Deep Multi-Task Learning for Joint Localization, Perception, and Prediction

John Phillips[1,2] Julieta Martinez[1] Ioan Andrei Bârsan[1,3]
Sergio Casas[1,3] Abbas Sadat[1] Raquel Urtasun[1,3]

[1]Uber Advanced Technologies Group [2]University of Waterloo [3]University of Toronto
{john.phillips,julieta,andreib,sergio.casas,asadat,urtasun}@uber.com

Abstract

Over the last few years, we have witnessed tremendous progress on many subtasks of autonomous driving including perception, motion forecasting, and motion planning. However, these systems often assume that the car is accurately localized against a high-definition map. In this paper we question this assumption, and investigate the issues that arise in state-of-the-art autonomy stacks under localization error. Based on our observations, we design a system that jointly performs perception, prediction, and localization. Our architecture is able to reuse computation between the three tasks, and is thus able to correct localization errors efficiently. We show experiments on a large-scale autonomy dataset, demonstrating the efficiency and accuracy of our proposed approach.

1. Introduction

Many tasks in robotics can be broken down into a series of subproblems that are easier to study in isolation, and facilitate the interpretability of system failures [75]. In particular, it is common to subdivide the self-driving problem into five critical subtasks: (i) Localization: placing the car on a high-definition (HD) map with centimetre-level accuracy. (ii) Perception: estimating the number and location of dynamic objects in the scene. (iii) Prediction: forecasting the trajectories and actions that the observed dynamic objects might do in the next few seconds. (iv) Motion planning: coming up with a desired trajectory for the ego-vehicle, and (v) Control: using the actuators (*i.e.*, steering, brakes, throttle, *etc.*) to execute the planned motion.

Moreover, it is common to solve the above problems *sequentially*, such that the output of one sub-system is passed as input to the next, and the procedure is repeated iteratively over time. This *classical* approach lets researchers focus on well-defined problems that can be studied independently, and these areas tend to have well-understood metrics that measure progress on their respective sub-fields. For simplicity, researchers typically study autonomy subproblems under the assumption that its inputs are correct. For example, state-of-the-art perception-prediction (P2) and motion planning (MP) systems often take HD maps as input, thereby assuming access to accurate online localization. We focus our attention on this assumption and begin by studying the effect of localization errors on modern autonomy pipelines. Here, we observe that localization errors can have serious consequences for P2 and MP systems, resulting in missed detections and prediction errors, as well as bad planning that leads to larger discrepancies with human trajectories, and increased collision rates. Please refer to Fig. 1 for an example of an autonomy error caused by inaccurate localization.

In contrast to the classical formulation, recent systems have been designed to perform multiple autonomy tasks jointly. This *joint* formulation often comes with a shared neural backbone that decreases computational and system complexity, while still producing interpretable outputs that make it easier to diagnose system failures. However, these approaches have so far been limited to jointly performing perception and prediction (P2) [11,13,41,43], P2 and motion planning (P3) [52,69,70], semantic segmentation and localization [51] or road segmentation and object detection [60].

In this paper, and informed by our analysis of the effects of localization error, we apply the joint design philosophy to the tasks of localization, perception, and prediction; we refer to this joint setting as *LP2*. We design an LP2 system that shares computation between the tasks, which makes it possible to perform localization with as little as 2 ms of computational overhead while still producing intepretable localization and P2 outputs. We evaluate our proposed system on a large-scale dataset in terms of motion planning metrics, and show that the proposed approach matches the performance of a traditional system with separate localization and perception components, while being able to correct localization errors online, and having reduced run time and engineering complexity.

