Deep Kinematic Models for Kinematically Feasible Vehicle Trajectory Predictions

Henggang Cui, Thi Nguyen, Fang-Chieh Chou,
Tsung-Han Lin, Jeff Schneider, David Bradley, Nemanja Djuric
Uber Advanced Technologies Group
{hcui2, thi, fchou, hanklin, jschneider, dbradley, ndjuric}@uber.com

Abstract—Self-driving vehicles (SDVs) hold great potential for improving traffic safety and are poised to positively affect the quality of life of millions of people. To unlock this potential one of the critical aspects of the autonomous technology is understanding and predicting future movement of vehicles surrounding the SDV. This work presents a deep-learning-based method for kinematically feasible motion prediction of such traffic actors. Previous work did not explicitly encode vehicle kinematics and instead relied on the models to learn the constraints directly from the data, potentially resulting in kinematically infeasible, suboptimal trajectory predictions. To address this issue we propose a method that seamlessly combines ideas from the AI with physically grounded vehicle motion models. In this way we employ best of the both worlds, coupling powerful learning models with strong feasibility guarantees for their outputs. The proposed approach is general, being applicable to any type of learning method. Extensive experiments using deep convnets on real-world data strongly indicate its benefits, outperforming the existing state-of-the-art.

I. INTRODUCTION

Self-driving vehicles (SDVs) have transformative potential benefits [1], not least of which is traffic safety. A recent study by NHTSA showed that human-related errors such as distracted driving, illegal maneuvers, poor vehicle control, or drowsy driving are linked to nearly 94% of crashes [2]. SDV technology could prevent such avoidable accidents, and help reverse negative safety trends observed on the US roads in the past decade [3]. Beyond road safety, SDVs may provide social benefits due to easier and cheaper access to transportation, and environmental benefits through increased traffic efficiency and a transition from infrequently used private vehicles to shared vehicle fleets [4].

A number of involved problems need to be solved in order to operate an SDV in a safe and effective manner in a complex traffic environment [1]. These include perceiving and understanding the current state of the traffic actors around the SDV, accurately predicting their future state, and efficiently planning motion for the SDV through an uncertain estimate of the future. In this work we focus on the problem of motion prediction of the surrounding vehicles, one of the key components necessary for safe autonomous operations.

There has been a significant recent interest by the research community in the task of predicting the future motion of nearby actors using deep learning models. Most published work models the traffic actors as points and only predicts their center positions directly through a deep neural network
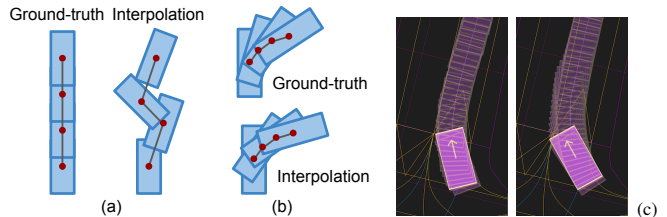

Fig. 1: Interpolating headings from positions results in suboptimal heading predictions: (a) large heading errors caused by small position errors; (b) kinematically infeasible headings even with perfect position predictions (as rear wheels can not turn); (c) real-world example before and after introduction of the kinematic constraints

with no motion constraints [5], [6], [7], [8], [9], [10], [11], [12]. Such unconstrained motion prediction works well for pedestrian actors [13], however it falls short in the case of vehicle actors whose shapes are more rectangular and motions are constrained by vehicle kinematics [14]. First, for vehicle actors we need to predict not only their center positions but also the headings, in order for the SDV to better reason about their occupancy in space. In previous work, interpolation is often used to estimate the headings from the sequence of predicted future positions [7], [8]. As illustrated in Figure 1, the obtained headings are often suboptimal as 1) they are very sensitive to position prediction errors; and 2) motion direction of vehicle center is not always the same as its true heading. Second, the unconstrained approach may predict kinematically infeasible movement (e.g., a too small turning radius). The resulting heading prediction errors and inaccurate motion predictions can have a significant negative impact on the SDV's future motion plans.

To address this issue we propose *deep kinematic model* (DKM) that embeds the two-axle vehicle kinematics [14] in the output layer of a deep learning trajectory prediction model, and jointly predicts vehicle actors' motion state (comprising positions, headings, and velocities) in a more accurate and kinematically feasible way (Figure 2b). This approach achieves more accurate position and heading predictions, and guarantees that the predictions are kinematically feasible, thus allowing for safer and more efficient SDV operations.

