MapLite 2.0: Online HD Map Inference Using a Prior SD Map

Teddy Ort[1], Jeffrey M. Walls[2], Steven A. Parkison[2], Igor Gilitschenski[2,3], and Daniela Rus[1]

***Abstract*—Deploying fully autonomous vehicles has been a subject of intense research in both industry and academia. However, the majority of these efforts have relied heavily on High Definition (HD) prior maps. These are necessary to provide the planning and control modules a rich model of the operating environment. While this approach has shown success, it drastically limits both the scale and scope of these deployments as creating and maintaining HD maps for very large areas can be prohibitive. In this work, we present a new method for building the HD map online by starting with a Standard Definition (SD) prior map such as a navigational road map, and incorporating onboard sensors to infer the local HD map. We evaluate our method extensively on 100 sequences of real-world vehicle data and demonstrate that it can infer a highly structured HD map-like model of the world accurately using only SD prior maps and onboard sensors.**

***Index Terms*—Mapping, Autonomous Vehicle Navigation, HD Maps, High Definition Maps, Mapless Driving, Localization**

I. Introduction

MANY automated vehicle systems rely on information encoded in a High Definition (HD) map that is transformed into the vehicle reference frame at run time to aid decision making. HD maps typically represent road topology, geometry, and attributes such as speed limits. Such systems require precise knowledge of the vehicle position with respect to the map. HD map creation is nontrivial and often includes immense resources to supervise the curation and annotation process. Moreover, map data becomes outdated as the structure of the environment evolves requiring costly maintenance procedures. At the same time, reliance on HD maps presents a safety risk during the period between the structural change and the next map update cycle.

Addressing this challenge requires a conceptually different approach to the AV perception stack. Human drivers do not rely on highly detailed maps and directly combine real-time scene understanding with knowledge provided by classical, Standard Definition (SD), street maps. Mimicking this ability requires novel perception techniques that combine raw sensor data with coarse prior SD maps to obtain a representation similar to current HD maps.

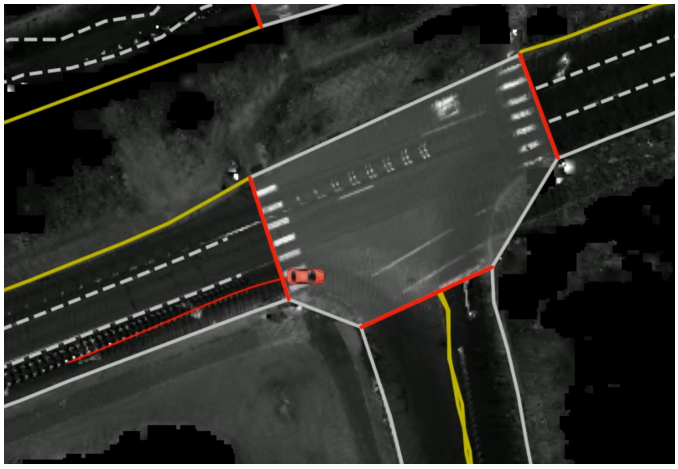

Fig. 1. An online HD map overlaying lidar intensity as a vehicle traverses an intersection. The white and yellow lines indicate inferred road and lane boundaries while the red lines denote intersection entry/exit ports. The background shows a projection of the lidar intensity data into the birds-eye-view (BEV) that was used to help generate the HD map online.

Existing online perception techniques focus on estimation of certain map attributes such as, lane boundaries or road topology. These often perform classification or segmentation in a dense sensor representation such as pixels or pointclouds. Those that do infer vectorized features, typically do not provide a sufficiently rich representation comparable to modern HD map data. Generally, downstream planning and control modules require the HD map to query important details such as the shape of the road ahead, the location, direction, and connectivity of adjacent road lanes, and the feasible traversals through an intersection. Feature detectors that merely detect the features in the scene, without inferring the underlying structure cannot satisfy this requirement.

We aim to enable the much richer HD map-like representation by leveraging an SD prior map. While the SD map provides valuable topological information, integrating it is particularly challenging because these maps were designed for humans and the annotation cannot be assumed to be sufficiently precise. Thus, enabling SD map based driving using an online computed HD map-like representation is a challenging and largely open problem.

In this paper, we present a system to replace or augment meticulously generated HD maps. Our method infers road geometry and topology, e.g., lane and intersection sizes, given a publicly available coarse road graph (in our case OpenStreetMap [1]) and online perception data. We convert the

Manuscript received: February, 25, 2022; Revised May, 11, 2022; Accepted June, 06, 2022.

This paper was recommended for publication by Editor Javier Civera upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Toyota Research Institute (TRI). However, the views contained herein are solely those of the authors.

[1]T. Ort and D. Rus are with the Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology

[2]J.M. Walls, S.A. Parkison, and I. Gilitschenski are with the Toyota Research Institute (TRI)

[3]I. Gilitschenski is with the Department of Computer Science, University of Toronto

Correspondence to `teddy@mit.edu`

Digital Object Identifier (DOI): see top of this page.