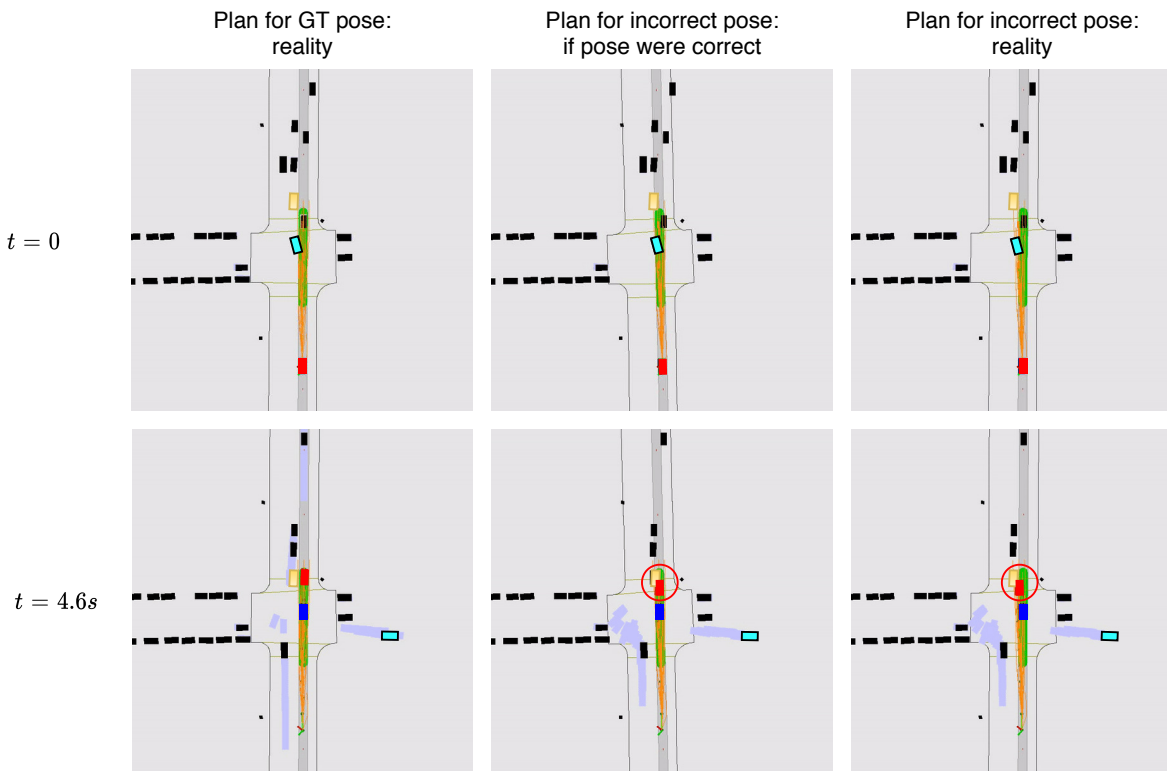

Figure 1: A scenario where a small amount of localization error results in a collision. The top row visualizes the first time step, and the bottom row visualizes a later time step where a collision occurs. GT labels are black rectangles, and the pale blue rectangles are forecasted object trajectories. The SDV is the red rectangle, with its GT trajectory in a dark blue rectangle. The samples predicted by the motion planner are shown as orange lines. The 3 columns visualize different variants of the same scenario. (Left) The planned trajectory of the SDV when there is no localization error. (Middle) What the SDV "thinks" is happening, based on its estimated pose that has error (x, y, yaw) = (10 cm, 0 cm, 1.5 deg). (Right) What the SDV is actually doing when subject to the pose error; this is the same trajectory as shown in the middle image, but rigidly transformed so that the initial pose agrees with the GT pose. The collision (red circle) occurs because the yellow vehicle is not perceived at $t = 0$ due to occlusion (by the cyan vehicle); the localization error then causes the SDV to go into the lane of opposite traffic which results in a collision.

2. Related Work

We provide a brief overview of existing approaches for the tasks that we study (perception, prediction, and localization), followed by a discussion of common multi-task paradigms for deep learning, and a review of recent work in characterizing and addressing the system-level challenges in perception.

Object Detection and Motion Prediction: Detecting actors and predicting their future motion from sensor data is one of the fundamental tasks in autonomous driving. While object detection and motion forecasting can be modeled as independent tasks [14, 19, 24, 49, 58, 66, 74], models that jointly perform both tasks [13, 41, 43, 72] have been shown to provide a number of benefits, such as fast inference, uncertainty propagation, and overall improved performance.

Localization: The objective of localization is to accurately and precisely determine the position of the ego-vehicle with respect to a pre-built map. Localization methods can be based on a wide variety of sensors, such as differential GPS in the form of Real-Time Kinematic systems [32, 63], LiDAR [8, 38, 42, 47, 68], cameras [31, 35, 54], RADAR [4, 59] or combinations of such sensors [44, 65, 76]. While purely geometric algorithms for LiDAR localization such as iterative closest-point [68] have been shown to be effective, recent work has shown that learned representations [8, 22, 42, 55, 76] can lead to improved robustness and scalability.

Multi-Task Learning: Compared to end-to-end approaches for autonomous agents that learn to directly map sensor readings to control output [3, 5, 36], multi-task modular approaches have been shown to perform better empirically [75], while also being more interpretable thanks to human-readable intermediary representations like semantic segmentation [75], object detections [70], occupancy forecasts [52] and planning cost maps [69]. Furthermore, Liang *et al*. [40] have shown the benefits of jointly perform-

ing mapping, object detection, and optical flow from LiDAR and camera data.

The wide range of recent multi-task learning approaches can be divided into two major areas. One line of work is focused primarily on developing and understanding network architectures, such as those leveraging a common backbone with task-specific heads [10, 17, 22, 33, 34, 43, 54, 60], cascaded approaches where some tasks rely on the outputs of others [20, 28, 29, 69, 70], or *cross-talk* networks such as Cross-Stitch [46] which have completely separate per-task networks, but share activation information.

Modular learning approaches such as Modular Meta-Learning [1], aim to construct reusable modular architectures which can be re-combined to solve new tasks. Side-Tuning [71] proposes an incremental approach where new tasks are added to existing neural networks in the form of additive side-modules that are easy to train, and have the advantage of leaving the weights of the original network unchanged, bypassing issues such as catastrophic forgetting.

Another line of work is concerned with the optimization process itself. The most straightforward approach is sub-task weighting, which may be based on uncertainty scores [34], learning speed [15] or performance [26]. Other methods have explored multi-task learning by performing multi-objective optimization explicitly [57], by regularizing task-specific networks through soft parameter sharing [67], or through knowledge distillation [6, 16]. Please see Crawshaw [18] for a detailed survey of multi-task deep learning.

Planning Under Pose Uncertainty: The task of planning robust trajectories under pose uncertainty has been studied in the past, with previous methods formulating it as a continuous POMDP which can be solved with an iterative linear-quadratic-Gaussian method [62], or as an optimal control problem solved using model-predictive control [30]. More recently, Artuñedo *et al*. [2] focus on autonomous vehicles and incorporate the pose uncertainty in a probabilistic map representation that is then leveraged by a sampling-based planner, while Zhang and Scaramuzza [73] propose an efficient way of estimating visual localization accuracy for use in motion planning. However, none of these approaches model other dynamic actors and the uncertainty in their own motion, and they do not study the complex interplay between pose uncertainty and state-of-the-art perception systems.

System-Level Analysis: A number of recent papers have studied the correlations between task-level metrics, such as object detection, and system performance [50]. This line of work has shown that while task-level metrics serve as good predictors of overall system performance, they are unable to differentiate between similar errors that can however lead to very different system behaviors. A related line of work analyzed the impact of sensor and inference latency on object detection in images [39] and LiDAR [23, 27]. At the same time, the simultaneous localization and mapping (SLAM) community [7, 21, 48] has recently proposed extending SLAM evaluation beyond trajectory accuracy [25], towards system-level metrics like latency, power usage, and computational costs.

3. The Effects of Localization Error

Since state-of-the-art perception-prediction (P2) and motion planning (MP) stacks make extensive use of accurate localization on high-definition maps (often assuming perfect localization [3, 11, 13, 69]), we study the effects of localization error on a state-of-the-art P2 and MP pipeline. We begin by describing how these modules work and how they use localization.

