MapLite: Autonomous Intersection Navigation without a Detailed Prior Map

Teddy Ort[1], Krishna Murthy[2], Rohan Banerjee[1], Sai Krishna Gottipati[2], Dhaivat Bhatt[2], Igor Gilitschenski[1], Liam Paull[2], Daniela Rus[1]

***Abstract*— In this work, we present MapLite: a one-click autonomous navigation system capable of piloting a vehicle to an arbitrary desired destination point given only a sparse publicly available topometric map (from OpenStreetMap). The onboard sensors are used to segment the road region and register the topometric map in order to fuse the high-level navigation goals with a variational path planner in the vehicle frame. This enables the system to plan trajectories that correctly navigate road intersections without the use of an external localization system such as GPS or a detailed prior map. Since the topometric maps already exist for the vast majority of roads, this solution greatly increases the geographical scope for autonomous mobility solutions. We implement MapLite on a full-scale autonomous vehicle and exhaustively test it on over 15km of road including over 100 autonomous intersection traversals. We further extend these results through simulated testing to validate the system on complex road junction topologies such as traffic circles.**

I. Introduction

At present, the majority of industry-led approaches to autonomous driving involve either building and maintaining detailed maps, or making structure assumptions about the environment such as the existence of road markings on highways and major roads. The latter approach is very appealing since building and maintaining maps is time-consuming, expensive, and suffers from perceptual aliasing in some rural environments. However, road markings are not always present. In fact, only about 2/3 of the roads in the United States are paved [1]. Additionally, these approaches based on local infrastructure generally only enable road following and not navigation (i.e., traversing intersections to reach a goal). As a result, there is a significant portion of the road network that cannot be easily used with state-of-the-art autonomous driving systems.

One class of approach addressing this problem is based on using neural network-based driving approaches [2]–[5] as they have demonstrated lane-stable driving without detailed maps. However, those approaches lack explainability, verifiable robustness, and require a high amount of training data to generalize. Early approaches incorporating topometric map data for navigation [6] usually rely on building structure information and are thus not applicable to most rural settings. In a previous work [7], we demonstrated lane following behaviors without detailed metric maps. This approach required human-driver disengagements at intersections, owing to their complex topologies and, thus, did not allow for fully autonomous point-to-point navigation.

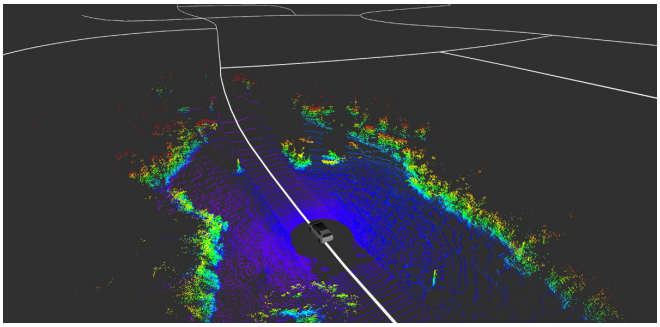

Fig. 1. **MapLite**: A visualization of our system operating on a rural road in Central Massachusetts. The white lines show the topometric map obtained from OpenStreetMap.org. The colored pointcloud is obtained from the lidar scanner used in MapLite to navigate without the need to localize on a detailed prior map. Note that the noisy topometric map has been registered to the local sensor data.

In this work, we propose a "one-click" solution for navigation using only a weak topological prior, such as OpenStreetMap (OSM), that is available for the entire road network and significantly less memory-intensive than a full dense map. The main advantage of this approach is that it requires *only* freely existing infrastructure (the OSM) and therefore scales to rural environments without the need for any map maintenance or updating of neural network models.

Our proposed method comprises three main parts: First, we segment the scene using on-board sensors. Second, we perform a topometric registration to fit the OSM to the local sensor data frame. Finally, we solve the navigation problem at the topological level and then the path planning problem at the metric level. To the best of our knowledge this is the first deployment of an autonomous navigation pipeline that can autonomously plan and execute high-level navigation tasks without the need for a precise georeferenced localization estimate either from high-precision GPS, or highly detailed prior maps. In summary, we claim the following contributions of our method:

1) A novel topometric map registration algorithm that approximates the maximum a posteriori estimate of the registration between the vehicle and the map.
2) A route planning method we call "smart replanning" that allows for efficiently solving the topological navigation problem on a frequently updating map
3) Extensive evaluations on a full-scale autonomous car in a rural driving setting where the roads lack structure and through simulations in CARLA [8] with more complex road topologies such as traffic circles.

[1] Computer Science and Artificial Intelligence Lab, MIT {teddy, rohanb, igilitschenski, rus}@mit.edu
[2] Département d'informatique et de recherche opérationelle, Université de Montréal {krishna, sai, dhaivat, paulll}@iro.umontreal.ca

II. Related Work

In this section, we briefly review some closely related approaches.

Map-based autonomous driving. Most of the existing approaches to autonomous driving rely on detailed prior maps of the environment, often constructed using precision laser scanners. Typical driving strategies on pre-built maps involve some form of localization: lidar-based [9]–[11] or vision-based [12], followed by local trajectory and motion planning. Evidently, such approaches need extremely precise prior maps, and cannot easily adapt to environment changes (e.g., repaved roads, detours, etc.). Furthermore, such approaches are not suitable for driving beyond confined areas such as cities/towns.