Main contributions of our work are summarized below:

- We combine powerful deep methods with a kinematic vehicle motion model to jointly predict the complete motion states of nearby vehicle actors, thus achieving more accurate position and heading predictions, as well as guarantee of kinematically feasible trajectories;

- While the idea is general and applicable to any deep architecture, we present an example application to a recently proposed state-of-the-art motion prediction method, using raster images of vehicle context as an input to convolutional neural networks (CNNs) [7], [8];
- We evaluate the method on a large-scale, real-world data set collected by a fleet of SDVs, showing that the system provides accurate, kinematically feasible predictions that outperform the existing state-of-the-art;
- Following extensive offline testing, the model was successfully tested onboard SDVs.

II. RELATED WORK

A. Motion prediction in autonomous driving

Accurate motion prediction of vehicles surrounding an SDV is a critical component in many autonomous driving systems [15], [16], [17]. Prediction provides an estimate of a future world state, which can be used to plan an optimal SDV path through a dynamic traffic environment. A common approach for short-term prediction of future motion is to assume that the driver will not change any control inputs (such as steering and acceleration). Then, one can simply propagate vehicle's current estimated state over time using a physical model (e.g., a vehicle motion model) that captures the underlying kinematics [15], where the current state (e.g., position, speed, acceleration) can be estimated using techniques such as a Kalman filter (KF) [18], [19]. For longer time horizons performance of this approach degrades as the underlying assumption of constant controls becomes increasingly unrealistic. An alternative method explicitly models lane-following driving behavior by using map information as a constraint on future vehicle positions [16]. This approach first associates detected vehicles with one or more lanes from the map. Then, all possible paths are generated for each *(vehicle, associated lane)* pair based on map topology, lane connectivity, and vehicle's state. While the resulting predictions are reasonable for commonly encountered cases, unusual traffic scenarios are difficult to model.

B. Learned prediction models

An alternative to manually engineered prediction methods mentioned above is to learn a motion prediction model from the large amounts of available driving data. Classical machine learning approaches such as Hidden Markov Model [20], Bayesian networks [21], or Gaussian Processes [22] have been applied to motion prediction in autonomous driving. However, these methods require manually designed features and no longer provide state-of-the-art performance.

Most recent research on motion prediction employs deep networks. In one line of research, recurrent neural networks (RNN) with Long Short-Term Memory (LSTM) or gated recurrent unit (GRU) were applied to predict future trajectories from past observed positions [9], [23], [24], [25], [26], [27]. Going beyond just using past observed positions as inputs, [8], [17], [7] rasterize actor's surrounding context and other scene information in a bird's-eye view (BEV) image. Some trajectory prediction works also include social layers

in their models to model interactions among actors [11], [12], [28], [29], [30]. To model the multimodality of future trajectories, [9], [10], [11], [29] used generative models to transform a latent space into trajectory predictions. These methods often need to draw many trajectory samples at inference time to have good coverage (e.g., as many as 20 for Social-GAN [29]), which could be impractical for time-critical applications. In [8] the authors tackled the multi-modal problem by having the model predict for each actor a small fixed number of trajectories along with their probabilities, and learning them jointly with a specially designed loss function. However, the above methods model each actor as a point, and predict only their future center positions without considering kinematic constraints. As heading information is often used in downstream modules, actor headings are approximated from the positions through interpolation [8].

Recently proposed TrafficPredict [25] and TraPHic [31] recognize the usefulness of actor type and size info and propose to include them in the model input. However, they also predict only the actor center positions and do not enforce any kinematic constraints (e.g., minimum turning radius). Luo et al. [6] explicitly predict future headings in addition to the center positions by encoding them in the regression target. However, these outputs are decoupled and inferred independently, and as our experiments show the predicted positions and headings may be kinematically infeasible.

Another related line of research is imitation learning, which by learning to imitate expert demonstrations addresses problems where future observations depend on previous predictions (or actions) executed by the actor. Since previous predictions influence the future observation/state distributions encountered by the learner, [32] showed that a learner that makes mistakes with probability ϵ under the distribution of states visited in the expert demonstrations, may instead make as many as $T^2\epsilon$ mistakes in expectation when executing its own predictions for T steps. As a consequence, the value of exploiting the regularization provided by an explicit motion model has long been recognized in the imitation learning community. For instance, inverse optimal control approaches [33] reframe the problem as a learned reward function combined with an optimization function and an explicit state transition model (such as the vehicle kinematics model used in this work). The reward function can also be interpreted as a probability distribution over future action sequences, as done by [34] for taxi driver route prediction. More recently [35] presented R2P2, a novel normalizing-flows version of this approach that predicts a multimodal distribution over future vehicle trajectories, in conjunction with a simple constant velocity motion model with acceleration control inputs. Unlike our work, R2P2 also predicts only center positions without kinematic constraints.