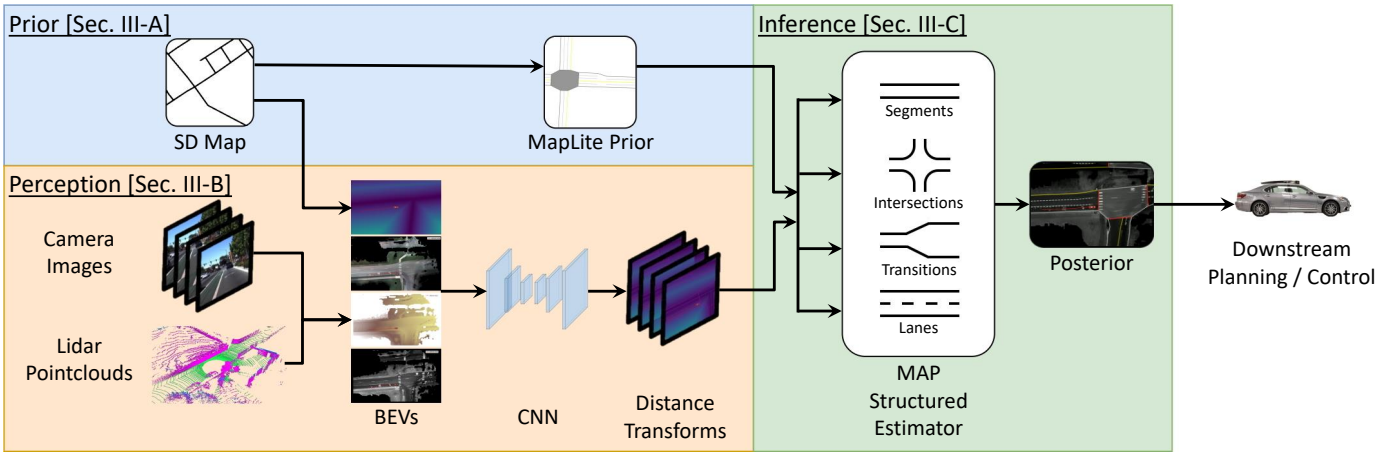

Fig. 2. In the proposed system, an SD prior map is fused with onboard sensor data to perform an online estimation of the HD map near the vehicle.

coarse road graph into an HD map prior and a rasterized birds-eye-view (BEV) representation. Additional BEV images are generated from a combination of sensor inputs. These are used as inputs to a Convolutional Neural Network (CNN) trained to predict their semantic labeling using existing HD map data for training. The semantic segmentation is converted into a set of label-specific distance transforms. Finally, a structured estimator maintains the local map estimate and integrates the SD map prior. In summary, the contributions of this work include:

- A new method for online HD map inference from raw sensor data and a prior SD map.
- A fully relative HD map representation that is designed specifically to enable both the incorporation of an SD prior map, as well as online perception-based updates.
- Evaluation of our approach on real-world scenes with diverse road layouts and appearance.

II. Related Work

A. Offline map generation

Automatically generating HD maps in full or in part as an offline inference task is an attractive problem for its ability to reduce manual supervision. [2] introduced a method for estimating road and lane boundaries given a coarse road graph. They formulated the inference task as an energy minimization problem where potentials encoded aerial image appearance and prior knowledge, e.g., lane smoothness. [3] designed a system to estimate crosswalk geometry given accumulated LIDAR intensity BEV rasters and approximate intersection location determined from a coarse SD map. The authors trained a network to provide input to a structured estimator that determines a consistent configuration of crosswalk parameters. [4] proposed a method for inferring lane boundaries for highway scenes given accumulated LIDAR intensity BEV rasters. They structured their estimate as a directed acyclic graphical model where each node in the graph encoded geometric and topological information, and the nodes were iteratively explored to expand the graph. [5] developed a lane graph estimation engine using vision and LIDAR data aggregated within a BEV representation. Their method is capable of handling arbitrary intersection topologies, but does not incorporate prior SD map information.

B. Online map inference

An alternative approach (and the one we take here) to dealing with the challenges of annotating HD maps, is to infer lane geometry and topology *online*. [6] trained a neural network to segment lanes in stereo images to produce BEV lane geometry and topology. Their method focused on mid-roadway estimation and was not extended into intersections or other complex topologies. [7] proposed an end-to-end system for lane line estimation where the architecture includes image-based instance segmentation followed by trained perspective transformation and a least-squares fit of a parameterized curve. They suggest that fitting parameterized lines in a pseudo-ground plane (akin to a BEV frame) results in a more simple parameterization compared to an image frame. PiNet [8] can produce quick and accurate lane boundary estimates by clustering image frame keypoints into instances. HDMapNet [9] estimates road boundaries, lane lines, and other semantic features in combined LIDAR and camera data to produce a BEV semantic map around the vehicle. Note that the above methods typically produce class instances, e.g., lane lines, but do not provide the topological information that many AV stacks expect.

Similar to the work we present here, MapLite [10] uses the topological information from an SD map to navigate. It was capable of piloting a vehicle with no prior HD map. But Maplite focused on simple road geometries in rural scenes – it did not create an HD map representation to enable use on multi-lane roads or in urban areas.

III. Method

A. State Description

1) SD Map Model: In this work, we assume the availability of SD prior maps. Unlike HD Maps, SD maps are widely available and have minimal storage and transmission requirements. For comparison, the HD map of a small city with 20,000 miles of road could require hundreds of gigabytes [11]. In contrast, an SD map of the same region could be stored in just 3.5

gigabytes. These savings come at a cost however. The SD map contains much less information than the HD equivalent.

We consider an SD map with a graph-like data structure $M_{SD} = \{N, E\}$ where the nodes, N, are intersections and the edges, E, are polylines representing road centerlines. Each edge also has associated semantic attributes such as the road name, speed limit, and number of lanes in each direction. Note that such maps are commonly used for navigation and OpenStreetMap.org [1] makes them freely available for much of the globe. In this work, we do not require a high level of accuracy in the road centerline geometry. For example, it is common for OpenStreetMap centerline features to have errors on the order of $10m$ and the current system is designed to handle that. However, we do require the topology (e.g. road connectivity) of the SD map to be correct.