Map-less autonomous driving. To overcome some of the scalability issues with map-based autonomous driving, there have been several attempts to go *mapless* (at least in part). Such approaches leverage environment structure for local perception. While a majority of such approaches rely on color/grayscale images to detect structures such as roads [13]–[15], lane markings [16], [17], and road boundaries [18], these suffer the common pitfalls of any image-based segmentation algorithm: poor generalization across varying illumination, scene dynamics, and infrastructural changes. A few other approaches [7] use lidar scans to detect drivable regions. However, most such approaches [16], [17] make assumptions about the local structure of the environment (eg. availability of lane-markings, etc.).

Another popular set of approaches take an *end-to-end* learning approach to mapless driving. These are either based on imitation learning [2]–[4], or on reinforcement learning [5]. While such approaches alleviate the need for precision maps, they are still laborious to implement, as they require enormous amounts of data/demonstrations from a (human) expert.

Road segmentation. In contrast to mapped approaches, where the driveable road surface is taken as a prior from the map, real-time road surface segmentation is a crucial requirement for mapless navigation. The basic problem of road segmentation is simple: for each element in the incoming sensor data, label it as either "road" or "off-road". Fig. 2 "Road Segmentation" shows an example pointcloud that has been segmented showing the points lying on the road surface in black and the off-road points in blue. This problem has been studied extensively in the literature including model-based approaches such as [15] and those that rely on camera images [14], [19], lidar [20], and combined vision and lidar [21], [22]. However, the MapLite scenario imposes some unique constraints. First, since a major benefit of our independence from prior maps is the ability to operate in large-scale rural regions where roads are missing structures such as lane markings or curbs, we abstain from utilizing these features in our approach. Furthermore, deployment on a real-world vehicle necessitates fast processing time. Finally, rural roads are often unlit, and have highly varying illumination even during the daytime due to seasonal vegetation. Therefore, we prefer methods that rely solely on lidar since these sensors are immune to changes in ambient lighting.

Here, we implement and compare three different model choices for our road segmentation module: a linear Support Vector Machine (SVM) and two convolutional networks: SparseConvNet (SCN) [23] and PointNet [24].

Topometric localization/registration. Another class of approaches to autonomous driving uses *topometric* maps (predominantly OpenStreetMaps [25]). In a series of works, Ballardini et al. [26]–[28] leverage the OSM for localization and intersection detection. In a similar vein, OpenStreetSLAM [29] aligns OSM data to local maps as an additional cue in a Monte-Carlo localization framework. However, all these approaches demonstrate the use of OSM in urban areas, where the precision of OSM is significantly higher than for rural areas. In particular, our approach can efficiently deal with missing roads, incorrect intersection annotations, and large deviations in road shapes.

III. MapLite Method

An overview of our proposed system is shown in Fig. 2. Note that our approach requires only a topometric map, which does not contain the detail necessary for precise localization. However, we demonstrate that it suffices for specifying the goal location and onboard sensors can be used to plan safe trajectories on the fly. The following subsections describe the five main components that enable this system to safely navigate to the desired location without the need for a detailed map: Topometric Map, Road Segmentation, Map Registration, Route Planner, and Motion Planner.

A. Topometric Map

The topometric maps used by MapLite are simple graph-like data structures with the extension that each node is associated with a two dimensional point on the surface of the earth described by its longitude and latitude. We utilize the flat-earth approximation with the UTM transform [30] which places the vertices in a plane. Thus for each map M we have

$$M = \{V, E\}$$

where each vertex $v_i \in \mathbb{R}^2$ describes a waypoint and each edge $e_i \in E \subset |V| \times |V|$ describes a road segment. However, while the connectivity of the network can be assumed to be relatively correct, the same cannot be said for either the relative or global accuracy of the specific waypoints.

Topometric maps of the roads throughout the world are readily available from a number of sources. For this work, we used maps downloaded from OpenStreetMap [25] which provides open access for the public to both download and upload mapping data. Fig. 1 shows an example of the topometric map (in white) used.

These topometric maps differ from the detailed maps typically used for localization in a number of important ways. First, detailed maps typically need to be recorded with a human driver prior to deployment and often require further labor-intensive manual editing to ensure the maps are accurate [31]. For this reason, detailed maps are currently

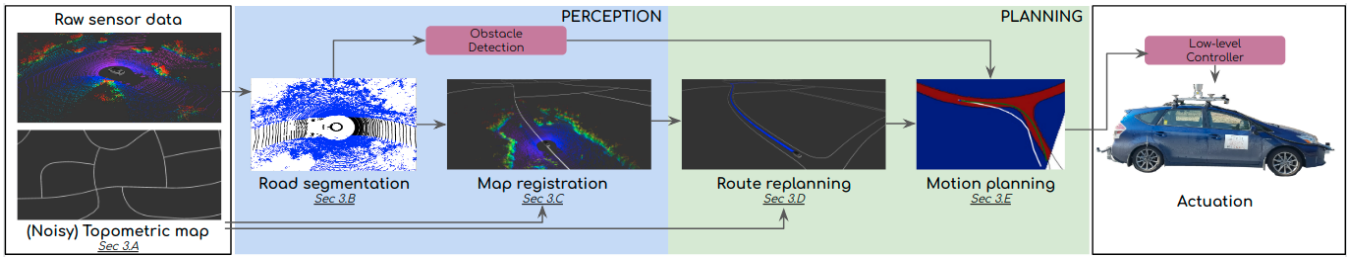

Fig. 2. The MapLite system differs from a typical autonomous navigation pipeline in five components: Topological Map, Road Segmentation, Map Registration, Route Replanning, and Motion Planning

available for only a handful of cities and even those are often only useful for a particular autonomous vehicle sensor setup. Topometric maps on the other hand, are already freely available for most roads throughout the world.

