Abstract

Knowledge of the road network topology is crucial for autonomous planning and navigation. Yet, recovering such topology from a single image has only been explored in part. Furthermore, it needs to refer to the ground plane, where also the driving actions are taken. This paper aims at extracting the local road network topology, directly in the bird’s-eye-view (BEV), all in a complex urban setting. The only input consists of a single onboard, forward looking camera image. We represent the road topology using a set of directed lane curves and their interactions, which are captured using their intersection points. To better capture topology, we introduce the concept of minimal cycles and their covers. A minimal cycle is the smallest cycle formed by the directed curve segments (between two intersections). The cover is a set of curves whose segments are involved in forming a minimal cycle. We first show that the covers suffice to uniquely represent the road topology. The covers are then used to supervise deep neural networks, along with the lane curve supervision. These learn to predict the road topology from a single input image. The results on the NuScenes and Argoverse benchmarks are significantly better than those obtained with baselines. Code: https://github.com/ybarancan/TopologicalLaneGraph.

1. Introduction

How would you give directions to a driver? One of the most intuitive ways is by stating turns, instead of distances. For example, taking the third right turn is more intuitive and robust than going straight for 100 meters and turn right. This observation motivates us to model road networks using the involved lanes and their intersections. We model the lane intersections ordered in the direction of traffic. Given a reference centerline $L$ and two lines $I_1, I_2$ intersecting $L$, we consider the set of all possible lines intersecting $L$ between the intersection points $L - I_1$ and $L - I_2$ as an equivalence class. Such modelling allows us to supervise the learning process explicitly using the topological structure of the road network. In turn, the topological consistency during inference is improved. Consider a car moving from the green point $P$ upwards in Fig 1, which needs to take the first left turn. In the two estimates of [6], the first left leads to different lanes. While the underlying directed graphs have the same connectivity in all estimates, they have very different topological structures which play an important role in decision making.

For autonomous driving, the information contained in the local road network surrounding the car is vital for the decision making of the autonomous system. The local road network is both used to predict the motion of other agents [15, 23, 36, 45] as well as to plan ego-motion [3, 12]. The most popular approach to represent the road network is in terms of lane graph based HD-maps, which contain both the information about the centerlines and their connectivity. Most existing methods address the problem of road network extraction by using offline generated HD-maps in combination with a modular perception stack [10, 24, 29, 35, 39]. However, offline HD-maps based solutions have two major issues: (i) dependency on the precise localization in the HD-map [29, 43], (ii) requirement to construct and maintain such maps. These requirements severely limit the scalability of autonomous driving to operate in geographically restricted areas. To avoid offline mapping [6] proposed to directly estimate the local road network online from just one onboard image. Inspired by this approach and given the importance of topological consistency for graph based maps we propose to directly supervise the map generation network to estimate topological consistent road networks.
Figure 2. The road networks and the resulting closed regions (minimal cycles, shown in different colors). In the road network, traffic flows from green to red dots and the yellow dots are connection points of two centerlines. Proposed formulation learns to preserve the identities of the centerlines enclosing the minimal cycles. Some interesting regions are shown by white circles on colored images.

Starting from [6], we represent the local road network using a set of Bezier curves. Each curve represents a driving lane, which is directed along the traffic flow with the help of starting and end points. However, compared to [6] we also consider the topology of the road network, which is modeled by these directed curves and their intersections. More precisely, we rely on the following statement: if the intersection order of every lane, with the other lanes, remains the same, the topology of the road network is preserved. Directly estimating the order of intersections is a hard problem. Therefore, we introduce the concept of minimal cycles and their covers. A minimal cycle is the smallest cycle formed by the directed curve segments (between two intersections). In Fig. 2, each minimal cycle is represented by different colors. The cover is the set of curves whose segments are involved in forming a minimal cycle. Given two sets of curves (one estimation and one ground truth), if all curves in both sets intersect in the same order, then the topology defined by both sets are equivalent. In this work, we show that such equivalence can be measured by comparing the covers of the minimal cycles. Based on our findings, we supervise deep neural networks that learn to predict topology preserving road networks from a single image. To our knowledge, this is the first work studying the problem of estimating topology of a road network, thus going beyond the traditional lane graphs.

