Bézier Everywhere All at Once: Learning Drivable Lanes as Bézier Graphs

Supplementary Material

6. Bézier Fitting

6.1. Fitting Procedure

In this section, we provide a more detailed description of the Bézier fitting process initially introduced in Sec. 3.1.

As previously described, the Bézier Graph nodes $\mathcal{V}_b$ are first chosen as a subset $\mathcal{V}_b \subseteq \mathcal{V}_l$ of the nodes of the input graph, $\mathcal{G}_l$. For each directed path in $\mathcal{G}_l$ between these nodes, we then add a corresponding edge to $\mathcal{G}_b$. This results in the surjective mapping

$$\text{EDGEMAP} : \mathcal{V}_l \rightarrow \mathcal{E}_b.$$

For each path

$$p = (v_1, v_2, \ldots, v_n) \in \mathcal{V}_l \times \cdots \times \mathcal{V}_l$$

for each node v_i in the path, we can define a mapping to Bézier parameter t. Consider that each node v_j has corresponding 2D position x_j, and the path therefore represents a sequence of positions $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. The mapping from node v_i to t is therefore defined as its fractional distance along the path, i.e.

$$\text{LENGTHMAP}(v_i) = \begin{cases} \frac{\sum_{j=2}^{i} \|\mathbf{x}_j - \mathbf{x}_{j-1}\|_2}{\sum_{j=2}^{n} \|\mathbf{x}_j - \mathbf{x}_{j-1}\|_2} & i \geq 2 \\ 0 & i = 1 \end{cases}$$

Note since each node exists in only one path, this is a unique mapping for each node.

Therefore, we have for each node a unique mapping to an edge in the Bézier Graph and a (normalised) length along the corresponding Bézier curve:

$$\text{NODEMAP}(v) = (\text{EDGEMAP}(v), \text{LENGTHMAP}(v)).$$

This allows us to now define the function $\mathbf{F}_{\mathcal{G}_l}(\mathbf{D}_b, \mathbf{E}_b)$ mapping an input lane graph and Bézier parameters to a matrix of reconstructed node positions: this is provided in Algorithm 2.

Note we have an additional requirement that the values of $\mathbf{E}_b$ are positive, i.e. $\mathbf{E}_b \in \mathbb{R}_+^{|\mathcal{E}_b| \times 2}$. To account for this, we actually perform gradient descent on the logarithm of these Bézier parameters, and exponentiate them to obtain the final $\mathbf{E}_b$ array.

6.2. Additional Fit Results

In this Appendix we present additional quantitative and qualitative results of our Bézier fitting method.

Algorithm 2 Bézier Graph reconstructed node positions

```
function $\mathbf{F}_{\mathcal{G}_l}(\mathbf{D}_b, \mathbf{E}_b)$
    $\mathbf{X} \leftarrow []$
    for $v \in \mathcal{V}_l$ do
        $e, t \leftarrow$ NODEMAP($v$)
        $x \leftarrow$ BEZIERPOSITION($\mathbf{D}_b[e[0]], \mathbf{D}_b[e[1]], \mathbf{E}_b[e], t$)
        $\mathbf{X}$.append($x$)
    end for
    return $\mathbf{X}$
end function
```

The full results for the Bézier fit for the Succ-LGP and Full-LGP experiments are shown in Tabs. 4 and 5 respectively. These show a good fit on average, as discussed in our paper. However, the high maximum Hausdorff distance values also demonstrate that in some edge cases, the fit fails.

Note that here the Hausdorff distance is computed per Bézier Graph edge (a cubic Bézier curve) as compared to the ground truth nodes to which it was fit. Therefore the maximum Hausdorff per tile is the maximum Hausdorff distance between any Bézier curve and its ground truth nodes. Mean Hausdorff is the mean over all curves. The mean Hausdorff distances reported in Tabs. 4 and 5 are computed over the mean values from all tiles. All Hausdorff distances were computed with the `directed_hausdorff` function from the `scipy.spatial.distance` Python library.