Perception-Prediction (P2): P2 models are tasked with perceiving actors and predicting their future trajectories to ensure that motion planning has access to safety-critical information about the scene for the entire duration of the planning horizon. We study the state-of-the-art Implicit Latent Variable Model [11] (ILVM), the latest of a family of methods that use deep neural networks with voxelized LiDAR inputs to jointly perform detection and prediction [13, 43, 69]. ILVM encodes the whole scene in a latent random variable and uses a deterministic decoder to efficiently sample multiple scene-consistent trajectories for all the actors in the scene. Besides LiDAR, the ILVM backbone takes as input a multi-channel image with semantic aspects of the rasterized map (*e.g*., one channel encodes walkways, another encodes lanes, and so on, for a total of 13 layers [11]), which the model is expected to use to improve detection and forecasting. While the LiDAR scans are always processed in the vehicle frame, the scans and the map are aligned using the pose of the car. Thus, localization error results in a *misalignment* between the semantic map and the LiDAR scan.

Motion Planning (MP): Given a map and a set of dynamic agents and their future behaviours, the task of the motion planner is to provide a route that is safe, comfortable, and physically realizable to the control module. We study the state-of-the-art Path Lateral Time (PLT) motion planner [53], a method that samples physically realizable trajectories, evaluates them, and selects the one with the minimal cost.

The PLT planner receives $S = 50$ Monte Carlo samples from the joint distribution over the trajectories of *all* actors $\{Y^1, \ldots, Y^S\}$ from the P2 module. It then samples a small set of trajectories $\tau \in \mathcal{T}(\mathcal{M}, \mathcal{R}, \mathbf{x}_0)$ given the map $\mathcal{M}$, high-level routing $\mathcal{R}$ and the current state of the SDV $\mathbf{x}_0$. The planned trajectory $\tau^* = \operatorname{argmin}_{\tau \in \mathcal{T}(\mathcal{M}, \mathcal{R}, \mathbf{x}_0)} \sum_{s=1}^{S} c(\tau, Y^s)$ is then computed to be the one with the minimum expected cost over the predicted futures, as defined by a cost function c that takes into account safety and comfort. In this case, bad

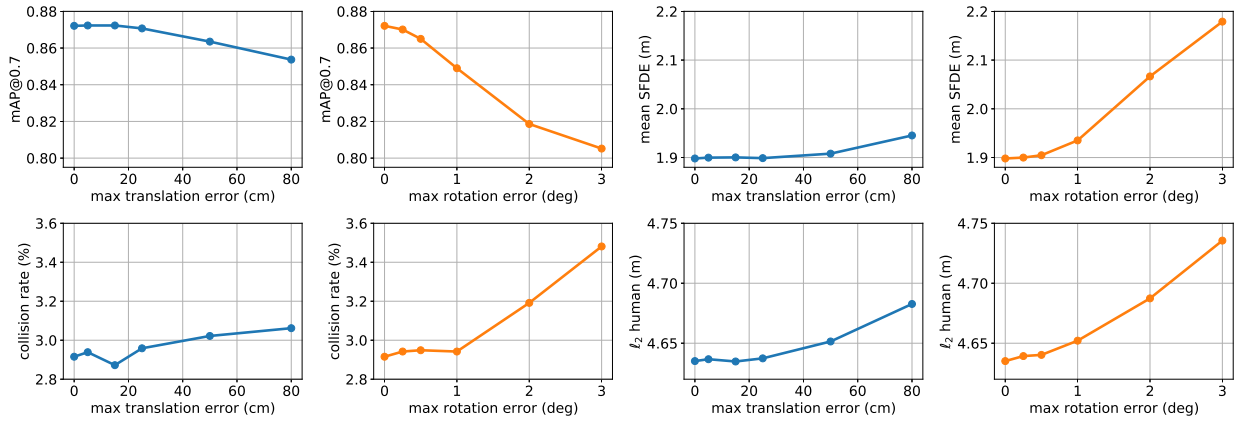

Figure 2: The effects of localization error on perception-prediction and motion planning. (Top) The effects of perturbing the ego-pose on P2. SFDE is the mean displacement error across all samples at the 5s mark as defined in [11], and mAP@0.7 is the mean average precision evaluated at an IOU of 0.7. (Bottom) The effects of perturbing the ego-pose on planning. Collision rate is the percentage of examples for which the planned path collides with another vehicle or pedestrian within the 5s simulation, and ℓ_2 human is the distance between the planned path and the ground truth human-driven path at the 5s mark.

localization gives the planner a wrong idea about the layout of the static parts of the scene.

3.1. Experimental setup

LP3 Dataset: Evaluating the localization, perception, prediction, and motion planning tasks requires a dataset that contains accurately localized self-driving segments, together with the corresponding HD appearance maps (to evaluate localization), as well as annotations of dynamic objects in the scene, their tracks, and their future trajectories (to evaluate P3). To the best of our knowledge, no current public dataset satisfies all these criteria.[1] Therefore, we use our own LP3 dataset. The LP3 dataset is named after the ability it provides to evaluate Localization (L) as well as Perception, Prediction, and motion Planning (P3). LP3 is a subset of the ATG4D dataset [11, 13, 69], that also has appearance maps available. In particular, our maps have 2d images of aggregated LiDAR intensity that summarize the appearance of the ground (*c.f.* the top left of Fig. 3), and are between 6 and 12 months old by the time the SDV traverses the scene. The dataset is comprised of 1858 sequences of 25 seconds each, all captured in a large North American city.

Besides bounding boxes for vehicles, pedestrians and bicycles in the scene, the dataset provides *semantic map* annotations, such as lanes, traffic signs and sidewalks. Importantly, LP3 also provides a map *appearance layer* comprised of the LiDAR intensity of the static elements of the scene as captured by multiple passes of LiDAR scans through the area (please refer to the top left of Figure 3 for an example). Our LP3 dataset makes it possible to evaluate methods that jointly perform appearance-based LiDAR localization and P2, and to quantify motion planning metrics.

Simulating Localization Error: We simulate localization error and study its effects on downstream autonomy tasks. Given a maximum amount of noise $m \in \mathbb{R}$ (which we call *maximum jitter*), we perturb the ground truth pose on evaluation frames by sampling translational or rotational noise from a uniform distribution $\varepsilon \sim \mathcal{U}(-m, m)$. To understand the effects of different types of noise, we evaluate translational noise and rotational noise independently.

Metrics: For perception, we focus on the mean average precision metric with at least 70% overlap between the predicted and the ground truth boxes (mAP@0.7) [13]. For prediction, we report the mean scene final displacement error (mean SFDE[2]) between the ground truth and the predicted trajectory after 5 seconds (*i.e.*, planning horizon) [11].

