Generation of Accurate Lane-Level Maps from Coarse Prior Maps and Lidar

Avdhut Joshi¹ and Michael R. James¹

Abstract—While many research projects on autonomous driving and advanced driver support systems make heavy use of highly accurate lane-level maps covering large areas, there is relatively little work on methods for automatically generating such maps. Here, we present a method that combines coarse, inaccurate prior maps from OpenStreetMap (OSM) with local sensor information from 3D Lidar and a positioning system. The algorithm leverages the coarse structural information present in OSM, and integrates it with the highly accurate local sensor measurements. The resulting maps have extremely good alignment with manually constructed baseline maps generated for autonomous driving experiments.

I. INTRODUCTION

Many existing approaches [1], [2] for autonomous driving systems make heavy use of maps that encode lane-level information at high levels of precision. The lane-level information is used in a variety of situations, from generating smooth trajectories for path planning [1], to predicting the behavior of other vehicles [3], [4], and for planning and reasoning about proper behavior in intersections [5]. In many cases, such maps are generated either through a tedious manual annotation process [2], or by driving the exact lane layout with a test vehicle [6] or by analyzing a collection of GPS tracks [7]. These methods require significant amounts of manual work, either through annotation or in the amount of data collection required. In this paper, we present a method for overcoming these limitations, which opens the door for creating more robust systems with the ability to create their own high-fidelity maps with less manual work.

While there has been extensive work in lane detection [8], [9], [10], [11] and lane tracking [12], [13] using a variety of sensors, we are unaware of any previous work that combines coarse prior information with sensor data to result in consistent, high-quality, large-scale maps such as we generate.

This paper presents a method for estimating the structure and layout of lanes within real-world road scenes, by combining

1) structurally informative, easily obtained, coarse maps used as prior information from the Open Street Map (OSM) [14] project, with

2) sensor data obtained from a test vehicle with 3D Lidar as well as a high-precision positioning system.

We are typically able to infer lane structure for an entire road by driving the road once, and not once for each lane.

Fig. 1: TRI-NA test vehicle, showing the sensors used for various advanced safety and autonomous driving research projects. Details of the sensors used for lane mapping are in the text.

This method is robust to differing styles of driving of the test vehicle, as the algorithm is based only on road-paint detection using intensity returns from the Lidar, and does not use the path of the test vehicle in lane estimation. Specifically, we contribute the following:

• Modeling of the inference problem that combines coarse structural prior map data with precise Lidar measurements in a (number of) tractable inference algorithms
• Algorithm for MAP inference of lane positions and identity management
• Evaluation on a real-world dataset

II. APPROACH

The lane estimation algorithm is based on a dataset that has first been refined using Slam in order to ensure consistent position estimates on loop closures. In Section III, we provide an overview of this algorithm; a variant of GraphSlam [15] run on datasets gathered from multiple runs on only partially overlapping roads, see Figure 2. This serves to both align the laser scans, as well as to ground the data in a consistent and physically meaningful reference frame. However, our main contribution, the lane estimation algorithm, can be run on any dataset that has been refined using this or similar methods.

The lane estimation algorithm Section IV is comprised of two phases, the first of which is the generation of mid-level lane features, which then serve as observations for the measurement function within the second phase which uses particle filtering to estimate the lanes. We experiment with
three variations of particle filters, a conventional approach, a dual formulation, and a mixture of the two. We also extend the mixture PF to leverage prior map information.

Our test vehicle (Figure 1) has a very similar hardware setup to many other autonomous vehicle projects [15], [1], [16]. The pose and motion estimates come from an Applanix POS-LV 220 inertial GPS navigation system. This system generates pose estimates at 100 Hz. The Lidar data is from a Velodyne HDL-64E, which uses 64 laser beams and spins at 10 Hz. In addition to 3D position for each Lidar return, the HDL-64E also measures an 8 bit intensity. Appropriate subsets of each Lidar spin are timestamped to the Applanix data, so that refinement of the vehicle poses using Slam will also result in refined Lidar positions, as is outlined next.