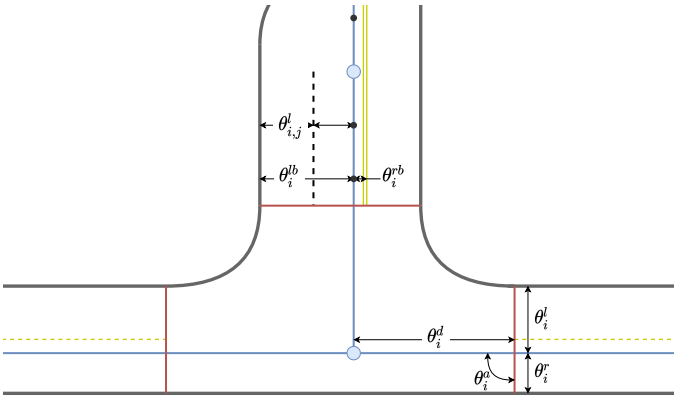

Fig. 3. The HD map model state consists of a set of parameters which define the geometries relative to an SD prior basemap (shown in blue). The red lines indicate intersection entry/exit ports and the black circles indicate knot points. In this illustrative example, there would be many more parameters required to define the full shape of the map elements shown. However for figure clarity, we only include one of each type.

2) HD Map Model: Although HD maps have been widely adopted for autonomous driving, there is currently no universal format for HD maps [12]. We take inspiration from several widely used standards in designing our HD map model. Similar to the Navigation Data Standard (NDS) [13] we utilize separate layers for routing and lane level geometry which we generate online. As in the OpenDrive [14] model we parameterize lane geometry with respect to road centerlines. Since we are estimating the model online, and do not assume the SD map centerlines to be highly accurate, we parameterize even the lane boundaries that coincide with the centerlines in the SD map which allows the online map to not only encode lane geometries that aren't in the SD map, but also shift the position of a road from the centerline in the SD map. Our primary objective in designing the HD map model is to provide a data structure with the geometric information needed for downstream planning and control while allowing it to be estimated online from the SD prior and sensor data.

The HD map model consists of a hierarchy with three primary layers.

1) The *road* layer consists of roads and intersections. Each road may contain multiple lanes and directions of travel and must begin and terminate with either an intersection (node degree, $D > 2$) or a road terminal ($D = 1$).
2) The *section* layer consists of road sections with a constant number of lanes. There are also transition regions ($D = 2$) where two sections with different numbers of lanes meet. Each section begins and ends at either a transition or the end of a road.
3) The *segment* layer consists of road segments with a constant number of lanes in a single direction. Thus, a section which allows travel in two directions would contain two road segments. Every road segment begins and ends at the nodes of its corresponding road section.

The HD map state is defined as $M_{HD} = \{M_{SD}, \mathbf{\Theta}\}$ which combines an SD basemap with a set of parameters $\mathbf{\Theta}$ that encode the HD map geometry relative to the SD map. For example, if the SD map contains an edge representing a road centerline, the HD map will include that edge, along with a set of parameters that encode the real-world road geometry with respect to that SD map edge as will be described next.

Fig. 3 shows an example intersection with three adjacent roads. The SD map edges are shown as blue lines, while the nodes are blue circles. Each intersection node has a degree D corresponding to the number of roads at the junction. The HD map model locates the entry and exit ports of the intersection (shown as red lines in Fig. 3) using a set of four parameters per port θ^d_i, θ^a_i, θ^r_i, and θ^l_i for $i \in [1, D]$. These define the distance along the SD centerline to the junction port, the angle of the road-intersection boundary, and the widths in the right and left directions. Thus, these $4D$ parameters fully describe the shape and position of all of the entry and exit ports for an intersection.

Each road segment is discretized into K knot points with fixed offset distance $knot\_dist$ along the SD map edge. A set of parameters θ^{rb}_i and θ^{lb}_i for $i \in [1, K]$ define two polylines which represent the right and left boundaries of the road segment. Next, for a segment with L lanes, there are an additional $L \cdot K$ parameters $\theta^l_{i,j}$ for $i \in [1, K], j \in [1, L]$. These define the perpendicular distance from each knot point to the corresponding lane line in multi-lane segments. Together, these parameters are used to augment an SD basemap with the precise lane-level geometry and topology typically offered by a prior HD map.

B. Bird's Eye View (BEV) Semantic Segmentation

The semantic segmentation pipeline combines multiple raw sensor streams and processes them to generate features in the Bird's Eye View (BEV) for input to the HD map state estimation update step as shown in Fig. 2 *Perception*. The BEV frame is a natural choice given that the SD and HD maps are also typically represented in this frame.