Another difference between topometric maps and detailed maps lies in the storage size. While a detailed map used to localize over the 20,000 miles of roads in a small city takes about 200GB [31], a similar topometric map could be stored in about 3.5GB. Considering the US alone contains over 4,000,000 miles of roadways, this can have a large impact on the storage and data transmission required for autonomous navigation.

Finally, since detailed maps include many transient surface features, any changes to the road surface, seasonal vegetation, or building construction can render a detailed map obsolete often in just a few months. Topometric maps, on the other hand, only need updating when the road network itself changes which means these maps are typically accurate for many years.

B. Road Surface Segmentation

We use a linear SVM model for road surface segmentation since its extremely fast runtime is needed for real-time operation at speed, and its accuracy was close to that of the much more performance-heavy CNNs (see section IV-B for a comparison). Our linear SVM relies on five features extracted from the lidar data. The first feature is the elevation z measured directly by the sensor. Next, a measure of the surface texture is calculated using an approach similar to [32] where the local variance is calculated at each point using a sliding window for all points $\boldsymbol{p}_i$ in the neighborhood N_p. Then the local variance v_p for each point is calculated using

$$v_p = \frac{1}{|N_p|} \sum_{i \in N_p} |\boldsymbol{p}_i - \overline{\boldsymbol{p}}|^2$$

where $\overline{\boldsymbol{p}}$ is the mean over all points in N_p. This feature yields a measure of the local surface texture that is larger for rough surfaces (e.g. grass and trees) and smaller on driveable surfaces. Next, the intensity of each return is also included to account for the difference in reflectivity of road surfaces. The fourth feature is a unique laser ID in $1-64$ to account for physical differences in the laser transmitters. Finally, an indicator feature in $\{0, 1\}$ is used to indicate any points that did not return. This is important as the presence of an out-of-range value (either too far or too close) contains valuable information for segmentation. These five features are used to together to classify each point as either road or not road.

C. Topometric registration

In the map registration step, we aim to obtain an estimate of the robot location on the map using only odometry measurements and lidar scans. Formally, our goal is to obtain the estimate

$$\hat{\boldsymbol{x}}_t = \arg\min_{\boldsymbol{x}_t} \left(-\ln P\left(\boldsymbol{x}_t | Z^o_{1:t}, Z^L_{1:t}\right)\right) \tag{1}$$

where $\boldsymbol{x}_t = [x_t, y_t, \theta_t]$ is the robot's pose in the topometric map frame at time t while $Z^o_{1:t}$ and $Z^L_{1:t}$ are the sets of all odometry and lidar measurements up to time t respectively. An approximation of this likelihood is required to allow for real-time operation.

1) Likelihood Approximation: Using Bayes' rule, the probability in (1) can be represented in terms of its prior, the lidar observation likelihood, and the odometry-based state prediction as

$$\begin{aligned} &P\left(\boldsymbol{x}_t | Z^o_{1:t}, Z^L_{1:t}\right) \\ &\quad \propto_x P\left(Z^L_t | \boldsymbol{x}_t\right) \cdot P\left(\boldsymbol{x}_t | Z^o_{1:t}, Z^L_{1:t-1}\right) \\ &\quad = P\left(Z^L_t | \boldsymbol{x}_t\right) \int P\left(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, Z^o_t\right) \\ &\qquad\qquad \cdot P\left(\boldsymbol{x}_{t-1} | Z^o_{1:t-1}, Z^L_{1:t-1}\right) \, \mathrm{d}\boldsymbol{x}_{t-1} \, . \end{aligned}$$

We approximate the prior as a discrete distribution concentrated at the last estimate $\delta(\boldsymbol{x}_{t-1} - \hat{\boldsymbol{x}}_{t-1})$ which results in

$$P\left(\boldsymbol{x}_t | Z^o_{1:t}, Z^L_{1:t}\right) \propto_x P\left(Z^L_t | \boldsymbol{x}_t\right) \cdot P\left(\boldsymbol{x}_t | \hat{\boldsymbol{x}}_{t-1}, Z^o_t\right) \, .$$

2) Lidar Observation Likelihood $P\left(Z^L_t | \boldsymbol{x}_t\right)$: Each lidar scan Z^L_t is represented as a set of n 3-tuples $\boldsymbol{z}_i = (x^L_i, y^L_i, l_i)$ where $x^L_i, y^L_i \in \mathbb{R}$ give the position of each measured lidar point in the sensor frame and l_i is a binary classification label obtained in the previously described segmentation module. We define the signed distance function $f_D\left(\boldsymbol{z}_i, \boldsymbol{x}_t, M\right)$ representing the distance from the point $\boldsymbol{z}_i$ to the nearest point on any edge in the topological map M (assuming the vehicle is in location $\boldsymbol{x}_t$). We model the probability of observing each point $\boldsymbol{z}_i$ at location $\boldsymbol{x}_t$ using a sigmoid function

$$P\left(\boldsymbol{z}_i | \boldsymbol{x}_t\right) = \begin{cases} \frac{1}{1+\exp(f_D(\boldsymbol{z}_i, \boldsymbol{x}_t, M) - r_w)} & l_i = 1 \\ 1 - \frac{1}{1+\exp(f_D(\boldsymbol{z}_i, \boldsymbol{x}_t, M) - r_w)} & l_i = 0 \end{cases} \tag{2}$$