Our proposed vehicle kinematic layer is agnostic to the model architecture and the learning method, and it can be used to replace the trajectory decoding module (e.g., LSTM decoder) or policy model of the above related work to improve accuracy and guarantee kinematic feasibility.
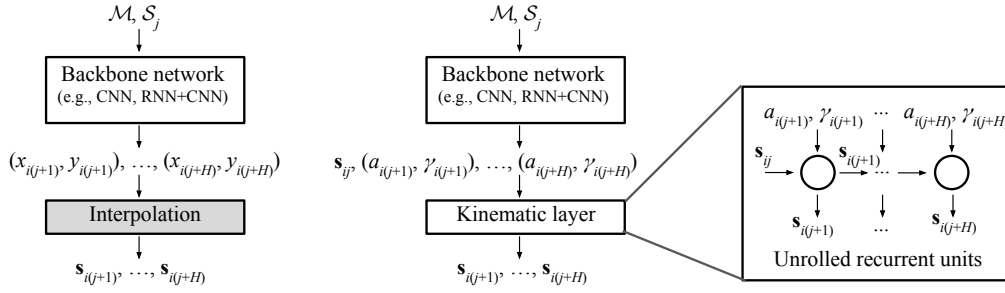
Fig. 2: Illustration of (a) the existing state-of-the-art unconstrained model; and (b) the proposed deep kinematic model; modules that are not used during training are shown in gray

Fig. 3: Input raster overlaid with output trajectories and probabilities

C. Vehicle motion models

Extensive work has been done on creating mathematical models of non-holonomic vehicle motion for the automotive industry [14]. Motion models for vehicles come in two broad classes, namely kinematic models that assume no wheel slip, and dynamic models that attempt to model complex interactions that occur when the tire forces are insufficient to fully overcome the inertia of the vehicle (i.e., the wheels are slipping). Past work on Model Predictive Control (MPC) has shown that a simple kinematic bicycle model is sufficient for modeling non-articulated vehicles performing normal driving maneuvers [36]. MPC typically uses known cost functions (e.g., behavior preferences of the predicted actors) and known dynamics models to define an optimization problem over potential vehicle trajectories [37], [38], and a solver is used to find the sequence of vehicle control inputs that produces a minimum-cost vehicle trajectory (when integrated through the chosen vehicle model). By contrast, in our approach we leave both the cost functions and the optimization algorithm implicit in a deep neural network, similarly to an unrolled optimization approach described in [39].

III. PROPOSED APPROACH

We assume there exists a tracking system onboard an SDV, ingesting sensor data and allowing detection and tracking of traffic actors in real-time (e.g., using KF taking lidar, radar, and/or camera data as inputs). State estimate contains information describing an actor at fixed time intervals, including bounding box, position, velocity, acceleration, heading, and turning rate. Moreover, we assume access to map data of an operating area, comprising road locations, lane boundaries, and other relevant info. The resulting tracking and map data is then used as an input to the proposed deep system.

We denote overall map data by $\mathcal{M}$, and a set of discrete times at which tracker outputs state estimates as $\mathcal{T} = \{t_1, t_2, \ldots, t_T\}$, where time gap Δ_t between consecutive time steps is constant (e.g., $\Delta_t = 0.1s$ for tracker running at frequency of $10Hz$). State output of a tracker for the i-th actor at time t_j is denoted as $\mathbf{s}_{ij}$, where $i = 1, \ldots, N_j$ with N_j being a number of tracked actors at time t_j. Then, given data $\mathcal{M}$ and all actors' state estimates up to and including t_j (denoted by $\mathcal{S}_j$), the task is to predict future states $[\mathbf{s}_{i(j+1)}, \ldots, \mathbf{s}_{i(j+H)}]$, where $\mathbf{s}_{i(j+h)}$ denotes state of the i-th actor 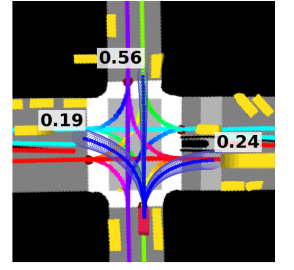at time t_{j+h}, and H is a number of consecutive steps for which we predict states (or *prediction horizon*). Past and future states are represented in an actor-centric coordinate system derived from actor's state at time t_j, where forward direction defines x-axis, left-hand direction defines y-axis, and actor's bounding box center defines the origin.

A. Unconstrained motion prediction

In this section we revisit the current state-of-the-art for vehicle motion prediction. The existing methods (e.g., [7], [8], [10]) follow the same basic idea as illustrated in Figure 2a. More specifically, the deep backbone network is trained to directly predict x- and y-positions of future trajectory points, and full actor states are then derived from the inferred positions (i.e., the headings and velocities are decided from the displacement vectors between consecutive positions), without being constrained by vehicle kinematics. As we show in the experiments, this is suboptimal as it may lead to positions and higher derivatives (e.g., heading and velocity) that are inaccurate and/or infeasible. In the remainder, we use a recent method by [8] as an example of a state-of-the-art unconstrained method for motion prediction, using convnets as the backbone network operating on BEV rasters.

