Appendix of "DFA3D: 3D Deformable Attention For 2D-to-3D Feature Lifting"

A. Closer visualization of the expanded 3D image features.

In Sec. 3.1, for a specific view (the n^{th} view for example), we obtain the expanded 3D image feature map by conducting outer product at last dimension between the estimated depth distribution $\boldsymbol{D}_n \in \mathbb{R}^{H\times W\times D}$ and 2D image feature maps $\boldsymbol{X}_n \in \mathbb{R}^{H\times W\times C}$. To further explain the resulting expanded 3D image feature $\boldsymbol{F}_n \in \mathbb{R}^{H\times W\times D\times C}$, we show a closer visualization in Fig. 4. For the features with the same (u_i, v_j) coordinate (zoomed in), they share the same 2D image feature $\boldsymbol{X}_{n,u_i,v_j}$ and depth distribution $\boldsymbol{D}_{n,u_i,v_j}$. However, since they have different depth values d_k, they select different depth confidence scores $\boldsymbol{D}_{n,u_i,v_j,d_k}$, resulting in different features $\{\boldsymbol{D}_{n,u_i,v_j,d_k} \cdot \boldsymbol{X}_{n,u_i,v_j}\}$.

B. Can 3D deformable attention be trivially implemented by feature weighting followed with a common 2D deformable attention?

As each sampling point in 3D deformable attention has its own 3D location, it will lead to its own depth score for 4 adjacent image features when conducting weighted bilinear interpolation. Inevitably, there will be features ($\boldsymbol{X}_{n,u_1,v_0}$ and $\boldsymbol{X}_{n,u_1,v_1}$ in Fig. 5) referred by more than one sampling point when sampling points are located in neighboring grids. In such cases, feature weighting will results in conflicts. Thus, we can not simply prepare 2D features by depth-weighting and then conduct the common 2D deformable attention. The depth-weighted feature computation should be conducted on the fly.

C. Applicability

As the comparison shown in Fig. 6, when integrating our DFA3D in any 2D deformable attention-based feature lifting only requires a few modifications in code. The main modification lies in the addition of DepthNet and replacing 2D deformable attention with our 3D deformable attention.

D. Visualization

We visualize the predictions of BEVFormer and BEVFormer-DFA3D in Fig. 7. The shaded triangles correspond to camera rays, in which BEVFormer makes more duplicate predictions behind or in front of ground truth objects compared with BEVFormer-DFA3D. It demonstrates the negative effects of depth ambiguity in 2D deformable attention. After integrating with DFA3D, the wrong predictions caused by the depth ambiguity problem are reduced.

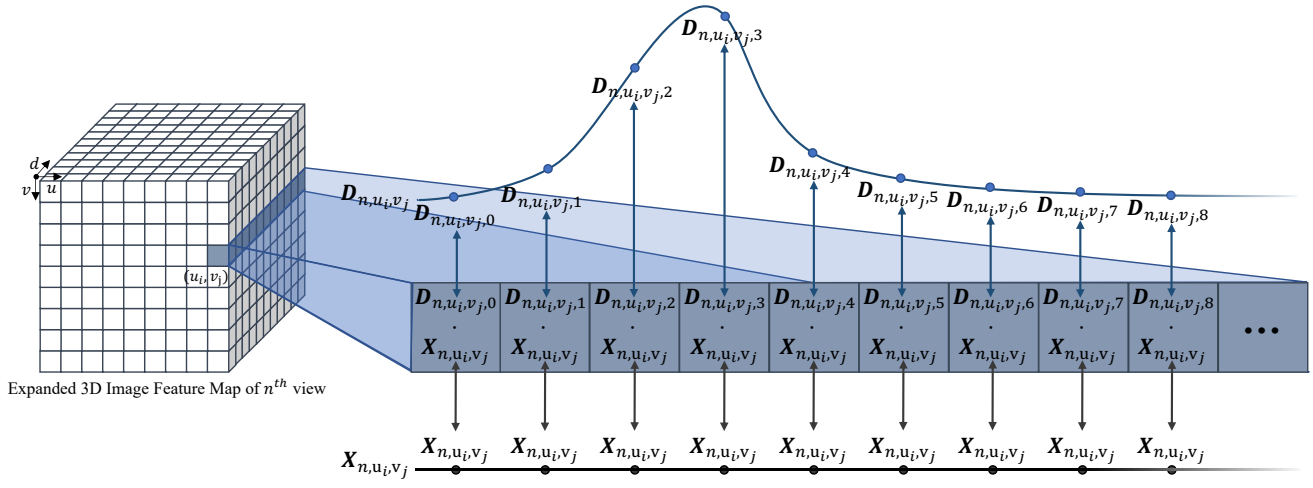

Figure 4: A closer visualization of the expanded 3D image features.

