BEV-LaneDet: An Efficient 3D Lane Detection Based on Virtual Camera via Key-Points

Ruihao Wang[1,2], Jian Qin[*1], Kaiying Li[1], Yaochen Li[2], Dong Cao[1], Jintao Xu[1]
[1]HAOMO.AI Technology Co., Ltd.
[2]Xi'an Jiaotong University
{wangruihao, qinjian, likaiying, caodong, xujintao}@haomo.ai, yaochenli@mail.xjtu.edu.cn

1. Appendix

1.1. Inference

In the inference phase, the four branches of the network output, including the confidence M_{conf}, the embedding M_{emb}, the offset M_{off}, and the lane height M_Z, are synthesized into the instance-level lanes by a fast unsupervised clustering method which we refer to mean-shift. As shown in Alg 1. This process consists of four steps: 1) Filtering the positive points of confidence mask M_{conf} by the $S_{threshold}$ to obtain the E_{list}. 2) Clustering points of M_{emb} to obtain the clustering points of each lane R_{point} and cluster centers of each lane R_{center} by inter-class distance D_{gap}. 3) Adding offset M_{off} and lane height M_Z to points in each lane to obtain 3D lanes R_{lines}. 4) Fitting the key points of the lanes R_{lines} to yield the lane equations R_{fit}.

Algorithm 1 Post-processing algorithm of our method

Input: $M_{conf}, M_{emb}, M_{off}, M_Z \leftarrow Model(I_v)$;
Output: 3D lanes after fitting, R_{fit}.

Filtering the point of confidence mask by $S_{threshold}$ to get the E_{list}:
for $x = 0; x < M_{conf}.cols; x++$ **do**
  for $y = 0; y < M_{conf}.rows; y++$ **do**
    if $M_{conf}[x, y] >= S_{threshold}$ **then**
      $E_{list}.append([x, y, M_{emb}[:, y, x]])$;
    end if
  end for
end for
Clustering points to get the R_{point} and R_{center} by D_{gap}:
for $i = 0; i < E_{list}.length; i++$ **do**
  $x, y, value = E_{list}[i]$;
  $min\_gap = D_{gap} + 1$;
  $min\_cid = -1$;
  for $j = 0; j < R_{center}.length; j++$ **do**
    $center\_id, (center, num) = R_{center}[j]$;
    $diff = Euclidean(value, center)$;
    if $diff < min\_gap$ **then**
      $min\_gap = diff$;
      $min\_cid = center\_id$;
    end if
  end for
  if $min\_gap < D_{gap}$ **then**
    $R_{point}.append([x, y, min\_cid])$;
    $center, num = R_{center}[min\_cid]$;
    $R_{center}[min\_cid] = [(center \times num + value)/(num + 1), num + 1]$;
  else
    $R_{center}.append([value, 1])$;
    $R_{point}.append([x, y, R_{center}.length - 1])$
  end if
end for
Adding offset to points in each lane to get R_{lanes}:
for $k = 0; k < R_{point}.length; k++$ **do**
  $x, y, id = R_{point}[i]$;
  $off\_y = M_{off}[:, y, x]$;
  $z = M_Z[y, x]$;
  $R_{lanes}[id].append([x, y + off\_y, z])$;
end for
$R_{fit} = FitFunc(R_{lanes})$;

1.2. Visualization

We analyse the comparison between BEV-LaneDet and PersFormer [1] in different scenarios by visualization. In Figure 1, the first column is the input images; the second column is the results of PersFormer on BEV; the third column is the results of our method on BEV; the fourth column is the results of PersFormer in 3D space; the fifth column is the results of our method in 3D space. The visualization results demonstrate that our method is more stable and accurate in different scenarios. At the same time, our method is more suitable to represent the diversity of lane structures compared with PersFormer.