where r_w is a tunable parameter that represents the likelihood of finding points labeled road, far from the map M which

contains road centerlines. Notice that at the road center where $f_D(z_i, M) = 0$, $P(z_i) \approx 1$ if $l = 1$ $P(z_i) \approx 0$ if $l = 0$ while far from the road, the converse is true. Also, in this work, a constant value was sufficient for r_w since all the roads traversed were of similar width ($5 - 7$m) However, given that the lanecount and roadway class is available for most roads on OpenStreetMap, this parameter could easily be extended to depend on the road type. Finally, the overall probability of an entire lidar scan is computed as the product over the probabilities of the individual measurement tuples $P\left(Z_t^L|\boldsymbol{x}_t\right) = \prod_i P(z_i|\,\boldsymbol{x}_t)$.

3) Odometry based prediction $P(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t-1},\, Z_t^o)$*:* Each odometry measurement $Z_t^o = [\Delta x_t,\, \Delta y_t,\, \Delta\theta_t]$ with respect to the previous pose is obtained by fusing both the wheel encoder information and the IMU measurements using an Extended Kalman Filter [33]. The odometry based likelihood of the future state is then modeled as

$$P(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t-1},\, Z_t^o) \propto_x \exp\left(-\frac{||\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t-1} + Z_t^o||}{b}\right)$$

with scale factor b that is obtained, for computational reasons, through parameter tuning rather than by estimating it along with the prior.

4) Implementation for Real-Time Operation: The main remaining driver of computational cost is evaluating $P\left(Z_t^L|\boldsymbol{x}_t\right)$. This is due to the fact that $f_D({}^o z_i, M) \in O(|E|)$ (with $|E|$ being the number of edges in M) and thus $P\left(Z_t^L|\boldsymbol{x}_t\right) \in O(n \cdot |E|)$. To achieve a speed-up we employ a number of further approximations. First, we discretize the space containing the map M into cells of size r_c (0.5m for our experiments) and precompute the distance transform. This computation only needs to happen once for each map, and in practice takes <1s to compute for a map of size 1km^2. This approximation turns distance computations into a simple lookup with runtime independent of number of edges or scanned points. Next, we randomly subsample the input lidar scans using two tunable parameters $s \in \mathbb{N}_+$, $b \in [0, 1]$ where s is the number of points to sample, and b is the desired share of points from the road (i.e. where $l_i = 1$) to account for high label-imbalance. Finally, we simplify (2) by approximating

$$\begin{aligned}(1 + \exp(f_D(\boldsymbol{z}_i, \boldsymbol{x}_t, M) - r_w))^{-1} \\ \approx 1 - \min\left(\frac{f_D(\boldsymbol{z}_i, \boldsymbol{x}_t, M)}{r_w}, 1\right)\end{aligned}$$

In practice, we found these approximations to have a negligible effect on system performance, while decreasing the time required to solve the problem from $10s$ to $20ms$ on a standard laptop with an Intel i7 processor.

D. Route Planning

The route planner is responsible for choosing the shortest route from the start location of the vehicle to the goal. It takes the registered topometric map as input, converts it into a graph structure and calculates euclidean distances as edge weights. Next A* [34] is used to plan the shortest path from the start to the goal. Given the heavy computation required by the other modules, we cannot afford to recalculate the entire route at a high frequency. However, since the topometric registration frequently updates the map we cannot simply reuse the original route plan. To account for this, we use a "fast_replanning" method that allows the positions of the waypoints in the map to be changed in the metric space without recalculating the route in the topological space. Furthermore, since it is possible the vehicle could deviate from the planned path (e.g. if the map registration hasn't updated recently) we also check for deviations from the plan and recalculate the A* algorithm only as necessary. Algorithm 1 describes how the fast replanner determines when a replan of the route plan is necessary or simply an update of the existing path to the new map. In our testing, we found that this reduced the average latency of the route planning module from $100ms$ to $1.5ms$. In section IV-C we show that this optimization has a substantial impact on real-world system performance.

Algorithm 1 Fast Route Replanning

```
Inputs: (goal, map, pose, route_plan)
while pose not equal to goal do
  if route_plan is empty then
    Store a new route_plan from A*(pose, goal, map)
    Set path using route_plan
  else
    Set path using fast_update(route_plan, map)
    if distance(pose, path) > recalculate_threshold
    then
      Recalculate a new route_plan with A*
    end if
  end if
  return path
end while
```

Additionally, we utilize a Finite State Machine to break the entire route plan into sub-tasks (e.g. drive to the next intersection) which enables the vehicle to wait for a goal, drive, make the correct navigation choice at each intersection, and finally stop at the goal destination.