Our model predicts lane curves and how they connect as well as the minimal cycles with their covers. During the learning process, both curves and cycles are jointly supervised. The curve supervision is performed by matching predicted curves to those of ground truth by means of the Hungarian algorithm. Similarly, the cycle supervision matches the cover of the predicted cycles to those of the ground truth. Such joint supervision encourages our model to predict accurate and topology preserving road networks during inference, without requiring the branch for cycle supervision.

For online mapping in autonomous driving and other robotic applications, it is crucial to directly predict the road network in the Birds-Eye-View (BEV) since the action space of the autonomous vehicles is the ground. This stands in contrast to traditional scene understanding which mainly takes place in the image plane. It was recently shown that performing BEV scene understanding in the image plane, and then project it to the ground plane is inferior to directly predicting the output on the ground plane [7, 9, 33, 34, 38, 42, 43]. Compared to our approach these methods do not provide the local road network, but a segmentation on the BEV. Note that the road regions alone do not provide the desired topological information. Similar methods that perform lane detection are limited to highway-like roads where the topology of the lanes is trivial, and are not able to predict road networks in urban scenarios with intersections which is the setting we are interested. As said [6] is able to predict such road networks but does not consider the topology, and we will show that our topological reasoning can improve the methods proposed in [6].

Our predicted connected curves provide us a full lane graph HD map. To this end, our major contributions can be summarized as follows.

1. We propose a novel formulation for the topology of a road network, which is complementary to the classic lane graph approach
2. We show that a neural network can be trained end-to-end to produce topologically accurate lane graphs from a single onboard image
3. We propose novel metrics to evaluate the topological accuracy of an estimated lane graph
4. The results obtained by our method are superior to the compared methods in both traditional and topological structure metrics

2. Related Works

Existing works can be roughly grouped in two distinct groups; first, offline methods, that extract HD map style road networks from aerial images or aggregated sensor data. Second, online methods, which either estimate lane boundaries or perform semantic understanding on the BEV plane,
only given current onboard sensor information. Our method is located between the two approaches, estimating HD map style lane graphs, however, based on onboard monocular images.

Road network extraction: Early works on road network extraction use aerial images [2, 37]. Building upon the same setup, recent works [4, 40, 41] perform the network extraction more effectively. Aerial imaging-based approaches only provide coarse road networks. Such predictions may be useful for routing, however, they are not accurate enough for action planning.

High definition maps: HD maps are often reconstructed offline using aggregated 2D and 3D visual information [21, 26, 27]. Although these works are the major motivation behind our work, they require dense 3D point clouds for accurate HD map reconstruction. These methods are also offline methods which recover HD maps in some canonical frame.

The usage of the recovered maps requires an accurate localization, in many cases. A similar work to ours is [20], where the lane boundaries are detected on highways in the form of polylines. An extension of [20] uses a RNN to generate initial boundary points in 3D point clouds. These initial points are then used as seeds for a Polygon-RNN [1] that predicts lane boundaries. Our method differs from [20] in: (i) point clouds vs. single image input, (ii) highway lane boundaries vs. lane centerlines in an unrestricted setting.

Lane estimation: There is considerable research in lane estimation using monocular cameras [17, 31]. The task is either performed directly on the image plane [18, 25] or in the BEV plane by projecting the image to the ground plane [16, 31, 44]. However, this line of research mainly focuses on highways and country roads, without intersections. In such cases the topology of the resulting road lane work is often trivial since lines to not intersect. Our approach focuses on urban traffic with complex road networks where the topology is fundamental.

BEV understanding: Visual scene understanding on the BEV has recently become popular due to its practicality [7, 34, 38]. Some methods also combine images with LIDAR [19, 32]. Maybe the most similar to our method are [7, 30, 38], which use a single image or monocular video frames for BEV HD-map semantic understanding. However, these methods do not offer structured outputs, therefore their usage for planning and navigation is limited.

In summary, in our paper we work in a setting similar to [22], where the output is a directed acyclic graph. However, the input is not an aggregated image and LIDAR data, but just one onboard image. Thus, the same sensor setup as existing lane estimation works, these however are not designed to work in urban environments. In fact our setting is identical to [6], but our work does focus on the topology of the lane graph and proposes a method to directly supervise the network to estimate topologically correct graphs.