To understand the causes of the poor fits we visualise two of the worst performing Bézier Graph fits for the Succ-LGP training set and Full-LGP training set in Fig. 7. These four visualisations demonstrate three different failure cases: Fig. 7a demonstrates poor performance on unusual curves with multiple inflection points; Figs. 7b and 7d both demonstrate poor performance on U-turns and Fig. 7c visualises a case with two lanes meeting at the same point but going in nearly completely opposite directions (this is likely due to opposite direction turning lanes meeting at a two-way lane).

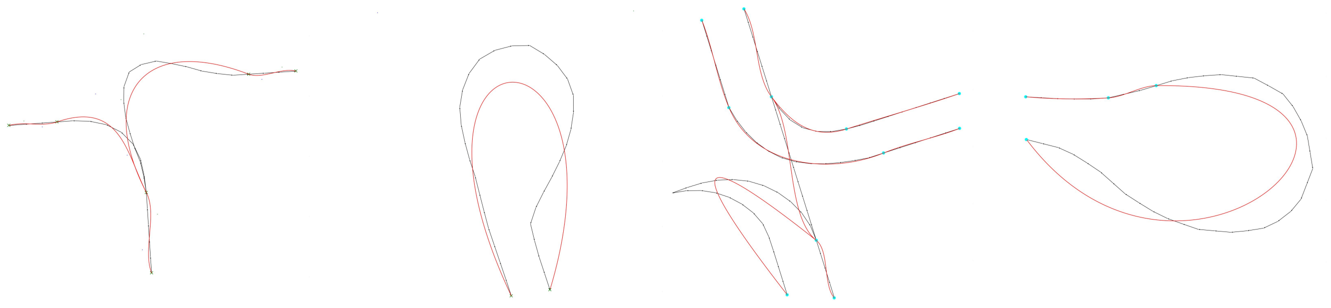
Future iterations of the Bézier Graph fitting process could be improved by better accounting for more complicated curves and bidirectional lanes.

7. Experiment Details

7.1. Full-LGP Dataset Tiling

For the Succ-LGP experiments, we use the training and evaluation tiles provided by the UrbanLaneGraph dataset.

For the Full-LGP experiments, we generate our own



(a) Succ-LGP Austin fitting failure case.

(b) Succ-LGP Washington fitting failure case.

(c) Full-LGP Austin fitting failure case.

(d) Full-LGP Washington fitting failure case.

Figure 7. Examples of where the Bézier Graph fitting process fails. Fitted Bézier Graph in red, ground truth lane graph in black.

training tiles, since we use a different tile resolution (512×512) and train on tiles containing all lanes, rather than just those which are descendants of the bottom centre node.

In order to generate these tiles from the 5000×5000 images in the Full-LGP training set, we use an agglomerative clustering algorithm, fit on node positions. For this we use `AgglomerativeClustering` from the `sklearn.cluster` Python library, using a `distance_threshold` of twice the output crop size ($2 * 512$).

These images are incompletely annotated, so this method resulted in training images containing incomplete lane networks. Therefore, to avoid incomplete training examples we further removed all clustered tiles containing any nodes with a degree of 1, thus removing "end-point" nodes.

This results in tiles that effectively follow the lane graph structure, with some overlap between them. In this way the training data is efficiently used and examples of incomplete annotation in the training set are minimised. A visualisation of this tiling approach can be found in Fig. 8.

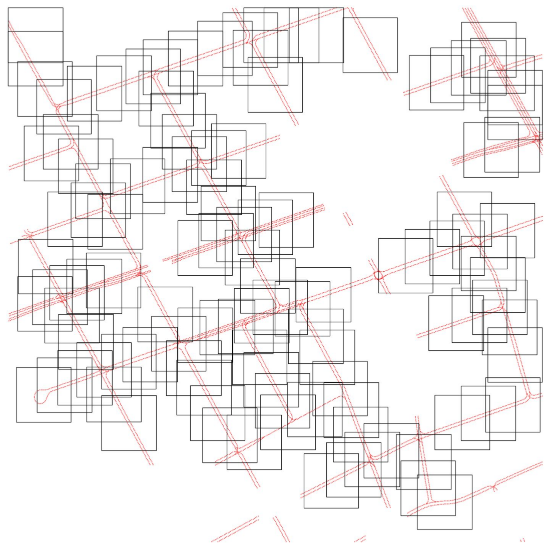


Figure 8. Agglomerative clustering of the node positions in an example Full-LGP image, to generate our 512×512 training tiles. Ground truth lane graph shown in red, tiles shown as black square boxes.