E. Trajectory Planner

The trajectory planner generates the trajectory the vehicle must follow to reach its destination. As seen in Fig. 2 it takes as input a segmented lidar scan as well as the high-level navigation reference path from the Route Planner. We utilize a variational trajectory planner to generate safe trajectories. However, unlike typical [35] implementations, we do not obtain road boundaries from a detailed map. Instead, the road boundaries are obtained from the road segmentation module (Section III-B) and this planner integrates the high-level navigation information obtained from the route planner to plan a locally safe path in the sensor frame. Fig. 3 shows an example of the typical problem this planner solves. The blue point is the current vehicle location, and the red and blue represents the road/not-road segmentation result. The white path, is the result from the topometric registration and route

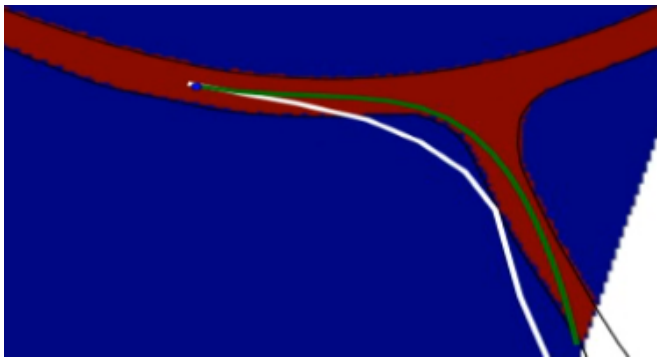

Fig. 3. An example result from the trajectory planner. The vehicle is located at the blue dot, and received the white path from the Route Planner to indicate a right turn. The red (road) and blue (off-road) road segmentation is used to plan the green trajectory which follows the high-level navigation goal while safely remaining on the road.

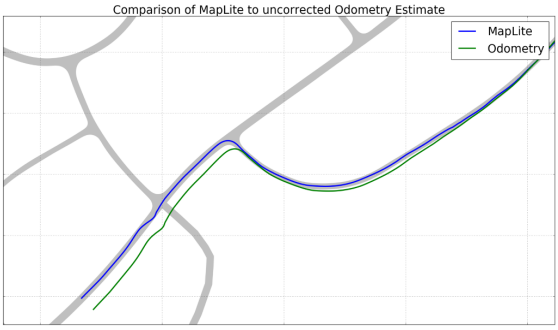

Fig. 4. A comparison between the trajectory autonomously driven by MapLite (blue) with the path estimated by odometry (green). The shaded area is the groundtruth road surface.

planning modules, which, while reliably providing navigation information (e.g. in this case "turn right") we cannot assume it to be precise at a metric level, since the topometric map does not contain such detail. Instead, the planner chose the green route, as it safely remains on the road, while also obeying the high-level navigation requirement.

The variational trajectory planner utilizes B-splines [36] to parameterize smooth trajectories. For a given reference trajectory r the planner completes a two-stage optimization process. The first stage *local goal-point selection* consists of determining the closest point to the high-level navigation goal that is both in the vehicle field of view and also lies on a road surface. The second stage *trajectory optimization* seeks an "optimal" path from the current vehicle location to the goal-point based on a number factors described shortly.

1) Local Goal Point Selection: The reference path is first clipped to the sensor range $\mathcal{X} \in \mathbb{R}^2$. Then, for a reference path r consisting of n waypoints, we denote the final point r_n. Note, although by construction r_n is in the sensor range, it may not lie on the road. To obtain the local goal point we define a cost function $J(x) = [d(x), d_e(x, r_n)]$ for $x \in \mathcal{X}$ where $d(x)$ is the signed distance function from x to the free space in $\mathcal{X}$ and $d_e(x, r_n)$ is the Euclidean distance from x to r_n. Then the goal-point is found by

$$x_g = \arg\min_{x \in \mathcal{X}} J(x) W_g^T$$

where W_g is a weight vector for tuning the relative importance of the cost function terms.

2) Trajectory Optimization: The second stage of the variation planner creates a B-spline trajectory that begins at the vehicle's location and ends at the goal-point x_g. This optimization utilizes a three part cost function composed of a road-distance cost

$$J_d(\boldsymbol{q}) = \frac{1}{k} \sum_{i=1}^{k} d(q_i)^2$$

where $\boldsymbol{q}$ is a candidate B-spline with k waypoints and $d(q_i)$ is, once again, the signed distance function to the road points. Next, in order to ensure that shorter paths are preferred to equally feasible longer ones, a relative-path-length cost is computed as

$$J_l(\boldsymbol{q}) = \frac{\sum_{i=2}^{k} d_e(q_i, q_{i-1})}{d_e(x_g, x_0)}$$

where x_0 is the current vehicle location. Finally, we also seek to minimize the maximum curvature of trajectories ride comfort. Since B-splines allow for analytic curvature calculation we impose a maximum curvature cost as

$$J_\kappa(\boldsymbol{q}) = \max \frac{\|\sigma' \times \sigma''\|}{\|\sigma'\|^3}$$

where σ is the B-spline representation of $\boldsymbol{q}$ and σ', σ'' are the first and second derivatives. Finally, the optimal trajectory is obtained using a numerical non-linear optimization routine based on [37] to solve

$$\boldsymbol{q}_{opt} = \arg\min_{\boldsymbol{q} \in \mathcal{X}} \left[J_d(\boldsymbol{q}), \quad J_l(\boldsymbol{q}), \quad J_\kappa(\boldsymbol{q}) \right] W_q^T$$

where W_q is a weight vector for tuning the relative importance of the cost function terms. Note that the roads utilized for testing in our rural testing environment are unmarked and often single-width. Therefore, the cost function prioritized driving down the center of the road. For use on roads with multiple lanes, we could easily extend this cost function to include a cost term based on lane boundaries instead.

IV. Performance Evaluation

A. Experiment Setup

1) Ground Truth: In order to train and evaluate these road segmentation methods, we utilized a Real-Time-Kinematic GPS Inertial Navigation System (Model: OXTS-RT3003) with a base-station transmitting Differential GPS corrections. The base-station provides groundtruth position and orientation with accuracy of $2cm$ with a range of $10km$.

