\max_{\mathbf{u}_2} \; G_2(\mathbf{x}^0, \mathbf{u}_{1,2}, \mathbf{u}_{\neg 1,2}^*(\mathbf{u}_{1,2}), \varphi_2) \\ \text{s.t. } \mathbf{u}_3^*(\mathbf{u}_{1,2}) &= \arg\max_{\mathbf{u}_3} \; G_2(\mathbf{x}^0, \mathbf{u}_{1,2,3}, \mathbf{u}_{\neg 1,2,3}^*(\mathbf{u}_{1,2,3}), \varphi_3) \\ &\vdots \\ \text{s.t. } \mathbf{u}_m^*(\mathbf{u}_{\neg m}) &= \arg\max_{\mathbf{u}_m} \; G_m(\mathbf{x}^0, \mathbf{u}_{\neg m}, \mathbf{u}_m, \varphi_m). \end{aligned} \tag{s10}$$

In comparison to the above two agent Stackelberg game formulation, we have omitted equality constraints, such as dynamics, and inequality constraints, $c_i(\mathbf{x}, \mathbf{u}) \leq 0$, for compactness. Note that in Eq. (s10), each agent's control policy depends on all other agents, resulting in a recursion of dependencies that creates a m-level optimization problem even more challenging to solve than the bilevel optimization.

The Stackelberg game is inherently asymmetric and imposes a recursive hierarchy of leaders and followers. While desirable in certain traffic situations, such as recursive conflict resolution on highways (4), it does not completely define socially-compliant behavior. In many traffic scenarios, there are no defined leaders and followers. Thus, we need a more symmetric and simultaneous decision-making game.

Another limitation of the Stackelberg game is that it assumes the leader to have indirect control over the other agents and direct access to their control policies, which limits the follower's ability to negotiate and compromise on decisions. In the two-agent case, the follower chooses $\mathbf{u}_2(\mathbf{u}_1)$, but it may be desirable to have more levels of tacit-negotiation, such that $\mathbf{u}_2(\mathbf{u}_1(\mathbf{u}_2(\mathbf{u}_1(...))))$. This back-and-forth between agents removes the leader-follower dynamics and changes the sequential game to a simultaneous choice game. The generalized procedure for a multi-agent iterative best response game is described in Algorithm 1.

Algorithm 1 Iterated Best Response

```
1: Initialize $\mathbf{u}, \mathbf{u}_{\text{prev}}$
2: while $||\mathbf{u} - \mathbf{u}_{\text{prev}}|| > \epsilon$ do
3:     for $i \leftarrow 1$ to $m$ do
4:         $\mathbf{u}_i^* = \arg\max_{\mathbf{u}_i} \; G_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_{\neg i}, \varphi_i)$
5:     $\mathbf{u}_{\text{prev}} = \mathbf{u}, \mathbf{u} = \mathbf{u}^*$
6: Return u
```

Interestingly, the iterative best response is a numerical method to compute a Nash equilibrium game (5), since when the stationarity condition $||\mathbf{u} - \mathbf{u}_{\text{prev}}|| < \epsilon$ is satisfied, a Nash equilibrium is reached by definition. We will detail a solutuon method to the Nash equilibrium in the following section.

S2.D. Defining the Constrained Multi-Agent Nash Equilibrium. If the infinite recursion from Algorithm 1 yields a stationary point, this is the Nash equilibrium of the system. It can be formulated as

$$\begin{aligned} \mathbf{u}_i^*(\mathbf{u}_{\neg i}) = \arg\max_{\mathbf{u}_i} \; & G_i(\mathbf{x}^0, \mathbf{u}_{\neg i}^*(\mathbf{u}_i), \mathbf{u}_i, \varphi_i), \quad \forall i \in m, \\ \text{s.t. } & \mathbf{x}^{k+1} = \mathcal{F}(\mathbf{x}^k, \mathbf{u}^k), \\ & c_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_{\neg i}^*(\mathbf{u}_i)) \leq 0, \end{aligned} \tag{s11}$$

which contains m separate optimizations dependent on each other. Instead of solving for the Nash equilibrium in the iterative fashion described in Algorithm 1, we can reformulate the optimization with KKT constraints to

$$\mathbf{u}^* = \arg\max_{\mathbf{u},\mathbf{k}} \sum_{i=1}^{m} G_i(\mathbf{x}^0, \mathbf{u}, \varphi_i), \tag{s12}$$

$$\text{s.t. } \mathbf{x}^{k+1} = \mathcal{F}(\mathbf{x}^k, \mathbf{u}^k),$$

$$c_j(\mathbf{x}, \mathbf{u}) \leq 0, \tag{s13}$$

$$\nabla_{\mathbf{u}_j} G_j(\mathbf{x}^0, \mathbf{u}, \varphi_j) + \mathbf{k}_j^\top \nabla_{\mathbf{u}_j} c_j(\mathbf{x}, \mathbf{u}) = 0, \tag{s14}$$

$$\mathbf{k}_j^\top c_j(\mathbf{x}, \mathbf{u}) = 0 \;, \tag{s15}$$

$$\mathbf{k}_j \geq 0, \qquad \forall j \in m \tag{s16}$$

where Eq. (s14) defines the stationarity condition, Eq. (s15) the complementary slackness, Eq. (s16) the dual and Eq. (s13) the primal feasibility constraints, and $\mathbf{k}$ is the vector of dual variables. The sum over G_i in the objective ensures solving for a maximum. We therefore avoid to add an additional constraint to enforce negative definiteness of the Hessians to ensure the solver to yield maxima. The approach is similar to the previously described reformulation of the Stackelberg bilevel optimization in Section S2.B. The reformulation of the optimization enables solving with state-of-the-art nonlinear optimizers.

The Nash equilibrium formulation Eq. (s11) and its reformulation Eq. (s12) readily generalizes to the multi-agent case. Furthermore, it provides a more compact formulation than the recursive Stackelberg multi-agent game.

Unlike Eq. (s3), the system dynamics evolve according to

$$\mathbf{x}^{k+1} = \begin{bmatrix} \mathbf{x}_1^{k+1} \\ \dots \\ \mathbf{x}_m^{k+1} \end{bmatrix} = \begin{bmatrix} \mathcal{F}_1\left(\mathbf{x}_1^k, \mathbf{u}_1^{k,*}(\mathbf{u}_{\neg 1}^k)\right) \\ \dots \\ \mathcal{F}_m\left(\mathbf{x}_m^k, \mathbf{u}_m^{k,*}(\mathbf{u}_{\neg m}^k)\right) \end{bmatrix}, \tag{s17}$$

where the dynamics of each agent depends on the control policies of all other agents.

The Nash equilibrium has a number of direct advantages towards our goal of socially-compliant control design. Due to its non-hierarchical structure, no leader and followers need to be defined, creating symmetric interactions based on simultaneous choices between agents. Since all agents' policies depend on all other agents, the outcome yields a negotiation process that mimics the resolution of social dilemmas between agents.

S2.E. Nash Equilibrium Solver Performance.

S2.E.1. Computation Times in Experiments. We solve the optimization problem of Eq. (s11) in a receding horizon fashion, i.e., at any timestep k we find the optimal control policy for the autonomous vehicle incorporating the expected and observed controls of the other vehicles. The vehicle then executes the control action $\mathbf{u}^{k,*}$. We set up the optimization problem with CasADi (6), a software framework for nonlinear optimization and optimal control including automatic differentiation. We employed IPOPT (7), a widely used interior point solver, to solve the resulting nonlinear optimization problem. All experiments were conducted on a single core of an AMD Ryzen 7 1700X @3.4Ghz.

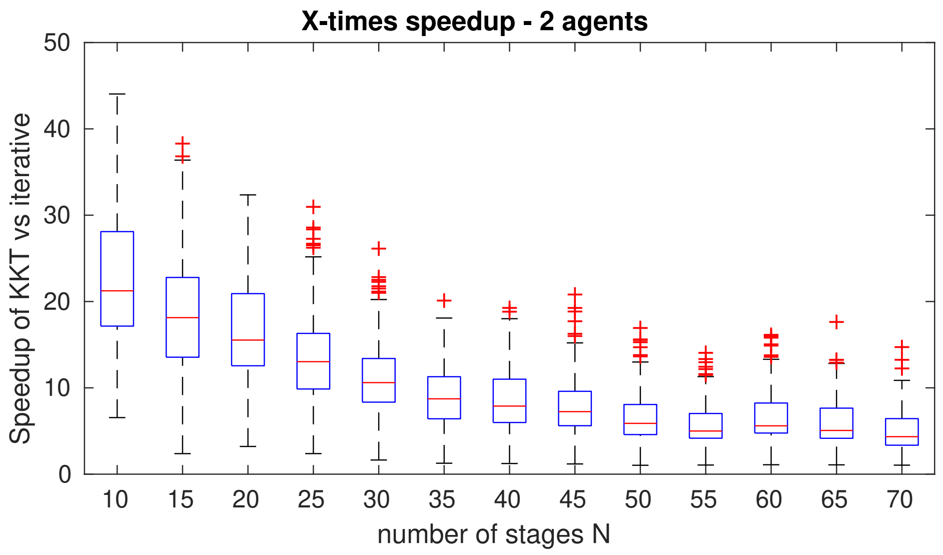
For experiments in simulation (left-turn across traffic and 3- to 2-lane highway merge), the interior point solver was capable of solving the optimization problem in less than $100ms$. The planning horizon consists of 20 steps of $\Delta t = 0.2s$, for a total horizon time of $4s$. In our experiments on congested highway driving based on the NGSIM dataset the interior point solver was capable of solving the optimization problem in less than $100ms$ for up to 4 controlled vehicles and up to 10 dynamic obstacles. A sensitivity analysis on total number of vehicles versus ego vehicle solver improvement showed that adding more cars did not have a major influence on the controlled ego vehicles. Thus, we account for the closest neighboring vehicles to the ego vehicle, which are the vehicles involved in the interaction. Furthermore, adding additional vehicles will increase the solution computation time, and may compromise real-time solutions without contributing a performance increase. We can quantify the level of interaction between vehicles by observing the norms of the Hessian blocks corresponding to the pairs of cars in the experiment. Details of this Hessian-based analysis are presented in Section S4. We observe that the norms of the Hessian blocks corresponding to cars very far away from the ego vehicle are very small, such that their level of interaction is low and they can be approximately treated as independent, which supports our argument.

S2.E.2. KKT vs Iterative Best Response Performance Comparison. To compare the performance of the KKT-based approach (see Section S2.D) to the iterative best response approach to compute the multi-agent Nash equilibrium, we created a series of benchmark problems. An exemplary problem is shown in Figure S7. A variable number of vehicles start from randomized initial conditions (position, heading, speed) with the goal to reach randomized goal positions. The cost function consists of control costs, collision avoidance, and distance to the goal position at the final time. These examples are used to illustrate the performance variations between our two methods of solving for the optimal control policies.

Both approaches were started with an initial guess computed based on independently optimized trajectories where all other agents' inputs were set to zero, which results in constant velocity. The computation time for the initial guess was included in the reported total solve time. Both approaches were run until stationarity conditions were met sufficiently (i.e., the magnitude of the gradients of the agents' cost functions were below a given threshold). Solutions of both solvers were checked for consistency and yielded the same trajectories up to some tolerance if initialized in the region of attraction of the same homotopy class. This verifies the correctness of our KKT-based solution approach. All experiments were repeated 500 times.

The KKT-based approach was consistently faster than the iterative based approach. While the solution time for the KKT-based approach was more than one order of magnitude faster on average, the difference was more prominent for shorter time horizons (10-20 times for $N < 30$) and lower for longer time horizons ($N > 40$), where N is the number of steps in the horizon. Figure S5 shows the relative speedup of the KKT-based approach for $m = 2$ and $m = 6$ agents, respectively. Figure S6 shows the solver times in seconds for both approaches across varying time horizons and number of agents.

While the iterative best response method approaches the Nash Equilibrium without the guidance of any gradients, the KKT-based method is able to use gradients to move towards the Nash Equilibrium faster. We find that as the number of interactions increases in a scenario, the greater is the performance advantage of the KKT-based approach. An explanation is that if all agents' solution trajectories are independent, then our method of computing an initial guess already yields the Nash Equilibrium solution. Thus, both approaches terminate immediately, the overall solution time is dominated by computing the initial guess, and both methods are equally fast. The amount of interaction of agents can be quantified by observing the norm of the non-diagonal blocks of every agents' Hessian, described in further detail in Section S4.



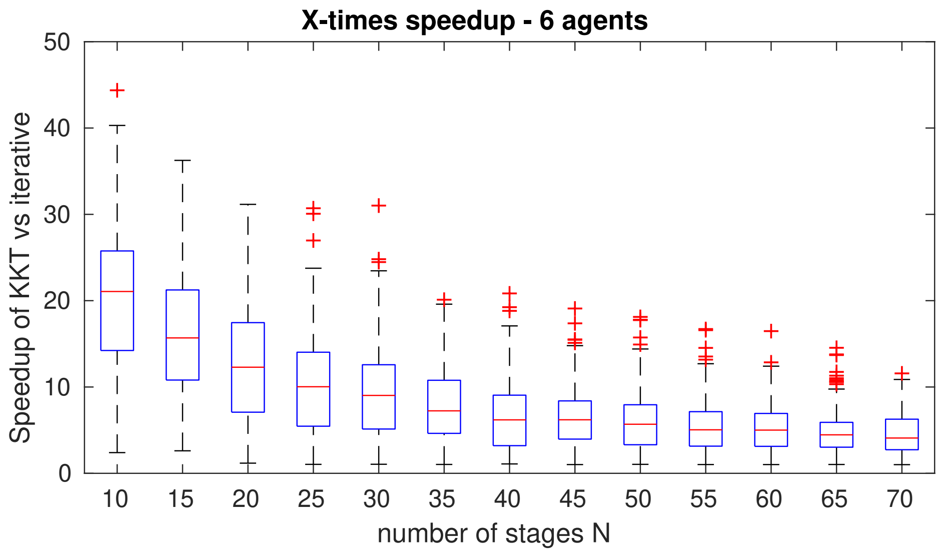


Fig. S5. Solver performance speedup of the KKT approach over the iterative best response approach to solving the multi-agent Nash equilibrium problem. **TOP:** Performance speedup for $m = 2$ agents, with speedup indicated as an $X-$times factor for the KKT approach over the iterative best response approach across varying time horizons of N stages. **BOTTOM:** Performance speedups for $m = 6$ agents.