Let us assume a predictive model with parameter set $\boldsymbol{\theta}$, taking map $\mathcal{M}$ and state info $\mathcal{S}_j$ as inputs at time t_j. The inputs are encoded as an overhead raster image to represent surrounding context for the i-th actor, used to predict H trajectory points at Δ_t-second gap [7]. We denote model outputs using the hat notation $\hat{\bullet}$, and for simplicity do not explicitly specify $(\boldsymbol{\theta}, \mathcal{M}, \mathcal{S}_j)$ as input arguments. Then, we write overall loss for the i-th actor at time t_j as the Euclidean distance between predicted and observed positions,

$$L_{ij} = \sqrt{(x_{i(j+h)} - \hat{x}_{i(j+h)})^2 + (y_{i(j+h)} - \hat{y}_{i(j+h)})^2}. \quad (1)$$

As described in [8], this formulation can be extended to multimodal predictions. In particular, instead of predicting one trajectory, the model outputs M modes and their associated probability p_{ijm} modeled by a soft-max, with $m \in \{1, \ldots, M\}$. At each step during training we find a predicted mode closest to the ground-truth trajectory, indexed by m^*. Then, the final loss is defined as

$$\mathcal{L}_{ij} = \sum_{m=1}^{M} I_{m=m^*}(L_{ijm} - \alpha \log p_{ijm}), \quad (2)$$

where I_c is a binary indicator function equal to 1 if the condition c is true and 0 otherwise, L_{ijm} is computed as (1) taking only the m-th mode into account, and α is a hyper-parameter used to trade-off between the two losses (i.e., the
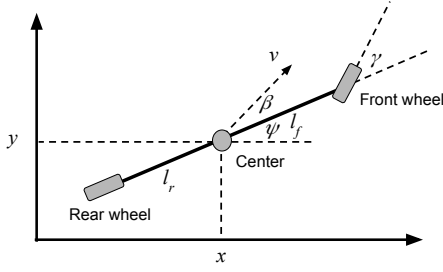
Fig. 4: Vehicle kinematic model embedded into DKM

difference between the observed data and the winning mode on one side, and a mode selection cross-entropy loss on the other). Note that, according to (2), during training we update the position outputs only for the winning mode and the probability outputs for all modes due to the soft-max.

We can optimize equation (2) over all actors and times in the training data to learn the optimal model parameters $\boldsymbol{\theta}^*$,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L} = \arg\min_{\boldsymbol{\theta}} \sum_{j=1}^{T} \sum_{i=1}^{N_j} \mathcal{L}_{ij}. \tag{3}$$

We refer to the method as Unconstrained Model (UM), producing multiple modes and their corresponding probabilities (example of input raster and outputs is shown in Figure 3).

B. Deep kinematic model

In this section we introduce our *deep kinematic model* (DKM). We propose to add a new *kinematic layer* between the final hidden layers of UM and its output layer to decode the trajectories, which explicitly embeds the kinematics of a tracked two-axle vehicle. Assuming H prediction horizons, the kinematic layer consists of H *kinematic nodes* in a sequence (see Fig. 2b), each implementing the same update equations of the kinematic vehicle model introduced below.

When predicting motion of actor i at time t_j, a kinematic node at horizon h predicts the actor's longitudinal acceleration $a_{i(j+h)}$ and steering angle $\gamma_{i(j+h)}$ at time t_{j+h}, and then uses them to roll out the actor's next state,

$$\mathbf{s}_{i(j+h+1)} = f(\mathbf{s}_{i(j+h)}, a_{i(j+h)}, \gamma_{i(j+h)}, \kappa_i), \tag{4}$$

where $\mathbf{s}_{i(j+h)}$ comprises x- and y-positions (i.e., $x_{i(j+h)}$ and $y_{i(j+h)}$), heading $\psi_{i(j+h)}$, and velocity $v_{i(j+h)}$, κ_i is a set of fixed kinematic parameters of actor i, including distances from the center to the rear and front wheels (l_{ri} and l_{fi}) and the maximum allowed acceleration and steering angle.

To model two-axle vehicles considered in this work, we implement f following vehicle kinematics introduced in [36],

$$\mathbf{s}_{i(j+h+1)} = \mathbf{s}_{i(j+h)} + \dot{\mathbf{s}}_{i(j+h)} \Delta t, \tag{5}$$

where the state derivatives are computed as follows,

$$\begin{aligned}
\dot{x}_{i(j+h)} &= v_{i(j+h)} \cos(\psi_{i(j+h)} + \beta_{i(j+h)}), \\
\dot{y}_{i(j+h)} &= v_{i(j+h)} \sin(\psi_{i(j+h)} + \beta_{i(j+h)}), \\
\dot{\psi}_{i(j+h)} &= \frac{v_{i(j+h)}}{l_{ri}} \sin(\beta_{i(j+h)}), \\
\dot{v}_{i(j+h)} &= a_{i(j+h)}, \\
\beta_{i(j+h)} &= \tan^{-1}\left(\frac{l_{ri}}{l_{fi} + l_{ri}} \tan \gamma_{i(j+h)}\right),
\end{aligned} \tag{6}$$

where β is an angle between velocity and heading ψ, and l_r and l_f can be estimated from the tracked state. Figure 4

illustrates this kinematic model, with variable subscripts representing actor and time step removed for clarity.