III. GRAPHSLAM

In order to generate a consistent dataset on which to run the lane estimation algorithm, we first apply a variant of GraphSlam presented in [15], but modified to optimize the vehicle trajectory with respect to a number of distinctive features that can be readily identified from Lidar data. The features include lane markers/paint, curbs, road signs, and poles. The features corresponding to lane markers are of special interest to us, as they are our primary sensor data used for lane estimation. The construction of this feature is done first by applying a 1D Laplacian filter to the intensity (reflectivity) returns for each beam of the 3D laser. This gives a response due to road-paint having different reflectivity characteristics than the road surface. Then a RANSAC algorithm is applied using the 3D positional data associated with each response, to remove outliers and to generate line-segment features \( \lambda \). See Figure 3 (Left).

The Slam algorithm iterates between a data association step and an optimization step. The data association step uses a threshold based on distance. On each data association step, new positions are calculated for each feature based on the new vehicle positions generated from the optimization step. The threshold is increasingly tightened, and this process is repeated until convergence. The output of the algorithm is a refined vehicle trajectory, which is subsequently used to re-project the Lidar scans.

IV. PROBABILISTIC LANE ESTIMATION

Both the lane marker estimation and the particle filtering portions of the approach make use of our weak prior information, in the form of an OSM map. This information is limited in that it first has large positional inaccuracy (up to few meters for our test scenario), and second it does not contain data about the number of lanes. Thus we have to leverage significant local data from our Lidar scans to infer precise positional estimates of all lanes in the scene. Our approach to particle filtering is unconventional in that the state of each particle is tied to nodes within the OSM map, rather than being defined in an Euclidean space. Similarly, the OSM map defines the subset of local sensor data to be processed at each step, as explained next.

A. Feature-based Lane Marker Estimation

We use OSM map data as a weak prior on position and existence of a road. A way in OSM is defined by \( \{p_1, p_2, ..., p_n\} \): a set of \( n \) nodes along the way. These nodes are evenly spaced at \( \delta_x = 1 \) meter intervals. The lane marker estimation process uses the same lane marker/paint features that were used in the GraphSlam algorithm from Section III, which take the form of relatively short (0.5 to 2 meter) line segments \( \lambda \) specified by 3d coordinates of both endpoints. For each OSM node \( p_i \), first the angle \( \theta_i \) to node \( p_{i+1} \) is computed, and then a search is performed to find the line markers lateral to the OSM node. The search is over a rectangle defined by the vectors of length \( \delta_x/2 \) forward and backward along \( \theta_i \), and for a fixed distance in both directions perpendicular to \( \theta_i \). All line segments falling within this rectangle are collected into a set \( \Lambda_i = \{\lambda_j\} \). This set is then filtered based on each segments’ alignment to \( \theta_i \), resulting in \( \Lambda_i = \{\lambda_j : \lambda_j \in \Lambda_i, ||\theta_{\lambda_j} - \theta_i|| < \theta_{\text{thresh}}\} \), where \( \theta_{\lambda_j} \) is the angle of the line segment. We then cluster all the line segments in \( \lambda \) using a greedy approach based on separation distance, and call the resulting clusters lane markers. For each lane marker, we compute the mean offset distance \( z_i \) from the OSM node \( p_i \). This offset distance will be used as an observation tied to this particular OSM node within the particle filter. See Figure 3 (Middle).

Next, we group the lane markers longitudinally, using a greedy, flood-fill algorithm in the longitudinal direction. The

---

1 Thanks to Masahiro Harada (TRI-NA) for use of his SLAM implementation and feature detectors.
purpose of this step is to generate additional binary features for each lane marker. Some groups of lane markers, such as those corresponding to solid, well-painted lines, will extend for a long longitudinal distance (tens or hundreds of meters) on rural or semi-rural roads, while in other cases such as dashed lines, or areas with many intersections, the groups will be short, on the order of meters.

Given these groupings, three additional features are computed which prove to be useful for lane estimation. First we calculate two binary features which encode on which side(s) of the lane marker a lane can exist (e.g. for a right most lane marking, a lane on the right cannot exist). We compute these binary features namely, \( has-l \) and \( has-r \) by looking at the entire lane segment data. For the entire lane segment, we count the number of lane marking observations \( z_n \) that lie on the either side \( (c^l_k \geq \delta_c, c^r_k) \). Then,

\[
\text{has-}j = (c^j_k \geq \delta_c), j \in \{l, r\}
\]

where \( \delta_c \) is a threshold parameter. The third binary variable encodes whether a lane marker is dashed. We first filter out all the lanes which are bigger than a standard dashed lane found in US. Then we connect lane marker groups which are at a set distance apart and have similar orientation. These are marked as a dashed.