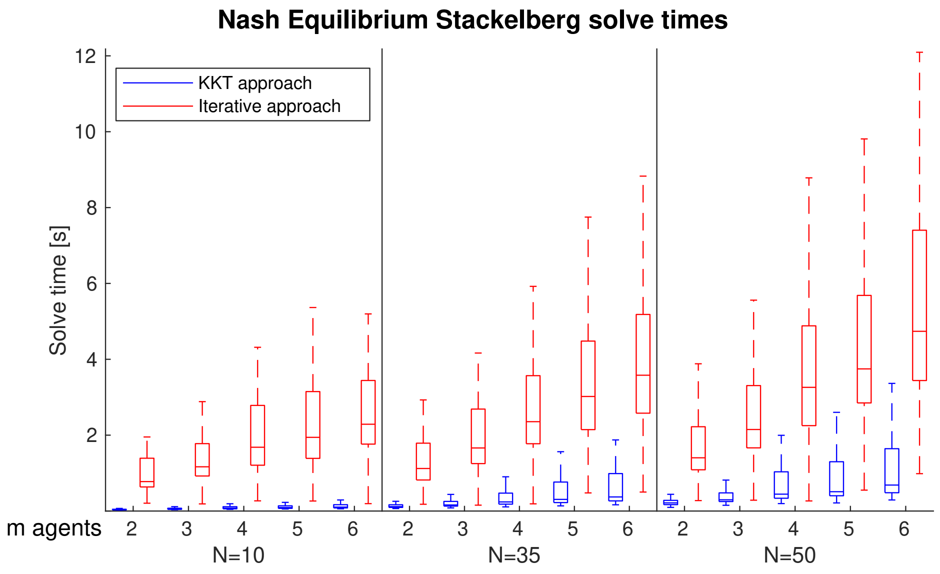


Fig. S6. Solver runtimes for both the KKT and iterative best response approaches for varying N-stage time horizon and $m = \{2, .., 6\}$ agents. Each approach is plotted with its median solver time and edges of the error bars indicating the 25th and 75th percentiles over the 500 trials. The KKT approach for solving the multi-agent Nash Equilibrium problem is significantly faster than using an iterative best response solver.

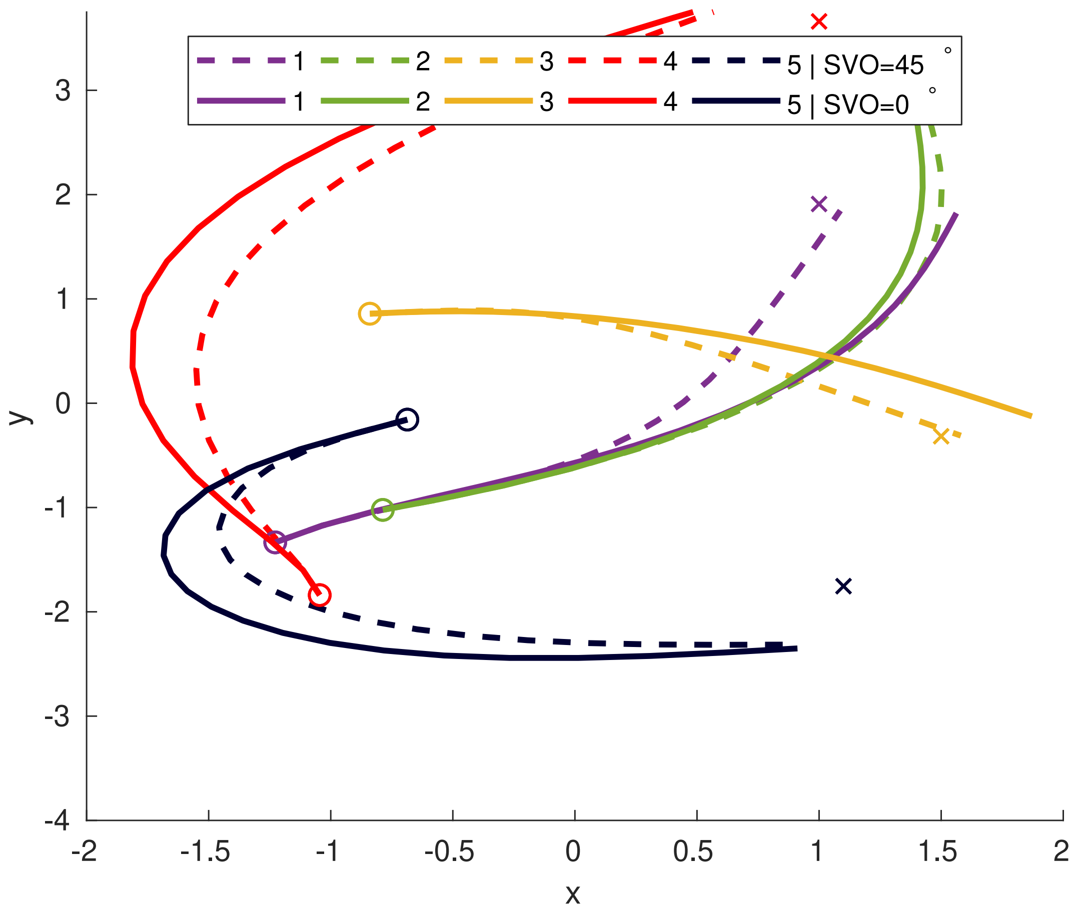


Fig. S7. Visualization of solution of the Nash Equilibrium problem. 5 vehicles start from their start positions and initial headings. Start positions are indicated by circles with the goal of reaching their respected goal locations shown by crosses. Visualized are the planned trajectories over the planning time horizon. The objective function consists of control costs (acceleration and steering), collision avoidance, and distance to goal location at the final time. Due to the trade off of control costs and distance to goal location, as well as dynamic constraints on maximum steering angles and acceleration, the vehicles may not always have reached the goal location at the end of the planning horizon yet. We also compare the influence of changing the SVO of all vehicles from egoistic ($\varphi = 0$) to prosocial ($\varphi = \frac{\pi}{4}$). We find that the vehicles execute cooperative trajectories, where some agents modify their actions to allow others to improve their performance. The overall reward increases by 24%. Interestingly, all agents improve, especially 1 and 2 since they switch sides. All other agents receive cascading improvements: 3 can move directly to its goal location and does not have to wait until 1 and 2 have passed. 4 can take a more direct path since 1 and 3 are already out of its way. Agent 5 is only very weakly influenced by the interaction and does not change its trajectory and reward.

S3. Learning Human Behavior and Human Decision Making Policies

In Section 3, we introduced our model of human drivers' decision making using a utility-maximizing policy. Here, we elaborate in more detail on the specifics of our model. To compute the utility function, we need an underlying reward function r_i for the human driver. In general, one can find r_i through Inverse Reinforcement Learning (IRL) (8–11) by learning from human demonstrations. We briefly describe our approach of learning the reward and utility function in Section S3.B. While we define the reward functions as linear combinations of weighted features, the concept of SVO is general enough to utilize reward functions of any form, such as popular general function approximators like neural networks.

S3.A. Reward Function Features in Human Driving. The agents define their utility function based on a reward function. Consistent with reinforcement learning literature, we define the reward functions $r_i(\cdot)$ as linear combinations of weighted features of the environment,

$$r_i(\mathbf{x}, \mathbf{u}) = \theta^T \psi(\mathbf{x}, \mathbf{u}), \qquad [s18]$$

where $\psi_i(\cdot)$ define the features with weights θ_i. We employ features that allow us to quantify

- **road progress**, found by projecting the driven velocity onto the road's tangent;
- **comfort**, defined by quadratically penalizing high steering and acceleration controls;
- **desired velocities** within speed limits;
- **penalizing tailgating** of other vehicles, in the form of orientation aligned Gaussians;
- **collision avoidance**, as in avoiding close lateral and longitudinal distances to other vehicles;

- **centered positions** within the lanes; and
- **road departures** for leaving the drivable space.

The cost function encoding the road in the six-lane NGSIM merge scenario is visualized in Figure S9. The merge lane angles into the adjacent lane. This costmap has lowest values within the lanes, with higher values indicating features to avoid. The cost function of the three-lane to two-lane merge including lane termination is pictured in Figure S8. We use this example in our simulations in Section 4.B.

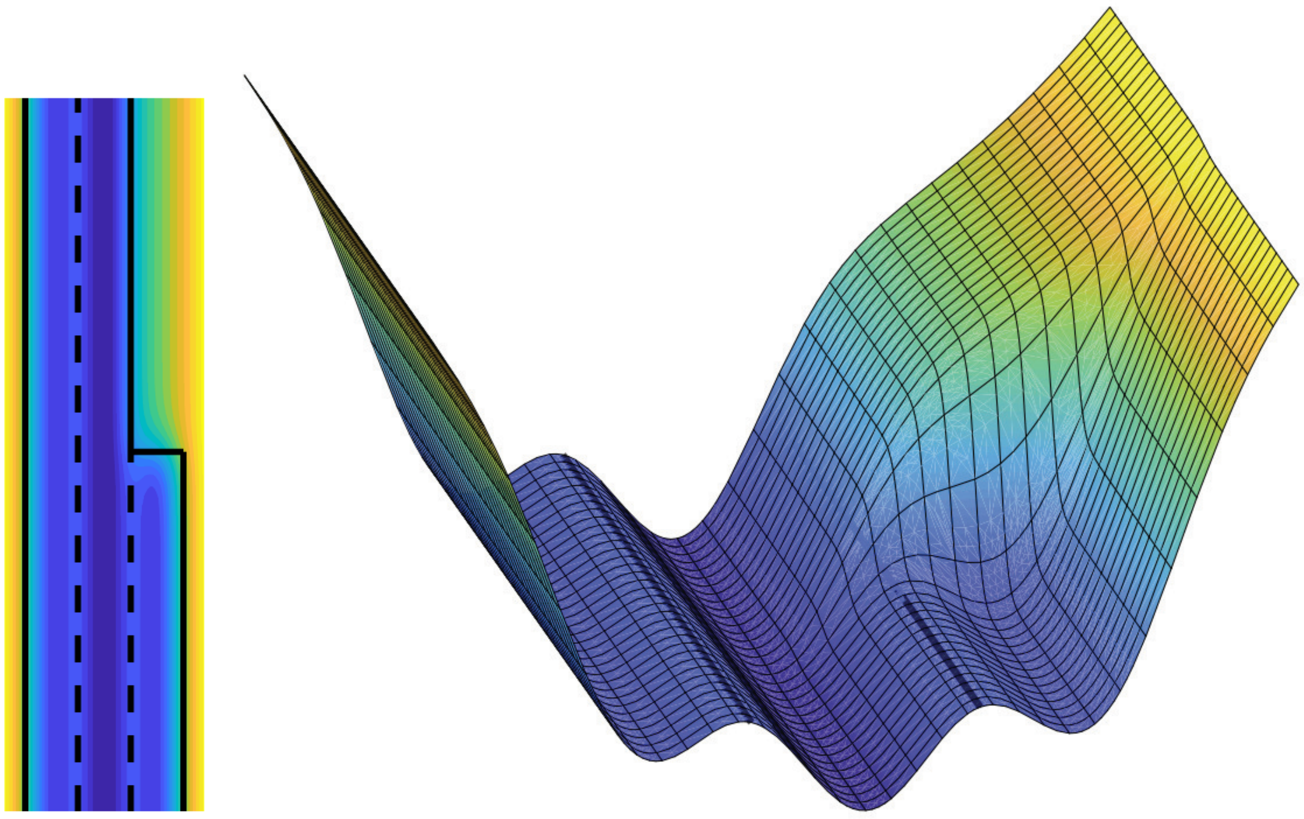


Fig. S8. An illustration of the cost map encoding the reward function Eq. (s18) for a three-lane to two-lane merging scenario. Less-desirable actions incur a higher cost. Note the area outside of the lanes increases in cost. This cost map is used in our simulations of autonomous merging where all cars are autonomous.

S3.B. Maximum Entropy Inverse Reinforcement Learning Model. Inverse Reinforcement Learning, also referred to as inverse optimal control, is the problem of recovering an unknown reward or utility function from a Markov decision process. As discussed in Section 3.A, human decision-makers are reasonably modeled as utility maximizing agents. Following this direction, the Maximum Entropy Inverse Reinforcement Learning model (11) models the probability of actions or controls $\mathbf{u}$ to be proportional to the exponential of the rewards encountered along the trajectory:

$$P(\mathbf{u}_i|\mathbf{x}^0, \mathbf{u}_{\neg i}, \varphi_i, \theta) = \frac{1}{Z}\exp\left(G_i(\mathbf{x}^0, \mathbf{u}, \varphi_i)\right), \tag{s19}$$

Therefore, less rewarding actions are exponentially less likely. Here, Z is the normalization function which can be evaluated by dynamic programming (11) which poses a practical challenge due to high computational complexity. This is particularly true for long time horizons and high dimensional systems with continuous control inputs.

