Ellipse Loss for Scene-Compliant Motion Prediction

Henggang Cui[1*], Hoda Shajari[2*], Sai Yalamanchi[1], Nemanja Djuric[1]

Abstract— **Motion prediction is a critical part of self-driving technology, responsible for inferring future behavior of traffic actors in autonomous vehicle's surroundings. In order to ensure safe and efficient operations, prediction models need to output accurate trajectories that obey the map constraints. In this paper, we address this task and propose a novel *ellipse loss* that allows the models to better reason about scene compliance and predict more realistic trajectories. Ellipse loss penalizes off-road predictions directly in a supervised manner, by projecting the output trajectories into the top-down map frame using a differentiable trajectory rasterizer module. Moreover, it takes into account actor dimensions and orientation, providing more direct training signals to the model. We applied ellipse loss to a recently proposed state-of-the-art joint detection-prediction model to showcase its benefits. Evaluation on large-scale autonomous driving data strongly indicates that the method allows for more accurate and more realistic trajectory predictions.**

I. INTRODUCTION

An autonomous vehicle is a highly involved system, supported by several input sensors (such as LiDARs, radars, and cameras), required to handle a large number of complicated situations in an uncertain environment [1]. Following the completion of the perception step where the current state of the world is inferred, one of the most critical parts of the self-driving vehicle (SDV) framework is predicting how will the world state then evolve in the near future, a task addressed by the motion prediction module of the SDV. Understanding the motion of surrounding actors, as well as the uncertainty of that motion, is necessary for ensuring safe and efficient routing of the SDV through a stochastic world. Despite its importance, the task of motion prediction is still far from being solved, and significant research efforts are being invested in furthering the current state-of-the-art.

An important aspect of the motion prediction problem is ensuring that the inferred trajectories are *scene compliant* and that they obey the constraints imposed by the underlying map [2], [3], [4]. This is however not a simple task. The map data presents highly structured information, and it is not immediately clear how to utilize it in the learning model to help constrain the outputs and improve scene compliance. Recent work proposed to provide it as an extra input [5], [6], to incorporate it as a part of the loss [4], or as a post-processing step [7]. Nevertheless, the existing methods commonly struggle with this requirement, and more work is

required to improve the scene compliance performance of the state-of-the-art motion prediction models.

We focus on this important problem and propose a method to enhance scene compliance of predicted trajectories. We propose a *Box-aware Differentiable Trajectory Rasterizer* (BDTR) module to project the output trajectories into the top-down map, extending ideas proposed in [4]. However, unlike the previous work that treats each actor as a point [4], BDTR explicitly incorporates actor's bounding box dimensions and orientation information. We propose a novel *ellipse loss* that employs BDTR to align the map and trajectory representations and penalize off-road trajectory predictions in a supervised manner to improve their scene-compliance.

We summarize the contributions of our work below:

- we propose the BDTR module that projects output trajectories into a top-down map while incorporating actor's bounding box and orientation, extending beyond point processes used by the current state-of-the-art;
- we propose an ellipse loss that uses BDTR in a supervised manner to directly penalize off-road predictions;
- the proposed ellipse loss is general, and we demonstrate its effectiveness by applying it to a state-of-the-art joint detection-prediction model, resulting in improved prediction performance on a large-scale real-world autonomous driving data set.

II. RELATED WORK

A. Trajectory prediction for autonomous driving

Trajectory prediction is one of the most critical tasks in an autonomous system, allowing SDVs to safely navigate in an uncertain real-world environment. Given the historical observations, the trajectory prediction module predicts future positions of the given actors, which can then be consumed by the motion planning module of the SDV system. The current state-of-the-art approaches rely on the deep neural network models for this task. They commonly use an encoder network to generate embeddings from the historical observations (e.g., by using recurrent layers) and a decoder network to generate future trajectories as well as their uncertainties (e.g., [8], [9], [10], [11], [12], [13], [14], [15]). In the cases when there is map information available, a common approach is to rasterize the map polygons (e.g., lanes and other drivable surfaces) into a multi-channel bird's-eye view (BEV) image, and use a convolutional neural network to extract scene context features from the BEV image which can then be fused with the embeddings pertaining to the historical observation [5], [6], [10], [11], [16], [17], [18].

[1]Uber Advanced Technologies Group (ATG), 50 33rd Street, Pittsburgh, PA 15201; emails: {hcui2, syalamanchi, ndjuric}@uber.com

[2]University of Florida, work done during internship at Uber ATG; email: shajaris@ufl.edu

*Authors contributed equally.