The above binary features illustrated in Figure 3 (Right), give important cues to interpreting the lane data, as will be shown in the development of the measurement function for the particle filters described in Section IV-C

B. Particle Filtering

We have experimented with multiple approaches for particle filtering for this domain. In the following sections, we outline these approaches starting here with some basic definitions that are common to all. As noted above, the particle filter evolution is based on the structure of the OSM nodes, with successive steps in the filter transitioning from one OSM node to the next. The state of each particle is based on its relation to the OSM node (which then ties it to a physical location). With this in mind, we now derive our filtering equations starting from a basic definition of the state of the map that we want to estimate:

\[
X_n : \{x_1^n, x_2^n, \ldots, x_m^n\};
\]

where \( m \) is number of lanes estimated at \( n^{th} \) node in OSM way and \( x_i^n \) is state of the lane estimate. The state of each lane is its offset from the OSM node and its width \( \{o_i^n, w_i^n\} \).

Using the observations \( z_n \rightarrow \{ \text{Lane markers observed at } n^{th} \text{ OSM node} \} \) from Section IV-A, our belief state is

\[
\text{Bel}(x_n) = p(x_n | z_n, z_{n-1}, \ldots, z_0)
\]

Using recursive Bayes filtering as defined in [17] for equation (1) we have

\[
\text{Bel}(x_n) \propto p(z_n | x_n) \int p(x_n | x_{n-1}) \text{Bel}(x_{n-1}) dx_{n-1}
\]

To implement a particle filter, we need to estimate the quantities \( p(z_n | x_n) \) and \( p(x_n | x_{n-1}) \text{Bel}(x_{n-1}) \). For all algorithms, we represent \( \text{Bel}(x_n) \) as a set of \( m \) weighted particles

\[
\text{Bel}(x) \approx \{x(i), \phi(i)\}_{i=1,\ldots,m}
\]

where \( x(i) \) is a sample of state (lane estimate) and \( \phi(i) \) is a non-negative parameter called the importance factor or weight. The other necessary quantities are described in depth in each of the following sections.

C. Conventional Particle Filter

Our implementation of the conventional particle filter follows the following three steps:

1) Sampling: Sample \( x_{n-1}^{(i)} \sim \text{Bel}(x_{n-1}) \) from the weighted sample set representing \( \text{Bel}(x_{n-1}) \).
2) Proposal Distribution: We sample \( x_n^{(i)} \sim p(x_n | x_{n-1}^{(i)}) \)

Since the particle state only evolves in relation to OSM nodes, and OSM maps are highly inaccurate in both position and direction, we sample \( x_n \) by adding Gaussian noise to \( x_{n-1}^{(i)} \).

\[
x_n^{(i)} : \{o_{n-1} + N(0, \sigma_o), w_{n-1}^{(i)} + N(0, \sigma_w)\}
\]
and so the filter fails to capture narrow bike lane on the top. The number of new proposed particles may be added each step, but importance. To make this approach tractable, only a limited number of new proposed particles may be added each step, and so the filter fails to capture narrow bike lane on the top.

Now pair \((x_n^{(i)}, o_n^{(i)})\) is distributed according to

\[ p(x_n|x_{n-1})Bel(x_{n-1}) \]

3) Update Function: We update the weight of each sample according to following distribution.

\[ \phi_n^{(i)} = p(z_n|x_n^{(i)}) \]

\[ z_n : \{l_1, l_2, \ldots l_k\}, \] where \(l_j\) are lane markers observed at \(n^{th}\) node.

a) For each \(x_n^{(i)}\), perform data association with lane observations, i.e. determine associated lane markings for \(x_n^{(i)}\).

b) Compute new observed lane offset and lane width from the observations \(\{o_n^{(i)}, w_n^{(i)}\}\).

c) Compute \(\phi_n^{(i)}\) using following equation

\[ \phi_n^{(i)} = \frac{1}{2\sigma_o} e^{-\frac{\phi (x_n^{(i)} - o_n^{(i)})^2}{2\sigma_o^2}} \cdot \frac{1}{2\sigma_w} e^{-\frac{\phi (x_n^{(i)} - w_n^{(i)})^2}{2\sigma_w^2}} \]

where \(\sigma_o\) and \(\sigma_w\) are parameters selected to fit typical standard deviations on width and location based on our data.