S3.C. Learning Human Reward Functions from Driving Data. The following outlines how we learn the human reward function from the utility function. We use the notation $G(\mathbf{u})$ to refer to the sum of utilities g_i along the trajectory defined by $(\mathbf{x}^0, \mathbf{u})$. For the purpose of this section, we use G instead of G_i, as the process is general to all agents. Consider

$$P(\mathbf{u}|\mathbf{x}^0) = \exp\left(G(\mathbf{u})\right)\left[\int \exp\left(G(\tilde{\mathbf{u}})\right) d\tilde{\mathbf{u}}\right]^{-1}. \tag{s20}$$

To approximate the intractable normalizer, the authors in (9) apply the Laplace transform, which corresponds to assuming that the demonstration performs a local optimization when choosing the actions $\mathbf{u}$. A local approximation of the utility function as

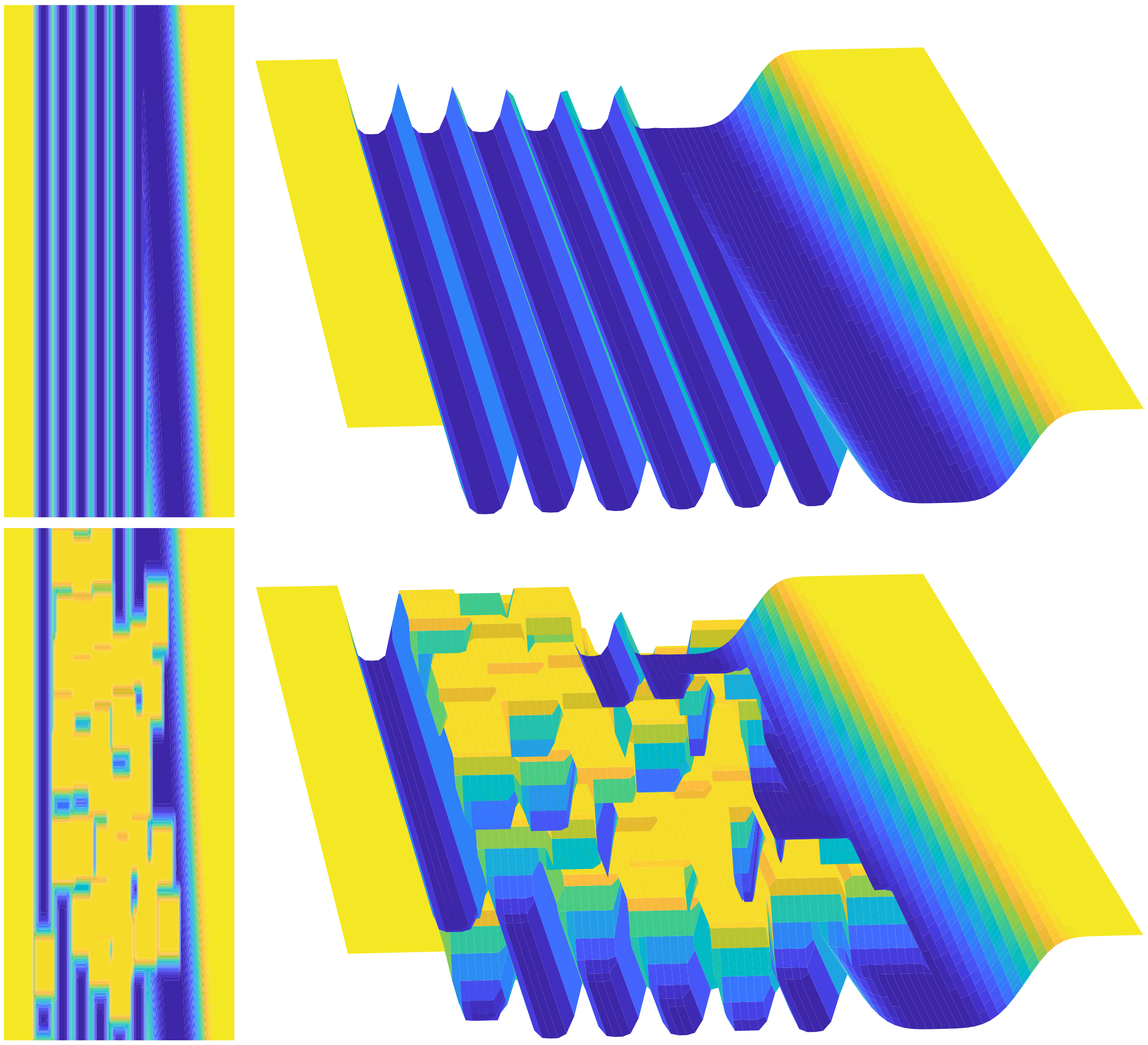


Fig. S9. An illustration of the cost map encoding the reward function Eq. (s18) for a six-lane highway and an adjacent merge lane from the NGSIM data set. Less-desirable actions incur a higher cost. **TOP:** Illustration of the cost map without vehicles. Lanes are naturally encoded with a lower cost, which rewards lane-keeping. The boundaries of the highway have a significantly higher cost. The merge lane is wider and therefore the cost basin is wider as well. **BOTTOM:** Cost map illustrated with vehicles and obstacles. For an autonomous ego vehicle in congestion, it will avoid collisions with other vehicles by keeping to low-cost areas of the map.

a second order Taylor expansion of $G(\mathbf{u})$ around $\mathbf{u}$ yields

$$G(\tilde{\mathbf{u}}) = G(\mathbf{u}) + (\tilde{\mathbf{u}} - \mathbf{u})^T \frac{\partial G}{\partial \mathbf{u}} + \frac{1}{2}(\tilde{\mathbf{u}} - \mathbf{u})^T \frac{\partial^2 G}{\partial \mathbf{u}^2}(\tilde{\mathbf{u}} - \mathbf{u}). \tag{s21}$$

We refer to $\frac{\partial G}{\partial \mathbf{u}}$ as $\mathbf{g}$ and $\frac{\partial^2 G}{\partial \mathbf{u}^2}$ as $\mathbf{H}$. Inserting the approximation in Eq. (s21) into the exponent in Eq. (s20) allows us to evaluate the integral of the normalization factor in closed form. This yields a tractable way of evaluating the likelihood including normalization factor,

$$\begin{aligned} P(\mathbf{u}|\mathbf{x}^0) &= \exp(G(\mathbf{u})) \left[\int \exp\left(G(\tilde{\mathbf{u}})\right) \mathrm{d}\tilde{\mathbf{u}} \right]^{-1} \\ &\approx \exp(G(\mathbf{u})) \left[\int \exp\left(G(\mathbf{u}) + (\tilde{\mathbf{u}} - \mathbf{u})^T \mathbf{g} + \frac{1}{2}(\tilde{\mathbf{u}} - \mathbf{u})^T \mathbf{H}(\tilde{\mathbf{u}} - \mathbf{u}) \right) \mathrm{d}\tilde{\mathbf{u}} \right]^{-1} \\ &= \exp\left(\frac{1}{2}\mathbf{g}^T \mathbf{H}^{-1} \mathbf{g} \right) |-\mathbf{H}|^{\frac{1}{2}} (2\pi)^{-\frac{\dim(\mathbf{u})}{2}}. \end{aligned} \tag{s22}$$

The assumption of local optimality is strictly less restrictive than the assumption of global optimality. In contrast to global methods, the local method described in (9) scales well with task dimensionality and long time horizons. Furthermore, by only updating the utility function locally, only locally optimal demonstrations are sufficient. The log-likelihood of Eq. (s22) is

$$\mathcal{L} = \frac{1}{2}\mathbf{g}^T \mathbf{H}^{-1} \mathbf{g} + \frac{1}{2} \log |-\mathbf{H}| - \frac{\dim(\mathbf{u})}{2} \log 2\pi. \tag{s23}$$

The learning process consists of maximizing the likelihood Eq. (s22) for parameters θ in G and therefore $\mathcal{L}(\theta)$,

$$\theta^* = \arg\max_{\theta} \mathcal{L}(\theta). \tag{s24}$$

This can be done with standard gradient and non-gradient based optimization techniques. The resulting θ^* are the maximum Entropy fit parameters best explaining the observed trajectories in the given dataset. Employing the learned parameters allows not only to observe the utility best explaining the observed behavior but also to predict and replicate human driver trajectories by optimizing their utility over their future actions. The result is a learned utility maximizing prediction of human behavior.

In practice, additional regularization schemes to ensure convergence and invertability of the Hessian $\mathbf{H}$ are needed. A method of solving for the computationally challenging Hessian inversion in linear-time under the restriction of linearized dynamics is described in (9).

S3.D. SVO-Based Utility Function Formulation. Recall that φ denotes the angle of SVO preference of an agent, as illustrated by the SVO ring in Figure 1, and we can quantify a persons preference to trade off own rewards for other people's rewards as a measure of their SVO preference. To incorporate the SVO preference into the utility function, we generalize the utility function from Eq. (1) to m agents in Eq. (s1) and repeated here as

$$g_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_{\neg i}, \varphi_i) = \frac{1}{m-1} \sum_{j \in \neg i} \left[\cos(\varphi_i) r_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_j) + \sin(\varphi_i) r_j(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_j) \right],$$

where $\mathbf{x}$ is the state of all agents, and $r_i(\cdot)$ and $r_j(\cdot)$ are the reward functions. Note that Eq. (s1) weights agent i's reward against the other agents' rewards according to their SVO preference φ_i. For egoistic human drivers with $\varphi_i \approx 0$, the utility function will rely mostly on their own reward $r_i(\cdot)$ with little weight given to other agent rewards. On the other hand, altruistic human drivers with $\varphi_i \approx \frac{\pi}{2}$ will choose control policies that prioritize other agents' rewards. When considering adversarial or sadistic agents, this behavior manifests as minimizing the rewards of the other agents. Note that mapping of the SVO preference into the utility function is quite intuitive: In the case of two agents it encodes how willing the agent is to give up an increase in their personal reward, $\frac{\partial r_1(\mathbf{x}, \mathbf{u}_1^*, \mathbf{u}_2^*)}{\partial \mathbf{u}_1}$, for an increase in the autonomous system's reward, $\frac{\partial r_2(\mathbf{x}, \mathbf{u}_1^*, \mathbf{u}_2^*)}{\partial \mathbf{u}_1}$ for some $\mathbf{u}_1$. We can see this by investigating the case for two agents, a horizon of $N = 1$: The utility-maximization yields

$$\left. \frac{\partial g(\mathbf{x}, \mathbf{u}_1, \mathbf{u}_2, \varphi_1)}{\partial \mathbf{u}_1} \right|_{\mathbf{u}=\mathbf{u}^*} = \left. \left(\cos(\varphi_1) \frac{\partial r_1(\mathbf{x}, \mathbf{u}_1, \mathbf{u}_2)}{\partial \mathbf{u}_1} + \sin(\varphi_1) \frac{\partial r_2(\mathbf{x}, \mathbf{u}_1, \mathbf{u}_2)}{\partial \mathbf{u}_1} \right) \right|_{\mathbf{u}=\mathbf{u}^*} = 0, \tag{s25}$$

and solving for the SVO preference

$$\varphi_1 = \arctan 2 \left. \left(-\frac{\partial r_1(\mathbf{x}, \mathbf{u}_1, \mathbf{u}_2)}{\partial \mathbf{u}_1}, \frac{\partial r_2(\mathbf{x}, \mathbf{u}_1, \mathbf{u}_2)}{\partial \mathbf{u}_1} \right) \right|_{\mathbf{u}=\mathbf{u}^*}. \tag{s26}$$

This relationship describes the tangent on the SVO circle corresponding to the SVO preference φ_1, which is another intuitive interpretation. In the multi-car case over any horizon N, the same argument can be made: φ indicates how willing an agent is to give up their reward for the reward of (multiple) others.

A visualization of the solution of the Stackelberg Nash Equilibrium problem with varying SVOs can be found in the main text. We show another example in Figure S7 where 5 agents start from their start positions indicated by circles with the goal of reaching their respective goal locations shown by crosses. As in the previous experiments on solving the multi-agent game theoretic formulation the objective function consists of control costs (acceleration and steering), collision avoidance, and distance to goal location at the final time. We can now compare the impact of changing the SVO of all vehicles from egoistic ($\varphi = 0$) to prosocial ($\varphi = \frac{\pi}{4}$). We find that the vehicles execute cooperative trajectories, where some agents modify their actions to allow others to improve their performance. As displayed in Table S2, the overall reward increases by 24%. Interestingly, all agents improve. The largest improvement is observed in agents 1 and 2 since they switch sides. Because of this, all other agents receive cascading improvements: 3 can move directly to its goal location and does not have to wait until 1 and 2 have passed. Subsequently, 4 can take a more direct path since 1 and 3 are already out of its way. Agent 5 is only very weakly influenced by the interaction and does not change its trajectory and reward.

For our simulation experiments we set the SVO of the autonomous vehicle to $\varphi = \frac{\pi}{4}$, as it is beneficial to design a prosocial autonomous vehicle and encourage cooperativeness among all agents.