Note that the RTK-GPS system was utilized only to obtain ground truth, and to provide an initial position when loading the map before the vehicle begins moving. While navigating, the vehicle only had access to odometry measurements, and corrected for drift entirely using the MapLite system. Fig. 4 shows an example trajectory that was driven autonomously. The blue path is the one chosen by MapLite, while the green is the path estimated by the odometry. Clearly, the drift in the

TABLE I
EVALUATION OF ROAD SURFACE SEGMENTATION METHODS

	Precision	Recall	F1	Accuracy	Runtime (s)
SparseConvNet	0.92	0.84	0.88	0.97	0.27
Linear SVM	0.91	0.84	0.87	0.96	0.19
PointNet	0.45	0.79	0.57	0.83	2.32

odometry would quickly cause the vehicle to deviate from the road without the MapLite corrections.

2) System Integration: We integrate the MapLite system described previously into a pipeline for fully autonomous navigation capable of operating the vehicle from an arbitrary starting point to a user specified goal location. To execute motion plans generated by the Variational Planner, we implement a pure pursuit controller [38], which enables us to choose a lookahead distance $d = 8m$ that was large enough to ensure rider comfort, while small enough to closely follow the reference path.

We set the vehicle speed using a dynamic speed controller such that it conforms to the following constraints: 1) Maximum Speed, 2) Maximum Linear Acceleration, 3) Maximum Linear Deceleration, and 4) Maximum Centripetal Acceleration. The speed controller first generates a predicted path by simulating the pure pursuit controller forward a fixed distance, and then analytically solves for the maximum speed that will remain within the constraints over the length of the predicted path.

The output of the Pure Pursuit and Dynamic Speed controllers are a command steering angle and velocity. In order to achieve those commands using robust closed-loop control, a pair of PID controllers was implemented. Each of these PID controllers was tuned to obtain fast responses to speed and steering commands while minimizing overshoot and steady state error.

B. Road Segmentation Results

We trained and evaluated three segmentation approaches: 1) the linear SVM, (section III-B), 2) A CNN based on SparseConvNet [23], and 3) another CNN based on PointNet [24]. Performing dense convolutions over sparse data is inefficient and computationally expensive due to fill-in. Both SparseConvNet and PointNet are specifically designed to achieve accuracy typical of CNN's on the sparse pointcloud data structure of lidar sensors.

To utilize the groundtruth system to evaluate the road segmentation methods, the vehicle was first driven manually along the border of each road included in the test set. This included approximately $10km$ of roads in rural Massachusetts. Next, the road boundary GPS traces were collected and used to create a georeferenced polygon of the road boundaries. Finally, at test time, the location of the vehicle at the time of each scan was used to project each scan into the georeferenced map. In this manner, the groundtruth class of each point was calculated to enable evaluation of each road segmentation method. Table I shows the results of evaluating the road segmentation methods on a set of 500 representative laser scans comprising over $55M$ segmented points. Notice that as expected, the SVM model is by far the fastest to segment a scan, although it does not perform quite as well as the SparseConvNet. The vastly larger number of parameters in the SparseConvNet network is also a consideration as that potentially allows it to learn a more complex variety of environments. However, that also makes it susceptible to overfitting. For our systems level evaluations presented in section IV, we chose to use the Linear SVM as its low runtime is paramount for enabling real-time operation at speed.

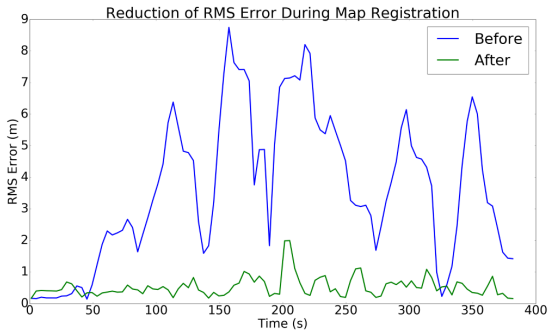

Fig. 5. A comparison between the RMSE of the estmated map before (Blue) and after (green) topometric registration.

1) Registration Evaluation: We evaluate the topometric registration by comparing the location of the road centerline in the registered topometric map, to the actual road center as measured by the groundtruth. We compute the Root-Mean-Square-Error (RMSE) along the portion of the map that is in the sensor range (as the topometric registration only considers that portion of the map. As a baseline, we also compute the RMSE using the prior to map registration (e.g. based solely on the odometry estimate).

Fig. 5 shows RMSE before (blue) and after (green) topometric registration computed during $380s$ of driving. As expected the error based solely on odometry starts out quite low. However, due to the inherent drift in dead-reckoning measurements, it quickly drifts to $> 5m$ RMSE which is much too large to use for autonomous operation. The maps corrected by topometric registration on the other hand, have a consistently smaller RMSE which reduces by 85.7% on average. Furthermore, it reduces the maximum RMSE by 79.7%. The corrected maps based on the registration alone are not sufficient for blindly following the topmetric map because the topometric map does not contain the detailed lane-level metric information needed for autonomous driving. However, these corrected maps are sufficient to generate a reference path that incorporates the high-level navigation information into the motion planner which generates the actual control trajectory. This incorporation of the reference path from the topometric registration is described in section III-E.