1) BEV Projection: To generate a BEV feature, we first accumulate a set of N consecutive pointclouds $P^V_{t-N}, \ldots, P^V_t$ in the vehicle frame where each pointcloud combines all of the lidar sensor measurements in the indicated time interval $[t-1, t]$. Each time interval, is also associated with a set of N images from each of M cameras with image field-of-view spanning some (possibly overlapping) portion of the pointclouds P^V. Each point is then "painted" with RGB values by projecting the point into the image frame. Next, each of the

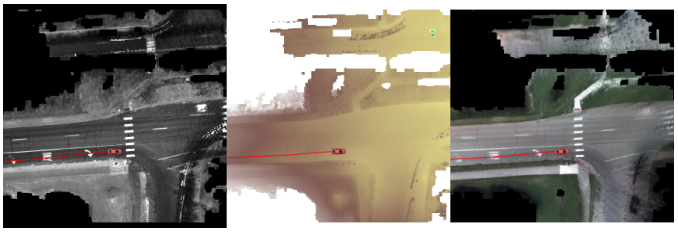

Fig. 4. The first step in the perception pipeline is to project the sensor data into the Bird's Eye View (BEV) frame. On the left is a lidar intensity frame, next is an elevation map, and last is an RGB frame. The red trace shows the path of the vehicle in each frame.

colored pointclouds for $P^V_{t-N}, \ldots, P^V_{t-1}$ are projected into the current vehicle frame. Although each frame requires $N-1$ transformations, each pointcoud P^V need only be painted once using this scheme. Finally, the accumulated, painted, pointcloud P^V_t is projected into the BEV frame by discretizing the space around the vehicle into a grid in the $X-Y$ plane, and aggregating the intensity, height, and RGB color values in each grid.

Not all cells in the grid will have data points due to both occlusions and the sparsity of lidar. Therefore, there is a trade-off between a very high resolution BEV grid with sharp features but many empty cells, and a low resolution grid with fewer cells missing data. To obtain a good balance, we compute the BEV at several resolutions, and then select the highest resolution available cell at each position which yields a variable resolution BEV image with higher resolution in locations with more dense sensor data.

We also choose specific targeted aggregation functions to best match the desired feature type. Namely, for the intensity feature, we use a mean aggregation, for the RGB features we choose the median for outlier rejection, and for the height feature, we use the minimum which efficiently filters out moving objects and gives an elevation map of the ground surface. Fig. 4 shows an example set of BEV features accumulating four seconds of data from four lidar sensors and six cameras arranged around the ego vehicle.

2) Semantic Segmentation: In order to estimate the HD map online we generate semantic features to infer the location of important map objects such as road boundaries and lane lines. To this end, we train a semantic segmentation model based on a ResNet-101 backbone [15]. The input to this network is a six channel image of the BEV frame with channels = $\{I, Z, R, G, B, SDT\}$ where I is the mean intensity of the lidar points, Z is the minimum height, R,G, and B are the median color channels from camera images, and SDT is a distance transform where each pixel gives the distance to the road centerlines in the SD prior map.

To reduce training requirements, we use transfer learning to fine-tune a DeepLabv3 [16] image segmentation model pretrained on the COCO [17] dataset. We modify this network by increasing the input channels to match the six channels in our features and initialize the weights using the pretrained input layer. Next, we modify the head of the network to have an output shape of $[B, C, W, H]$ where B is the batch size, C is the number of classes to predict, and W,H are the width and height of the features. We use $C = 4$ for classes=$\{background, roads, intersections, lines\}$.

We compile a training dataset using a ground truth (human annotated) HD map to generate segmented BEV labels of the four classes. We use a cross-entropy loss with the modification that any invalid (Nan-valued) pixels in the input BEV (corresponding to cells occluded or out of range of the vehicle sensors) are excluded from the loss calculation.

3) Signed Distance Transform: The final step in the perception pipeline entails generating signed euclidean distance tranforms [18] for each of the four classes in the semantic segmentation network output. The distance transforms contain the metric distance to the nearest feature of each class.

We require a modification to the standard distance transform to account for occlusions. Recall that the distance transform $DT(f)$ for a binary image f evaluates to 0 everywhere $f = 0$ (the background) and the euclidean distance to the nearest background point everywhere $f = 1$ (the foreground). Note that the signed distance transform is calculated as the sum of the forward and negative inverse distance transforms $SDT = DT(f) - DT(1-f)$. This transform has exactly the same values in the foreground as $DT(f)$ (the distance to the nearest background point), but has negative values in the background giving the distance to the nearest foreground point. Simply treating occluded points as background would have the undesired effect of spurious edges near missing data. Instead, we use an indicator function to generate the forward distance transform treating the missing data as foreground and add the negative inverse distance transform treating the missing values as background. This ensures the final distance transform only contains measurements that represent distances to other valid data and removes the artifacts missing data would otherwise introduce.

Discarding the background and missing data cells, we ultimately construct three output signed distance transforms: road regions, intersections regions, and lane lines. These are the inputs used to estimate the state of the HD map online as described next.

C. Online HD Map State Estimation

In the online HD map state estimation step, the HD map model parameters are updated to incorporate new information from the vehicle sensors. We formulate this problem as a maximum likelihood estimation in which we seek to find Θ^* such that

$$\Theta^* = \arg\max_{\Theta} P\left(\Theta | Z_{1:t}\right) \tag{1}$$

Where Θ is the set of decision variables defining an HD map, and $Z_{1:t}$ is the set of all sensor measurements. With Bayes rule we can decompose this as

$$P\left(\Theta | Z_{1:t}\right) \propto_{\Theta} P\left(Z_t | \Theta, Z_{1:t-1}\right) \cdot P\left(\Theta | Z_{1:t-1}\right) \tag{2}$$

The second term represents our prior belief over the state given our prior measurements, while the first term is an update step that integrates a new measurement to obtain an updated belief. We design a tracking algorithm to perform the state estimation updates online. In the next subsection, we describe how we initialize the prior $P\left(\Theta\right)$ for the first time before

Parameter	Notation	Unit	Distance Transform
Segment Boundary	$\theta_i^{\{rb/lb\}}$	[m]	Road Regions
Junction Distance	θ_i^d	[m]	Intersections
Junction Angle	θ_i^a	[rad]	Intersections
Junction Width	$\theta_i^{\{r/l\}}$	[m]	Intersections
Lane Boundary	$\theta_{i,j}^l$	[m]	Lane Lines

TABLE I
PARAMETER TYPES IN THE HD MAP MODEL

obtaining any measurements, and then we discuss how the update term is calculated to integrate new sensor data.