7.2. Hyperparameters and Training

Both models were trained with a ResNet-50 backbone and Deformable-DETR transformer architecture. Both used transformer layer dimension 256, 6 layers and 8 attention heads in both the encoder and decoder. Both used the same loss weightings λ, shown in Tab. 6.

The Succ-LGP model was trained for 150 epochs, Full-LGP for 250. We applied colour jitter dataset augmentation to both, and random rotation by integer multiples of $90°$ to the Full-LGP model. Both models were trained using the Adam optimiser with a cosine varying learning rate, beginning at 10^{-4} and ending at 10^{-5}.

λ_c^n	λ_p^n	λ_d^n	λ_{prob}^e	λ_a^e
1	5	2	0.2	1

Table 6. Loss weightings used for both models.

7.3. Reproducing LaneGNN

We encountered difficulties when trying to use the public UrbanLaneGraph LaneGNN code to reproduce their Succ-LGP experiments. Our understanding of the provided codebase was that additional "context" RGB files were required, which at the time of writing were not included in the downloaded UrbanLaneGraph successor `eval` data. We attempted to create our own `eval` data using the full tiles and the preprocessing scripts provided in their codebase, but this resulted in a significant fraction of the resulting graphs containing errors such as incorrectly connected overlapping lanes, which particularly harmed the successor graph generation. Therefore, to be as fair as possible, we used the LaneGNN numbers provided by the original paper for our Succ-LGP comparisons. Our method did not require the additional "context" RGB files, so we were able to use the provided `eval` data for evaluation.

For the Full-LGP task we were able to use the provided data and code to evaluate the UrbanLaneGraph model. However, as we highlight in the main body of our paper, the original UrbanLaneGraph paper used a pretrained LaneExtractor model to provide initial positions and directions for LaneGNN. We did not have access to this trained model, so we instead initialised with 180 sampled positions and directions from the ground truth data. Our Full-LGP comparisons therefore use metrics obtained from our own reproduction of the LaneGNN model across all cities, rather than those reported in the original paper; note the original paper evaluated only on the two Miami files in order to compare with LaneExtractor.