The *waypoint representation* is the most popular trajectory representation used by most state-of-the-art prediction models (e.g., [10], [11], [16], [17], [18]), where each trajectory is represented as a sequence of future position coordinates $\{(x_t, y_t)\}_{t=1}^{T_f}$, where T_f is the prediction horizon. This representation is compact and allows for an efficient interface between the prediction and the motion planning modules of the autonomous driving system. Some researchers proposed to use an *occupancy grid representation* that encodes the future motion prediction as an occupancy grid [2], [3], [19]. However, while the trajectory representation can be exactly cast into an occupancy grid the opposite is not true, leading to the loss of detailed information that some online systems rely on. Moreover, the grid representation is less compact, resulting in larger memory and latency costs which makes this approach less commonly used in applied systems.

The state-of-the-art prediction models are often trained in a supervised fashion, using some sort of a distance measure (e.g., ℓ_2 norm, smooth ℓ_1, KL divergence, or log-likelihood) between the output trajectories and the ground-truth trajectory as their loss function (e.g., [6], [9], [10], [11], [16], [17], [18]). In recent years Generative Adversarial Networks (GANs) have shown to reach good performance [4], [12], [14], [15], where the trajectory decoder is trained with the help of a discriminator network. However, GAN-based models are rarely used in real-world systems as they require one to draw many samples to sufficiently cover the output space (e.g., as many as 20 samples in [4], [12], [14]), which makes them impractical for time-sensitive applications.

When it comes to model inputs, some works assume access to a separate detection and tracking module that infers historical actor tracks ingested by the model (e.g., [8], [10], [15], [16]). On the other hand, recently proposed approaches directly take in the raw LiDAR point clouds as their inputs and jointly perform object detection and trajectory prediction in the same network (e.g., [5], [6], [20], [21], [22], [23], [24]). The benefit of such joint models is that there is no loss of information between the detection and prediction modules, where the prediction module has access to richer information from the raw sensor data. Our work focuses on these models and proposes a novel loss that can be directly applied to improve prediction performance.

B. Scene-compliant trajectory prediction

Traditional loss functions such as ℓ_2 norm, smooth ℓ_1, or KL divergence used in previous work focus on minimizing the distance from the prediction to the ground truth, however, they do not have any explicit mechanism to enforce the trajectories to be scene-compliant (e.g., stay within the drivable region). In order to encourage the predicted trajectories to be scene-compliant, some authors have proposed various types of scene-compliant loss functions to explicitly penalize off-road predictions, which we discuss in this section.

ChauffeurNet [19], DRF-NET [3], and Ridel et al. [2] proposed to represent each predicted trajectory waypoint as an occupancy grid and computed the product of the occupancy grid and a non-drivable region mask as one

of their loss terms to penalize off-road predictions. To obtain trajectories in the form of waypoint representation, Ridel et al. [2] sampled the waypoint positions from each predicted occupancy grid as a post-processing step. This approach, however, cannot guarantee that the waypoints sampled from a scene-compliant occupancy grid will remain scene-compliant. Niedoba et al. [25] proposed a scene-compliant loss that can be directly applied to the waypoints. In their work, they identified predicted waypoints that are not scene-compliant and upweighted their loss by a fixed factor, so the model focuses more on such cases. Wang et al. [4] proposed a *Differentiable Trajectory Rasterizer* (DTR) module that bridges the waypoint and the occupancy grid representations. DTR converts a waypoint coordinate (x, y) into an occupancy grid in a differentiable manner, so the losses previously only applicable to the occupancy grids can be directly applied to the waypoint representations as well. Different from our work, the DTR module from Wang et al. [4] treated each actor as a point and focused on keeping the actor centroid inside the drivable region instead of the whole bounding box. Moreover, Wang et al. [4] used a GAN architecture and applied their DTR in a discriminator network to make it more sensitive to trajectories that are not scene-compliant. On the other hand, we apply our Box-aware DTR module in a supervised manner and compute our novel ellipse loss as the product of the occupancy grid and the non-drivable region mask, similarly to [2], [3], [19], resulting in a more flexible and easier-to-use architecture.

Instead of treating each actor as a point as in previous works [2], [3], [4], [25], we fully consider the actor bounding box dimensions by proposing a novel Box-aware DTR module. The BDTR module has two important distinctions compared to DTR. First, we use a 2D Gaussian distribution parameterized by an actor's dimensions, as opposed to a fixed distribution. Second, we truncate the rasterized occupancy grid based on the object's bounding box dimensions. These extensions allow the proposed ellipse loss to make the entirety of the actor's bounding box scene-compliant instead of just the centroid. The evaluation results indicate that the introduced extensions are critical in improving the prediction performance in a supervised setting.