During the data association, we check for the appropriate binary variable has-l and has-r and remove ambiguous data associations. (e.g. if the state of a particle is to the left of the left most lane, then it is not associated with the any lane markings). If the above data association fails, we penalize the \(\phi_n^{(i)}\) by a penalty factor \(\gamma\). We relax this penalty factor if dashed lane markings are present as we expect them to be missing periodically.

In order to recover from lane additions or exits in natural road environment, we extend the sampling scheme stated above. We introduce a new parameter \(t\) which is percentage of new particles introduced in the system on every update. Hence we sample according to Algorithm 1, where \(\mu_w\) is the expected lane width. Note that selecting large standard deviations means that a large number of new proposed particles (and corresponding computational cost) are required to sufficiently cover the state space. Further, having large standard deviations increases the chance of proposing an erroneous particle that matches noise in the data.

Figure 4 illustrates the output of Regular Particle Filter estimating lanes at one of the OSM nodes.

**Algorithm 1:** Modified Re-sampling algorithm

**Input:** \(m \rightarrow \) number of particles

**Input:** \(t \rightarrow \) percentage of new particles

for \(i = 1; m \ast (1 - t)\) do

Sample \(x_{n-1}^{(i)} \sim Bel(x_{n-1})\);

Add \(x_{n-1}^{(i)} \rightarrow Bel(x_{n-1})\);

end

for \(i = 1; m \ast t\) do

Generate new state \(x_{n-1}^{(i)} : \{N(0, \sigma_o)N(\mu_w, \sigma_w)\}\);

Set \(\phi_n^{(i)} : \epsilon\);

Add \(x_{n-1}^{(i)} \rightarrow Bel(x_{n-1})\);

end

Replace Bel\((x_{n-1})\) with \(\hat{Bel}(x_{n-1})\);

**D. Dual Particle Filter**

One major limitation observed when applying the conventional particle filter is its failure to capture lanes with abnormal specifications (like biking lanes or extra wide
ramps) as shown in Figure 4. While this could be addressed by increasing the standard deviation of new particles, this solution is suboptimal for reasons discussed above. We will now describe a dual method in order to tackle this problem formally. In the dual configuration, we reverse the role of proposal distribution and measurement function as stated above. At every iteration we sample new particles based on their agreement with the observations

\[ x_n^{(i)} \sim p(z_n|x_n) \]

and importance factors are set using

\[ \phi_n^{(i)} = \int p(x_n^{(i)}|x_n^{(i)}) Bel(x_{n-1}) dx_{n-1} \]

The algorithm is then:

1) Proposal Distribution: We propose new particles based on the observations. Let \( z_n : \{l_1, \ldots, l_k\} \) be \( k \) lane markers observed at \( n^{th} \) OSM node, sorted by location. We uniformly select \( j \in \{1, \ldots, k\} \) and propose

\[ x_n^{(i)} : \{l_j + l_{j+1}/2 + N(0, \sigma_u); (l_{j+1} - l_j) + N(0, \sigma_w)\} \]

2) Update Function: Importance factors for each particle are then corrected using prior belief \( Bel(x_{n-1}) \). To approximate this distribution over the continuous state space, we take a kernel density approach. We first generate \( m \) samples as done for the proposal distribution in a conventional particle filter.

\[ \hat{x}_n^{(i)} \sim p(x_n|x_{n-1}) Bel(x_{n-1}) \]

Writing \( h(\{x_n\}; x) \) to denote the parameterized kernel density function approximating this distribution, the importance factor for each particle is given by

\[ \phi_n^{(i)} = h(\{x_n\}; x_n^{(i)}) \]

As shown in Figure 5, the Dual Particle Filter is able to estimate non-standard bike lane which the Conventional Particle Filter failed to capture.

E. Mixture Particle Filter

While the pure Dual Particle Filter is able to capture abnormal lane specifications, it will fail in the situation where new lanes are added. Proposed particles for new lanes cannot be matched to any in the previous distribution, thus getting essentially zero weight. The approach described in [17] fixes this problem using a combination of both Conventional and Dual Particle Filter. In the Mixture approach, we use a variable mixture ratio \( \theta(0 \leq \theta \leq 1) \) and sample from the Conventional method with probability \( 1 - \theta \) and with probability \( \theta \) using the Dual.