We run the planner at the beginning of the segments, and let the trajectory unfold for 5 seconds. We then measure the percent of segments for which there is a collision, and the ℓ_2 distance between the predicted trajectory and the trajectory followed by the human driver after 5 seconds.

Note that in our setting all the actors are "passive", in the sense that they follow their pre-recorded trajectory independently of the actions taken by the SDV. This is often called an *open-loop* evaluation. While evaluating our task on a *closed-loop* setting would be more desirable, building a simulator of reactive agents and counterfactual sensor inputs comes with its own set of challenges (*e.g.*, realistic LiDAR

[1] A few days after the submission deadline, the nuScenes dataset [9] added support for an appearance layer, thereby enabling similar experiments to the ones we present in this work.

[2] We use meanSFDE instead of minSFDE because for large numbers of samples ($S = 50$ in our case), minSFDE is overly optimistic. The presence of unrealistic false positive trajectory samples can interfere with the SDV, causing it to break or swerve, creating a dangerous situation. However, the minSFDE will not capture this dangerous behavior as long as there is at least one good sample. Please refer to [12] for a more detailed discussion.

simulation, realistic controllable actors, *etc*.) and is out of the scope of our work.

Results: We show the effects of perturbing the pose of the ego-vehicle on P2 in the top row of Figure 2. We observe that the performance of ILVM is barely affected by translational jitter up to 25 cm, and rotational jitter up to 0.5°. Larger amounts of translational noise have little effect (∼2% mAP, .05 mean SFDE) up to 80 cm, while the effect is stronger for rotational error (∼7% mAP, .30 mean SFDE) up to 3°.

We show the effects of perturbing the ego-vehicle pose on motion planning in the bottom row of Figure 2. Similar to P2, MP performance does not degrade much until there is translational noise above 25 cm (or rotational noise above 0.5°). We also observe that large translation errors have small effects relative to rotational noise for both collision rate and distance to human route. While this is somewhat expected (as rotational error can cause straight paths to run into sidewalks or incoming traffic), it is interesting to formally quantify these effects.

4. Joint Localization, Perception, & Prediction

We now formulate a model that performs joint localization and P2 (LP2). First, we lay out the key challenges that we would like our system to overcome, and then explain our design choices in detail.

4.1. System Desiderata

Low Latency: In order to provide a safe ride, a self-driving car must react quickly to changes in its environment. In practice, this means that we must minimize the time from perception to action. In a naïve, cascaded autonomy system, the running time of each component adds up linearly, which may result in unacceptable latency.

To reduce the latency of the localization system, it is common to use Bayesian filtering to provide high-frequency pose updates. In this case, a belief about the pose is maintained over time and updated through the continuous integration of different levels of evidence from wheel autoencoders, IMUs, or camera and LiDAR sensing. In this context, the external sensing step (*e.g*., carried out via iterative closest point alignment between the LiDAR reading and an HD map) typically carries the strongest evidence, but is also the most expensive part of the system. Therefore, it is critical to keep the latency of the sensing step of the localization filter low.

Learning-Based Localization: Localization systems with learned components are typically better at discerning semantic aspects of the scene that are traditionally hard to discriminate with purely geometric features (*e.g*., growing vegetation, tree stumps, and dynamic objects), and have the potential of being more invariant to appearance changes due to season, weather, and illumination [45]. Therefore, we would like to incorporate a learning-based localization component in our system. Moreover, since P2 systems are typically heavily driven by learning, it should be possible to incorporate learning-based localization by sharing computation between the two modules, resulting in reduced overhead to the overall LP2 system.

Simple Training and Deployment: We would like our joint LP2 system to be easier to train and deploy than its classical counterpart. Given the large amounts of ML infrastructure invested around P2 systems (*e.g*., on dataset curation, labelling, active learning, and monitoring), it makes sense to design a localization subsystem that can be trained as a smaller addition to a larger P2 model [56]. This should also make it easier to iterate on the more lightweight localization module without the need to retrain the more computationally-expensive P2 component.

4.2. Designing an LP2 System

We now explain our model design choices, highlighting the ways they overcome the aforementioned challenges and achieve our design goals. We show an overview of the proposed architecture in Figure 3.

Input Representation: Our system receives LiDAR as input, which is then converted to a bird's-eye view (BEV) voxelization with the channels of the 2D input corresponding to the height dimension [66]. Despite P2 and localization models both relying on some form of voxelized LiDAR input, perception-prediction models often use a coarser LiDAR resolution (*e.g*., 20cm [43, 69]) to accomodate larger regions, while matching-based localizers typically require a finer-grained resolution to localize with higher precision [8].

Using only a fine resolution voxelization for an LP2 model would be simplest, but imposes large run-time efficiency costs. Therefore, to accommodate these resolution differences, our method simultaneously rasterizes the incoming LiDAR point cloud $\mathbf{x}$ into two tensors of different resolutions, $\tilde{\mathbf{x}}_{\text{coarse}}$ for perception and $\tilde{\mathbf{x}}_{\text{fine}}$ for localization.

Perception and Prediction: For our P2 subsystem, we rely on ILVM [11], whose robustness to localization error we quantified in Section 3 – this corresponds to the lower part of Figure 3. The proposed P2 approach contains four main submodules. (i) A lightweight network processes a rasterized semantic map centred at the current vehicle pose. We pass our estimated pose to this module. (ii) Another neural backbone h extracts features from a coarsely voxelized LiDAR sweep $\tilde{\mathbf{x}}_{\text{coarse}}$. These two features maps are concatenated and passed to (iii) a detector-predictor that encodes the scene into a latent variable Z, and (iv) a graph neural network where each node represents a detected actor, and which deterministically decodes samples from Z into samples of the joint posterior distribution over all actor trajectories.