Agent	1	2	3	4	5	Σ
Cost SVO 0	1,419	1000	762	686	870	4,737
Cost SVO $\frac{\pi}{4}$	1064	533	458	679	866	3,600
Δ Improvement	355	467	304	7	4	1137
% Improvement	25.0	46.7	39.9	1.0	0.46	24.0

Table S2. Change in reward for all agents due to change in SVO from egoistic to prosocial. Due to the cascading beneficial effect prosocial behavior has on all agents, by modifying their actions to allow others to improve performance, we can observe a decrease in the sum of all agents' costs. The result is cooperative behavior.

S3.E. Estimating SVO by Observing Driving. In the previous sections, we established how to interactively plan and predict given an agent's SVO φ. If the autonomous vehicle does not know the other agent's SVO, it will need to estimate this quantity to act accordingly.

We present two probabilistic measurement functions to estimate the likelihood of an agent's SVO and integrate both into recursive filters to achieve good estimation results. We compare results, and give an intuition why actively controlling the SVO of an autonomous vehicle can be beneficial in traffic negotiation. The two methods for formulating the measurement likelihood are:

1. Employing the previously described method to solve for Nash Equilibria and planning trajectories with variable SVOs from a past point of view and estimating which SVO fits closest to the observed part of trajectories. Intuitively, proposed SVOs closer to the true SVOs will yield closer matching trajectories.

2. Using a Maximum Entropy model, frequently employed in Inverse Reinforcement Learning (IRL) to generate the SVO estimate based on past observations.

While the Maximum Entropy likelihood function only employs past measurements, the prediction based method takes into account both past and *future* trajectories. The motivation is intuitive: Human drivers plan their actions into the future, such that they have their own prediction about the future in mind when deciding how to allocate resources among themselves and others. Therefore, it is reasonable to assume that it is necessary to take this effect into consideration when estimating a driver's SVO. The advantage comes at the cost of increased computational burden since for each prediction based likelihood evaluation the game theoretic optimization needs to be executed. The Maximum Entropy likelihood function on the other hand is computationally efficient since no optimization is necessary. Additionally, SVO likelihoods can be evaluated independently, therefore scaling linearly with the number of vehicles. Whereas in the prediction based approach likelihoods can only be evaluated jointly over all SVOs and the method scales exponentially in the number of agents.

To integrate both likelihood functions in a recursive filtering framework we need to formulate the SVO dynamics, which can be also considered as the SVO transition probability. We want to make minimal assumptions on how SVO preferences may change over time, thus we formulate the SVO dynamics for both approaches as a Gaussian distribution on a circle, or more precisely, according to the von Mises distribution

$$p(\varphi_i^k | \varphi_i^{k-1}) \propto \mathcal{M}(\varphi_i^k | \varphi_i^{k-1}, \sigma^2). \qquad \text{[s27]}$$

The von Mises distribution is a close approximation to the wrapped normal distribution, which is the circular analogue to the normal distribution.

We start from the classical filtering problem and formulate the nonlinear filtering equations over r state measurements instead of a single state measurement. To update our predictions about the SVO state, we write

$$p(\varphi^{k-r} | \mathbf{x}^{0:k-1}) = \int p(\varphi^{k-r} | \varphi^{k-r-1}) p(\varphi^{k-r-1} | \mathbf{x}^{0:k-1}) \, \mathrm{d}\varphi^{k-r}, \qquad \text{[s28]}$$

which is used in both of our approaches to estimate the SVO.

The update function to get the new distribution $p(\varphi^{k-r} | \mathbf{x}^{0:k})$ by updating $p(\varphi^{k-r} | \mathbf{x}^{0:k-1})$ based on a new measurement $\mathbf{x}^k$ in both filters is constructed as

$$p(\varphi^{k-r} | \mathbf{x}^{0:k}) = \frac{p(\mathbf{x}^{k-r:k} | \varphi^{k-r}) p(\varphi^{k-r} | \mathbf{x}^{0:k-1})}{\int_{-\pi}^{\pi} p(\mathbf{x}^{k-r} | \varphi^{k-r}) p(\varphi^{k-r} | \mathbf{x}^{0:k-1}) \mathrm{d}\varphi^{k-r}}, \qquad \text{[s29]}$$

where the measurement function $p(\mathbf{x}^{k-r:k}|\varphi^{k-r})$ is evaluated over the last r state measurements $\mathbf{x}^{k-r:k}$ instead of a single state measurement $\mathbf{x}^{k-r}$ to generate a likelihood of the SVO φ^{k-r}. We have found this modification to be necessary to generate accurate SVO estimates. The measurement function is approximated as

$$\begin{aligned}
p(\mathbf{x}^{k-r:k}|\varphi^{k-r}) &= \int p(\mathbf{x}^{k-r:k}, \varphi^{k-r+1:k}|\varphi^{k-r})\mathrm{d}\varphi^{k-r+1:k} \\
&= \int p(\mathbf{x}^{k-r:k}|\varphi^{k-r:k})p(\varphi^{k-r+1:k}|\varphi^{k-r})\mathrm{d}\varphi^{k-r+1:k} \\
&= \int p(\mathbf{x}^{k-r:k}|\varphi^{k-r:k}) \prod_{j=k-r}^{k-1} p(\varphi^{j+1}|\varphi^{j})\mathrm{d}\varphi^{k-r+1:k} \\
&\approx p(\mathbf{x}^{k-r:k}|\varphi^{k-r+1:k} = \varphi^{k-r}, \varphi^{k-r}) \\
&\approx p(\mathbf{x}^{k-r:k}|\varphi^{k-r}),
\end{aligned} \qquad [\text{s30}]$$

where, for the sake of computational tractability, we assume only a small change in SVO over the observed horizon r, $\varphi_{k-r+1:k} \approx \varphi_{k-r}$. Note that this assumption is only made in evaluating the measurement function. In general, the SVO dynamics are given by $p(\varphi_i^k|\varphi_i^{k-1})$.

Next, we present our prediction-based SVO estimation, then present our Maximum Entropy based SVO estimation. We use the notation $\hat{\mathbf{x}}$ to denote observations, and the notation $\check{\mathbf{x}}$ to denote predicted states.

S3.E.1. Prediction-Based SVO Estimation. The general idea is to observe the states $\mathbf{x}^{k-r:k}$ for r time steps in the past, we denote these measurements as $\hat{\mathbf{x}}^{k-r:k}$, to find SVO estimates $\check{\varphi}_i$ for all cars that can best explain the observations

$$p(\mathbf{x}^{k-r:k}|\boldsymbol{\varphi}^{k-r}) \propto \mathcal{N}(\mathbf{x}^{k-r:k}|\check{\mathbf{x}}^{k-r:k}(\boldsymbol{\varphi}^{k-r}), \Sigma) \qquad [\text{s31}]$$

with variance Σ, given predicted trajectories $\check{\mathbf{x}}^{k-r:k}(\boldsymbol{\varphi}^{k-r})$, following from SVOs $\boldsymbol{\varphi}^{k-r}$. Based on Eq. (s12), for given SVO values $\boldsymbol{\varphi}^{k-r} = \varphi_{1:m}^{k-r}$, we can compute estimated control trajectories $\check{\mathbf{u}}^{k-r:k+q}(\boldsymbol{\varphi}^{k})$ starting from an initial state $\mathbf{x}^{k-r}$ by solving the multi-agent game theoretic problem. Followingly we can roll out the control trajectories to arrive at the predicted state trajectories $\check{\mathbf{x}}^{k-r:k+q}(\check{\mathbf{u}}^{k-r:k+q}(\boldsymbol{\varphi}^{k-r}))$ from time $k-r$ until $k+q$. The process is detailed in Algorithm 2. Note that the state $\mathbf{x}^{k-r}$ of the system r time steps in the past from the current time k indicates the initial state of the optimization Eq. (s12); $\check{\mathbf{u}}^{k-r:k+q}$ therefore denotes the control trajectory propagated $r+q$ steps forward from the initial state of the optimization. Interestingly, trajectories are predicted q steps into the future, further than the current time k. This reflects the idea that human drivers take the reward-to-go accumulated over a future prediction into account and thus will affect their actions in the short term based on their current SVO.

Inserting the predicted states from time $k-r$ to k of $\check{\mathbf{x}}^{k-r:k+q}(\check{\mathbf{u}}^{k-r:k+q}(\boldsymbol{\varphi}^{k-r}))$, i.e. $\check{\mathbf{x}}^{k-r:k}(\check{\mathbf{u}}^{k-r:k}(\boldsymbol{\varphi}^{k-r}))$, into Eq. (s31) yields

$$\begin{aligned}
p(\mathbf{x}^{k-r:k}|\boldsymbol{\varphi}^{k-r}) &\propto p(\mathbf{x}^{k-r:k}|\check{\mathbf{x}}^{k-r:k}(\check{\mathbf{u}}^{k-r:k}(\boldsymbol{\varphi}^{k-r}))) \\
&\propto \mathcal{N}(\mathbf{x}^{k-r:k}|\check{\mathbf{x}}^{k-r:k}(\check{\mathbf{u}}^{k-r:k}(\boldsymbol{\varphi}^{k-r})), \Sigma).
\end{aligned} \qquad [\text{s32}]$$

Algorithm 2 Prediction Based SVO Measurement

```
1: Input: Observed states $\hat{\mathbf{x}}^{k-r:k}$, proposed SVO $\check{\boldsymbol{\varphi}}^{k-r}$
2: Output: $p(\mathbf{x}^{k-r:k} = \hat{\mathbf{x}}^{k-r:k}|\check{\boldsymbol{\varphi}}^{k-r})$
3: $\check{\mathbf{u}}^{k-r:k+q} \leftarrow$ Predict input based on $\check{\boldsymbol{\varphi}}^{k-r}$ and $\hat{\mathbf{x}}^{k-r}$, Eq. (s12)
4: $\check{\mathbf{x}}^{k-r:k} \leftarrow$ Forward propagate $\check{\mathbf{u}}^{k-r:k}$ from initial state $\hat{\mathbf{x}}^{k-r}$ based on dynamics
5: $p(\mathbf{x}^{k-r:k} = \hat{\mathbf{x}}^{k-r:k}|\check{\boldsymbol{\varphi}}^{k-r})$   Evaluate likelihood Eq. (s32)
```

We can now formulate a particle filter as described in Algorithm 3. We chose a particle filter to make the least assumptions about the posterior distribution. Additionally, we expect the posterior to be multimodal, since actions can not always be interpreted unambiguously. Future work may explore other, potentially more efficient filtering methods of estimating the SVO of other drivers.

Algorithm 3 SVO Particle Filter Update

```
1: Input: $m$ particles $\check{\boldsymbol{\varphi}}^{k-r-1}$, corresponding weights $w^{k-r-1}$, and observations $\hat{\mathbf{x}}^{k-r:k}$
2: Output: $\check{\varphi}, \sigma^2_\varphi, \check{\boldsymbol{\varphi}}^{k-r}, w^{k-r}$
3: for all m particles do
4:     Sample $\check{\boldsymbol{\varphi}}^{k-r}_{[i]} \leftarrow \mathcal{M}(\check{\boldsymbol{\varphi}}^{k-r}_{[i]} | \check{\boldsymbol{\varphi}}^{k-r-1}_{[i]}, \sigma^2)$
5:     Update $w^{k-r}_{[i]} \leftarrow w^{k-r-1}_{[i]} \times p(\hat{\mathbf{x}}^{k-r:k}_{[i]} | \check{\boldsymbol{\varphi}}^{k-r}_{[i]})$, Eq. (s32)
6: Normalize $w^{k-r} \leftarrow w^{k-r} / \sum_{i=1}^{N} w^{k-r}_{[i]}$
7: if $1/\sum_{i=1}^{N} (w^{k-r}_{[i]})^2 < 0.5N$ (Sample impoverishment) then
8:     Resample($\check{\boldsymbol{\varphi}}^{k-r}$, $w^{k-r}$)
9: Compute $\check{\varphi} \leftarrow \sum_{i=1}^{N} w^{k-r}_{[i]} \check{\boldsymbol{\varphi}}^{k-r}_{[i]}$
10: Compute $\sigma^2_\varphi \leftarrow \sum_{i=1}^{N} w^{k-r}_{[i]} (\check{\boldsymbol{\varphi}}^{k-r}_{[i]} - \mu_\varphi)^2$
```

We first initialize the particles $\check{\boldsymbol{\varphi}}^0$ with random weights w^0. During the particle filter update step at time k, the particles are then perturbed in Step 3 according to the dynamics $\mathcal{M}(\check{\boldsymbol{\varphi}}^{k-r} | \check{\boldsymbol{\varphi}}^{k-r-1}, \sigma^2)$ and scored with the measurement function $p(\mathbf{x}^{k-r:k} | \check{\boldsymbol{\varphi}}^{k-r})$ detailed above. Step 6 triggers resampling if the effective number of particles $1/\sum_{i=1}^{N} (w^{k}_{[i]})^2$ is lower than half of the total number of particles. Subsequently, we compute the weighted mean and weighted standard deviation of the posterior distribution in Steps 8 and 9 respectively.