III. ELLIPSE LOSS

In this section we introduce our novel scene-compliant *ellipse loss*. While the proposed approach is generic and can be applied to any motion prediction model, we implemented and evaluated it on top of MultiXNet [6], a state-of-the-art joint object detection and motion prediction model.

A. Backbone network and output representation

The MultiXNet model takes a total of T_h current and historical LiDAR sweeps as input, as well as the high-definition map of SDV's surroundings (containing M different map elements). Assuming the area of interest centered on SDV to be of length L, width W, and height V, the method voxelizes the LiDAR points into a 3D tensor $\mathcal{V}$ of shape $L \times W \times (VT_h)$ with a voxel size of $\Delta_L \times \Delta_W \times \Delta_V$, and rasterizes the map

onto a multi-channel binary image $\mathcal{M}$ of shape $L \times W \times M$ using the same resolution. Moreover, we assume that we are provided with a binary *drivable region mask* $\mathcal{D}$ of the same dimensions and resolutions as the LiDAR and map rasters, with drivable cells encoded as 1 and non-drivable cells as 0.

The model uses a multi-scale convolutional network as the *first-stage* network, which runs on the concatenation of $\mathcal{V}$ and $\mathcal{M}$ and produces the probability that each grid cell contains an object and the corresponding object bounding box parameters. The detections (i.e., bounding boxes with high objectness probabilities) are then passed as input to the *second-stage* network, which then outputs their future trajectory predictions (represented as a sequence of future waypoints) for a total of T_f time steps. The waypoint of the i-th object at each time step t is parameterized as $\mathbf{s}_t^{(i)} = (x_t^{(i)}, y_t^{(i)}, l_t^{(i)}, w_t^{(i)}, \theta_t^{(i)})$, representing its center position, box length, box width, and box orientation, respectively. We assume that the bounding box remains constant over time, and use the same box size $(l^{(i)}, w^{(i)})$ for all time steps t.

B. Loss function

The first-stage and second-stage networks are trained end-to-end. A focal loss term is applied to the object detection score of each grid cell from the first-stage network. Since we applied our ellipse loss to the trajectory predictions from the second-stage network, we will focus on discussing the second-stage prediction losses in the remainder of the paper.

The baseline MultiXNet model optimizes the following loss on the second stage outputs,

$$
\begin{aligned}
\mathcal{L}_{\text{vanilla}} = \sum_{i=1}^{N} \sum_{t=1}^{T_f} & \ell_1\big(x_t^{(i)} - x_t^{(i)*}\big) + \ell_1\big(y_t^{(i)} - y_t^{(i)*}\big) \\
& + \ell_1\big(l_t^{(i)} - l_t^{(i)*}\big) + \ell_1\big(w_t^{(i)} - w_t^{(i)*}\big) \\
& + \ell_1\big(\sin(\theta_t^{(i)}) - \sin(\theta_t^{(i)*})\big) \\
& + \ell_1\big(\cos(\theta_t^{(i)}) - \cos(\theta_t^{(i)*})\big),
\end{aligned} \tag{1}
$$

where N is the number of true positive detections, ℓ_1 represents the smooth-ℓ_1 loss, and $(x_t^{(i)*}, y_t^{(i)*}, l_t^{(i)*}, w_t^{(i)*}, \theta_t^{(i)*})$ are their ground-truth waypoints.

We extend MultiXNet by adding the proposed ellipse loss as an additional loss term to make trajectory predictions more scene-compliant[1]. Given a predicted waypoint $\mathbf{s}_t^{(i)} = (x_t^{(i)}, y_t^{(i)}, l_t^{(i)}, w_t^{(i)}, \theta_t^{(i)})$, we use our novel Box-aware DTR module to convert it to a 2D occupancy grid $\mathcal{G}(\mathbf{s}_t^{(i)})$ (referred to as *Gaussian raster* and discussed in detail in the following section) with the same dimensions $L \times W$ and grid cell size $\Delta_L \times \Delta_W$ as the drivable region mask $\mathcal{D}$. Then, we compute the ellipse loss $\mathcal{L}_{\text{ellipse}}$ as follows,

$$
\mathcal{L}_{\text{ellipse}} = \sum_{i=1}^{N} \sum_{t=1}^{T_f} I_t^{(i)*} \sum_{\text{cells}} \mathcal{G}(\mathbf{s}_t^{(i)}) \circ (1 - \mathcal{D}), \tag{2}
$$

where $1 - \mathcal{D}$ is the non-drivable region mask, $\circ$ denotes the Hadamard product, and $I_t^{(i)*}$ is an indicator term equal