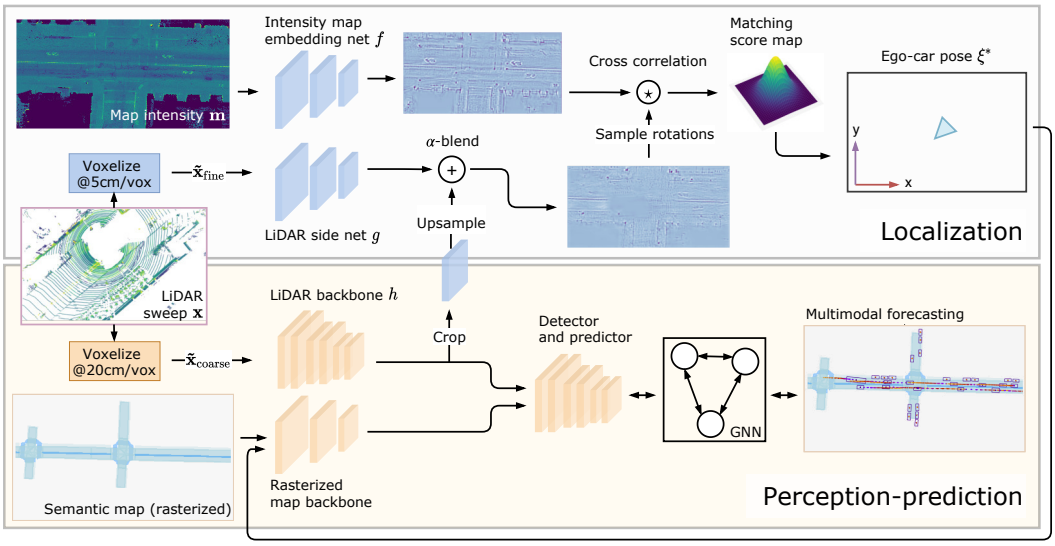

Figure 3: The architecture of the combined localization and perception-prediction (LP2) model.

Localization: We approach localization using ground intensity localization with deep LiDAR embeddings [8]. The idea behind ground intensity localization [38] is to align the (sparse) observed LiDAR sweep $\mathbf{x}$ with a pre-built (dense) map of the LiDAR intensity patterns of the static scene, $\mathbf{m}$. This localizer learns deep functions that produce spatial embeddings of both the map $f(\mathbf{m})$ and LiDAR sweep $g(\tilde{\mathbf{x}}_{\text{fine}})$ before alignment. Following existing work [8] we parameterize the vehicle pose using three degrees of freedom (DoF), x, y, and yaw, represented as $\xi \in \mathbb{R}^3$.

Given a small set of pre-defined translational and rotational offsets, we compute the dot product between the transformed sweep and the map embeddings, and choose the pose candidate ξ^* from a pre-defined set (near the original pose estimate) with the highest correlation as the maximum-likelihood estimate of the vehicle pose:

$$\xi^* = \underset{\xi}{\operatorname{argmax}}\, \pi(g(\tilde{\mathbf{x}}_{\text{fine}}), \xi) \cdot f(\mathbf{m}) \triangleq \underset{\xi}{\operatorname{argmax}}\, \mathbf{p}(\xi) \quad (1)$$

where π is a function that warps its first argument based on the 3-DoF offset ξ, and $\cdot$ represents the dot product operator. In practice, this matching is done more efficiently by observing that the dot products can be computed in parallel with respect to the translational portion of the pose candidates by using a larger-region $\mathbf{m}$ and performing a single cross-correlation rather than multiple dot products for each DoF in the rotation dimension.

Multi-Resolution Feature Sharing: An important advantage of localizing using LiDAR matching is that in contrast to, *e.g.*, point cloud-based localizers [22], it uses the same BEV input representation as P2, enabling a substantial amount of computation to be shared between both systems. However, as discussed earlier, the inputs to the P2 and localization backbones use different resolutions, which can make information fusion difficult. We address this issue by upsampling a crop of the LiDAR feature map computed by the coarse perception backbone to match the resolution of the finer features in the localization backbone. We then add the feature maps together using a weighted sum to produce the final localization embedding, as depicted in Fig. 3. This allows localization LiDAR embeddings to be computed with very little run-time or memory overhead compared to the base perception-prediction network.

4.3. Learning

We optimize the full model using side-tuning [71]. We first train the heavier perception-prediction module, and then add the LiDAR branch of the localizer as a side-tuned module. In the second stage, we freeze the weights of the perception-prediction network (yellow modules in Fig. 3), and only learn the map and online branches of the localizer (purple modules in Fig 3). There are three benefits to this approach: first, there is no risk of catastrophic forgetting in the perception-prediction task, which can be problematic as it typically requires 3–5$\times$ more computation to train than the localizer alone; second, we do not need to balance the loss terms of the localization vs. perception-prediction, eliminating the need for an additional hyperparameter; third, training the localizer network can be done much faster than the full system, since the P2 header no longer needs to be evaluated and fewer gradients needs to be stored.

Perception and Prediction: We train the P2 component using supervised learning by minimizing a loss which combines object detection with motion forecasting, while accounting for the multimodal nature of the trajectory predictions. The P2 loss is therefore structured as

$$\mathcal{L}_{\text{P2}} = \mathcal{L}_{\text{DET}} + \alpha \mathcal{L}_{\text{PRED}}, \quad (2)$$

where $\mathcal{L}_{\text{DET}}$ optimizes a binary cross entropy term for the object detections and one based on smooth-ℓ_1 for the box regression parameters [66], $\mathcal{L}_{\text{PRED}}$ optimizes the ELBO of the log-likelihood of the inferred n trajectories over t time steps, conditioned on the input [11], and α represents a scalar weighting term selected empirically.

Localization: Learning f and g end-to-end produces representations that are invariant to LiDAR intensity calibration, and ignore aspects of the scene irrelevant to localization. Learning is performed by treating localization as a classification task and minimizing the cross-entropy between the discrete distribution of $\mathbf{p}(\xi)$ and the ground truth pose offset $\mathbf{p}^{\text{GT}}$ expressed using one-hot encoding [8]:

$$\mathcal{L}_{\text{LOC}} = -\sum_{\xi} \mathbf{p}(\xi)^{\text{GT}} \log \mathbf{p}(\xi). \tag{3}$$

The online and map embedding networks f and g use an architecture based on the P2 map raster backbone and do not share weights. In Section 5 we also show that it is possibly to significantly improve run time by down-sizing g while keeping f fixed with little impact on overall performance. We refer to this architecture as a Pixor backbone [66].