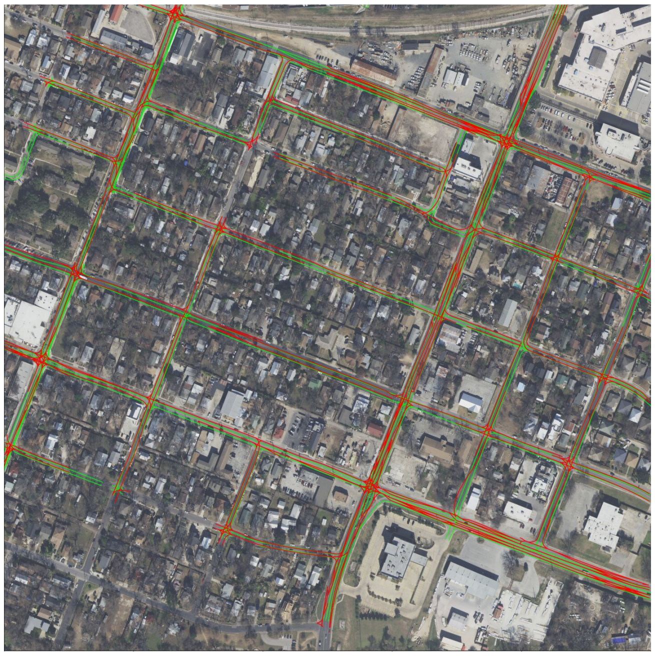
8. Additional Full-LGP Results

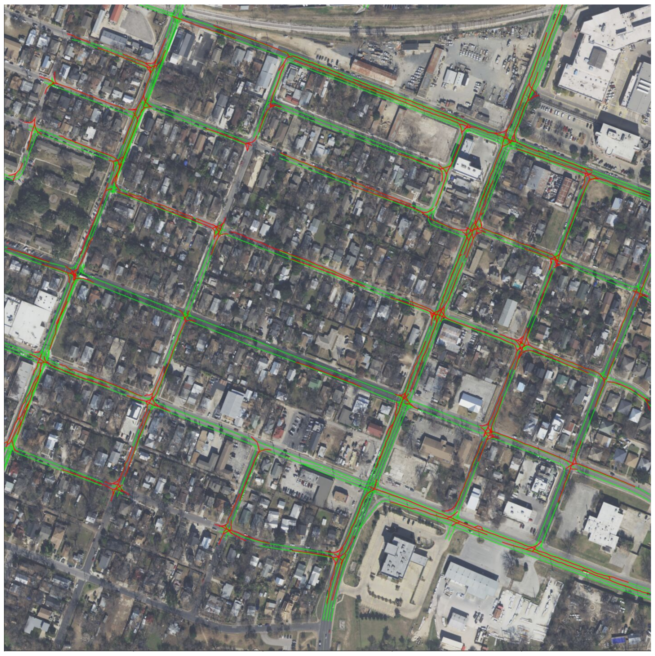
For clarity, and given space limitations, we visualised only crops of the full 5000×5000 tiles in the body of our paper. In Fig. 9 we show the full results for the Full-LGP model on the corresponding tiles.

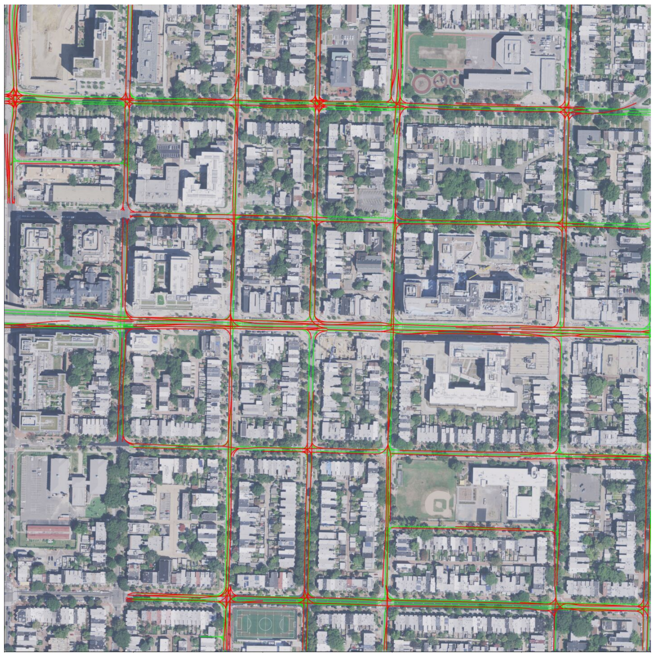
Note to account for the incomplete ground truth annotation and following the experimental setup of Büchner *et al.* [3], we filter our predicted graph by removing any curve components for which there does not exist a ground truth node within 50 pixels.

8.1. Curvature Distributions

In Fig. 10 we visualise several example curvature distributions, comparing BGFormer to LaneGNN on several of the Full-LGP aggregated graphs. We observe that BGFormer captures the curvature distribution of the ground truth more closely, and it seems that there is a common trend for LaneGNN to predict a higher proportion of larger curvatures; we believe this may be exacerbated by the 'wobbling' of the LaneGNN predicted lanes due to the node lattice subtractive sampling.