[1]The ellipse loss only encourages the predictions to stay inside the drivable region, so the original prediction loss (1) is still required to encourage the predicted trajectories to be close to the ground truth.
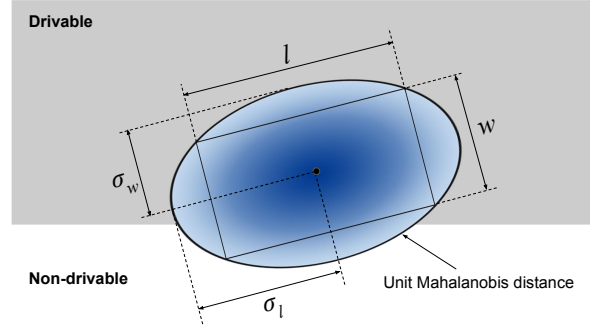

Fig. 1: Illustration of the ellipse loss; inscribed rectangle denotes the actor's bounding box, blue ellipse is the unit Mahalanobis distance range in which the loss is computed

to 1 when the ground-truth box $\mathbf{s}_t^{(i)*}$ is inside the drivable region and 0 otherwise. We included $I_t^{(i)*}$ in the loss to avoid penalizing off-road predictions when the ground truth is already off-road (e.g., parked cars on the side of the road).

Then, the final second-stage loss is computed as

$$
\mathcal{L}_{\text{ours}} = \mathcal{L}_{\text{vanilla}} + \lambda \mathcal{L}_{\text{ellipse}}, \tag{3}
$$

where the ellipse loss is weighted by a fixed parameter λ. Note that BDTR is only used during training for the loss computation, and does not impact model inference latency.

C. Box-aware Differentiable Trajectory Rasterization

In this section we present the novel Box-aware DTR module that converts a waypoint $\mathbf{s}_t^{(i)}$ into an occupancy grid $\mathcal{G}(\mathbf{s}_t^{(i)})$. Since the operation is applied independently to each actor and waypoint, for clarity we omit the actor index i and time index t in the remainder of the section.

BDTR computes a value of each cell (i, j) in the grid $\mathcal{G}(\mathbf{s})$ (denoted as $\mathcal{G}_{i,j}(\mathbf{s})$) as the density of a 2D Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ evaluated at $\mathbf{d}_{i,j}(x, y)$,

$$
\mathcal{G}_{i,j}(x, y, l, w, \theta) = \mathcal{N}\big(\mathbf{d}_{i,j}(x, y); \mathbf{0}, \mathbf{\Sigma}(l, w, \theta)\big). \tag{4}
$$

Value of $\mathbf{d}_{i,j}(x, y)$ is computed as a displacement vector from the waypoint center position (x, y) to the grid cell (i, j), where the grid cell position is in the same coordinate frame as the waypoint coordinate (x, y).

Covariance matrix as a function of actor state. Different from the original DTR module [4] where the covariance matrix is symmetric and hard-coded as $\mathbf{\Sigma} = \text{diag}(\sigma, \sigma)$ with a predefined σ, we define $\mathbf{\Sigma}$ in (4) as a function of the actor's predicted dimensions l and w, as well as its predicted orientation θ. In particular, for each actor we set the standard deviation of each of the two axes of the Gaussian distribution as a linear function of the bounding box size (l, w) as

$$
(\sigma_l, \sigma_w) = (kl, kw), \tag{5}
$$

where k is a fixed positive scaling factor. We further rotate the covariance matrix to be aligned with the predicted actor's orientation θ, computed as

$$
\mathbf{\Sigma}(l, w, \theta) = \mathbf{R}(\theta)^\top \text{diag}(\sigma_l, \sigma_w) \mathbf{R}(\theta), \tag{6}
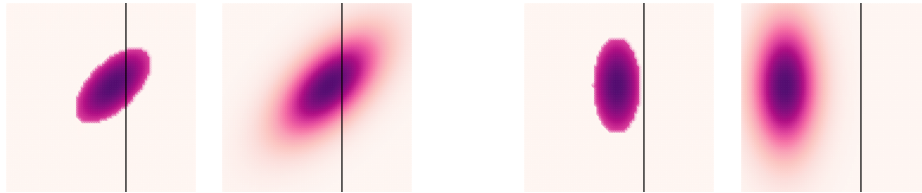$$

Fig. 2: Characterization of the ellipse loss on a toy example, showing that the loss pushes the actor away from the non-drivable region (right to the boundary line) by correcting both the center position and the orientation of the bounding box; the left two figures show the initial actor state, and the right two figures show the state after 1,000 iterations of gradient updates for the ellipse loss with and without truncation, respectively

where $\mathbf{R}(\theta)$ represents a rotation matrix constructed from the actor's orientation θ.