The first kinematic node at $h = 0$ takes the state at current time t_j as input, and each subsequent node takes output of a previous node and computes the next state according to the controls a and γ predicted by a fully-connected layer (see Fig. 2b). To ensure the trajectories are kinematically feasible, the controls are clipped to be within the allowed ranges[1].

It is important to emphasize that the DKM model still outputs positions and that the kinematic computations presented in equations (5) and (6) are fully differentiable. This allows DKM to be trained end-to-end by minimizing exactly the same losses as UM, given in (2) and (3), without requiring ground-truth labels for the predicted controls a and γ.[2] Furthermore, while we consider two-axle vehicles, the proposed approach can be applied to other vehicle types by using appropriate kinematics and updating (6). In addition, while we use tracked state to obtain the kinematic parameters κ_i, they can also be directly learned by the network. Lastly, the proposed method is a general solution, and while we introduce and evaluate it in the context of supervised learning and feed-forward convnets used in UM, it is straightforward to plug in the kinematic layer into other types of learning problems (e.g., imitation learning) and other types of model architectures (e.g., GAN-based models). These considerations are however beyond the scope of our current work.

IV. EXPERIMENTS

We collected 240 hours of data by manually driving in various conditions (e.g., varying times of day, days of the week, in several US cities), with data rate of $10Hz$. We ran a state-of-the-art detector and Unscented KF (UKF) tracker [40] with the kinematic state-transition model (5) and (6) on this data to produce a set of tracked vehicle detections. Each vehicle at each discrete tracking time step amounts to a data point, and once we removed non-moving vehicles there remained 7.8 million points. We considered horizons of up to $6s$ (i.e., $H = 60$) with $3:1:1$ split for train/validation/test data. We compared the following motion prediction methods:

- UKF, non-learned model that simply propagates the tracked state forward in time;
- Social-LSTM[3] which considers social interactions between the actors [28];
- poly-n, modeling output trajectory with an n-degree polynomial curve, inferring scalar parameters q and r of the polynomial in the last hidden layer, as

$$\begin{aligned}
x_{j+h} &= \sum_{d=1}^{n} q_d(\boldsymbol{\theta})(t_{j+h} - t_j)^d, \\
y_{j+h} &= \sum_{d=1}^{n} r_d(\boldsymbol{\theta})(t_{j+h} - t_j)^d, \tag{7}
\end{aligned}$$

where we experimented with $n \in \{1, 2, 3\}$;

---

[1] Maximum absolute acceleration and steering angle are set to $8m/s^2$ and $45°$, respectively, roughly following characteristics of a midsize sedan.

[2] Extra loss terms on ψ, v, a, and γ can be added if the labels are available.

[3] We used the open-sourced code from https://github.com/quancore/social-lstm, which only evaluates the ℓ_2 errors.

TABLE I: Comparison of prediction errors (lower is better)

Method	Position ℓ_2 [m] @3s	Position ℓ_2 [m] @6s	Heading [deg] @3s	Heading [deg] @6s	Infeasible [%]
UKF	3.99	10.58	7.50	18.88	**0.0**
S-LSTM [28]	4.72	8.20	-	-	-
poly-1	1.71	5.79	3.97	8.00	**0.0**
poly-2	1.37	4.39	10.19	31.44	20.8
poly-3	1.45	4.66	10.16	16.09	15.5
UM	**1.34**	4.25	4.82	7.69	26.0
UM-velo	1.37	4.28	4.76	7.55	27.3
UM-LSTM	1.35	4.25	4.22	7.20	22.9
UM-heading	1.37	4.32	4.65	7.09	20.7
CTRA	1.56	4.61	3.60	8.68	**0.0**
DKM	**1.34**	**4.21**	**3.38**	**4.92**	**0.0**

- UM, unconstrained model from Section III-A;
- UM-velo, where instead of actual positions the model predicts differences between two consecutive positions, amounting to predicting velocity at each horizon without constraints (note that this is logically the same as the policy model used in R2P2 [35]);
- UM-LSTM, UM with an additional LSTM layer of size 128 that is used to recursively output positions instead of simultaneously outputting positions for all horizons;
- UM-heading, unconstrained model that additionally predicts headings (similarly to Luo et al. [6]);
- constrained model with the Constant Turning Rate and Acceleration (CTRA) motion model [41] at the output;
- DKM, the proposed deep vehicle kinematic model.