3. Method

3.1. Lane Graph Representation

Following [6], we represent the local road network as a directed graph of lane centerline segments which is often called the lane graph. Let this directed graph be $G(V, E)$ where $V$ are the vertices of the graph (the centerlines) and the edges $E \subseteq \{(x, y) \mid (x, y) \in V^2\}$ represent the connectivity among those centerlines. The connectivity can be summarized by the incidence matrix $A$ of the graph $G(V, E)$. A centerline $x$ is connected to another centerline $y$, i.e. $(x, y) \in E$ if and only if the centerline $y$’s starting point is the same as the end point of the centerline $x$. This means $A[x, y] = 1$ if the centerlines $x$ and $y$ are connected. We represent centerlines with Bezier curves.

3.2. Topological Representation

While the directed graph builds an abstract high level representation of the traffic scene, the graph also introduces fundamental topological properties about the road scene. The topological properties depend on the intersections of the centerlines whereas the lane graph depends on the connectivity of the centerlines\(^1\). Thus, also considering the topology gives complementary information which we can use to estimate better representations.

We assume that the target BEV area is a bounded 2D Euclidean space, where the known bounding curves represent the borders of the field-of-view (FOV). Identical to the lane graph, each curve has a direction which represents the flow of traffic, while boundary curves have arbitrary directions. We denote the set of all curves including the border curves as $C$. To establish our later results, we assume that any two curves can intersect at most once and a curve does not intersect with itself. Due to the restricted FOV and relative short curve segments this assumption is not restrictive. Moreover, in a lane graph no curve is floating, since every end of a curve either connects to another curve or leaves the bounded space, which also results in an intersection. Let $c \in C$ be a curve and $I_c$ be the ordered sequence of intersections along the direction of curve $c$ and $I_c(m) \in P$ be the $m$th intersection of the sequence, where $P$ is the set of all intersection points. The set of $I_c$ for all curves $c$ is denoted by $I$. Combining the curves $C$ and intersection order $I$ we can form our topological structure $T(C, I)$ that together with $G(V, E)$ define the local road network (see a) of Fig. 3). In this example with linear curves, the order of intersection $I_c$ for all curves is given.

When estimating the lane graph of a traffic scene, or in fact any graph structures formed by curves, we would not only like to correctly estimate the lane graph $G(V, E)$ but also the topological properties $T(C, I)$. However, es-

\(^1\)A connection is defined by the incidence matrix of $G(V, E)$ whereas an intersection between two curves is defined in the geometric sense.
Figure 3. The graphical illustrations of our basic definitions are shown. Gray dots show the intersection points. Every curve part between any two consecutive intersection point is a curve segment. a) The complete network with all the curves \( \{c_1\} \) and the order of intersections for each curve. b) An example polycurve (above), a closed (but not minimal) polycurve (below). c) Two minimal closed polycurves (minimal cycles) with the set of curves enclosing them, i.e. their minimal covers \( B \). d) Another configuration of the same curves and the resulting minimal cycles.

Estimating the ordering of the intersection points directly is very challenging. In the following, we will show that under some assumptions the intersection order \( I \) for each curve is equivalent to the covers of minimal cycles of curves. This equivalence allows us to efficiently add a topological reasoning to our network. Let us first define minimal cycles and covers. A curve segment \( S_c(i, j) \) is the subset of the curve \( c \) between successive intersection points \( i \) and \( j \), known due to \( I \). We define a polycurve \( PC \) as a sequence of curve segments \( PC_S = (S_m|S_m(j) = S_{m+1}(i)) \). A closed polycurve \( CC \) is a polycurve with no endpoints, which completely encloses an area (see b) of Fig. 3). A minimal closed polycurve or minimal cycle \( MC \) is a closed polycurve where no curve intersects the area enclosed by \( MC \), see Fig. 3 c). Note that minimal cycles form a partition of the bounded space. Finally, given a polycurve that forms a \( MC \), we can also define the corresponding minimal cover \( B \), which is the set union of the curves that the segments in that polycurve belong to, or in other words the list of curves that form the minimum cycle, see Fig 3 c) and d). What makes minimal covers \( B \) so interesting is that although they are relatively simple, we will show in the following that they still hold the complete topological information of the road graph and are equivalent to the intersection order \( I \).

To establish this equivalence, let us first state the following results that link intersection orders to minimal cycles and covers, which holds under mild conditions detailed in the supplementary material.