Truncation by bounding box dimension. Multiplying the entire Gaussian raster with the non-drivable region mask in the loss (2) has a tendency to over-compensate and push the predictions too far away from the non-drivable region, which often results in worse performance, as illustrated in the following Section III-D and empirically shown in Section IV-D. To address this issue, we propose to truncate the Gaussian raster based on the actor's bounding box to limit its range.

Earlier work has shown that the gradient magnitude of the Gaussian raster reaches its maximum at the unit Mahalanobis distance [4]. Based on this property, we propose to truncate the Gaussian raster at this distance, which corresponds to an ellipse with radii σ_l and σ_w. For each actor, we intend this ellipse to circumscribe the actor's bounding box (l, w), to ensure scene-compliance while minimizing over-compensation. As a result, we set $k = \frac{\sqrt{2}}{2}$ in (5) and zero out all the values outside the truncation range. We stop the gradients of $\mathcal{L}_{\text{ellipse}}$ w.r.t. l and w to prevent the model from minimizing the ellipse loss by incorrectly reducing the bounding box size.

Figure 1 illustrates the range of the truncated Gaussian raster. The intuition is that we only want to penalize predictions that cause the actor bounding box to be outside the drivable region, and when the bounding box is already outside the non-drivable region it should have zero loss.

D. Characterization on a toy example

We run simple toy experiments to exemplify how the proposed ellipse loss pushes away the non-scene-compliant actors from the non-drivable region, shown in Figure 2. In particular, we mark the region to the right of the black boundary line as non-drivable and place the actor's bounding box so as to straddle the boundary. Two leftmost figures show the actor's initial position and orientation, as well as its Gaussian raster. We then use the gradients from the ellipse loss to update the actor's position and orientation, running for 1,000 iterations. We performed two sets of experiments, one with the Gaussian raster truncated at unit Mahalanobis distance and another experiment with no truncation.

The final states after the gradient updates are plotted in the two rightmost figures. Both the truncated and non-truncated ellipse losses were able to push the actor away from the non-drivable region by correcting both the center position and

the orientation. We can see that the truncated loss pushed the actor to be exactly at the boundary of the non-drivable region. The non-truncated ellipse loss, on the other hand, kept pushing the actor further away from the boundary. Although it may seem that the loss resulting in a state that is further from the boundary might be a better choice, traffic actors in the real world do not behave in that way. This is also confirmed by the improved experimental performance of the truncated loss, presented later in Section IV-D.

IV. EVALUATION

A. Experimental setup

Data set. We evaluated the proposed ellipse loss on the ATG4D autonomous driving data set [6], [26]. The data was collected in several North American cities by a fleet of SDVs, each equipped with a 64-beam roof-mounted LiDAR, containing over 1 million frames captured at $10Hz$ from 5,500 different scenarios. It also includes manually labeled object bounding boxes and detailed map polygons. Since the data set does not have an explicit drivable region flag, we treated the union of all map components (including lanes, intersections, etc.) as the drivable region. Note that an exact definition of the drivable region is orthogonal to our method, and one can also customize the drivable region for each individual actor to include only the lanes that the actor has access to, if such information is available in the data.

Implementation details. We used the same model architecture as MultiXNet [6] for all of our experiments, with the only difference among the competing models being their loss functions. We set the BEV voxel, map raster, and drivable region mask input shape and resolution to $L = 150$m, $W = 100$m, $V = 3.2$m, $\Delta_L = 0.16$m, $\Delta_W = 0.16$m, $\Delta_V = 0.2$m. We used 1s of historical LiDAR inputs ($T_h = 10$ at 10Hz) to predict 3s-long future trajectories ($T_f = 30$ at 10Hz). For the models employing the ellipse loss we set $\lambda = 0.03$, obtained through cross-validation. The models were implemented in PyTorch [27] and trained using the same setup as MultiXNet [6].

Evaluation metrics. While the method is general and can be applied to any actor type, in this work we focus on evaluating vehicle trajectory prediction. For each competing model, we evaluated ℓ_2 error of its trajectory predictions, as well as the *off-road false positive* (ORFP) metrics [4] for both the final 3-second waypoint and averaged over the

TABLE I: Comparison of the competing methods; confidence intervals are computed over 3 runs