All models except UKF and Social-LSTM were trained using the multimodal loss (3), outputting trajectories and their probabilities (we set $M = 3$). They were implemented in TensorFlow using the same backbone architecture and training setup described in [8], with α set to 1, per-GPU batch size of 64, and Adam optimizer [42]. Learning rate was initialized to 10^{-4} and decreased by a factor of 0.9 every 20,000 iterations, trained for 400,000 iterations. The poly-n, UM, UM-velo, and UM-LSTM models predict only positions, and we computed headings through interpolation.

A. Results

For evaluation we considered each method's top-ranked trajectory according to the predicted probabilities[4], and report position ℓ_2 errors and heading errors. We also report the percentage of kinematically infeasible trajectories. We define a trajectory as kinematically infeasible if for any of its points the turning radius is less than the threshold for the given actor category[5]. The results are presented in Table I.

Expectedly, the naïve non-learned UKF baseline gave the largest errors, but its predictions are all kinematically feasible as it simply rolls out the current track states. Social-LSTM [28] also underperformed compared to other learned models, explained by the fact that it does not use scene

[4] The top-ranked metric requires the model to accurately predict both the trajectories and their probabilities. Some GAN-based models (e.g., Social-GAN [29]) use the min-over-N error metric when probabilities are not available. Our findings still stand when using this metric (results not shown).

[5] Minimum feasible turning radius can be computed from the maximum steering angle and vehicle length, equaling around $3m$ for a midsize sedan.
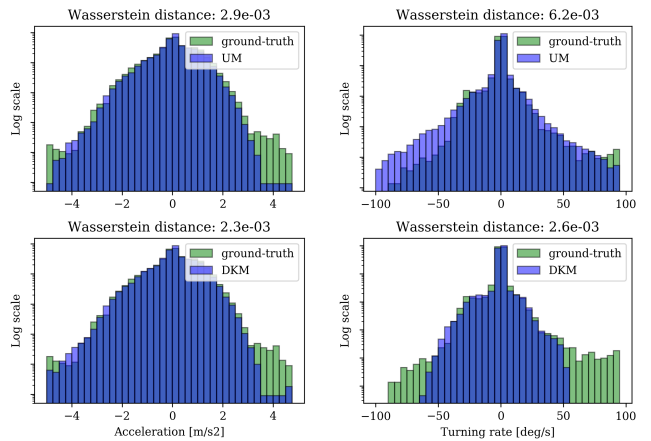

Fig. 5: Log-scale histogram of ground-truth and predicted states at $6s$ horizon (top: UM, bottom: DKM); also shown Wasserstein distance between the two distributions

context as input. Furthermore, we found that poly-n methods of varying degrees yielded very different results. The poly-1 approach amounts to a constant-velocity model that does not capture real-world behavior well, although it does guarantee predicted trajectories to be kinematically feasible. Nevertheless, all metrics improved across the board compared to the non-learned model, indicating that even simple curve fitting could provide benefits. The second- and third-degree polynomials (constant-acceleration and constant-jerk models, respectively) further improved position errors, although with degraded heading and feasibility metrics.