1) HD Map Initialization: We model the prior map parameters as gaussians where we require not only an initial estimate for the value of each parameter, but also an estimate of the uncertainty: $P(\Theta) = \mathcal{N}(\mu_\Theta, \Sigma_\Theta)$. Furthermore, while we can use the result from the last iteration in subsequent steps, in the initial estimate before the first sensor measurement is integrated, we only have the SD map available. Thus, we construct the HD map model, and initialize all of the parameters solely from the SD prior. We call this initial HD map estimate the *MapLite Prior* and it is constructed in an offline process without requiring any online sensor data. To obtain it, we first collect a training set of human annotated HD maps. Next, we compare these ground truth maps to the corresponding SD maps and calculate the parameter values that would generate an HD map that best matches the ground truth. Finally, we regress initial values and uncertainties from the distribution of these parameters over the training region. For example, we regress the value of the parameter θ^{rb}, a road boundary parameter, in our ground truth dataset as a function of the number of lanes in each segment. We then initialize the parameter values at the mean of this distribution, and the variance as the sum of the squared residuals.

In this way, we can generate an initial guess for the full HD map state. As expected, we found that parameters that represent features such as lane width which are fairly consistent initialize with relatively low uncertainty, while those that encode features like intersection shape, which are much more varied, initialize with relatively high uncertainty.

Note that since all of the parameters are relative measurements, the HD map is agnostic to the reference frame (e.g. transforming the basemap geometry transforms all of the HD map geometry but importantly, the parameter values are unchanged). This allows the inference to be performed in the local vehicle frame, which is constantly changing with respect to the map frame, without requiring any transformation of the estimated parameters and uncertainties in the update step.

2) Belief Update: In order to update the state estimate, we need to obtain the sensor likelihood from measurements encoded as the set of distance transforms (roads, intersections, and lane lines) obtained in the Semantic Segmentation step Sec. III-B. We can simplify the expression in Eq. (2) by assuming that the measurements are conditionally independent,

$$P(Z_t|\Theta, Z_{1:t-1}) = P(Z_t|\Theta) \tag{3}$$

First, note that most of the decision variables are not in view in a given update step. Therefore we define Θ_t as:

$$\Theta_t = \{\forall\theta \in \Theta \text{ s.t. } \theta \text{ is } in\_view \text{ at time } t\}$$

The update step is then reduced to only updating Θ_t as our estimate of all other state variables is unchanged.

To find the sensor update distribution, we pair each of the five parameter types found in our HD map representation with an associated distance transform as shown in Table I. Let $X(\theta)$ be a rendering function that returns one or more points in the vehicle frame representing the geometry encoded in each parameter (see Fig. 3). For example, $X\left(\theta_i^d\right)$ returns the point on the given junction line at the distance θ_i^d from the intersection node. Similarly, $X(\theta_i^a)$ returns a set of points distributed along the junction line rotated by the angle θ_i^a. Next, let $\delta(X)$ be the mean distance of the points in $X(\theta)$ to the associated distance transform in Table I. The likelihood component of the objective function is obtained as

$$P(Z_t|\mu_{\Theta_t}) = 1 - \frac{\delta(\theta)}{\delta_{max}}$$

where δ_{max} is the value at which the distance transforms are clipped (in our case $10\,\mathrm{m}$). Therefore, as the distance between the encoded geometry and its observation increases, the likelihood term will become smaller. Integrating this estimate to obtain its normalization constant would be computationally prohibitive. However, since its purpose in the objective function is to balance the observation with the prior, we instead multiply these two terms with parameter weights which we tuned for best performance.

We then solve for the optimal mean by combining Eqs (2, 3) to approximate Eq (1) as:

$$\mu_{\Theta_t} = \arg\max_{\mu_{\Theta_t}} \left[P(Z_t|\mu_{\Theta_t}), \quad P(\mu_{\Theta_t}|\Theta_{t-1})\right] W_\mu^T$$

where W_μ^T is a weight vector that balances between the importance of the prior and the update.

The associated uncertainty is approximated as

$$\Sigma_{\Theta_t} = \left[P(Z_t|\mu_{\Theta_t}), \quad P(\mu_{\Theta_t}|Z_{1:t-1})\right] W_\Sigma^T$$

where W_Σ^T balances between confidence in the prior, and the measurement.

While we cannot empirically measure this uncertainty, we tune this function to strike a balance between our confidence in new measurements versus prior measurements. Numerical optimization is then used online to obtain updated parameter means and uncertainties at each measurement update.

IV. EVALUATION

A. Dataset

We evaluate our approach by collecting a dataset of 100 real-world autonomous vehicle sequences in Michigan and California. The vehicle platform includes four lidar sensors and six cameras giving a 360-degree view around the vehicle. Throughout each sequence, the ground truth vehicle pose is obtained using a factor-graph based SLAM system implemented in Ceres [19] that computes an optimized trajectory by associating landmark observations to a prior fixed landmark

map. We estimate the trajectory estimation error in this system to be $< 10\,\mathrm{cm}$. This prior map was also the reference for the manually annotated ground truth.