**Lemma 3.1.** A minimal closed polycurve (minimal cycle) \( MC \) is uniquely identified by its minimal cover \( B \).

**Proof.** See supplementary material for proof.

For the statement to be wrong, the same curve \( c_i \) of the minimal cover \( B \) would need to generate another minimal cycle. Which intuitively becomes hard under the assumption that curves are only allowed to intersect once. For lines as shown in Fig. 3 this is not possible. For general curves the proof becomes more involved and needs some further assumptions which can be found in the supplementary.

Given that we have a link between the minimal cover and minimal cycles we now focus on to relationship between the intersection orders \( I \) and the minimal cycles.

**Lemma 3.2.** Let a set of curves \( C \) and the induced intersection orders \( I \) form the structure \( T \). Applying any deformations on the curves in \( C \), excluding removal or addition of curves, results in a new induced intersection order that creates \( T' \). Given these two typologies, \( I = I' \) if and only if the sets of minimal cycles are the same.

**Proof.** See supplementary material for proof.

Given this equivalence between intersection orders and minimal cycle we can state our main result.

**Corollary 3.2.1.** From Lemma 3.1 and Lemma 3.2, given a structure \( T \), \( I \) can be uniquely described by the set of minimal covers \( B \).

The remarkable fact about Corollary 3.2.1 is that we converted a global ordering problem into a detection problem. Instead of creating a sequence for each curve, it is enough to detect minimal cycles where each minimal cycle can be represented by a one-hot vector of the curves in \( T \) which shows whether a curve is in the minimal cover of the particular minimal cycle or not.

### 3.3. Structural Mapping

The previous theoretical results allow us to train a deep neural network that jointly estimates the curves and the intersection orders. Therefore, we build a mapping between the curves and minimal covers for both the estimated \( T_E \) and the ground truth \( T_R \) structures.

Using a neural network, we predict a fixed number of curves and minimum cycles, which is larger than the real number of curves and cycles in any scene. Thus, let there be a function \( U(x) \) that takes the input \( x \) (a camera image in our case) and outputs two matrices, \( V_e \) of size \( N \times D \) which is a \( D \) dimensional embedding for all \( N \) curve candidates.
and \( V_m \) of size \( M \times E \), which is a \( E \) dimensional embedding for all \( M \) minimal cycle candidates. The two embedding matrices are each processed by a function, \( F(V_c) \) and \( H(V_m) \). \( F(V_c) \) processes the curve candidate embedding \( V_c \) and generates a matrix output \( Z^c_t \in \mathbb{R}^{N \times \theta} \) containing the parameters for the \( N \) curves and \( Z^p_c \in \mathbb{R}^N \) the probability that the \( i^{th} \) curve exists. \( H(V_m) \) processes the minimal cycle candidate embedding, and generates three outputs each describing a property of minimum cycles. First, \( Z^a_m \in \mathbb{R}^{M \times (N+K)} \) the estimated minimal cover for each of the \( M \) candidates, describing the probability that one of the \( N \) candidate curves and \( K \) FOV boundary curves belongs to the cover. Second, \( Z^p_m \in \mathbb{R}^M \) the probability that a candidate minimal cycle exists and finally, \( Z^z_m(a) \in \mathbb{R}^{M \times 2} \) an auxiliary output estimating the centers of the candidate minimal cycles. Thus, our framework generates a set of curve and minimal cycle candidates, see Fig. 4 for an illustration.

### 3.4. Training Framework

The output of the network is, (i) a set of candidate curves and (ii) minimal cycles that are defined with respect to the candidate curves. In training we use Hungarian matching on the L1 difference between the control points of centerlines. However, it is more complex for the minimum cycles, where it is fundamental that the matching between the ground truth topology and the estimated topology is consistent. Let there be \( N' \) true curves and \( M' \) true minimal cycles with \( K \) boundary cycles. Similarly, \( Y^{rq}_c \in \mathbb{R}^{N' \times \theta} \) represents the true curve parameters, \( Y^{rq}_m \in \{0,1\}^{M' \times (N'+K)} \) the minimal covers with respect to the true curves, and \( Y^z_m \) the true centers of the minimum cycles.