(a) BGFormer Evaluated on an Austin Tile


(b) LaneGNN Evaluated on an Austin Tile


(c) BGFormer Evaluated on a Washington Tile

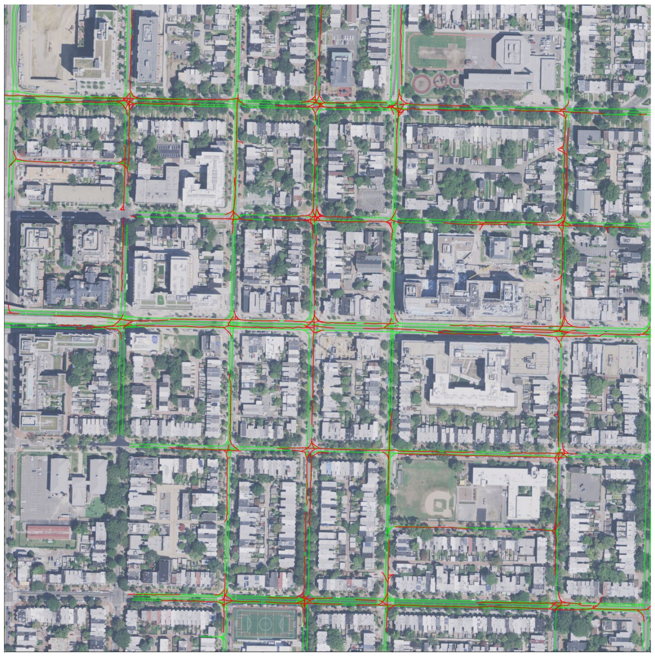

(d) LaneGNN Evaluated on a Washington Tile

Figure 9. Qualitiative comparison of uncropped Full-LGP results. Predicted lanes are depicted in red, ground truth lanes in green. Figs. 9c and 9d visualise the uncropped version of figures presented in the main body of our paper.

City	Max H		Mean	$\|\mathcal{V}_l\|$		$\|\mathcal{V}_b\|$		$\|\mathcal{E}_l\|$		$\|\mathcal{E}_b\|$	
	Max	Mean	H	Max	Mean	Max	Mean	Max	Mean	Max	Mean
Austin	11.4	1.1 ± 1.0	0.65	72.0	27.1 ± 7.8	12	4.3 ± 1.7	70	26.0 ± 7.8	11	3.3 ± 1.7
Detroit	20.9	1.2 ± 1.1	0.67	93.0	27.4 ± 8.5	17	4.4 ± 1.7	89	26.2 ± 8.4	16	3.4 ± 1.7
Miami	39.9	1.2 ± 1.3	0.68	94.0	27.5 ± 8.8	16	4.4 ± 1.8	91	26.4 ± 8.7	15	3.4 ± 1.8
Palo Alto	16.7	1.3 ± 1.4	0.76	76.0	27.6 ± 8.0	21	4.5 ± 1.7	74	26.5 ± 8.0	20	3.5 ± 1.7
Pittsburgh	40.8	1.2 ± 1.4	0.74	70.0	25.1 ± 8.5	11	4.0 ± 1.6	69	24.0 ± 8.4	10	3.0 ± 1.6
Washington	29.8	1.2 ± 1.3	0.67	77.0	27.4 ± 8.3	14	4.3 ± 1.6	74	26.2 ± 8.2	13	3.3 ± 1.6

Table 4. Complete results for the Bézier fit of the Succ-LGP training examples. Max H and Mean H respectively denote the maximum and mean Hausdorff distances between the ground truth and Bézier Graph for each training example - both are reported in pixels. To obtain distances in meters, multiply by 0.15. $|\mathcal{V}|$ denotes number of nodes, $|\mathcal{E}|$ denotes number of edges, l and b subscripts denote lane (source) graph and Bézier (fitted) graph respectively.

City	Max H		Mean	$\|\mathcal{V}_l\|$		$\|\mathcal{V}_b\|$		$\|\mathcal{E}_l\|$		$\|\mathcal{E}_b\|$	
	Max	Mean	H	Max	Mean	Max	Mean	Max	Mean	Max	Mean
Austin	32.1	3.3 ± 3.4	1.09	701	187 ± 98	60	14 ± 8	689	183 ± 96	50	10 ± 7
Detroit	22.0	3.4 ± 3.5	1.04	697	223 ± 120	60	17 ± 10	680	218 ± 117	58	12 ± 8
Miami	48.9	3.2 ± 3.2	1.10	666	192 ± 102	64	15 ± 9	650	188 ± 99	61	11 ± 8
Palo Alto	115.0	4.0 ± 5.5	1.30	1007	221 ± 124	77	16 ± 10	980	215 ± 121	60	11 ± 8
Pittsburgh	120.2	3.7 ± 5.1	1.25	546	152 ± 79	49	13 ± 8	532	148 ± 77	42	9 ± 7
Washington	29.3	3.1 ± 3.3	1.07	678	209 ± 119	52	15 ± 9	658	204 ± 116	47	11 ± 8

Table 5. Complete results for the Bézier fit of the Full-LGP training example tiles. See Tab. 4 for column descriptions.