Intersections represent the most complex component of the HD map inference problem as merely detecting lanes on a straight road is relatively much simpler. For this reason, we focus each sequence on an intersection traversal, beginning the sequence ten seconds prior to the intersection entrance, and terminating it five seconds after the exit. Each sequence is composed of a series of frames at 10Hz where each frame includes:

1) Six synchronized camera images (1936x1216) giving a surround view of the vehicle.
2) An accumulated lidar pointcloud from the four lidar sensors over a four second interval.
3) A local region from the SD prior map downloaded from OpenStreetMap.org.
4) A local region in the hand annotated HD map used for evaluation purposes only.

We collected a total of 100 sequences, ranging in duration from 10 to 25 seconds. These include traversals through 18 unique intersections. All sequences are non-overlapping such that any two sequences that traverse the same intersection do so in either a different direction, and/or at a different time. We randomly select 10 sequences to hold back for validation, with the remaining 90 used for training. Each frame is processed sequentially, by first projecting to BEV, performing the semantic segmentation to generate feature distance transforms, and then updating the HD map model parameters that are within the sensor view at each time step. The final HD map model obtained after a single traversal is then evaluated as described below. Since we are interested in online estimation in this work, we do not save the updated maps between sequences, instead, each sequence begins with a new model derived only from the SD prior map.

B. Semantic Segmentation Results

While semantic segmentation is not the main focus of this paper, we report the results of our segmentation pipeline here to evaluate the role of this component in the overall system. We train the semantic segmentation CNN using the 90 training sequences on a single Nvidia Titan XP GPU for 2.5 hours. The input features are 6 channel 1936x1216 BEV images obtained by concatenating {*Red, Green, Blue, Lidar Intensity, Height, OSM Distance Transform*}. The output includes per-pixel labels in {*Background, Road, Intersection, Lane Line, Out of Range*} where out of range refers to pixels in the BEV that were empty (e.g. due to occlusion).

We evaluate both overall and per-class accuracy on both the training and validation splits of the dataset. We achieve an overall accuracy of 93% on the validation sequences, with per-class accuracies of Background: 97%, Road: 82%, Intersection 96%, and Lane Line: 90%. Fig. 5 shows the confusion matrix over these classes. Notably, although the network performs well at predicting most classes, it does show some confusion between road and lane markings. We also compare the results over the training runs and find them to be similar at: 95% overall. This indicates that the training dataset is diverse enough to avoid significant overfitting.

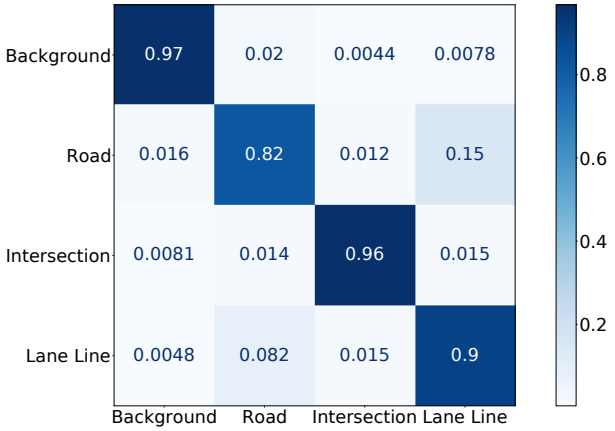

Fig. 5. The confusion matrix for the semantic segmentation. The true class is on the vertical axis, with the prediction on the horizontal.

C. MapLite 2.0 Results

In evaluating the performance of MapLite 2.0 we compare the HD map state estimate after a single traversal to the hand annotated ground truth map. We limit the evaluation to a $30\,\mathrm{m}$ swath around the vehicle to include only the portion of the map likely to have passed within range of the onboard sensors. Since the HD map is based on the underlying SD map prior, while the hand annotated ground truth map was generated entirely independently, there is no guaranteed one-to-one correspondence between features. For example, the decision to consider a divided road two separate one-way roads, as opposed to a single two-lane road has some ambiguity. Therefore, we propose two metrics for evaluating the overall quality of the HD map estimate.

1) **Vehicle Trace Accuracy** is a coarse metric that relies on the fact that the vehicle always drove on the road. We compute the portion of the driven path that falls within the road boundaries in the HD map estimate.
2) **Road-region Intersection Over Union (IOU)** is a finer metric that compares the regions defined as road in the annotated map, to those in the HD map.

We compute these metrics over all the validation sequences and also calculate the standard deviation to measure the consistency of the results.

For comparison, we also compute these metrics on a "Raw OSM" map, which is a naively generated road region obtained by inflating the OpenStreetMap road centerlines a fixed nominal lane width. Additionally, we run the same analyses on the "MapLite Prior" which is the HD map structure, derived from the OSM prior, along with the initial estimates of the state parameters, but before any onboard sensor data is integrated. Finally, to better understand how the semantic segmentation accuracy affects our results, we also show an "Oracle" result. This evaluates the same structured estimator algorithm, but using ground truth semantic segmentation instead of the CNN.