**Min Matching.** Since a ground truth (GT) minimal cycle is defined on GT curves while detected minimal cycles are defined on estimated curves, we must first form a matching between estimated and GT curves. Using Hungarian matching is not ideal since it does not consider the fragmentation of the estimated curves. Fragmentation is the situation when several connected estimated curves represent one GT curve. Therefore, often the estimated candidate minimal cycles will have more candidate curves than their GT counterparts. Due to the one-to-one matching in the Hungarian algorithm, a long GT curve can only be matched to one short, fragmented curve, even though combining the estimated fragments would result in a closer approximation. Thus, we instead match each candidate curve to its closest GT curve. This means every candidate curve is matched to exactly one GT curve, while a GT curve can be matched to any number (including zero) of candidate curves.

After min matching, we create a new target for minimal cycles estimates that we denote by \( Y'_m \in \{0,1\}^{M' \times (N+K)} \). An entry in \( Y'_m(i,j) \) is 1 if the GT curve to which the \( j^{th} \) estimated curve is matched is in the \( i^{th} \) true minimal cycle. In other words, we set all the matched estimated curves to one if their corresponding true curve is present in a minimal cycle. Given this modified GT minimal cycle label and the estimated minimal cycles, we run Hungarian matching to find the pairs used for the loss calculations. This allows a consistent training of the estimated topology.

For the connectivity of the curves, we explicitly estimate the incidence matrix \( A \) of \( V(G,E) \) in our network. This is done by extracting a feature vector for each candidate centerline and building a classifier \( A(C_i, C_j) \) that takes two feature vectors belonging to curves \( C_i \) and \( C_j \) and outputs the probability of their association. The training uses the Hungarian matched curves to establish the correct order. The estimated incidence matrix allows a merging post-processing step during test time, where the endpoints of the curves are modified, so that connected curves coincide.

The centerline spline control points and minimal cycle centers are trained with an \( L_1 \) loss, while we utilize the binary cross-entropy for centerline and minimal cycle probability. We also use the binary cross-entropy for the membership loss of minimal cycles, i.e. between
We measure the backbone\textsuperscript{a)}\textsuperscript{a)} and \textsuperscript{b)} process two sets of queries (curve and minimal cycle) jointly to produce corresponding feature vectors. These vectors are then fed to MLPs for final estimations. \textsuperscript{b)}\textsuperscript{b)}\textsuperscript{b)}\textsuperscript{b)} has three parts: 1) Initial point estimation, 2) Polygon-RNN that outputs the subsequent control points of a curve given the initial points, and 3) minimal cycle decoder.

\[ Z_m^c \text{ and } Y_m^c \text{ and for the connectivity. The total loss then becomes } L = L_{\text{curve}} + \alpha L_{\text{cycle}}, \text{ where } L_{\text{curve}} = L_{\text{splines}} + \beta_c L_{\text{exists}} + \beta_c L_{\text{connect}} \text{, and } L_{\text{cycle}} = L_{\text{member}} + \beta_c L_{\text{exists}} + \beta_c L_{\text{center}}, \text{ with } \alpha \text{ and } \beta_c \text{ hyperparameters.} \]

4. Network Architectures

Following [6], we focus on two different architectures to validate the impact of our formulation. The first architecture is based on transformers [8] while the second approach is based on Polygon-RNN [1].

4.1. Transformer

We modify the transformer-based architecture proposed in [6]. We use two types of learned query vectors: centerline (curve) and cycle queries. We concatenate centerline and cycle queries before being processed by the transformer. Therefore, curves and cycles are jointly estimated. The transformer outputs the processed queries that correspond to \( V_c \) and \( V_m \) in our formulation. Finally, we pass these vectors through two-layer MLPs to produce the estimates \( Z_c \) and \( Z_m \). The overview is given in Fig 5. Note that the addition of the MC formulation adds negligible parameters since the number of parameters in the transformer is fixed. We call the transformer model with MC, Ours/TR.

As a baseline, we added an RNN on the base transformer to estimate the order of intersections directly and provide supervision to the network. The RNN processes each centerline query output from the transformer independently and generates an \( N + K + 1 \) dimensional vector at each time step that represents the probability distribution of intersecting one of the \( N + K \) curves and one `end` token. The RNN is supervised by the true intersection orders converted to estimate centerlines through Hungarian matching. We named this method TR-Decoder, see Suppl. Mat. for details.