C. System Evaluation Results

We compare the MapLite system against that of an expert human driver to quantify performance metrics. First we chose a standardized test route of $1.2km$ in length requiring the traversal of a total of ten intersections (3 right-turns, 4 left-turns, and 3 straight). The test route was in a rural area near

Fig. 6. The vehicle at the test site in rural Massachusetts. The heavily wooded area has roads without markings or boundaries.

TABLE II
MAPLITE PERFORMANCE EVALUATION AND ABLATION STUDY

	Precision(m)	Accuracy(m)	Intervention
Human Expert	0.31	0.43	0
MapLite	0.17	0.49	0
1) Map Registration	0.34	0.81	0
2) Variational Planner	0.28	0.81	1.3
3) 1 & 2	-	-	>10
4) Smart Replanning	0.55	0.57	0

Devens, Massachusetts with overgrown, unmarked, mostly single-lane roads. (See Fig. 6)

Next, we utilize the GPS groundtruth to measure a baseline traversal of the test route. Finally, both the human driver and MapLite steer the vehicle along the test route twice. For these runs we compute two metrics: 1) Accuracy and 2) Precision. The accuracy metric is the RMSE of the test run compared to the baseline. The precision metric is the root mean square deviation (RMSD) of the second run compared to the first. This gives a measure of the repeatability of the system independent of how closely it navigates to the baseline.

We found that the expert human driver could navigate the test route with an Accuracy of $RMSE = 43cm$ while the MapLite system obtained $RMSE = 49cm$. While this was not quite as close to the baseline as the human driver, it is quite close considering the $6cm$ is quite a small difference with respect to the size of typical lanes. Perhaps surprisingly, when it came to precision, the MapLite system outperformed the human driver with $RMSD$ of only $17cm$ while the human had $31cm$. We attribute this to the unique ability of the autonomous system to precisely reproduce the same trajectory, while human drivers typically have more variability even when navigating safely.

Next, to exhaustively test the system, we also randomly selected GPS destinations and for each one, MapLite autonomously planned a route and piloted the vehicle from its starting position to the destination. Once the vehicle came to a stop, a new destination was input, and the process repeated. We executed this test for a total of 16.1km consisting of 107 intersection traversals. At no point throughout this test was safety driver intervention required, and the vehicle navigated safely to every one of the destination points.

Finally, to determine the value of each of the components described previously, we performed an ablation test wherein we removed a single component and computed the accuracy and precision measurements described previously by traversing the test route twice more. We did this for each of the following four ablation scenarios:

1) We remove the Topometric Registration component leaving the map fixed
2) We remove the Variational Planner, instead directly following the reference from the topometric map
3) We combine both 1) and 2)
4) We remove the Smart Replanning, instead recomputing a new route plan each time the map is updated

For each test, we also recorded the required interventions/km by the safety driver to prevent road departure. Table II compares the performance of the Human driver, MapLite, and each of the ablation tests. For 1) Removal of Topometric Registration, no interventions were needed. However, the performance of the system was worse in both accuracy and precision. While it might seem surprising that the system can operate at all with a fixed topometric map, it should be noted that in this system, the map is used mostly as a source of high-level navigation information, while the local trajectory plan is computed by the Variational Planner.

Next, for 2) removal of the Variational Planner the driver was required to intervene $1.3 interventions/km$. Furthermore, both the accuracy and precision metrics increased. Next, for 3) both the Topometric Registration and Variational Planner were removed. In this case however, the vehicle could not safely navigate autonomously, and the test was aborted after 10 interventions. This is interesting because it provides evidence that while the Topometric Registration and Variational Planner are supplementary (the vehicle performed best when they were both used) they are also complementary (each of them alone could enable at least some measure of autonomous navigation).

Finally, in 4) we remove the Smart Replanning feature of the route planner. In that test again, there were no interventions required. However, both the precision and accuracy were dramatically worse. This can be explained by the increased processing time required to recompute the route plan each time the map was updated, which caused a lag between when new sensor data arrived and when the vehicle was able to respond.

D. Simulations

While the testing location in Massachusetts provided a range of intersection topologies in which to test MapLite, we leveraged autonomous driving simulation to assess the performance of MapLite in more complex traffic topologies that were not represented at the testing site. We chose to use the CARLA [8] autonomous driving simulator for our analysis because it includes realistic environments with a larger diversity of road topologies. In particular, we selected the built-in Town03 urban environment in CARLA for testing, shown in Fig. 7, which includes a multi-lane traffic circle with multi-lane incoming roads.

We assessed the performance of MapLite in this environment in the same manner as we did the real-world vehicle.

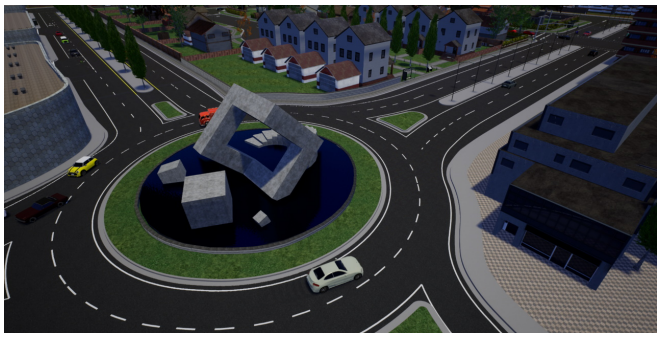

Fig. 7. A traffic circle in the CARLA simulator. We used CARLA to test more complex road topologies than were available at our test site.