Table II shows the results of these evaluations. It is interesting that simply applying the MapLite Prior already improves

	Vehicle Trace Accuracy	IOU Road Regions
Raw OSM	0.55 (0.40)	0.59 (0.13)
MapLite Prior	0.87 (0.18)	0.77 (0.04)
MapLite 2.0 (Ours)	0.98 (0.04)	0.88 (0.04)
Oracle	0.99 (0.01)	0.89 (0.05)

TABLE II
MAPLITE 2.0 EVALUATION RESULTS. VALUES REPRESENT MEAN (STD)

the Vehicle Trace Accuracy from 55% to 87% without requiring any additional real-world data. This indicates that applying the HD map structure with the data driven initialization is sufficient to obtain a strong prior for the HD map inference. The MapLite 2.0 Posterior, which includes the inference from the onboard perception increases this accuracy to 99%. Compared to the Vehicle Trace Accuracy metric the IOU metric is finer-grained as it compares the entire observed road-region rather than only the vehicle trace. Here, we once again see increasing accuracy with each step as well as decreasing standard deviation which indicates more consistency.

The reason the IOU does not reach unity is likely due to three factors: 1) The HD map is derived from the SD prior OpenStreetMap while the ground truth map is hand annotated independently. Therefore, there is some inherent ambiguity as to what should be labeled "road" (e.g. paved dividers in semi-divided roads, private driveways, etc.) 2) Model limitations that do not exactly represent reality. For example, our intersection model, links each entry/exit port with a straight line segment for simplicity, which is only a coarse approximation of real-world intersection regions used for evaluation. 3) The inference optimization could get caught in a local minimum. Computational resources preclude exhaustive search of the entire parameter space, therefore, a local minimum could prevent the model from reaching the optimal state.

While the oracle does outperform the MapLite 2.0 model slightly, it demonstrates that the majority of the remaining error lies in the inference step. Therefore, to improve these metrics, future work should focus on the three sources of inference error described previously, rather than on improving the semantic segmentation step to have the greatest impact.

Fig. 6 shows several illustrative real-world examples. The first two columns include the components that can be run offline as the MapLite prior is derived from the SD prior before integrating sensor data. The last three columns show how the sensor data is segmented to create the distance transforms used to update the HD Map state estimate. One issue of note occurs in row (4) where the road includes a center turn lane which our model does not explicitly handle. This causes the yellow centerlines to appear jagged. In contrast, in row (2) the divided center is correctly inferred online, even though this is not represented in the prior. In future work, we could expand our model to explicitly account for special lane scenarios such as this. In row (5) an error in the road edge at a single point in the road exiting the intersection at the top leads to a "kink" in the otherwise smooth road estimate. Providing a term to encode a "smoothness" constraint may provide more realistic results.

V. DISCUSSION AND CONCLUSION

In this work, we have presented a novel HD map model representation that is fully relative to an underlying SD map prior. We have shown that it can provide a reasonable HD map prior directly from an available SD map. Furthermore, we have designed a perception and inference system that can be used to estimate the HD map model parameters to generate an online HD map estimate from only an SD prior and onboard perception.

This can be useful not only for allowing autonomous vehicles to navigate in previously unmapped regions, but also for at least two other important applications. 1) An online HD map could be run in the background of a fully mapped solution to detect changes and suggest map maintenance, and 2) MapLite 2.0 could be used online to jumpstart mapping of unexplored regions autonomously, which could then be combined to create an optimized offline map for future use.

There are also some important limitations to this system that should be considered. The map model is inevitably an approximation and cannot capture all real-world complexity. While we found that our model is sufficient to represent a large variety of road types in multiple cities in our dataset, care must be taken to account for new road structures. For example, there is some judgement required when choosing the information that should be gleaned from the SD map, versus inferred online. In this work, we utilize the lane number attribute in the SD map as a prior, but detect the actual lane boundaries from onboard sensors. Conversely, we don't use crosswalk information, even though that is available in some regions. Therefore, care should be taken to select what to include in the prior and what to infer online based on the availability in the region of interest.

In [10], we focused on probabilistic estimation of the vehicle localization to a coarse SD map. In this work, we instead consider the problem of inferring the HD map online, while relying on our SLAM system for localization. This allows us to directly evaluate the HD map inference problem in isolation. In the future, we hope to combine these systems to enable both localization of the vehicle to an SD map, and online inference of the HD map simultaneously.

Finally, we chose a 2D map representation for simplicity which precludes the ability to handle multi-level roads (e.g. overpasses). Handling multiple levels through a map-splitting approach, or an extension to 3D would be needed for deployment on such roads.

REFERENCES

[1] M. M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *IEEE Pervasive Computing*, Oct. 2008. 1, 3
[2] G. Máttyus, S. Wang, S. Fidler, and R. Urtasun, "Enhancing Road Maps by Parsing Aerial Images Around the World," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1689–1697. 2
[3] J. Liang and R. Urtasun, "End-to-End Deep Structured Models for Drawing Crosswalks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 2
[4] N. Homayounfar, W.-C. Ma, J. Liang, X. Wu, J. Fan, and R. Urtasun, "DAGMapper: Learning to Map by Discovering Lane Topology," in *Proc. IEEE Int. Conf. Comput. Vis.*, October 2019. 2
[5] J. Zürn, J. Vertens, and W. Burgard, "Lane Graph Estimation for Scene Understanding in Urban Driving," *IEEE Robot. and Automation Lett.*, vol. 6, no. 4, pp. 8615–8622, 2021. 2