S3.E.2. Maximum Entropy Model for SVO Estimation. Inspired by the Maximum Entropy model popular in the inverse reinforcement learning literature described in Section S3.B, we can treat the SVO as a parameter to be estimated and pose the measurement likelihood function as

$$\begin{aligned} p(\mathbf{x}^{k-r:k} | \boldsymbol{\varphi}^{k-r}) &\propto p(\mathbf{u}^{k-r:k}(\mathbf{x}^{k-r:k}) | \boldsymbol{\varphi}^{k-r}) \\ &\propto \exp(G(\mathbf{u}^{k-r:k}, \boldsymbol{\varphi}^{k-r})) \left[\int \exp\left(G(\tilde{\mathbf{u}}, \boldsymbol{\varphi}^{k-r}) \right) \mathrm{d}\tilde{\mathbf{u}} \right]^{-1} \\ &\propto \exp\left(\frac{1}{2} \mathbf{g}^T \mathbf{H}^{-1} \mathbf{g} \right) | - \mathbf{H}|^{\frac{1}{2}} (2\pi)^{-\frac{\dim(\mathbf{u})}{2}}. \end{aligned} \qquad [s33]$$

Since the observed controls $\hat{\mathbf{u}}^{k-r:k}$, consisting of steering and acceleration inputs are not directly observable for other cars they have to be inferred from the state trajectory $\hat{\mathbf{x}}^{k-r:k}$. This can be done approximately but might come at the risk of increased noise since the inputs are essentially derivatives of the truly observed states which are themselves subject to noise. Integration into a filtering framework is therefore necessary. The inverse of the Hessian can be computed in linear time (9) with respect to the length of the time horizon r. Nonetheless, care needs to be taken since the second order Taylor expansion employed to make the evaluation of the partition function of the likelihood tractable is only valid close to the true value.

Due to its low computational complexity we can rely on a histogram filter to fully capture multiple hypothesis over the full SVO ring without the risk of sample impoverishment. The filtering update process is outlined in Algorithm 4. Line 3 propagates the dynamics forward, distributing probability mass from each histogram bin to all other histogram bins according to the dynamics $p(\varphi^{k-r} | \varphi^{k-r-1}, \sigma^2)$. Similarly, line 4 distributes probability mass from each histogram bin to all other bins according to the measurement function Eq. (s33).

Algorithm 4 SVO Histogram Filter Update

```
1: Input: $m$ discretizations $\check{\boldsymbol{\varphi}}^{k-r-1}$ , corresponding weights $w^{k-r-1}$, and observed states $\hat{\mathbf{x}}^{k-r:k}$
2: Output: $\check{\varphi}, \sigma^2_\varphi, \check{\boldsymbol{\varphi}}^{k-r}, w^{k-r}$
3: Dynamics update $w^{k-r} \leftarrow w^{k-r-1} \times p(\check{\boldsymbol{\varphi}}^{k-r} | \check{\boldsymbol{\varphi}}^{k-r-1}, \sigma^2)$, Eq. (s27)
4: Measurement update $w^{k-r} \leftarrow w^{k-r} \times p(\hat{\mathbf{x}}^{k-r:k} | \check{\boldsymbol{\varphi}}^{k-r})$, Eq. (s33)
5: Normalize $w^{k-r} \leftarrow w^{k-r} / \sum_{i=1}^{N} w^{k-r}_{i}$
6: Compute $\check{\varphi} \leftarrow \sum_{i=1}^{N} w^{k-r}_{i} \check{\boldsymbol{\varphi}}^{k-r}_{i}$
7: Compute $\sigma^2_\varphi \leftarrow \sum_{i=1}^{N} w^{k-r}_{i} (\check{\boldsymbol{\varphi}}^{k-r}_{i} - \mu_\varphi)^2$
```

S3.E.3. SVO Estimation Results and Interpretation. Additionally to the SVO estimation results based on the Maximum Entropy model in the main text (Figure 2), we present estimation results based on the prediction based likelihood in Figure S10. We can see that the overall prediction resembles the same characteristics. First, car 2 becomes egoistic and competitive to signal the intent to merge into car 2's lane, but car 2 is egoistic as well and does not allow for the merge to proceed. Second, car 2 becomes prosocial or even altruistic and increases the gap size to allow car 1 to complete the merge. Nonetheless, we have found that the prediction-based method displays higher uncertainty and slower adaption to new measurements than the Maximum Entropy model based method.

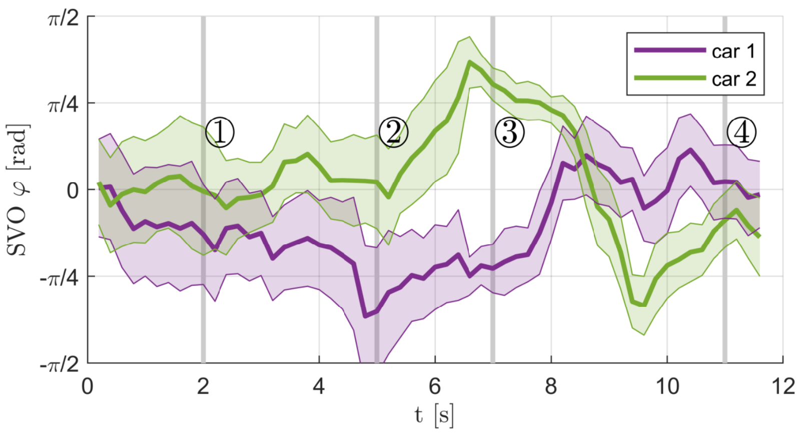


Fig. S10. Prediction based SVO estimates over time for the merge presented in Figure 2. The solid line indicates our estimate over time, with the shaded region representing the confidence bounds. Here, car 1 (purple) is attempting to merge into the lane with car 2 (green). We see that initially, car 2 does not cooperate with the merging car 1, and does not allow it to merge. After a few seconds, car 2 becomes more prosocial, which corresponds to it "dropping back" and allowing the first car to merge.

S4. Quantifying Interactions Between Agents

Here, we present a method of quantifying interactions between vehicles by computing and observing properties of an agent's Hessian. The amount of interactions between agents may vary from scenario to scenario. In traffic situations, vehicles are more likely to interact when they are closer together, but a heuristic on distance alone doesn't explain all interactions. Two vehicles driving in parallel in adjacent lanes with similar speeds are close in proximity, but do not necessarily need to interact if they choose to stay in their respective lanes. However, as soon as the cars need to change lanes or merge into a common lane, the interaction between the vehicles is much more significant.

We presented a game theoretic model of interactions, wherein each agent is modeled as maximizing a utility function G_i. Here, we show that by observing an agent's Hessian, $\mathbf{H}(G_i)$, we can quickly assess which agents are interacting. This is due to the face that the Hessian is computed as $\partial^2 G_i/\partial \mathbf{u}_i \partial \mathbf{u}_j$, which naturally encodes how one agent's utility gradient $\partial G_i/\partial \mathbf{u}_i$ (with respect to its own controls $\mathbf{u}_i$) depends on the inputs $\mathbf{u}_j$ of another agent j. Agent i's Hessian is written

$$\mathrm{H}(G_i) = \begin{bmatrix} \frac{\partial^2 G_i}{\partial u_1^2} & \frac{\partial^2 G_i}{\partial u_1 \partial u_2} & \cdots & \frac{\partial^2 G_i}{\partial u_1 \partial u_m} \\ \frac{\partial^2 G_i}{\partial u_2 \partial u_1} & \frac{\partial^2 G_i}{\partial u_2^2} & \cdots & \frac{\partial^2 G_i}{\partial u_2 \partial u_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 G_i}{\partial u_m \partial u_1} & \frac{\partial^2 G_i}{\partial u_m \partial u_2} & \cdots & \frac{\partial^2 G_i}{\partial u_m^2}, \end{bmatrix} \qquad [s34]$$

for a multi-agent game comprising m agents. Consider the multi-agent example shown in Figure S7, where agents move from initial locations (circles), to goal locations (crosses). Their cost function consists of control costs, collision avoidance costs, and a cost of the distance from the goal at the final time step. Figure S11 displays the Hessian (top) and its norms (bottom) for this scenario, with the color intensity encoding the magnitude of the norm. The larger the norm of the corresponding blocks, the stronger the interaction. Investigating agent 1's Hessian, we can see that the norm of $\frac{\partial^2 G_1}{\partial \mathbf{u}_1 \partial \mathbf{u}_3}$ is large (white) and $\frac{\partial^2 G_1}{\partial \mathbf{u}_1 \partial \mathbf{u}_2}$ is somewhat noticeable (gray). Since the Hessian is symmetric, their symmetric counterparts show the same values. We can therefore deduce that agent 1 strongly interacts with agent 3, slightly with agent 2, and not significantly with any other agents. We can verify this observation by seeing that agent 1 and 3 are on a collision path (cf. Figure S7), and agent 1 closely follows agent 2. If agent 1 would change its actions, agent 2 and 3 would also change their actions. Note that there is a difference between occurring costs because of close proximity and interactions (in the sense of influencing each others' actions): Although agent 1 and 4's paths are very close in the beginning, they are not actively choosing to interact. They start with an initial velocity and heading which can not be changed too quickly, such that they do not affect each others' actions although both agents are subject to high collision avoidance costs. Conversely, we find that agents 2 and 4 have strong interactions, as they have similar goal locations. The interaction between this pair agrees with our expectation when observing Figure S7. We also observe that agent 5 remains independent of the other agents, which moves in the opposite direction of the other agents.

We can also re-run the scenario in Figure S7, except imposing that all agents exhibit a prosocial SVO value. Figure S11 bottom shows the resulting Hessian and norms for this scenario. Here, we notice that there are more interactions across the group, indicated by the increase in brightness values across the images. In analyzing our traffic data from the NGSIM data set, we make similar observations. Two cars driving in close proximity does not necessarily yield a strong interaction. An example of this are two cars in adjacent lanes. Only if their actions influence each other, such as if one car chooses to merge into the lane of the other car, do the interactions become apparent.

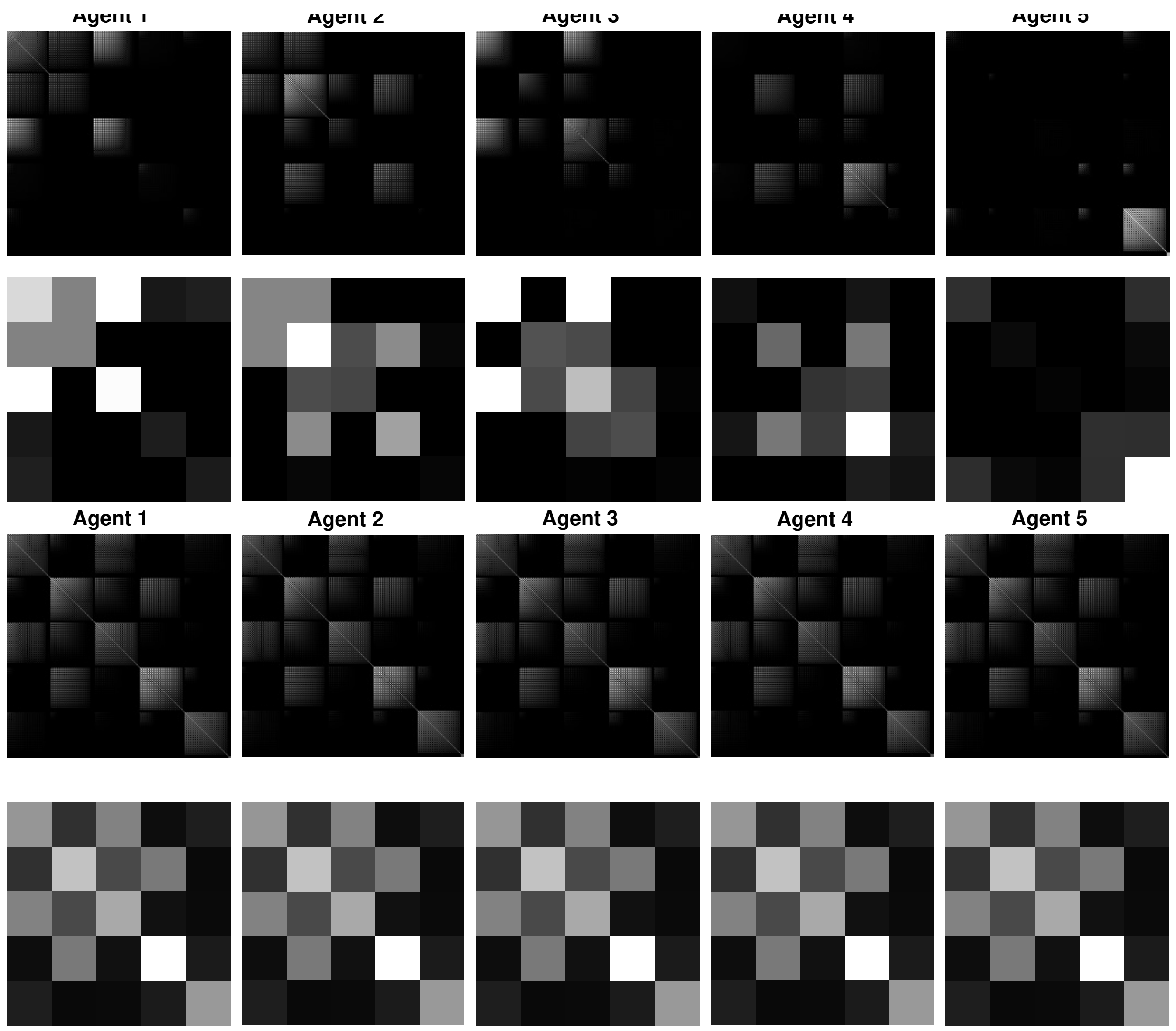


Fig. S11. TOP: Greyscale representation of the Hessian of scenario of Figure S7 at the top, and norm of block-Hessians at the bottom. Here, each pixel block corresponds to the values of the Hessian in Eq. (s34), with brighter values indicating a higher value. All agents show egoistic SVO preferences in this example. **BOTTOM:** Same greyscale representations of the Hessians and norm of block-Hessians of the same scenario as above and S7, but all agents exhibit prosocial SVO preferences. In contrast to above, the brighter squares indicate more interaction across the inter-agent pairings.