Moving forward, we see that the four unconstrained models, UM, UM-velo, UM-LSTM, and UM-heading gave strong results in terms of the center position prediction. This especially holds true in the short-term, where position errors are on par with the best-performing DKM method. Interestingly, although outputs of UM and UM-velo are quite different (where former directly predicts positions, and latter predicts velocities), the results are near-identical, and both have about 30% of their trajectories being kinematically infeasible, mostly due to the sensitivity of heading interpolation for slow-moving or stopping actors. Since the proposed DKM infers vehicle controls, we specifically included UM-velo as it gives outputs that can be interpreted as controls as well. Through this comparison we wanted to evaluate if such unconstrained controls outputs could lead to improvements over unconstrained position outputs. However, this is not corroborated by the empirical results. It is interesting to discuss UM-LSTM, which showed good performance in terms of position errors but poor in terms of heading and infeasibility metrics. The recurrent structure of LSTM mirrors the structure of the kinematic layer, and in theory LSTM could learn to approximate the update function (6) of the proposed vehicle model given enough data. Nevertheless, we can see that directly encoding prior knowledge in a form of the proposed kinematic layer leads to better performance. The UM-heading model, which predicts both positions and headings, has better heading and infeasibility metrics com-
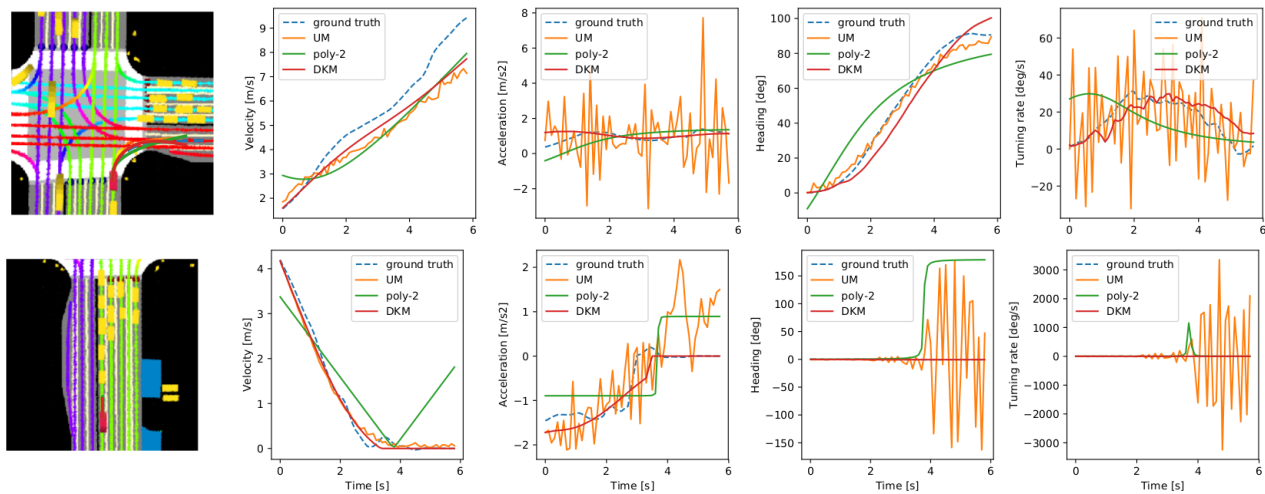
Fig. 6: Actor states predicted by competing methods for two commonly encountered traffic situations (right-turning and stopping); first column shows input raster with overlaid output trajectories, followed by columns showing profiles of output velocity, acceleration, heading, and turning rate, respectively (best seen in the electronic version of the manuscript)

pared to UM. However, around 20% of the predicted are still kinematically infeasible due to positions and headings being predicted independently without any kinematic constraints.

Lastly, we evaluated constrained approaches using two different vehicle models at their output. CTRA is a model shown to capture vehicle motion well [41]. However, when compared to unconstrained models it is clear that the method underperformed, although it is competitive in terms of kinematic feasibility. The DKM method gave the best performance, as it improves the heading errors by a large margin and its trajectories are guaranteed to be kinematically feasible. This strongly suggests benefits of combining powerful deep methods with well-grounded kinematic models.

In Figure 5 we provide a more detailed analysis of the higher derivatives of output trajectories for UM (top) and DKM (bottom) methods. In particular, we investigated the distributions of acceleration and turning rates, and how they compared to the distribution of the ground-truth values. We report Wasserstein distance (WD) to quantify differences between the distributions, given above each subfigure. While in both cases distribution of DKM-computed values was closer to the ground truth, this especially holds true for turning rate where the improvement was substantial, as indicated by more than two times smaller WD. Note that y-axes are given in log-scale, and both tails of the ground-truth distribution represent rare events and outliers.

B. Case studies

Next we provide an analysis of several common traffic situations, and compare performance of representative unconstrained method UM, and two best-performing constrained methods, poly-2 and DKM. We analyzed the following cases: a) actor in a right-turn lane; b) actor approaching red traffic light and braking for stopped vehicles. We present the results in Figure 6, where the first column gives input raster with overlaid color-coded top-ranked output trajectory of each method, followed by time series of predicted actor states.

In the first case all three models correctly predicted that the actor is going to make a right turn at the intersection. We can see that DKM, in addition to accurate position estimates, also outputs smooth acceleration and heading profiles. On the other hand, UM does predict reasonable positions, however acceleration and turning rate are very noisy. It is interesting to consider poly-2, which similarly to DKM also gives smooth outputs. While this results in more realistic profiles of all physical measures, the method is too constrained and gives a trajectory that cuts the corner and overshoots the turn (shown as a dark green trajectory in the first column). Similarly to CTRA (results not shown), these approaches may be overly constrained and not capable of fully capturing vehicle motion during more complex maneuvers.

In the second case all models predicted the actor to come to a stop. Note that poly-2, which amounts to a constant acceleration method, at one point starts predicting that the actor will move in reverse. We can see that the UM method again outputs noisy acceleration and heading estimates, while acceleration profile of DKM best corresponds to the ground truth with a very smooth deceleration.

V. CONCLUSION

We presented a method addressing the critical task of motion prediction of traffic actors, guaranteeing that predicted motion is kinematically feasible. The approach combines powerful deep learning algorithms on one side and vehicle models developed in robotics on the other. While the method is general and can be used with any machine learning algorithm, including convolutional and recurrent deep networks, we evaluated the approach using convnets as an example. Extensive experiments on real-world, large-scale data collected by an SDV fleet strongly indicated its benefits, with the proposed method outperforming the existing state-of-the-art. Following extensive offline testing, the framework was subsequently successfully tested onboard the SDVs.