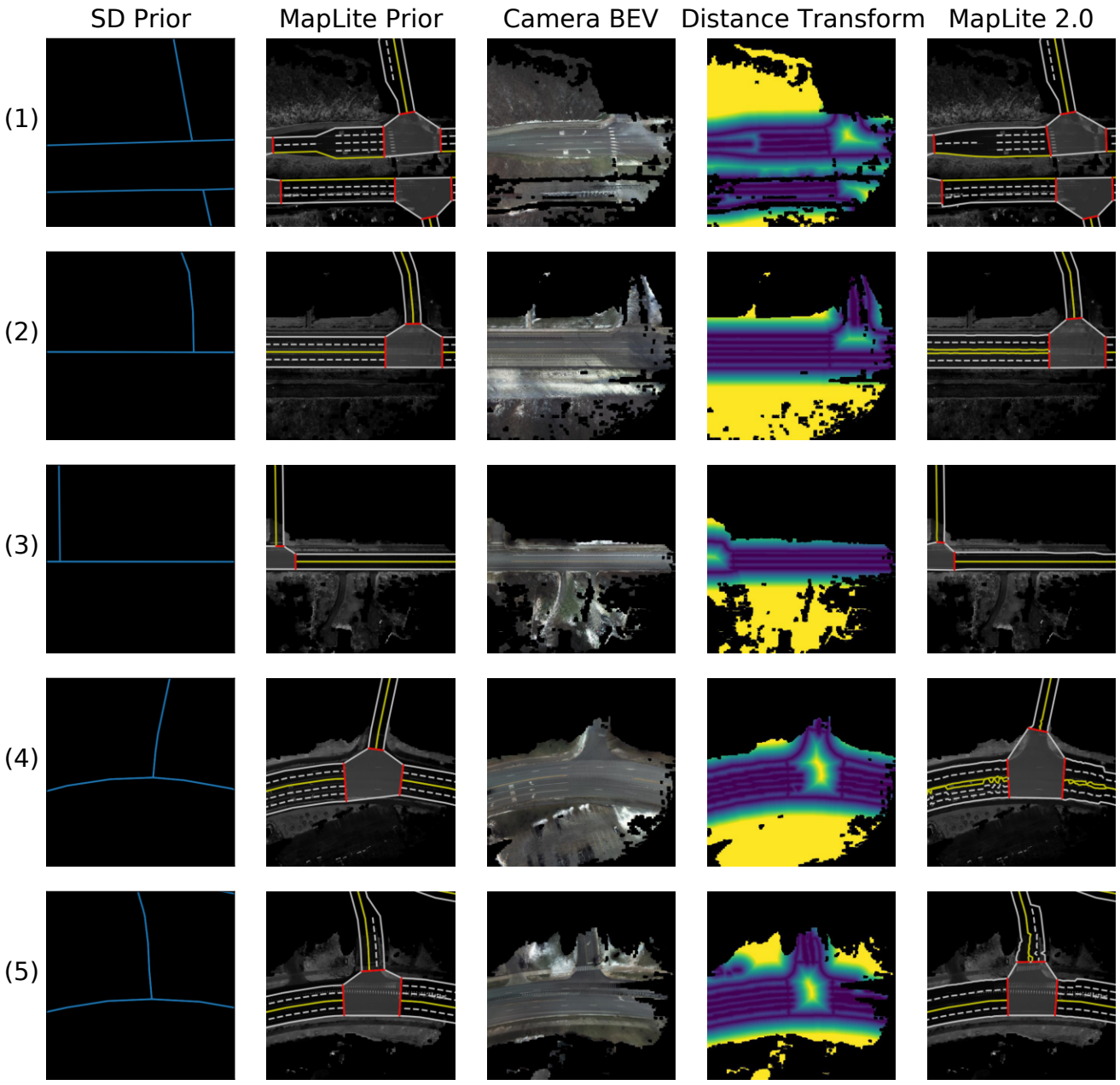

Fig. 6. Selected examples of the online estimated HD map. Rows indicate different scenarios while columns illustrate pipeline outputs.

[6] A. Meyer, N. O. Salscheider, P. F. Orzechowski, and C. Stiller, "Deep semantic lane segmentation for mapless driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.* IEEE, 2018, pp. 869–875. 2

[7] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: An instance segmentation approach," in *IEEE Intell. Vehicle Symposium*, 2018. 2

[8] Y. Ko, Y. Lee, S. Azam, F. Munir, M. Jeon, and W. Pedrycz, "Key points estimation and point instance segmentation approach for lane detection," *Proc. Int. Conf. on Intell. Transportation Syst.*, 2021. 2

[9] Q. Li, Y. Wang, Y. Wang, and H. Zhao, "HDMapNet: A Local Semantic Map Learning and Evaluation Framework," *arXiv preprint arXiv:2107.06307*, 2021. 2

[10] T. Ort, L. Paull, and D. Rus, "Autonomous Vehicle Navigation in Rural Environments Without Detailed Prior Maps," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2018, pp. 2040–2047. 2, 7

[11] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments," in *Robotics: science and systems*, vol. 4. Citeseer, 2007, p. 1. 2

[12] R. Liu, J. Wang, and B. Zhang, "High definition map for automated driving: Overview and analysis," *The Journal of Navigation*, vol. 73, no. 2, pp. 324–341, 2020. 3

[13] "Navigation Data Standard (NDS)," https://nds-association.org/, 2022, accessed: 2022-01-26. 3

[14] "ASAM OpenDRIVE," https://www.asam.net/standards/detail/opendrive/, 2022, accessed: 2022-01-26. 3

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 770–778. 4

[16] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017. 4

[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. European Conf. Comput. Vis.* Springer, 2014. 4

[18] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, "2D Euclidean distance transform algorithms: A comparative survey," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, pp. 1–44, 2008. 4

[19] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 3 2022. [Online]. Available: https://github.com/ceres-solver/ceres-solver 5