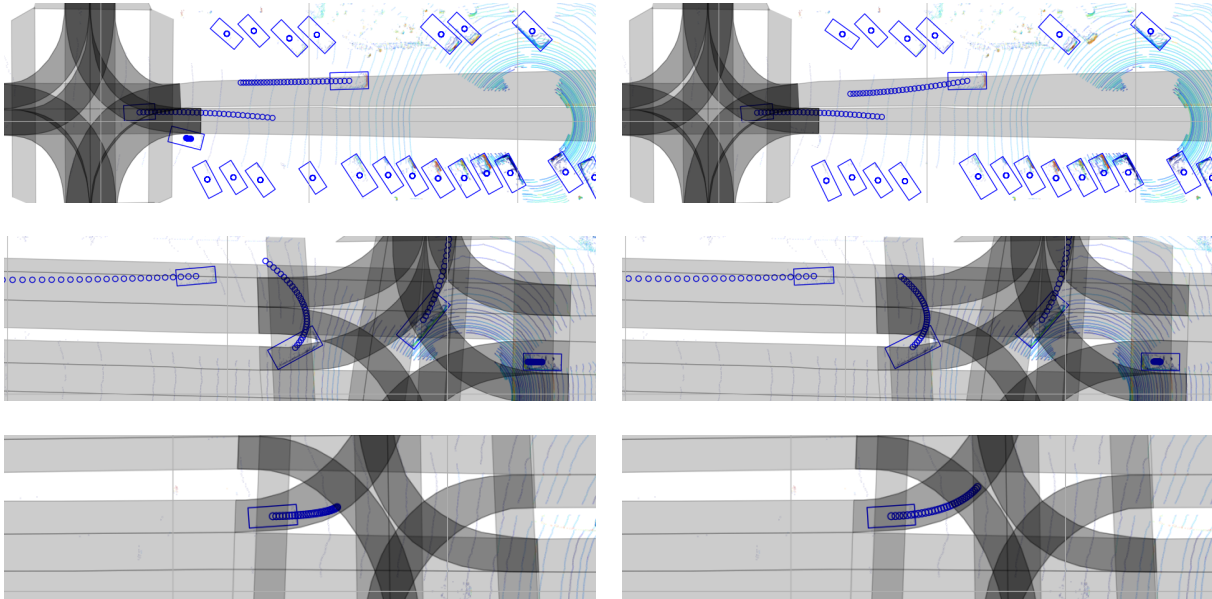| Method	ℓ_2 [m] $\downarrow$		CtrORFP [%] $\downarrow$		BoxORFP [%] $\downarrow$	
	Avg	@3s	Avg	@3s	Avg	@3s
MultiXNet [6]	$\mathbf{0.465} \pm \mathbf{0.001}$	$\mathbf{0.836} \pm \mathbf{0.002}$	0.061 ± 0.007	0.094 ± 0.018	0.624 ± 0.009	0.850 ± 0.029
Off-Road [25]	$\mathbf{0.465} \pm \mathbf{0.002}$	$\mathbf{0.837} \pm \mathbf{0.005}$	0.058 ± 0.001	0.086 ± 0.006	0.616 ± 0.015	0.853 ± 0.020
Ellipse-Loss	0.472 ± 0.004	0.846 ± 0.008	$\mathbf{0.031} \pm \mathbf{0.003}$	$\mathbf{0.043} \pm \mathbf{0.006}$	$\mathbf{0.445} \pm \mathbf{0.007}$	$\mathbf{0.485} \pm \mathbf{0.004}$

Fig. 3: Case studies for MultiXNet (left) and Ellipse-Loss (right) on three representative scenarios; the detected objects and trajectory predictions are plotted in blue

whole trajectory for each vehicle actor. We refer to an output trajectory waypoint as being off-road false positive if it is predicted to be off-road while the corresponding ground-truth waypoint is in-road. The ORFP ratio at a particular horizon is computed as the number of predictions that are ORFP at this horizon over the total number of predictions[2]. We used two policies to decide whether a waypoint is off-road or not. The *center-off-road* policy labels a waypoint off-road if the center (x, y) position of the bounding box is outside the drivable region, and the *box-off-road* policy labels a waypoint off-road if any of the four corners of the bounding box is outside the drivable region. We refer to these two metrics as CtrORFP and BoxORFP, respectively.

The competing models perform joint object detection and trajectory prediction, achieving nearly identical detection performance (result not shown). In order to make sure the prediction metrics are fairly compared, we set the detection score threshold for each model such that they all achieve the same 0.8 recall with an IoU threshold of 0.5 for vehicles, as suggested by earlier work [6], [22]. Then, the prediction ℓ_2 error and ORFP metrics are evaluated on true-positive detections computed at this threshold.

[2]If a predicted waypoint or a ground truth is outside the 150m × 100m raster range, we re-use the ORFP result of a previous waypoint.