REFERENCES

[1] C. Urmson and W. Whittaker, "Self-driving cars and the urban challenge," *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 66–68, 2008.

[2] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," National Highway Traffic Safety Administration, Tech. Rep. DOT HS 812 506, March 2018.

[3] NHTSA, "Early estimate of motor vehicle traffic fatalities for the first half (jan-jun) of 2018," National Highway Traffic Safety Administration, Tech. Rep. DOT HS 812 629, October 2018.

[4] D. Howard and D. Dai, "Public perceptions of self-driving cars: The case of Berkeley, California," in *Transportation Research Board 93rd Annual Meeting*, vol. 14, 2014.

[5] S. Casas, W. Luo, and R. Urtasun, "Intentnet: Learning to predict intention from raw sensor data," in *Conference on Robot Learning*, 2018, pp. 947–956.

[6] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proceedings of the IEEE CVPR*, 2018, pp. 3569–3577.

[7] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, N. Singh, and J. Schneider, "Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving," *arXiv preprint arXiv:1808.05819*, 2018.

[8] H. Cui, V. Radosavljevic, *et al.*, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[9] 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*, 2017, pp. 336–345.

[10] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, "Multi-agent tensor fusion for contextual trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 126–12 134.

[11] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "Sophie: An attentive gan for predicting paths compliant to social and physical constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1349–1358.

[12] P. Zhang, W. Ouyang, P. Zhang, J. Xue, and N. Zheng, "Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 085–12 094.

[13] F.-C. Chou, T.-H. Lin, H. Cui, V. Radosavljevic, T. Nguyen, T.-K. Huang, M. Niedoba, J. Schneider, and N. Djuric, "Predicting motion of vulnerable road users using high-definition maps and efficient convnets," in *Workshop on 'Machine Learning for Intelligent Transportation Systems' at Conference on Neural Information Processing Systems (MLITS)*, 2018.

[14] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[15] A. Cosgun, L. Ma, *et al.*, "Towards full automated drive in urban environments: A demonstration in gomentum station, california," in *IEEE Intelligent Vehicles Symposium*, 2017, pp. 1811–1818. [Online]. Available: https://doi.org/10.1109/IVS.2017.7995969

[16] J. Ziegler, P. Bender, M. Schreiber, *et al.*, "Making bertha drive—an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, pp. 8–20, 10 2015.

[17] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[18] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[19] S. Chen, "Kalman filter for robot vision: a survey," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, 2012.

[20] T. Streubel and K. H. Hoffmann, *Prediction of driver intended path at intersections*. IEEE, Jun 2014. [Online]. Available: http://dx.doi.org/10.1109/IVS.2014.6856508

[21] M. Schreier, V. Willert, and J. Adamy, "An integrated approach to maneuver-based trajectory prediction and criticality assessment in arbitrary road environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 10, p. 2751–2766, Oct 2016. [Online]. Available: http://dx.doi.org/10.1109/TITS.2016.2522507

[22] J. Wang, D. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *PAMI*, vol. 30, no. 2, p. 283–298, Feb 2008. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2007.1167

[23] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, Montreal, CA, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[24] B. Kim, C. M. Kang, S. H. Lee, H. Chae, J. Kim, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," *arXiv preprint arXiv:1704.07049*, 2017.

[25] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6120–6127.

[26] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti, "Visual interaction networks: Learning a physics simulator from video," in *Advances in neural information processing systems*, 2017, pp. 4539–4547.

[27] J. Liang, L. Jiang, J. C. Niebles, A. G. Hauptmann, and L. Fei-Fei, "Peeking into the future: Predicting future person activities and locations in videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5725–5734.

[28] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[29] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[30] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–7.

[31] R. Chandra, U. Bhattacharya, A. Bera, and D. Manocha, "Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8483–8492.

[32] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[33] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, pp. 1–. [Online]. Available: http://doi.acm.org/10.1145/1015330.1015430

[34] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." 2008.

[35] N. Rhinehart, K. M. Kitani, and P. Vernaza, "R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 772–788.

[36] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design." in *Intelligent Vehicles Symposium*, 2015, pp. 1094–1099.

[37] T. Howard, C. Green, and A. Kelly, "Receding horizon model-predictive control for mobile robot navigation of intricate paths," in *International Conferences on Field and Service Robotics*, 2009.

[38] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2013, pp. 2335–2340.

[39] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 8299–8310.

[40] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium*, 2000, pp. 153–158.

[41] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking." in *FUSION*. IEEE, 2008, pp. 1–6.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.