5. Experiments

We design our experiments to test the accuracy of our multi-task model on the joint localization and P2 (LP2) task. We also show how these improvements translate to safe and comfortable rides based on motion planning metrics. We refer to the task of doing localization, P2, and motion planning as LP3.

Dataset and Metrics: We use the LP3 dataset (*c.f.* Sec. 3), for all our experiments. To evaluate localization accuracy, following prior work [64], we report the percentage of frames on which the localizer matches the ground truth exactly, and where it matches the ground truth or a neighbouring offset as recall @ 1 (r@1) and recall @ 2 (r@2) respectively. In our setting, the former metric corresponds to exactly matching the ground truth, up to our state space resolution (5cm and 0.5°), while the latter corresponds to being inside a 15cm×15cm×1.5° region centered at the ground truth. For P2, we focus on mAP@0.7 for detection and mean SFDE for prediction, as in Sec. 3. For motion planning, besides collision rate and ℓ_2 distance to human trajectory (see Sec. 3), we also measure lateral acceleration, jerk, and progress towards the planning goal.

Experimental Setup: There are multiple ways to design an experiment that tests a localizer. One alternative is to start the state estimation at the identity and later align the produced trajectory with the ground truth (as is often done in SLAM [61]). Alternatively, online localization often initializes the robot pose at the ground truth location, and measures how far a localizer can travel before obtaining an incorrect pose [8, 44, 76]. By definition, these setups assume that the initial pose is correct, and do not test the ability to recover from localization failure – which is crucial for self-driving. Instead, we assume a self-driving scenario where the localization of the pose is initially incorrect. As such, we perturb the true pose of the vehicle and thus measure the ability of the localizer to recover from this failure, as well as the ability of P2 and MP to deal with localization failure. The perturbations are performed following the same uniform noise policy described in Section 3.1 with 0.5 metres for translation and 1.5 degrees for rotation.

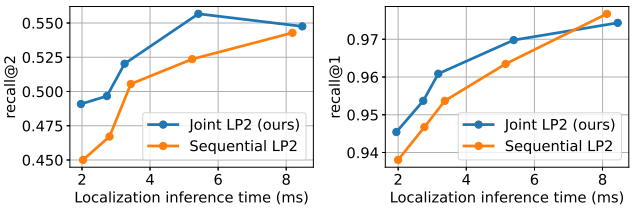

Figure 4: Localizer embedding runtime vs. recall. The localization performance and runtime of the single-task (*i.e.*, sequential) and multi-task (*i.e.*, joint) localization methods. Faster inference is achieved by narrower and shallower networks for the online LiDAR embedding.

Implementation Details: We train our model for 5 epochs using the Adam [37] optimizer using 16 GPUs. The coarse LiDAR tensor $\tilde{\mathbf{x}}_{\text{coarse}}$ is rasterized at 20cm/voxel, while the fine tensor $\tilde{\mathbf{x}}_{\text{fine}}$ uses 5cm/voxel. The spatial region corresponding to the coarse LiDAR voxelization is 144m×80m×3.2m, while the spatial region corresponding to the fine LiDAR voxelization is 48.05m×24.05m×3.2m. Reducing the spatial extent of the high-resolution rasterization reduces the run time of the system without sacrificing performance. The localization search range covers ±0.5m relative to the initial vehicle pose estimate in the x and y dimension, discretized at 5cm intervals, and [-1.5°, -1.0°, -0.5°, 0.0°, 0.5°, 1.0°, 1.5°] in the yaw dimension. We do not use any height information from the maps, which are encoded as BEV images. When training the localizer, we add uniform noise to the ground truth pose, up to the size of the search range.

Results: In Table 1, we evaluate localization and P2 performance through motion planning metrics. Notably, despite significant variation in localization metrics, both of our localization models perform similarly well when evaluated in terms of the motion planning metrics. These results further confirm the observations from our jitter experiments (Figure 2): both P2 and a short-term rollout of PLT perform similarly well when subject to a modest amount of localization error. This means that besides localization accuracy (which is important from an interpretability perspective), we

Model	P2 pose (GT, N, L)	Planning Pose (GT, N, L)	r@1 ↑ (%)	r@2 ↑ (%)	Collision ↓ (% up to 5s)	ℓ_2 human ↓ (m @ 5s)	Lat. acc. ↓ (m/s^2)	Jerk ↓ (m/s^3)	Progress ↑ (m @ 5s)
ILVM	GT	GT	-	-	**2.915**	**4.64**	**2.13**	**1.82**	**24.95**
ILVM	GT	N	-	-	3.168	4.68	2.21	1.83	**24.95**
ILVM	N	N	-	-	3.511	4.70	2.20	1.83	24.92
Joint LP2 – Ours (Tiny Pixor)	N	N	46.6	93.5	2.962	**4.64**	**2.13**	**1.82**	**24.96**
Joint LP2 – Ours (Big Pixor)	N	N	**52.5**	**96.9**	2.922	**4.64**	**2.13**	**1.82**	**24.95**

Table 1: Motion planning evaluation using pose estimate and actor predictions. For the P2 and Planning poses: GT denotes ground truth (the pose was not altered); N denotes that localization noise was added (translation and rotation sampled uniformly at random from [-0.5m,+0.5m] and [-1.5°,+1.5°], respectively). Big Pixor refers to the largest width Pixor Embedding Net from Fig 4, and Tiny Pixor refers to the smallest. Bold denotes the best results (within an epsilon threshold) and underlines second best results.

Model	Time (ms)	r@1	r@2
LiDAR Localizer [8]	25.92	0.52	0.95
LiDAR Localizer (Pixor-based)	2.79	0.47	0.95
Joint LP2 (Ours)	1.95	0.49	0.95

Table 2: Localization inference time comparison. While being nearly identical in terms of matching accuracy when comparing models with recall @ 2 performance similar to [8], the proposed approach is much faster, due to a more efficient architecture and sharing computation with the perception backbone.

have plenty of room to optimize for latency and simplicity when designing the localization component of an LP2 architecture. Our tiny Pixor-based localizer only takes 2ms of overhead on top of the P2 subsystem, while providing a robust learned localization signal to the autonomy system.

Ablation Study: We perform an ablation study to investigate the trade-off between matching performance and inference time in the localization part of our system. We show our results in Fig. 4.