S5. Vehicle Dynamics Model

We use a simplified car model for the vehicle dynamics, with state $\mathbf{x}_i = [x_i, y_i, \phi_i, \delta_i, v_i]^T$ consisting of position x_i and y_i, orientation ϕ_i, steering angle δ_i and speed v_i. The control inputs are acceleration $u_{i,\text{acc}}$ and steering angle velocity $u_{i,\text{steer}}$. The continuous-time dynamics are given by

$$\underbrace{\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\phi}_i \\ \dot{\delta}_i \\ \dot{v}_i \end{bmatrix}}_{\dot{\mathbf{x}}_i} = \begin{bmatrix} v_i \cos(\phi_i) \\ v_i \sin(\phi_i) \\ \frac{v_i}{L_i} \tan(\delta_i) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} u_{i,\text{steer}} \\ u_{i,\text{acc}} \end{bmatrix}}_{\mathbf{u}_i}. \qquad [\text{s35}]$$

S6. NGSIM Data Set Analysis and Validation

To validate our model with human driving data, we used the Next Generation Simulation (NGSIM) data set, provided by the US Department of Transportation and Federal Highway Administration[*]. The NGSIM data set comprises four highway and

[*] Available online at: https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm

city traffic scenarios from California and Atlanta.

Our results are computed using the Interstate 80 Freeway Dataset[†], which captures the eastbound traffic in April 2005 during rush hour. Vehicle tracker data is provided for 500 meters of the freeway. The freeway has six traffic lanes, with the leftmost lane being a high-occupancy vehicle (HOV), and an on-ramp for merging traffic. Motorcycle lane-splitting, where a motorcycle drives between two traffic lanes, is also visible in the data. In total, there is approximately 45 minutes of trajectory data, with a resolution of 10 frames per second. We focus on this sample due to the number of interactions that occur during highway driving and merging.

Due to errors and noise in the data set, we pre-processed the vehicle trajectories before performing the estimations and predictions detailed in this paper. For each vehicle, we filter for noise in local frames, smoothing the trajectories. We check for errors in the data set that arise from tracking errors, or in some cases, mis-attribution of vehicle id that results in duplicate or deleted cars.

The heading of each vehicle is not included in the raw data, and is instead extrapolated from the filtered trajectories. We also recalculate all velocities and accelerations from the filtered quantities. In comparisons between the "predicted" and "actual" trajectory, the actual trajectory is this processed data, not the raw data provided in the data set. Quantities such as vehicle class and size are taken directly from the data set.

To generate our trajectory predictions, the ego car computes a reward function that includes as a component the road network geometry. Lane geometry is not explicitly given in the raw data, thus we have reconstructed the lanes based on trajectories. When generating the reward function for our trajectory predictions, we re-generate lanes matched to the road network of the data set. An exemplary cost map is shown in Figure S9.

S7. Additional Prediction and Simulation Results

This section provides expanded analysis and results corresponding to the results presented in Section 4 on human driving data from the NGSIM data set and autonomous driving simulations. We provide further detail on our baseline algorithm and how we benchmark the accuracy of our prediction algorithms. We also discuss trends in the SVO observed from the data set. Finally, we include simulation results of the autonomous driving highway merge scenario.

S7.A. Prediction Accuracy on Interactive Merges. To better quantify our performance, we ran our algorithm against several variations of SVO preferences, as well as against a non-interactive baseline algorithm. For the baseline algorithm, each agent computes their policy as a single agent, and does not consider the interactions and rewards of the other agents in the system. Instead, all other agents are seen as simple dynamic obstacles, with simple lane-keeping actions and no predictions about their changes in acceleration. This baseline algorithm is analogous to current approaches in modeling multi-agent behavior without communication. We refer to this baseline algorithm as the "baseline" agent approach. For our other benchmarks, we compare our estimated SVO algorithm, which dynamically updates the SVO values of other agents, against our algorithm with SVO preferences held static throughout the interaction. This comparison highlights how the performance of the multi-agent game theoretic formulation increases with better SVO prediction. For each of these algorithms, we refer to them as "egoistic,", where the SVO is fixed to egoistic, which highlights the performance of the multi-agent game theoretic formulation without the use of SVO. The "static best" approach refers to a SVO that is held static during a scenario but is set to reflect the best possible SVO with respect to error. This reflects the benefit of employing the SVO metric, i.e. taking others' rewards into account during decision-making. And "estimated" refers to the dynamically online estimated SVO based on the estimation techniques presented in this work.

From the NGSIM data set, we examined 92 merge scenarios and compared performances across all scenarios. Here, we predict the trajectories of the cars through the merge and compute the mean square error (MSE) along the prediction horizon. We present errors in Table S3, corresponding to Table 1 presented in the main article. Here, a lower value corresponds to lower errors in prediction and thus, better performance of the algorithm. Through both the relative and absolute tables, we see that the estimated SVO multi-agent game theoretic approach has the best performance of the different algorithms. We also see that using a multi-agent game theoretic formulation reduces prediction errors over the baseline agent model. When normalized against the baseline, we note that including our SVO in the multi-agent formulation increases the accuracy of the prediction, indicated by the lower score. The best static SVO score corresponds to the best estimate when the SVO is held constant throughout the interaction. For different interactions, this may yield a different static SVO. The estimated SVO uses our proposed online algorithm, and a key difference from the static SVO is that the cars' SVO preference is allowed to change throughout the interaction. Overall, we see a performance increase of 5% of the multi-agent game theoretic approach with egoistic SVO compared to the baseline, a 18% improvement with a static best SVO, and 25% improvement with a dynamically estimated SVO. We can conclude that while the multi-agent game theoretic approach improves prediction results, the combination of multi-agent game theoretic and dynamically estimated SVO results in the largest benefit. The fact that the static best SVO is nearly 13% better than the egoistic SVO highlights the impact of SVO preferences in human decision-making.

[†] https://www.fhwa.dot.gov/publications/research/operations/06137/index.cfm

Prediction	baseline	multi-agent game theoretic		
SVO	-	egoistic	static best	estimated
MSE position [m]	1.559	1.476	1.279	**1.174**
MSE longitudinal [m]	1.451	1.370	1.170	**1.074**
MSE angle [rad]	0.149	0.139	0.139	**0.136**
MSE speed [m/s]	0.988	0.943	0.827	**0.803**

Table S3. Absolute mean squared error (MSE) over the prediction horizon during 92 interactive merges on the NGSIM dataset.

To visually illustrate the accuracy of the trajectory predictions, Figures S12, S13, and S14 show the ground truth trajectory (blue), the predicted trajectory using our estimated SVO (red dashed), and the predicted trajectories with varying static SVOs (black dotted) over many examples of the merge. Each subfigure in these figures represents a single merge example taken from the data set. A total of 17 different SVO values are compared for each merge, and as shown, this greatly impacts not only the lateral direction but also longitudinal direction of the predicted trajectory. Note the estimated SVO trajectory closely follows the ground truth trajectory. This is further validated by its minimal errors described in Table S3.

S7.B. Merging Drivers are More Competitive. From our experiment trials, we can compare the SVO values of the vehicles, and separate them into populations of merging and non-merging vehicles. Figure S16 illustrates the distribution of all estimated SVO values for all measurements at each time step in all merges. Figure S17 displays the distribution of the estimated SVO averaged over each scenario. Figure S18 plots the mean SVO estimates for each merge onto the SVO ring, with merging and non-merging vehicles indicated in red and blue, respectively. The radius of each observation indicates the consistency of each mean SVO estimate during a merge, computed from the variance of SVO measurements during the respective merge. We find from these distributions that merging vehicles show more competitive behavior than the non-merging vehicles, which exhibit more cooperative and prosocial SVO preferences.

We can test for statistical significance of this observation under the null-hypothesis that the mean SVO of merging cars is higher than that of non-merging cars. The one sided paired t-test rejects the null-hypothesis with $p = 6.5932e - 04$. Dropping the t-test's assumption that the variables in question are normally distributed in the two groups the non-parametric Wilcoxon signed-rank test can be applied. The non-parametric Wilcoxon signed-rank test as an alternative to the t-test also rejects the null-hypothesis with $p = 0.0018$. Since all p values are below the significance threshold $p < 0.005$ the null hypothesis can be rejected with high confidence and the alternative hypothesis holds. We conclude that merging drivers exhibit a lower SVO than non-merging drivers. Therefore, merging drivers are more competitive than non-merging drivers. While this statement is fairly intuitive for day-to-day drivers it can be grounded on objective observations of the SVO metric.

S7.C. Altruistic Driving Example. Unfortunately it is not possible to obtain ground truth labels for the SVO preferences of human drivers. The reason is that SVO preferences in traffic significantly change over time and even when consistent are hard to interpret by the eye. In Figure S15 we illustrate the trajectory predictions of a merging vehicle overlaid onto the actual vehicle trajectory for a single merge in which the yielding vehicle behaves fairly altruistic. We compare the baseline, and multi-agent game theoretic prediction results with different static SVOs with the estimated SVO approach. As expected, the multi-agent game theoretic approaches follow the actually executed trajectory closer than the baseline approach. The altruistic and estimated SVO trajectory predictions perform the best by closely following the ground truth trajectory.

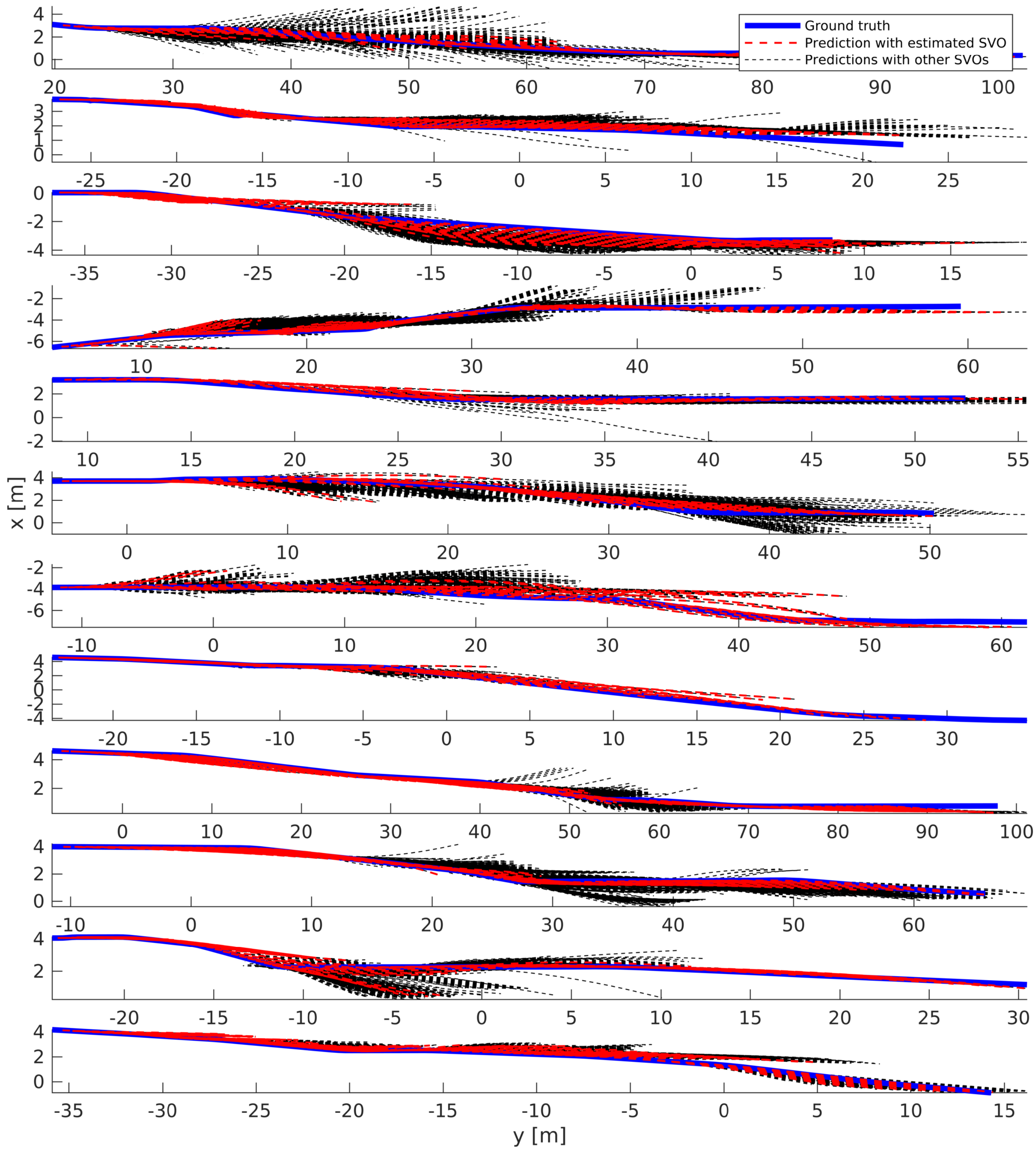


Fig. S12. Predicted and ground truth trajectories during lane-merging. Displayed are ground truth trajectories (blue), the predicted trajectories using our estimated SVO (red dashed), and the predicted trajectories with varying static SVOs (black dotted) for a variety of merge scenarios. Deviation of ground truth and varying static SVO predictions are largest during lateral motion of the merge, i.e. when the actual merge is executed. The predictions based on the estimated SVO are able to follow the ground truth trajectories very closely. By changing a single parameter alone, the SVO, a wide variety of trajectories can be predicted. Depending on what SVO is assessed, lane changes are predicted earlier or later which is shown by the spread of predictions. Since whole trajectories are predicted, predictions capture position, time, speed, as well as acceleration and steering angles. In the first example, predictions with other SVOs flare out to the left and right side, depending on which SVO is taken into account. The estimated SVO, displayed in red, follows the ground truh trajectory very closely.