Baselines. We compared our approach against recently proposed state-of-the-art methods: a MultiXNet baseline [6] that uses the $\mathcal{L}_{\text{vanilla}}$ term from (1) in the loss, and Off-Road method [25] that directly penalizes the positions of off-road false positive waypoints. In particular, following the setup from [25] we upweight the $\ell_1\left(x_t^{(i)} - x_t^{(i)*}\right)$ and $\ell_1\left(y_t^{(i)} - y_t^{(i)*}\right)$ terms of $\mathcal{L}_{\text{vanilla}}$ by a factor of 5.

B. Quantitative results

We trained Ellipse-Loss and the baselines three times, with means and standard deviations reported in Table I where the best-performing methods are marked in bold. We can see that the Off-Road method achieved better results than MultiXNet, where the additional penalty of off-road losses led to larger scene compliance without affecting prediction performance. In addition, the results show that Ellipse-Loss had significantly lower CtrORFP and BoxORFP ratios than either of the baseline methods. In particular, it reduced both CtrORFP and BoxORFP by nearly a factor of 2, while obtaining comparable ℓ_2 error metrics.

C. Qualitative results

In this section we present the case studies on three representative scenarios, shown in Figure 3 which compares the predictions from Ellipse-Loss and MultiXNet. In

TABLE II: Ablation study of the ellipse loss with different truncation settings

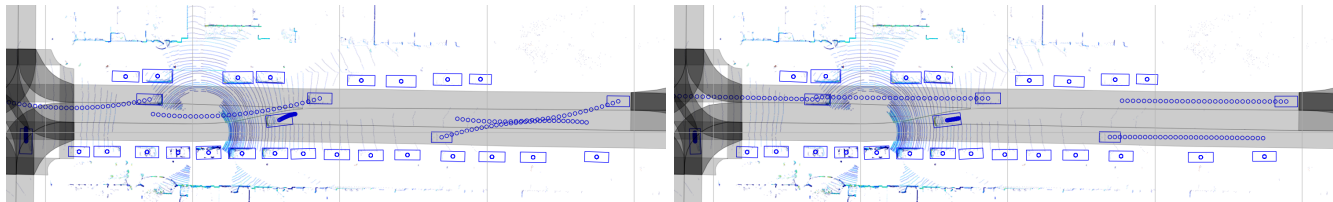| Method	ℓ_2 @3s [m] ↓ Val.	Rel.	CtrORFP @3s [%] ↓ Val.	Rel.	BoxORFP @3s [%] ↓ Val.	Rel.
`0.5Md-ellipse`	0.845	-0.1%	0.056	+30%	0.738	+52%
`1Md-ellipse`	0.846	0%	0.043	0%	0.485	0%
`2Md-ellipse`	0.899	+6%	0.033	-23%	0.348	-28%
`No-truncation`	1.719	+103%	0.017	-60%	0.139	-71%

Fig. 4: Comparison of predicted trajectories without truncation (left) and with truncation (right) of the Gaussian raster; in the case of no truncation the minimization of the ellipse loss pushes the trajectories towards the middle of a drivable region

the first scenario the actor was merging into the lane from roadside parking. We can see that the inferred trajectories from `MultiXNet` continued driving along the side of the road, while the prediction from `Ellipse-Loss` returned back to the road sooner, which was the ground-truth behavior as well. In the second scenario the actor was making a U-turn in an intersection. The left-hand figure shows that the prediction from `MultiXNet` went outside the road for the last few waypoints, while `Ellipse-Loss` trajectory was more scene-compliant and stayed inside the road. In the third scenario the actor was making a left turn following a left-turning lane. We see that the prediction from `MultiXNet` did not follow the lane well, resulting in half of the bounding box going outside of its lane for the last few waypoints. On the other hand, the prediction from `Ellipse-Loss` followed the lane much better, mirroring the ground-truth trajectory. We can conclude that the novel box-aware loss yields more scene-compliant predictions, as it encourages the entire bounding box to stay inside the drivable region.

D. Ablation study

We performed an ablation study with several variations of the ellipse loss to better characterize its effects. In particular, we truncated the Gaussian raster at different Mahalanobis distances, with the results summarized in Table II. For `2Md-ellipse` we truncated the loss at a Mahalanobis distance of 2, which results in an ellipse that is 4× the size of the regular ellipse, denoted as `1Md-ellipse`. Similarly, `0.5Md-ellipse` has 0.25× the ellipse size compared to `1Md-ellipse`. For all metrics, we show both the absolute values and the relative deltas as compared to `1Md-ellipse`. The results show that the off-road errors decreased but the ℓ_2 errors increased as we increased the ellipse size. This is expected, as due to increased ellipse loss the model focuses more on improving the scene compliance at the expense of less accurate predictions. On the flip side, the `0.5Md-ellipse` method had a lower ℓ_2 error but significantly higher BoxORFP. We can conclude that the

`1Md-ellipse` model achieved the best balance between the two competing objectives.