In this case, our test route included both entering and exiting the traffic circle, and the groundtruth GPS measurement came from the simulator itself. Using this measurement as the baseline, we obtained a precision $RMSD = 10cm$ and accuracy $RMSE = 38cm$. Both of these values are actually lower than the corresponding values for the real-world vehicle (see Table II). However, while the simulation does include random noise, and, in fact, when the vehicle navigates solely based on odometry it quickly drifts from the road, the simulation does not accurately model all of the various noise sources in a real vehicle, and thus its better precision and accuracy are not unexpected.

V. Conclusion

We have demonstrated a novel autonomous navigation system capable of using publicly sourced OpenStreetMap maps combined with onboard sensors for autonomous navigation on rural roads without detailed prior maps or GPS localization. This greatly increases the areas in which autonomous driving systems can be deployed. The presented system does not assume the publicly sourced maps to be perfectly accurate at a metric level. While the present work has focused on the driving task, future work will investigate how autonomous vehicles can be used to automatically update topometric maps.

References

[1] "Trading economics: Paved roads in the united states (as share of total roads)," http://www.tradingeconomics.com/united-states/roads-paved-percent-of-total-roads-wb-data.html, accessed: 2019-09-10.
[2] A. Amini, G. Rosman, S. Karaman, and D. Rus, "Variational end-to-end navigation and localization," *arXiv:1811.10119*, 2018.
[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv:1604.07316*, 2016.
[4] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *ICRA*, 2018.
[5] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *ICRA*, 2019.
[6] M. Hentschel and B. Wagner, "Autonomous robot navigation based on OpenStreetMap geodata," in *ITSC*, 2010.
[7] T. Ort, L. Paull, and D. Rus, "Autonomous vehicle navigation in rural environments without detailed prior maps," in *ICRA*, 2018.
[8] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *CoRL*, 2017.
[9] J. Levinson and S. Thrun, "Map-based precision vehicle localization in urban environments," in *RSS*, 2007.
[10] ——, "Robust vehicle localization in urban environments using probabilistic maps," in *ICRA*, 2010.
[11] R. W. Wolcott and R. M. Eustice, "Fast lidar localization using multiresolution gaussian mixture maps," in *ICRA*, 2015.
[12] ——, "Visual localization within lidar maps for automated urban driving," in *IROS*, 2014.
[13] A. Broggi and S. Bertè, "Vision-based road detection in automotive systems: A real-time expectation-driven approach," *JAIR*, vol. 3, pp. 325–348, 1995.
[14] G. Zhang, N. Zheng, C. Cui, Y. Yan, and Z. Yuan, "An efficient road detection method in noisy urban environment," in *IV*, 2009.
[15] J. M. Alvarez, A. Lopez, and R. Baldrich, "Illuminant-invariant model-based road segmentation," in *IV*, 2008.
[16] W. Liu, H. Zhang, B. Duan, H. Yuan, and H. Zhao, "Vision-based real-time lane marking detection and tracking," in *ITSC*, 2008.
[17] M. Aly, "Real time detection of lane markers in urban streets," in *IV*, 2008.
[18] Y. W. Seo and R. R. Rajkumar, "Detection and tracking of boundary of unmarked roads," in *FUSION*, 2014.
[19] J. Zhang and H.-H. Nagel, "Texture-based segmentation of road images," in *IV*, 1994.
[20] L. Caltagirone, S. Scheidegger, L. Svensson, and M. Wahde, "Fast lidar-based road detection using fully convolutional neural networks," in *IV*, 2017.
[21] Z. Chen, J. Zhang, and D. Tao, "Progressive lidar adaptation for road detection," *Journal of Automatica Sinica*, vol. 6, no. 3, pp. 693–702, 2019.
[22] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "Lidar–camera fusion for road detection using fully convolutional neural networks," *Robotics and Autonomous Systems*, vol. 111, pp. 125–131, 2019.
[23] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016.
[24] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.
[25] M. M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, Oct. 2008.
[26] A. L. Ballardini, S. Fontana, A. Furlan, D. Limongi, and D. G. Sorrenti, "A framework for outdoor urban environment estimation," in *ITSC*, 2015.
[27] A. L. Ballardini, D. Cattaneo, S. Fontana, and D. G. Sorrenti, "Leveraging the osm building data to enhance the localization of an urban vehicle," in *ITSC*, 2016.
[28] ——, "An online probabilistic road intersection detector," in *ICRA*, 2017.
[29] G. Floros, B. Van Der Zander, and B. Leibe, "Openstreetslam: Global vehicle localization using openstreetmaps," in *ICRA*, 2013.
[30] R. B. Langley, "The utm grid system," *GPS world*, vol. 9, no. 2, pp. 46–50, 1998.
[31] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments," in *RSS*, 2007.
[32] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *VIS*, 2002.
[33] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *IAS-13*, July 2014.
[34] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *SSC*, vol. 4, no. 2, pp. 100–107, 1968.
[35] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, *et al.*, "Making bertha drive—an autonomous journey on a historic route," *ITSM*, vol. 6, no. 2, pp. 8–20, 2014.
[36] C. De Boor, "On calculating with b-splines," *Journal of Approximation theory*, vol. 6, no. 1, pp. 50–62, 1972.
[37] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The computer journal*, vol. 7, no. 2, pp. 155–162, 1964.
[38] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," CMU Robotics Inst., Tech. Rep., 1992.