Additionally, the Mixture Particle Filter allows for more flexible modeling based on situational information. For instance we can vary the mixture ratio \( \theta \) based on structural information from the OSM map. Specifically, we reduce the ratio closer to intersections where performance of Dual is significantly bad due to the lack of lane markings. Variations on this theme, and whether such dependencies can be learned, are an interesting source of future work.

F. Clustering and Lane Indexing

Our generated map will have only a finite number of lanes, each with a single lane position and width estimate for each OSM node. Further, these lanes should be linked over iterations using IDs. This requires one further processing step. The particle filters above result in a discrete approximation of \( Bel(x_n|z_n) \) represented by a set of particles. This distribution can be observed in Figure 4. This distribution is multi-modal and number of modes are unknown apriori. We use a EM-based weighted clustering algorithm on the distribution to find the maximum-a-posteriori modes. These cluster centers are final lane estimates. This clustering is done in the space of \( x \) (i.e. on both offset and width).

To generate temporal links between iterations, we assign an index to each cluster using Algorithm 2.

\[
\text{Algorithm 2: Cluster Indexing algorithm}
\]

The parameter \( \theta \) of the mixture case. The number of missed estimates is significantly bad due to the lack of lane markings. Variations on this theme, and whether such dependencies can be learned, are an interesting source of future work.

V. Results

To evaluate the accuracy of our approach, we compare our lane estimates with hand labeled road network data for 28 km of road. Our hand-labeled road network data consists of rural roads, urban roads and freeways which is shown in Figure 2, but unfortunately, this does not include bicycle lanes. For all our experiments, we set the number of particles to 5000 and for mixture case, we had mixture ratio set to 0.3. As discussed above, the mixture ratio is set to zero near intersections, i.e. we relied only on regular particle filter in that case. Figure 6 illustrates qualitative results of lane estimates for each type of scene. We did not evaluate the dual approach on its own.

We evaluate our results using two metrics, mean positional error and number of nodes for which we incorrectly estimated a lane centers. Quantitative results are shown in Table I. These results show that both the regular and mixture approaches generate highly accurate lane-level maps, with the mixture approach being slightly better in some cases.

For the entire dataset, we were not able to associate our estimates with hand labeled data for 2.3% of the estimated lane center nodes for the regular particle filter, and for 6.2% for the mixture case. The number of missed estimates is higher for the mixture case as we do not have hand labeled
Fig. 6: Qualitative Results. Left two figures show lane estimation on country roads. Even though there is missing data at intersections, we are able to track lanes successfully. Third figure shows lane estimation on highway with dashed lanes. Last figure illustrates our results on multi-lane urban roads. Note that for all those roads, we did not have any prior information about number of lanes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean Error(m)</th>
<th>Max Error(m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Particle Filtering - urban</td>
<td>0.06</td>
<td>0.38</td>
</tr>
<tr>
<td>Regular Particle Filtering - highway</td>
<td>0.05</td>
<td>0.13</td>
</tr>
<tr>
<td>Regular Particle Filtering - all data</td>
<td>0.06</td>
<td>0.38</td>
</tr>
<tr>
<td>Mixture Particle Filtering - urban</td>
<td>0.06</td>
<td>0.22</td>
</tr>
<tr>
<td>Mixture Particle Filtering - highway</td>
<td>0.04</td>
<td>0.08</td>
</tr>
<tr>
<td>Mixture Particle Filtering - all data</td>
<td>0.05</td>
<td>0.22</td>
</tr>
</tbody>
</table>

TABLE I: Quantitative Results

data for bike lanes for which mixture particle filter is able to guess lane estimates correctly, in other words, the more flexible modeling capability of the mixture approach actually hurts it in this metric. Analysis of the locations where errors occur, indicates that errors mainly stem from noisy data at intersections where lanes markings are missing.

VI. CONCLUSIONS

In this work, we have shown how structural priors can be leveraged in a real-world outdoor mapping task requiring, and compared against, accurate lane-level maps. Our results are encouraging, this approach is able to generate maps that agree with hand-made maps to a high level of accuracy. Application of this work will allow our system to generate accurate large-scale lane-level maps suitable for advanced driver support and autonomous driving applications.

Additionally, we find the general approach of combining structural priors with accurate local information extremely interesting and are working on a number of ideas for extending it including:

- Improve the use of prior information in intersection handling
- Learning a model of how and when to modify inference algorithm based on situations
- Develop a method for detecting and correcting maps when world changes
- Use this approach in an on-line algorithm

REFERENCES