4.2. Polygon-RNN

The second network is based on Polygon-RNN [1] and is similar to [20], where the authors generate lane boundaries from point clouds. We adapt [20] to work with images and to output centerlines rather than lane boundaries. Following [6], we use a fully connected sub-network that takes \( V_c \) as input and outputs a grid. Each element represents the probability of an initial curve point of a curve starting at that location, i.e. \( Z_c^\alpha \).

Given the initial locations and the backbone features, Polygon-RNN [1] produces the next control points of the centerline. We fix the number of iterations of Polygon-RNN to the number of spline coefficients used to encode centerlines. The approach described so far forms the base Polygon-RNN. With Polygon-RNN producing \( Z_c^\alpha \), we add a transformer decoder to the architecture to detect the minimal cycles. We use a set of minimal cycle queries similar to our transformer architecture, where the queries are processed with final feature maps of Polygon-RNN. Therefore, in the transformer decoder, the query vectors attend the whole set of estimated centerlines to extract the minimal cycle candidates. For this process, we pad the RNN states to a fixed size and add positional encoding. This ensures that the decoder receives the information regarding the identity of the curves. The processed query vectors are passed to the same MLPs as in the transformer architecture to produce the set of minimal cycle estimates \( Z_m^c \). Fig 5 outlines this approach, which we call Ours/PRNN. Different to the transformer based method, this is a two stage process, where first the centerlines are estimated and then the minimal cycles.

5. Metrics

Several metrics were proposed in [6] to measure accuracy in estimating the lane graph. They are \textit{M-F-Score}, \textit{Detection ratio}, and \textit{Connectivity}. These metrics do not cover the topology of the road network. Thus, we propose two new metrics that capture the accuracy in estimating topology. The proposed metrics complement the existing lane graph metrics to give a full picture of the accuracy of the estimated road network.

\textbf{Minimal-Cycle Minimal Cover (B)}. We measure the...
minimal cycle accuracy in 2 inputs. First, minimal cycles are extracted from the estimations. We use the procedure of Section 3.4 to obtain $Y^q_m$. Then, these cycles are matched using Hungarian matching to calculate true positives, true negatives, and false positives. This metric is referred to as MC-F. We also measure the accuracy of the minimal cycle network. Similar to MC-F, first $Y^q_m$ is obtained, and second we threshold $Z^p_m$ to obtain the detected $Z^q_m$. Then, we apply Hungarian matching and calculate statistics on matched cycles. We call this metric H-GT-F, which is applicable only if the minimal cycles are detected. H-GT-F measures the MC-network’s performance in estimating the true cycles in the true topology. Finally, H-EST-F measures the MC-head’s performance in detecting the estimated cycles. Since the extracted MCs and the MC head estimations are with respect to estimated curves, we directly run Hungarian matching on the extracted and estimated MCs.

**Intersection Order** ($I$ of $G(C, I)$). To measure the performance of the methods in preserving the intersection order, we start with min-matching. Then for each true curve, we select the closest matched estimate. For a given true curve $C_i$, let the matched curve be $S_i$. We extract the order of intersections from both $C_i$ and $S_i$ and apply the Levenshtein edit distance between them. The distance is then normalized by the number of intersections of the true curve. We refer to this metric as I-Order.

### 6. Experiments

We use NuScenes [5] and Argoverse [11] datasets. Both datasets provide HD-Maps in the form of centerlines. We convert the world coordinates of the centerlines, to the camera coordinate system of the current frame, then resample these points with the target BEV resolution and discard any point that is outside the region-of-interest (ROI). The points are then normalized with the ROI bounds $[0, 1]^2$. We extract the control points of the Bezier curve for this normalized coordinate system. The ground truth and the estimations of the method are also represented in the same coordinate system. We use the same train/val split proposed in [38].

**Implementation.** We use images of size 448x800 and the target BEV area is from -25 to 25m in x-direction and 1 to 50m in z direction with a 25cm resolution. Due to the limited complexity of the centerlines, three Bezier control points are used. We use two sets of 100 query vectors for centerlines and minimal cycles: one for right (Boston & Argoverse) and one for left sided traffic (Singapore). The backbone network is Deeplab v3+ [13] pretrained on the Cityscapes dataset [14]. Our implementation is in Pytorch and runs with 11FPS without batching and including all association steps. When training Polygon-RNN, we use true initial points for training of the RNN, following [20]. To train the initial point subnetwork, we use focal loss [28].