Lastly, we also trained a `No-truncation` model without any truncation. Expectedly, this model had lower BoxORFP but significantly higher ℓ_2 errors than models with truncation. As discussed previously, less truncation of the loss has the tendency to push the predicted trajectories closer to the center of drivable regions, affecting the model accuracy. We visualize this characteristic in Figure 4 on an example scene, showing results aligned with the toy experiments presented in Section III-D. To avoid such undesired behavior, it is important to truncate the Gaussian raster by the size of the actor's bounding box. This allows the ellipse loss to not over-penalize predicted bounding boxes that are already inside the drivable region, resulting in output trajectories that are both accurate and scene-compliant.

V. CONCLUSION

We considered the problem of motion prediction, a critical component of an autonomous system ensuring safe and efficient operations. To improve the accuracy of output trajectories and ensure they obey the map constraints, we introduced a novel scene-compliant loss that takes into account the map and actor shape information. This is done in a supervised manner, by projecting the trajectories into a top-down raster image using a recently proposed differentiable trajectory rasterizer, that helps align trajectory and map representations and directly penalize off-road predictions. In addition, we showed how actor dimensions can be used to truncate the off-map loss, allowing for more precise loss computation. To evaluate the proposed approach we used a large, real-world autonomous driving data, and extended a state-of-the-art model that performs joint detection and prediction with the proposed loss. The results indicate that the method is significantly more realistic, while maintaining comparable prediction accuracy to the existing state-of-the-art approaches. Moreover, detailed ablation and case studies provided important insights into the method's performance.

REFERENCES

[1] 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.

[2] D. Ridel, N. Deo, D. Wolf, and M. Trivedi, "Scene compliant trajectory forecast with agent-centric spatio-temporal grids," *arXiv preprint arXiv:1909.07507*, 2019.

[3] A. Jain, S. Casas, R. Liao, Y. Xiong, S. Feng, S. Segal, and R. Urtasun, "Discrete residual flow for probabilistic pedestrian behavior prediction," *arXiv preprint arXiv:1910.08041*, 2019.

[4] E. Wang, H. Cui, S. Yalamanchi, M. Moorthy, F.-C. Chou, and N. Djuric, "Improving movement predictions of traffic actors in bird's-eye view models using GANs and differentiable trajectory rasterization," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2020.

[5] 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.

[6] N. Djuric, H. Cui, Z. Su, S. Wu, H. Wang, F.-C. Chou, L. S. Martin, S. Feng, R. Hu, Y. Xu, *et al.*, "Multixnet: Multiclass multistage multimodal motion prediction," *arXiv preprint arXiv:2006.02000*, 2020.

[7] S. Yalamanchi, T.-K. Huang, G. C. Haynes, and N. Djuric, "Long-term prediction of vehicle behavior using short-term uncertainty-aware trajectories and high-definition maps," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[8] 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.

[9] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2375–2384.

[10] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control," *arXiv preprint arXiv:2001.03093*, 2020.

[11] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," *arXiv preprint arXiv:1910.05449*, 2019.

[12] 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.

[13] 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.

[14] 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.

[15] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese, "Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 137–146.

[16] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2090–2096.

[17] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, and J. Schneider, "Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[18] 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 *IEEE Intelligent Vehicles Symposium (IV)*, 2020.

[19] 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.

[20] 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.

[21] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8660–8669.

[22] S. Casas, C. Gulino, R. Liao, and R. Urtasun, "Spatially-aware graph neural networks for relational behavior forecasting from sensor data," *arXiv preprint arXiv:1910.08233*, 2019.

[23] P. Wu, S. Chen, and D. Metaxas, "Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps," *arXiv preprint arXiv:2003.06754*, 2020.

[24] G. P. Meyer, J. Charland, S. Pandey, A. Laddha, C. Vallespi-Gonzalez, and C. K. Wellington, "Laserflow: Efficient and probabilistic object detection and motion forecasting," *arXiv preprint arXiv:2003.05982*, 2020.

[25] M. Niedoba, H. Cui, K. Luo, D. Hegde, F.-C. Chou, and N. Djuric, "Improving movement prediction of traffic actors using off-road loss and bias mitigation," in *Workshop on 'Machine Learning for Autonomous Driving' at Conference on Neural Information Processing Systems*, 2019.

[26] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "Lasernet: An efficient probabilistic 3d object detector for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 677–12 686.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.