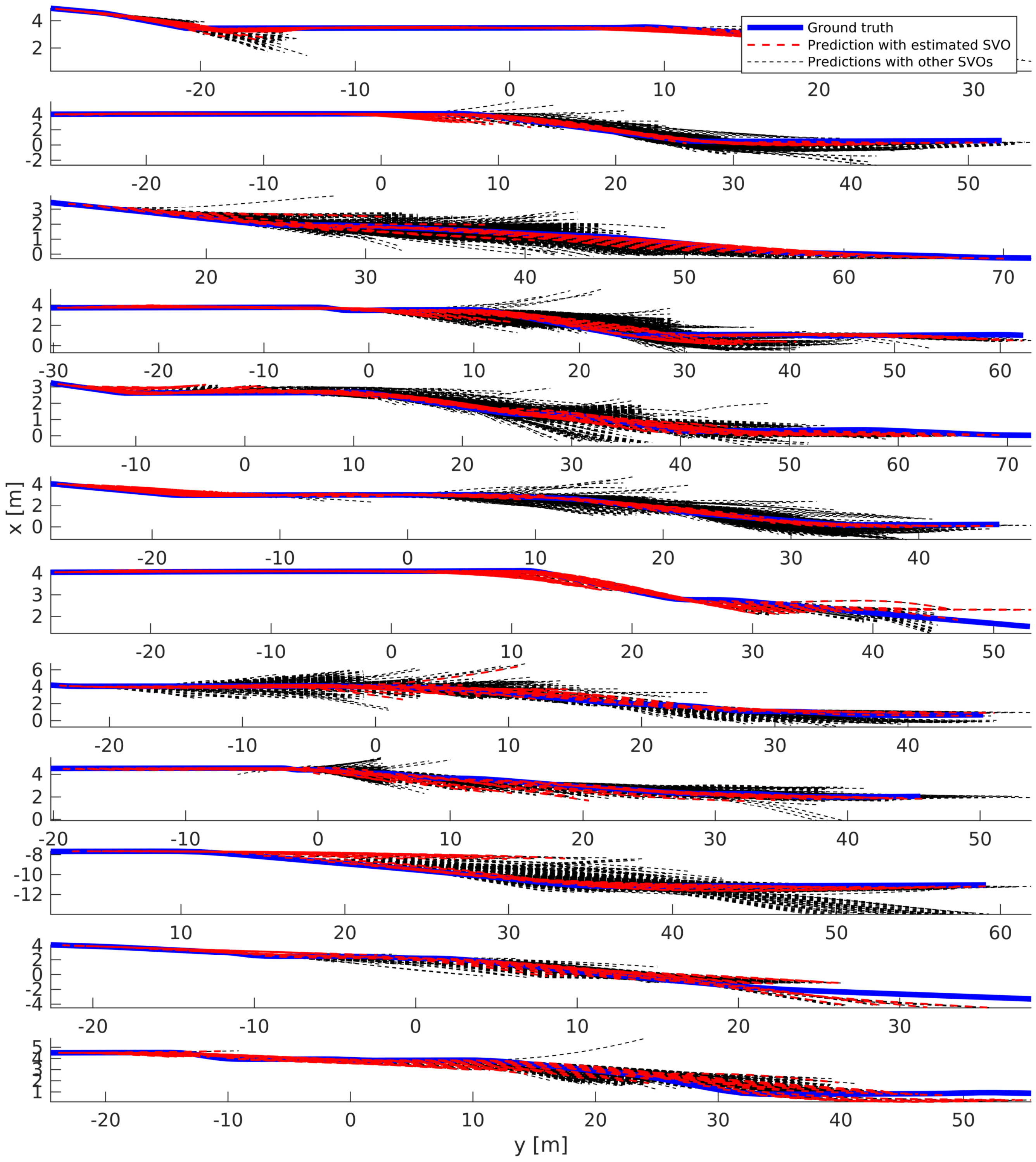


Fig. S13. We show the same information as in Figure S12 but for an additional number of merges. Displayed are ground truth trajectories (blue), the predicted trajectories using our estimated SVO (red dashed), and the predicted trajectories with varying static SVOs (black dotted) for a variety of merge scenarios. The impact of interaction may vary, as the 7th subfigure shows: The varying static SVO predictions yield very similar trajectories and only spread at the end of the merge. In other cases such as the 3rd scenario interactions are very strong throughout the whole merge, which shows the impact of the SVO on the predicted trajectories. By changing the SVO estimate alone, a wide variety of trajectories can be synthesized. The SVO therefore proves to have a high impact on trajectory predictions and an important metric for autonomous driving behavior.

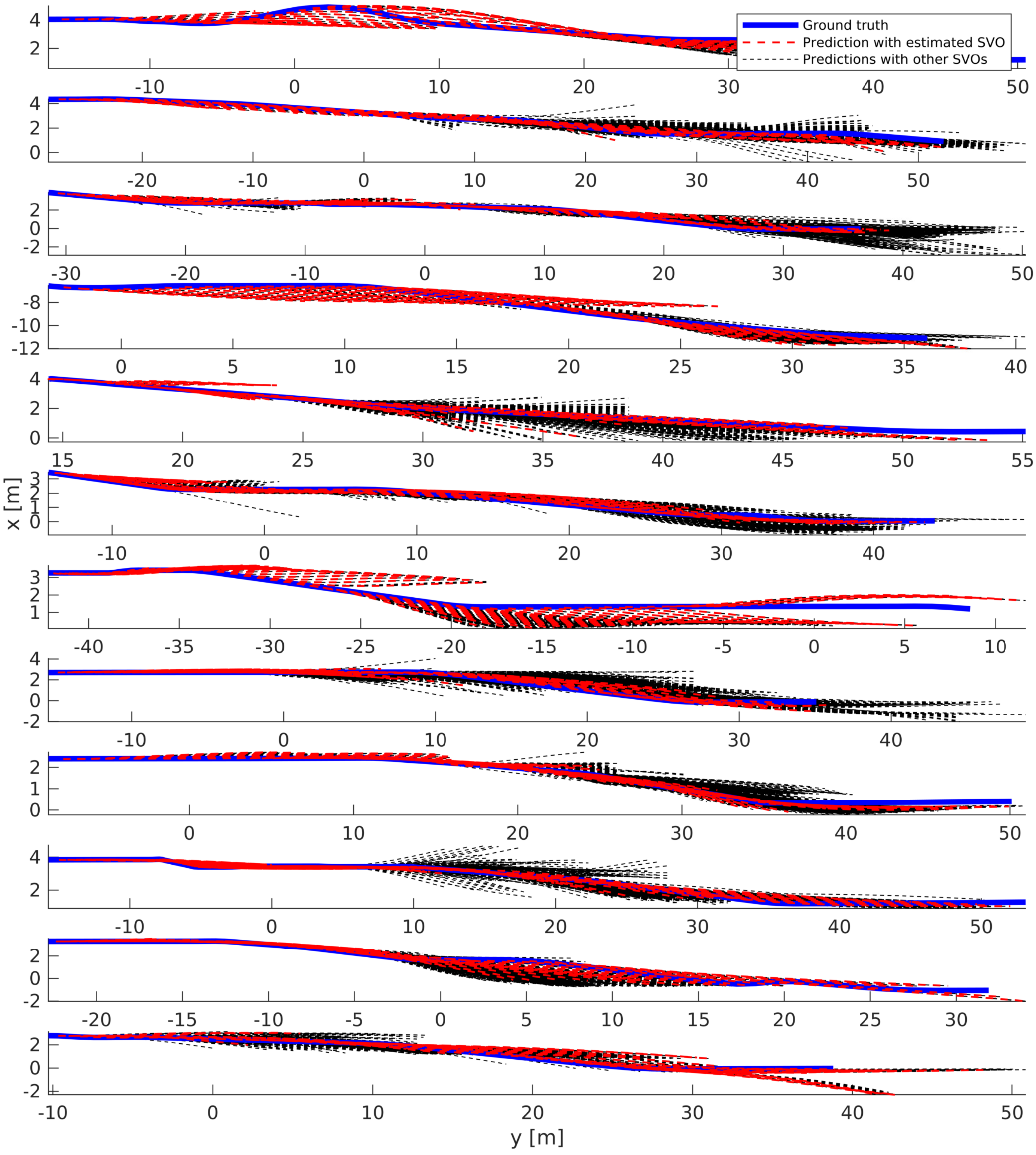


Fig. S14. We show the same information as in Figure S12 but for an additional number of merges. Displayed are ground truth trajectories (blue), the predicted trajectories using our estimated SVO (red dashed), and the predicted trajectories with varying static SVOs (black dotted) for a variety of merge scenarios.

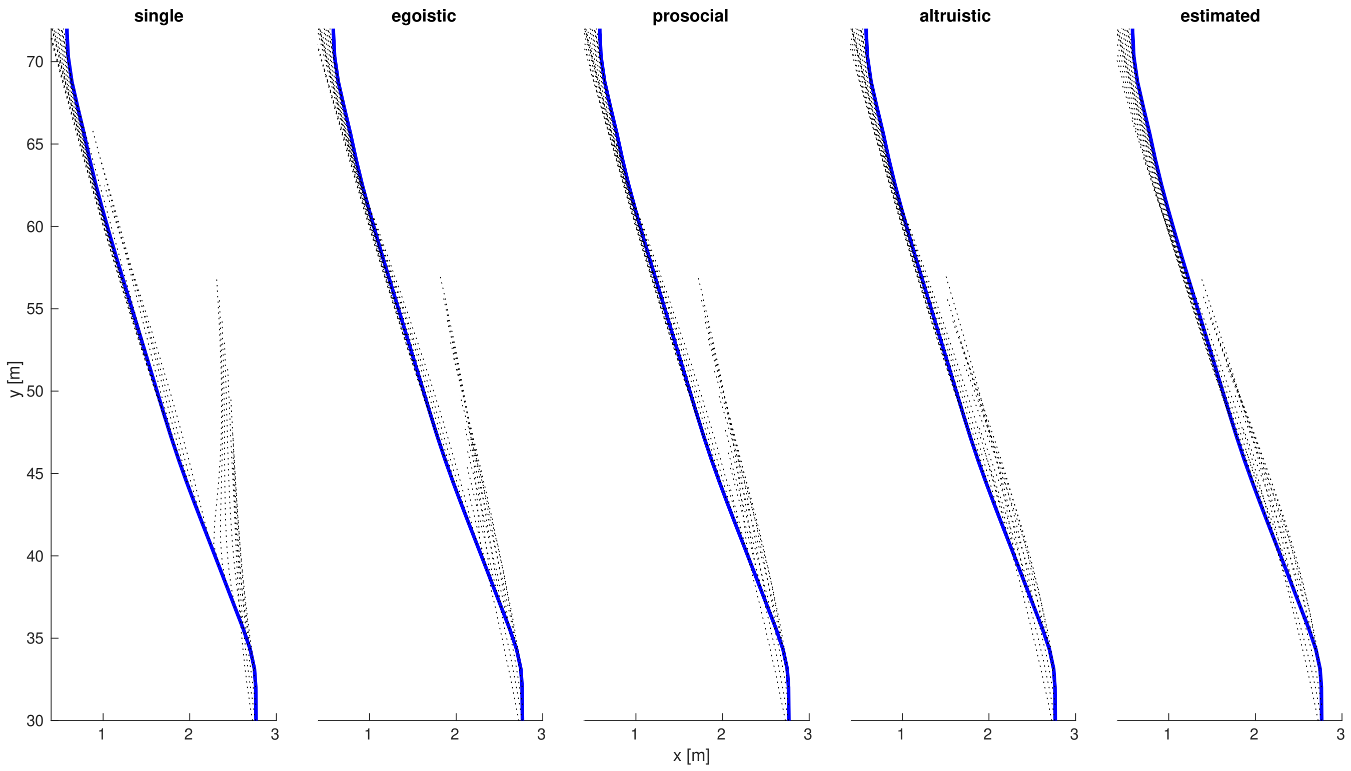


Fig. S15. Trajectory predictions of a merging vehicle. The vehicle leaves a highway merge lane and moves into the left adjacent lane where a vehicle yields to allow the lane change to succeed. The merging vehicle's actual trajectory is displayed in blue, whereas the predicted trajectories over time are overlaid as dotted lines. On the left all other vehicles are predicted with constant velocity while the merging car is predicted by a single car optimization, i.e. the baseline algorithm. All other plots to the right rely on the multi-agent game theoretic Nash Equilibrium formulation. The labels egoistic, prosocial, and altruistic refer to the SVO of the car that allows the predicted car to merge into its lane, whereas all other cars are assumed to be egoistic. Of these, the best fit is achieved with the yielding vehicles SVO to be altruistic. On the right, all SVOs of all vehicles are estimated based on the Maximum Entropy model resulting in an even closer fit. The combination of game-theoretic formulation and estimating the SVOs of other vehicles allows the vehicle to merge. Otherwise the vehicle would have acted too conservatively and avoided the merge, which is seen in the "single" figure by the estimated trajectories pointing straight when the car is actually moving laterally to merge.