We compare the effectiveness of our localization network trained to re-use P2 features (*i.e.*, the joint LP2 network), and a network trained to do localization from scratch (*i.e.*, as used in a sequential setting). In both cases, faster inference is achieved by shallower (fewer layers) and narrower networks (fewer channels) used for the online LiDAR embedding. The reported inference time does not include the map embedding branch, which can be pre-computed offline. The largest model corresponds to an architecture similar to the P2 rasterized HD map backbone (which is itself a smaller version of the P2 LiDAR backbone), while the faster and smaller models have fewer layers or fewer channels in each layer. The four largest models have 11 convolutional layers and a factor of $C = 1/2^0, 1/2^1, 1/2^3, 1/2^4$ the number of channels as the largest model. The smallest (fifth largest) has $C = 1/2^4$ and five layers rather than 11, which corresponds to one layer around each of the three pooling/upsampling stages followed by a final layer. We observe that, while reducing model size leads to a small drop in matching accuracy, this does not end up affecting motion planning, as shown in Table 1, while at the same time reducing the online embedding computation time four-fold.

Finally, Table 2 compares our proposed online LiDAR embedding networks to the state of the art. The U-Net-based approach from [8] was shown to outperform classic approaches like ICP-based localization, especially in challenging environments such as highways. Our results show that the original performance can already be matched with a much faster network architecture, while leveraging the perception feature maps allows even smaller models to perform at the same level. All inference times are measured on an NVIDIA RTX5000 GPU.

6. Conclusion

While prior research in autonomous driving has explored either full end-to-end learning or the joint study of tasks such as object detection and motion forecasting, the task of localization has not received as much attention in the context of perception and planning systems, in spite of the strong reliance of self-driving vehicles on HD maps for these tasks.

In this paper, we studied how localization errors affect state-of-the-art perception, prediction, and motion-planning systems. Our analysis showed that while perception is robust to relatively small localization errors, motion planning performance suffers more, especially in case of yaw errors, motivating the need to detect and correct such issues. We subsequently proposed a multi-task learning solution capable of jointly localizing against an HD map while also performing object detection and motion forecasting, and showed that localization errors can be successfully detected and corrected in less than 2 ms of GPU time.

Our work suggests multiple areas for improvement that may be addressed in future work, such as end-to-end system evaluation using a closed-loop simulator, more detailed comparisons to classic localizers (e.g., ICP-based [68]), the integration of a recursive Bayesian filter in the localizer, and a finer-grained evaluation of motion planning errors.

References

[1] Ferran Alet, Tomás Lozano-Pérez, and Leslie P Kaelbling. Modular meta-learning. In *CoRL*, 2018. 3

[2] Antonio Artuñedo, Jorge Villagra, Jorge Godoy, and Maria Dolores Del Castillo. Motion planning approach considering localization uncertainty. *IEEE Transactions on Vehicular Technology*, 69(6):5983–5994, 2020. 3

[3] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst. In *RSS*, 2019. 2, 3

[4] Dan Barnes, Rob Weston, and Ingmar Posner. Masking by moving: Learning distraction-free radar odometry from pose information. In *CoRL*, 2019. 2

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 2

[6] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006. 3

[7] Mihai Bujanca, Paul Gafton, Sajad Saeedi, Andy Nisbet, Bruno Bodin, Michael FP O'Boyle, Andrew J Davison, Paul HJ Kelly, Graham Riley, Barry Lennox, et al. SLAMBench 3.0: Systematic automated reproducible evaluation of SLAM systems for robot vision challenges and scene understanding. In *ICRA*, 2019. 3

[8] Ioan Andrei Bârsan, Shenlong Wang, Andrei Pokrovsky, and Raquel Urtasun. Learning to localize using a LiDAR intensity map. In *CoRL*, 2018. 2, 5, 6, 7, 8

[9] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 4

[10] Bingyi Cao, Andre Araujo, and Jack Sim. Unifying deep local and global features for efficient image search. In *ECCV*, 2020. 3

[11] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *ECCV*, 2020. 1, 3, 4, 5, 7

[12] Sergio Casas, Cole Gulino, Simon Suo, and Raquel Urtasun. The importance of prior knowledge in precise multimodal prediction. In *IROS*, 2020. 4

[13] Sergio Casas, Wenjie Luo, and Raquel Urtasun. IntentNet: Learning to predict intention from raw sensor data. In *CoRL*, 2018. 1, 2, 3, 4

[14] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. MultiPath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *CoRL*, 2020. 2

[15] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, 2018. 3

[16] Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D Manning, and Quoc V Le. BAM! Born-again multi-task networks for natural language understanding. In *ACL*, 2019. 3

[17] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008. 3

[18] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020. 3

[19] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *ICRA*, 2019. 2

[20] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016. 3

[21] Andrew J Davison. FutureMapping: The computational structure of spatial AI systems. *arXiv preprint arXiv:1803.11288*, 2018. 3

[22] Juan Du, Rui Wang, and Daniel Cremers. DH3D: Deep hierarchical 3D descriptors for robust large-scale 6DoF relocalization. In *ECCV*, 2020. 2, 3, 6

[23] Davi Frossard, Simon Suo, Sergio Casas, James Tu, Rui Hu, and Raquel Urtasun. StrObe: Streaming object detection from LiDAR packets. In *CoRL*, 2020. 3

[24] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. VectorNet: Encoding HD maps and agent dynamics from vectorized representation. In *CVPR*, 2020. 2

[25] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012. 3

[26] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *ECCV*, 2018. 3

[27] Wei Han, Zhengdong Zhang, Benjamin Caine, Brandon Yang, Christoph Sprunk, Ouais Alsharif, Jiquan Ngiam, Vijay Vasudevan, Jonathon Shlens, and Zhifeng Chen. Streaming object detection for 3-D point clouds. In *ECCV*, 2020. 3

[28] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. In *EMNLP*, 2017. 3

[29] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 3

[30] Vadim Indelman, Luca Carlone, and Frank Dellaert. Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments. *IJRR*, 34(7):849–882, 2015. 3

[31] Herve Jegou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *PAMI*, 34(9):1704–1716, 2011. 2