**Baselines.** We compare against state-of-the-art transformer and Polygon-RNN based methods proposed in [6] as well as another baseline which uses the method PINET [25] to extract lane boundaries. The extracted lane boundaries are then projected onto the BEV using the ground truth transformation. We then couple pairs of lane boundaries and extract the centerlines using splines. This baseline is not evaluated for connectivity.

### 7. Results

We report quantitative comparison with SOTA on the NuScenes dataset in Tab. 1. The proposed formulation provides substantial boost in almost every metric for Polygon-RNN based methods. Compared to transformer-based
Figure 7. The I-Order measures (lower is better) with number of centerlines, intersection points, and occlusion in scene. The x-axis measures reflect the complexity of scenes for the visual understanding.

Table 1. Results on NuScenes. See Section 5 for the metrics.

<table>
<thead>
<tr>
<th>Method</th>
<th>M-F</th>
<th>Detect</th>
<th>C-F</th>
<th>MC-F</th>
<th>I-Order</th>
<th>RNN-Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>PINET [25]</td>
<td>49.5</td>
<td>19.2</td>
<td>-</td>
<td>14.7</td>
<td>1.08</td>
<td>-</td>
</tr>
<tr>
<td>PRNN</td>
<td>52.9</td>
<td>40.5</td>
<td>24.5</td>
<td>45.9</td>
<td>0.894</td>
<td>-</td>
</tr>
<tr>
<td>Ours/PRNN</td>
<td>51.7</td>
<td>53.1</td>
<td>49.9</td>
<td>53.2</td>
<td>0.824</td>
<td>-</td>
</tr>
<tr>
<td>TR [6]</td>
<td>56.7</td>
<td>59.9</td>
<td>55.2</td>
<td>62.0</td>
<td>0.800</td>
<td>-</td>
</tr>
<tr>
<td>TR-RNN</td>
<td>58.0</td>
<td>59.7</td>
<td>52.3</td>
<td>60.3</td>
<td>0.791</td>
<td>0.939</td>
</tr>
<tr>
<td>Ours/TR</td>
<td>58.3</td>
<td>60.2</td>
<td>55.3</td>
<td>62.5</td>
<td>0.776</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2. Results on Argoverse

<table>
<thead>
<tr>
<th>Method</th>
<th>M-F</th>
<th>Detect</th>
<th>C-F</th>
<th>MC-F</th>
<th>I-Order</th>
<th>RNN-Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>PINET [25]</td>
<td>47.2</td>
<td>15.1</td>
<td>24.5</td>
<td>72.5</td>
<td>1.23</td>
<td>-</td>
</tr>
<tr>
<td>PRNN</td>
<td>55.7</td>
<td>40.2</td>
<td>51.3</td>
<td>42.8</td>
<td>1.05</td>
<td>-</td>
</tr>
<tr>
<td>Ours/PRNN</td>
<td>44.3</td>
<td>55.4</td>
<td>46.4</td>
<td>49.5</td>
<td>1.05</td>
<td>-</td>
</tr>
<tr>
<td>TR [6]</td>
<td>55.6</td>
<td>60.1</td>
<td>54.9</td>
<td>57.4</td>
<td>0.893</td>
<td>-</td>
</tr>
<tr>
<td>TR-RNN</td>
<td>55.8</td>
<td>53.6</td>
<td>54.4</td>
<td>52.2</td>
<td>0.953</td>
<td>0.951</td>
</tr>
<tr>
<td>Ours/TR</td>
<td>57.1</td>
<td>64.2</td>
<td>58.1</td>
<td>58.5</td>
<td>0.883</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3. Polygon-RNN with GT initial points results

Table 4. Performance of minimal cycle estimation head.

8. Conclusion

We studied local road network extraction from a single onboard camera image. To encourage topological consistency, we formulated the minimal cycle matching strategy by means of matching only their covers. Our formulation is then used to derive losses, to train neural networks of two different architectures, namely Transformer and Poly-RNN. Both architectures demonstrated the importance of the proposed MC branch, and thus the formulated loss function, on two commonly used benchmark datasets. The proposed formulation, and the method, have the potential to be used in many other computer vision problems which require topologically consistent outputs, for example, indoor room layout estimation or scene parsing.

Limitations. The theoretical assumptions are mild for most modern road networks. The extraction of minimal cycles for training is time consuming and should be done offline.
References