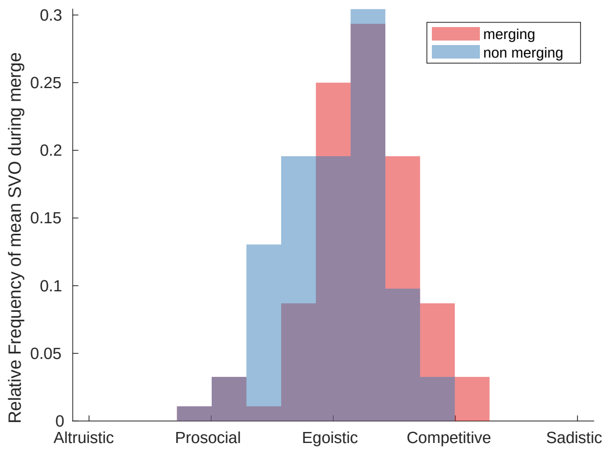


Fig. S16. Distribution of estimated mean SVO values for all evaluated merges. The mean is taken over all measurements captured during the full length of each merge interaction. Merging vehicles show more competitive behavior, while the non-merging vehicles exhibit more prosocial or even cooperative behavior. The histogram of merging vehicles is displayed in red, the histogram of non merging vehicles is shown in blue. Both are shown with 50% opacity such that the overlap appears in purple.

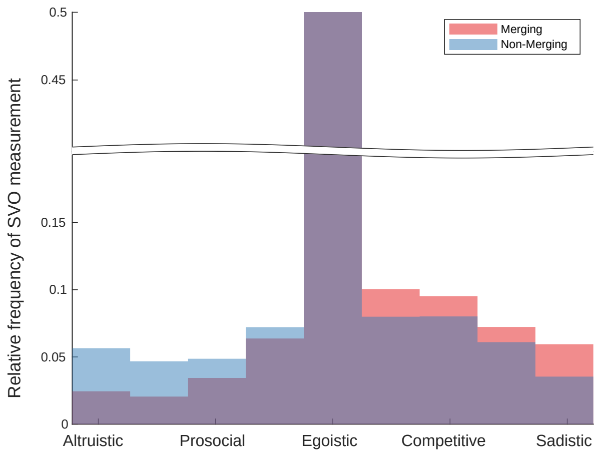


Fig. S17. Distribution of estimated SVO values for all evaluated measurements at all points in time of all merges. In contrast to Figure S16, the SVO measurements are not averaged over the length of a merge interaction but taken instantaneously at each point in time. Similarly to Figure S16, we osberve that merging vehicles show more competitive behavior, while the non-merging vehicles exhibit more prosocial or even cooperative traits. The histogram of merging vehicles is displayed in red, the histogram of non merging vehicles is shown in blue. Both are shown with 50% opacity such that the overlap appears in purple.

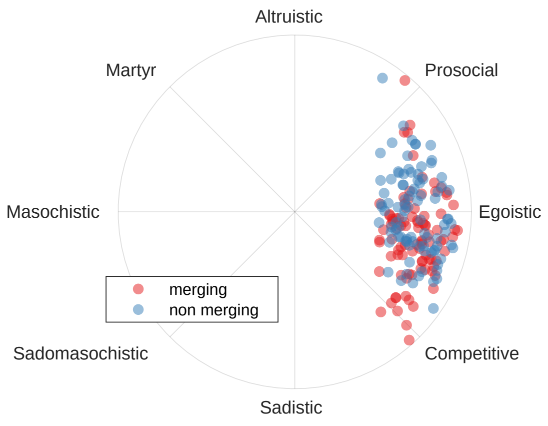


Fig. S18. Mean SVO estimates for each scenario shown on the SVO circle. Merging (red) and non-merging (blue) data points are shown separately. The radius of each measurement corresponds to the consistency of the SVO measurements during the respective merge. We observe that merging drivers show more competitive behavior.

S7.D. Additional Autonomous Driving Simulations on Highway Merging. The first scenario is the example of highway merging, where an autonomous vehicle must merge into another lane of traffic within a certain distance. Figures S19 and S20 illustrate the difference between egoistic and prosocial behavior, respectively. For each scenario, the autonomous vehicle is car $(i = 1)$ and shown in red, and must attempt to merge onto the highway around three other vehicles. In Figure S19, the magenta car $(i = 4)$ is an egoistic agent and thus is expected to not accommodate the autonomous vehicle. As expected, the egoistic agent does not make room for the autonomous vehicle, and the autonomous vehicle must slow down in order to merge after the cars have passed. Figure S21(a) shows the velocity profiles of the cars over time. We notice that the red vehicle must stop, while the egoistic vehicle actually accelerates slightly to prevent the autonomous vehicle from merging.

Conversely, Figure S20 illustrates the case with three other prosocial cars. In this scenario, we see the red autonomous car merge into a gap between the green $(i = 3)$ and magenta $(i = 4)$ cars. Examining the velocity profiles in Figure S21(b), we see that for this cooperative merge, the magenta vehicle slows down, but the green vehicle also speeds up in order to increase the gap, allowing the red car to merge.

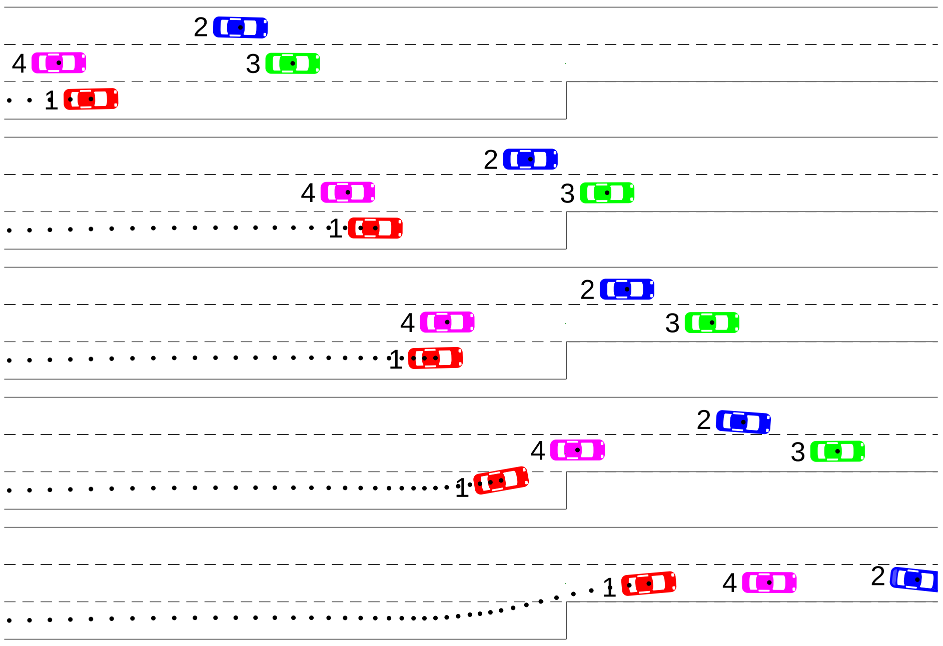


Fig. S19. Simulation of autonomous ego vehicle (red, $i = 1$) merging onto highway with three egoistic vehicles. The other cars do not allow the red vehicle to merge, and it must brake and wait for the other cars to pass to make the merge. In contrast, Figure S20, illustrates a prosocial merge.

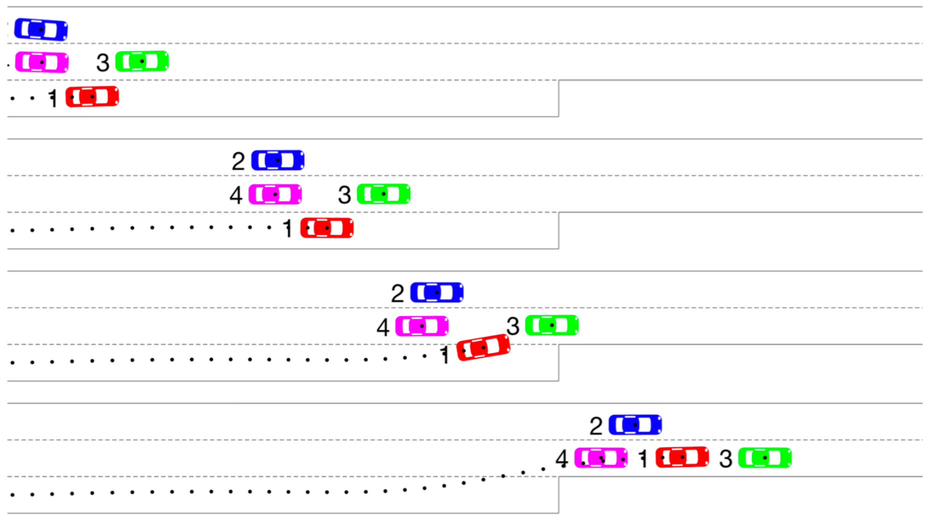
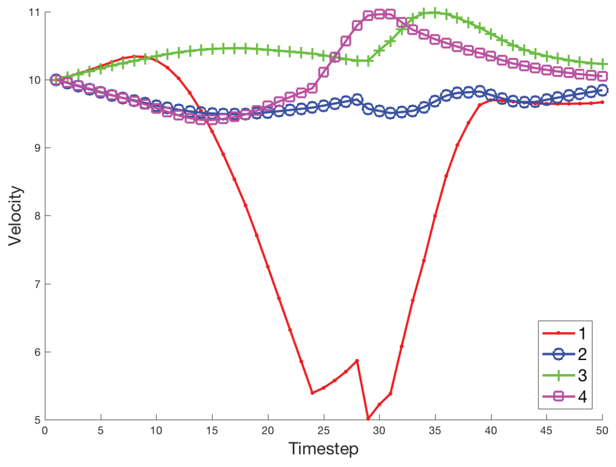
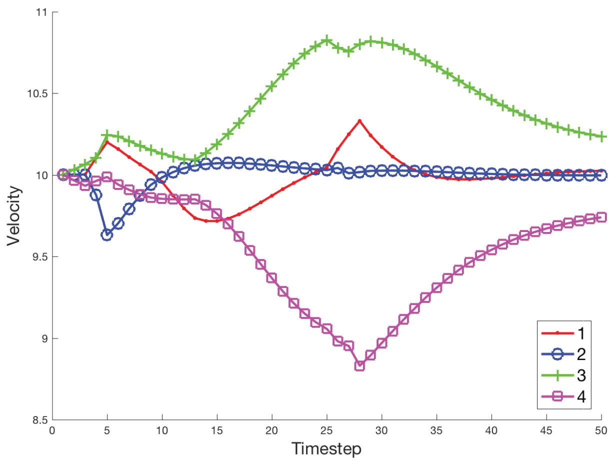


Fig. S20. Simulation of autonomous vehicle (red, $i = 1$) merging onto highway with three other prosocial vehicles. To allow the red vehicle to merge, the green vehicle $(i = 3)$ accelerates and the magenta vehicle $(i = 4)$ decelerates, increasing the gap. Here, each agent's individual reward function has penalties for acceleration and braking. Unlike Figure S19, because the agents are prosocial, they will slightly modify their actions in order to reduce the braking effort merging car 1.



(a) Egoistic



(b) Prosocial

Fig. S21. Comparison of the velocity profiles for the (a) egoistic merge in Figure S19 and (b) prosocial merge in Figure S20. In (a), the autonomous vehicle (red) must brake and wait for the other cars to pass. In (b), the cars cooperatively increase the gap, allowing the red car to merge between them. The decelerations in (b) are smaller than the decelerations in (a), showing the flow of traffic is more smooth with the prosocial group.

S8. Movie

Movie S1. In the movie, we provide animations of the problems presented here, to aid the reader in visualizing these interactions. The video shows the interactions in the highway merging scenario from the NGSIM data set, an animation of our dynamic SVO estimation, and animations of the autonomous driving simulations for both merging and left-turn scenarios.

References

1. Von Stackelberg H (2010) Market structure and equilibrium. (Springer Science & Business Media).
2. Sadigh D, Sastry S, Seshia SA, Dragan AD (2016) Planning for autonomous cars that leverages effects on human actions in Proceedings of the Robotics: Science and Systems Conference (RSS).
3. Pilecka M, , et al. (2012) Combined reformulation of bilevel programming problems. Schedae Informaticae 21:65–79.
4. Schwarting W, Pascheka P (2014) Recursive conflict resolution for cooperative motion planning in dynamic highway traffic in 17th International IEEE Conference on Intelligent Transportation Systems (ITSC). pp. 1039–1044.
5. Basar T, Olsder GJ (1999) Dynamic noncooperative game theory. (Siam) Vol. 23.
6. Andersson JAE, Gillis J, Horn G, Rawlings JB, Diehl M (In Press, 2018) CasADi – A software framework for nonlinear optimization and optimal control. Mathematical Programming Computation.
7. Wächter A, Biegler LT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106(1):25–57.
8. Abbeel P, Ng AY (2005) Exploration and apprenticeship learning in reinforcement learning in Proceedings of the 22nd international conference on Machine learning. (ACM), pp. 1–8.
9. Levine S, Koltun V (2012) Continuous inverse optimal control with locally optimal examples. arXiv preprint arXiv:1206.4617.
10. Ng AY, Russell SJ, , et al. (2000) Algorithms for inverse reinforcement learning. in Icml. pp. 663–670.
11. Ziebart BD, Maas AL, Bagnell JA, Dey AK (2008) Maximum entropy inverse reinforcement learning. in AAAI. (Chicago, IL, USA), Vol. 8, pp. 1433–1438.