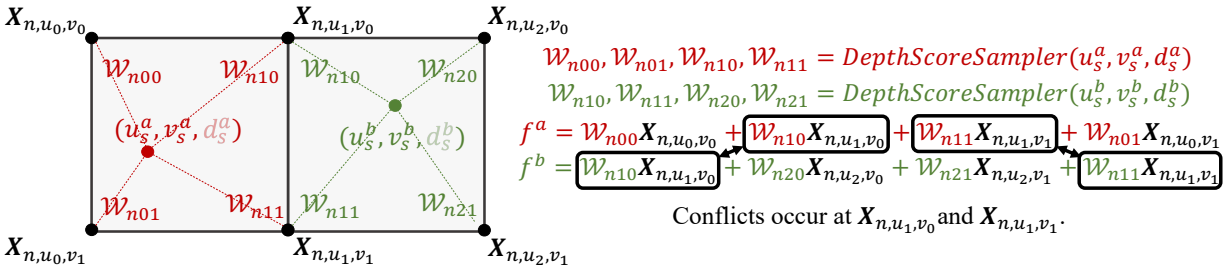

Figure 5: In 3D deformable attention, when sampling features for two sampling points (u_s^a, v_s^a, d_s^a) and (u_s^b, v_s^b, d_s^b) respectively, we first sample depth scores $(\mathcal{W}_{n00}, \mathcal{W}_{n10}, \mathcal{W}_{n11}, \mathcal{W}_{n01})$ and $(\mathcal{W}_{n10}, \mathcal{W}_{n20}, \mathcal{W}_{n21}, \mathcal{W}_{n11})$ for these two points respectively. After that, we conduct depth-weighted bilinear interpolation for these two points based on their own depth scores independently.

```
multiview_feature_maps = Backbone(multiview_input_RGB)

multiview_reference_point = ProjEgo2Pixel(
    positions_BEV,
    extrinsic_param,
    intrinsic_param
)
multiview_reference_point = multiview_reference_point[..., :2]  # [N 3] → [N 2]

for layer_id in range(num_layers):
    multiview_sample_offset = MLP_2D(content_BEV) # [N C] → [N K 2]
    multiview_sample_point = multiview_reference_point + multiview_sample_offset

    content_BEV = DeformableAttention_2D(
        multiview_feature_maps_2D,
        multiview_sample_point
    )
lifted_feature = content_BEV
```

2D Deformable Attention-based Feature Lifting

```
multiview_feature_maps = Backbone(multiview_input_RGB)
multiview_depth_dist = DepthNet(multiview_feature_maps)

multiview_reference_point = ProjEgo2Pixel(
    positions_BEV,
    multiview_extrinsic_param,
    multiview_intrinsic_param
)

for layer_id in range(num_layers):
    multiview_sample_offset = MLP_3D(content_BEV) # [N C] → [N K 3]
    multiview_sample_point = multiview_reference_point + multiview_sample_offset

    content_BEV = DeformableAttention_3D(
        multiview_depth_dist,
        multiview_feature_maps_2D,
        multiview_sample_point
    )
lifted_feature = content_BEV
```

3D Deformable Attention-based Feature Lifting

Figure 6: Integrate DFA3D into any 2D deformable attention-based feature lifting requires only a few modifications in code.

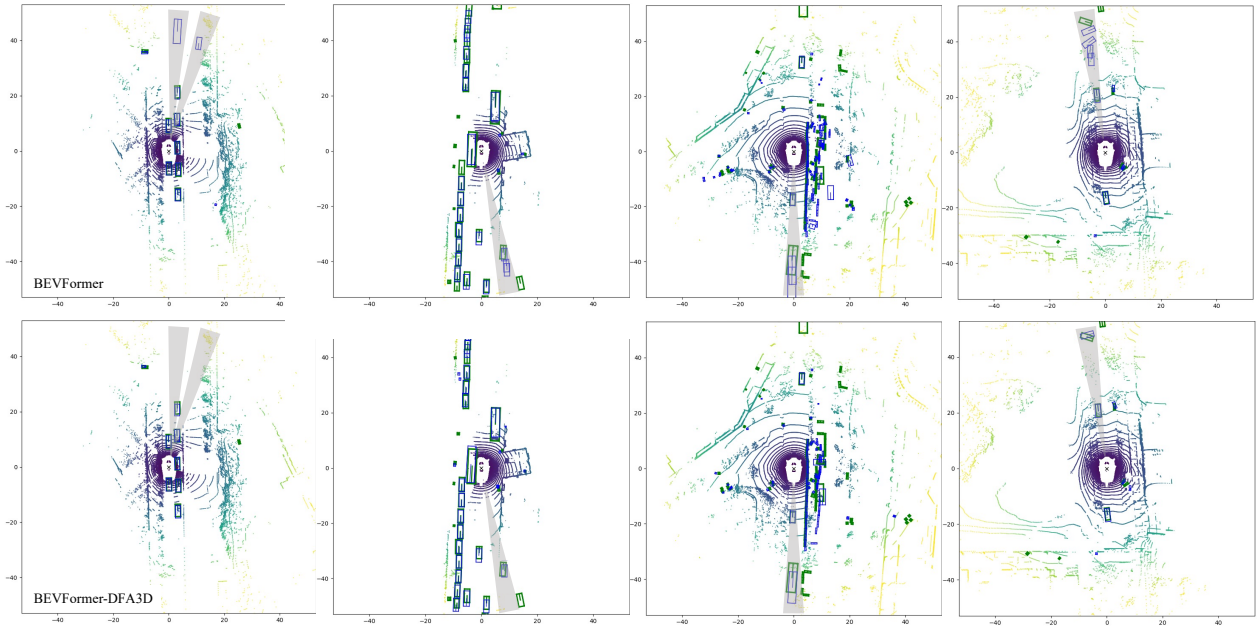

Figure 7: The visualization of predictions in BEV using green boxes to represent ground truth boxes and blue boxes to represent predicted ones. Duplicate predictions, which are caused by the depth ambiguity problem, are enclosed by shaded triangles.