References

[1] Li Chen, Chonghao Sima, Yang Li, Zehan Zheng, Jiajie Xu, Xiangwei Geng, Hongyang Li, Conghui He, Jianping Shi, Yu Qiao, et al. Persformer: 3d lane detection via perspective transformer and the openlane benchmark. *arXiv preprint arXiv:2203.11089*, 2022. 1, 2

*Corresponding author

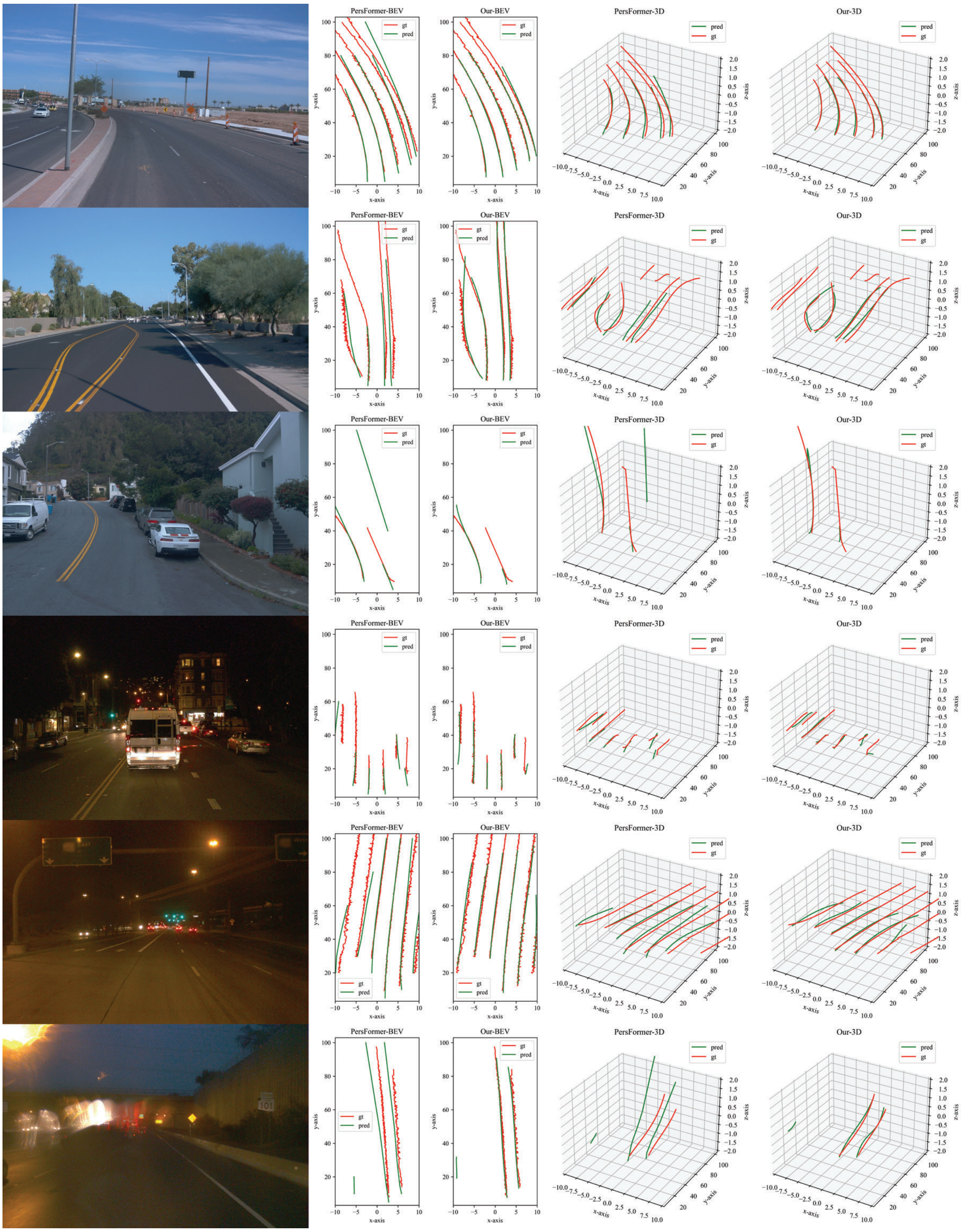

Figure 1. Qualitative results of PersFormer [1] and BEV-LaneDet in different scenarios of the OpenLane dataset. First row: Curve; second row: Merge&Split; third row: Up&Down; fourth row: Night; fifth row: Intersection; sixth row: Backlight.