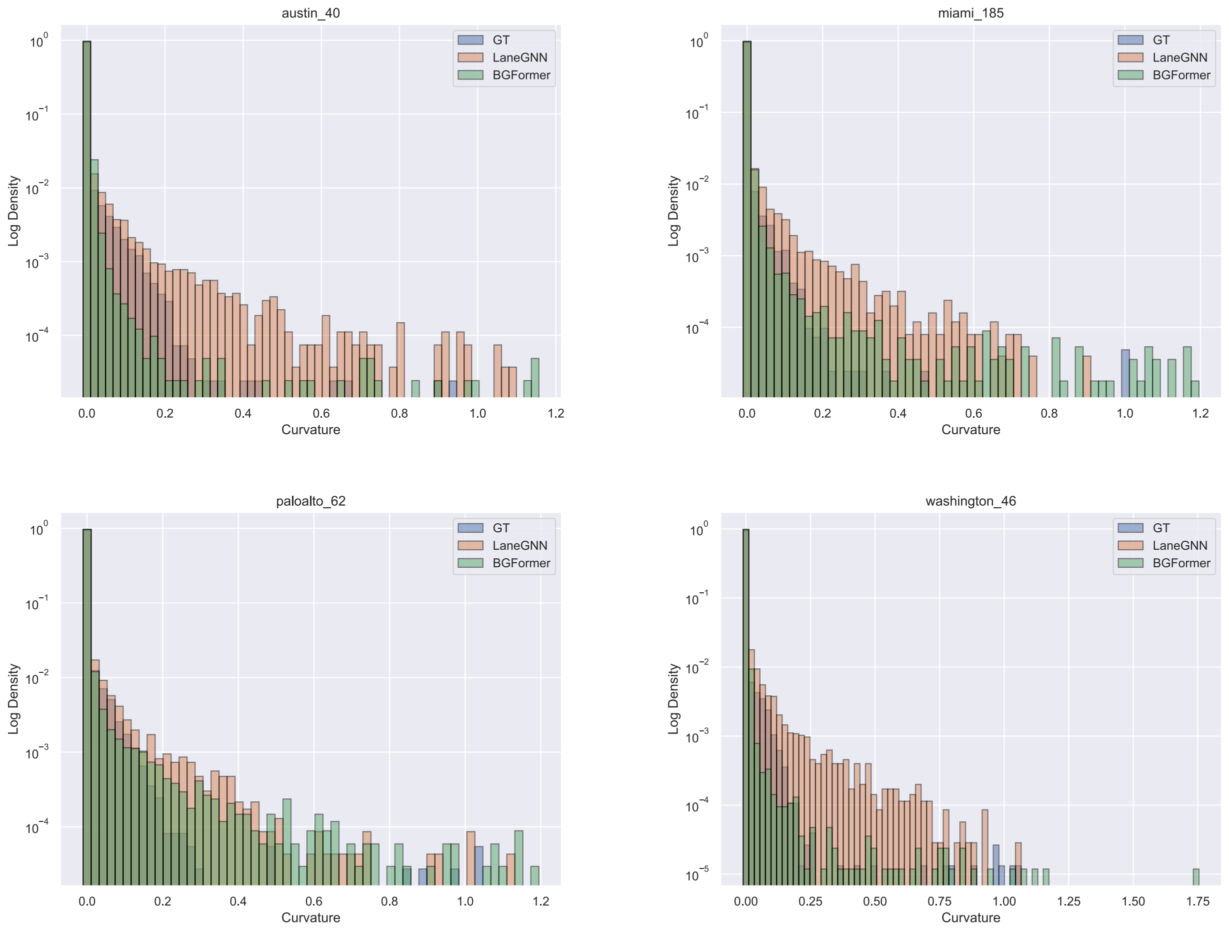


Figure 10. Curvature distributions for several example Full-LGP tiles. Ground truth (GT) distribution shown in blue; LaneGNN in orange; BGFormer in green.