[32] Niels Joubert, Tyler GR Reid, and Fergus Noble. Developments in modern GNSS and its impact on autonomous vehicle architectures. In *IEEE Intelligent Vehicles Symposium (IV)*, 2020. 2

[33] Łukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017. 3

[34] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018. 3

[35] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DoF camera relocalization. In *ICCV*, 2015. 2

[36] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *ICRA*, 2019. 2

[37] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 7

[38] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. Map-based precision vehicle localization in urban environments. In *RSS*, 2007. 2, 6

[39] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *ECCV*, 2020. 3

[40] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3D object detection. In *CVPR*, 2019. 2

[41] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. PnPNet: End-to-end perception and prediction with tracking in the loop. In *CVPR*, 2020. 1, 2

[42] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. DeepVCP: An end-to-end deep neural network for point cloud registration. In *ICCV*, pages 12–21, 2019. 2

[43] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net. In *CVPR*, 2018. 1, 2, 3, 5

[44] Wei-Chiu Ma, Ignacio Tartavull, Ioan Andrei Bârsan, Shenlong Wang, Min Bai, Gellert Mattyus, Namdar Homayounfar, Shrinidhi Kowshika Lakshmikanth, Andrei Pokrovsky, and Raquel Urtasun. Exploiting sparse semantic HD maps for self-driving vehicle localization. In *IROS*, 2019. 2, 7

[45] Julieta Martinez, Sasha Doubov, Jack Fan, Ioan Andrei Bârsan, Shenlong Wang, Gellért Máttyus, and Raquel Urtasun. Pit30M: A benchmark for global localization in the age of self-driving cars. In *IROS*, 2020. 5

[46] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016. 3

[47] Balázs Nagy and Csaba Benedek. Real-time point cloud alignment for vehicle localization in a high resolution 3D map. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018. 2

[48] Luigi Nardi, Bruno Bodin, M Zeeshan Zia, John Mawer, Andy Nisbet, Paul HJ Kelly, Andrew J Davison, Mikel Luján, Michael FP O'Boyle, Graham Riley, et al. Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM. In *ICRA*, 2015. 3

[49] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. CoverNet: Multimodal behavior prediction using trajectory sets. In *CVPR*, 2020. 2

[50] Jonah Philion, Amlan Kar, and Sanja Fidler. Learning to evaluate perception models using planner-centric metrics. In *CVPR*, 2020. 3

[51] Noha Radwan, Abhinav Valada, and Wolfram Burgard. VLocNet++: Deep multitask learning for semantic visual localization and odometry. *RA-L*, 3(4):4407–4414, 2018. 1

[52] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *ECCV*, 2020. 1, 2

[53] Abbas Sadat, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. In *IROS*, 2019. 3

[54] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, pages 12716–12725, 2019. 2, 3

[55] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 2

[56] D Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, and Dan Dennison. Hidden technical debt in machine learning systems. In *NIPS*, 2015. 5

[57] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *NIPS*, 2018. 3

[58] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *CVPR*, 2019. 2

[59] Tim Yuqing Tang, Daniele De Martini, Dan Barnes, and Paul Newman. RSL-Net: Localising in satellite images from a radar on the ground. *RA-L*, 5(2):1087–1094, 2020. 2

[60] Marvin Teichmann, Michael Weber, Marius Zoellner, Roberto Cipolla, and Raquel Urtasun. MultiNet: Real-time joint semantic reasoning for autonomous driving. In *IEEE Intelligent Vehicles Symposium (IV)*, 2018. 1, 3

[61] Sebastian Thrun. Simultaneous localization and mapping. In *Robotics and cognitive approaches to spatial mapping*, pages 13–41. Springer, 2007. 7

[62] Jur van den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *IJRR*, 31(11):1263–1278, 2012. 3

[63] Guowei Wan, Xiaolong Yang, Renlan Cai, Hao Li, Yao Zhou, Hao Wang, and Shiyu Song. Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes. In *ICRA*, 2018. 2

[64] Xinkai Wei, Ioan Andrei Bârsan, Shenlong Wang, Julieta Martinez, and Raquel Urtasun. Learning to localize through compressed binary maps. In *CVPR*, 2019. 7

[65] Ryan W Wolcott and Ryan M Eustice. Visual localization within LIDAR maps for automated urban driving. In *IROS*, 2014. 2

[66] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: Real-time 3D object detection from point clouds. In *CVPR*, 2018. 2, 5, 7

[67] Yongxin Yang and Timothy M Hospedales. Trace norm regularised deep multi-task learning. In *ICLR Workshop Track*, 2017. 3

[68] Keisuke Yoneda, Hossein Tehrani, Takashi Ogawa, Naohisa Hukuyama, and Seiichi Mita. Lidar scan feature for localization with highly precise 3-D map. In *IEEE Intelligent Vehicles Symposium (IV)*, 2014. 2, 8

[69] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *CVPR*, 2019. 1, 2, 3, 4, 5

[70] Wenyuan Zeng, Shenlong Wang, Renjie Liao, Yun Chen, Bin Yang, and Raquel Urtasun. DSDNet: Deep structured self-driving network. In *ECCV*, 2020. 1, 2, 3

[71] Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-tuning: Network adaptation via additive side networks. In *ECCV*, 2020. 3, 6

[72] Zhishuai Zhang, Jiyang Gao, Junhua Mao, Yukai Liu, Dragomir Anguelov, and Congcong Li. STINet: Spatio-temporal-interactive network for pedestrian detection and trajectory prediction. In *CVPR*, 2020. 2

[73] Zichao Zhang and Davide Scaramuzza. Fisher information field: an efficient and differentiable map for perception-aware planning. *arXiv preprint arXiv:2008.03324*, 2020. 3

[74] Stephan Zheng, Yisong Yue, and Jennifer Hobbs. Generating long-term trajectories using deep hierarchical networks. In *NIPS*, 2016. 2

[75] Brady Zhou, Philipp Krähenbühl, and Vladlen Koltun. Does computer vision matter for action? *Science Robotics*, 4(30), 2019. 1, 2

[76] Yao Zhou, Guowei Wan, Shenhua Hou, Li Yu, Gang Wang, Xiaofei Rui, and Shiyu Song. DA4AD: End-to-end deep attention aware features aided visual localization for autonomous driving. In *ECCV*, 2020. 2, 7