Linking Modality Isolation in Heterogeneous Collaborative Perception

Supplementary Material

6. Implementation Details

6.1. Validation

We adopt two evaluation types in our experiments, detailed as follows:

Type 1 (Tables 3, 4, 20, 21, following HEAL): For each scene in the validation set, vehicles are first assigned sequential IDs (1-4). The modalities to be evaluated are then mapped onto these vehicles in order, with the first modality always assigned to the ego vehicle (ID = 1). For example, in a scenario labeled 'm1 + m2', modality m_1 is assigned to vehicle 1 (ego), and m_2 to vehicle 2 (neighbor); higher-numbered vehicles remain unmapped, so there are up to 2 cars in this scenario. Note that this fixed ordering affects data distribution, that modalities appearing later in the sequence occur less frequently across scenes. In scenes with fewer than four vehicles, unmapped IDs are simply omitted.

Type 2 (all other tables): To more comprehensively evaluate performance in multi-agent collaborative settings, we adopt: for a two-modality collaboration scenario mi + mj, we assign mi to the ego vehicle (ID = 1) and set all other collaboratable vehicles to modality mj. For multi-modality scenarios (as in Table 7), we fix the ego modality (m2) and iteratively assign each remaining modality as mj to all non-ego vehicles, performing inference multiple times and averaging the results to assess pairwise cross-modal alignment. For single-modality evaluation, all collaboratable vehicles are assigned the same modality and participate jointly in the collaboration.

For OPV2V, both training and evaluation are conducted within the spatial range: $x \in [-102.4m, +102.4m], y \in [-102.4m, +102.4m]$. And for DAIR-V2X, both ranges are: $x \in [-102.4m, +102.4m], y \in [-51.2m, +51.2m]$

6.2. Module Structure and Parameter Statistics

Table 8. Parameter for modality-specific encoders and modality-agnostic modules.

Modality	Encoder Type	#Param
m1	PointPillar	0.23M
m2	Second	0.97M
m3	VoxelNet	0.56M
m4	PointPillar	0.23M
m5	PointPillar	0.23M
m6	LSS(ResNet-101)	1.77M
m7	LSS(EfficientNet)	14.95M

Module	#Param
Adapter	0.30M
Codebook	0.02M
Translator	0.12M
Backend	5.27M
- Pyramid Fusion	3.79M
- Shrink Net	1.48M
- Heads	5k

Our setting accommodates a wide spectrum of heterogeneous modalities, including different sensor types (which is the most influential), diverse encoder networks, and LiDARs with different beam counts and varying voxel size. These differences lead to substantially divergent intermediate feature representations, making direct fusion challenging. Notably, encoder sizes vary dramatically, especially for vision-based modalities like m7, rendering full retraining of all encoders impractical in terms of computation requirements. In contrast, our alignment components (Adapter: 0.30M, Codebook: 0.02M, Translator: 0.12M) are extremely lightweight, easy to expand on a large scale. Moreover, the backend is frozen during code space construction, which drastically reduces training overhead while preserving strong detection capability. This modular and parameter-efficient strategy enables scalable, plug-and-play collaboration across highly heterogeneous sensing setups.

6.3. Inference Latency

Table 9 compares the inference efficiency of different methods on a single RTX 3090. CodeAlign achieves 0.111 s latency with 702 MB memory usage and 0.210 T FLOPs, showing competitive efficiency. Compared with HM-ViT, CodeAlign requires much lower latency, memory, and computation. Although HEAL is slightly more efficient, CodeAlign maintains a favorable efficiency-accuracy trade-off overall.

Table 9. Inference efficiency on 1× RTX 3090.

Method	Latency(s)	Memory Usage(MB)	FLOPs(T)
Late Fusion	0.149	460	0.282
HM-ViT	0.210	1866	0.369
HEAL	0.110	580	0.188
CodeAlign	0.111	702	0.210

7. More Performance Results of CodeAlign

7.1. General Experiments

In collaborative perception, CodeAlign demonstrates considerable generalizability. In Table 10, CodeAlign maintains the best performance in normal heterogeneous settings without modality isolation and achieve strong results in homogeneous scenes.

We validate CodeAlign on both simulated (OPV2V) and real-world (DAIR-V2X) datasets, which are both widely used in collaborative perception. While prior works such as HEAL and PolyInter consider only 4 modalities, we evaluate 7 modalities spanning mainstream sensor types, sensor configurations, and encoders. Appendix Table 13 evaluates 6 additional modality combinations. To ensure representativeness, most experiments focus on the largest cross-modal gap between LiDAR–camera collaboration. We evaluate on

Table 10. Performance in more general settings.

Method	Heterogeneous (m1+m6)			Homogeneous (m1)		
	AP30	AP50	AP70	AP30	AP50	AP70
Single	81.18	79.44	68.26	81.18	79.44	68.26
Late Fusion	84.03	80.36	62.83	93.00	91.92	80.84
DiscoNet	85.21	82.60	68.34	89.57	86.04	70.22
Attention Fusion	84.58	82.14	67.21	89.50	85.15	68.06
CoBEVT	86.98	85.14	71.78	93.73	91.06	76.60
V2X-ViT	86.64	84.40	69.58	**93.75**	92.26	79.42
HM-ViT	81.92	80.04	66.82	88.48	87.37	65.33
HEAL	82.45	81.03	71.70	92.55	91.81	**86.66**
CodeAlign(FCF)	87.96	86.00	72.92	93.11	**92.39**	85.34
CodeAlign(group)	**89.04**	**87.54**	**79.63**	/	/	/

as many modalities as the current datasets allow; nevertheless, we acknowledge in limitations that these datasets do not yet support larger-scale modality diversity testing, and we will continue to advance in future work.

7.2. Robustness under Communication Latency

Table 11. Robustness under communication latency (m1+m2).

Method	Latency=0ms		Latency=100ms		Latency=200ms	
	AP30	AP50	AP30	AP50	AP30	AP50
Late Fusion	92.73	91.46	90.04	81.57	81.17	59.13
CodeAlign	**93.39**	**92.67**	**92.75**	**89.36**	**86.28**	**75.28**

CodeAlign maintains robustness under communication latency. Table 11 shows that CodeAlign remains more robust under communication latency than Late Fusion. Although both methods degrade as latency increases, CodeAlign consistently achieves better AP30 and AP50 across all settings. The advantage becomes larger at higher latency, especially at 200 ms, where CodeAlign outperforms Late Fusion by a clear margin. This indicates that CodeAlign is more tolerant to delayed communication and can better maintain detection performance in asynchronous scenarios.

8. Detailed Experiments on CodeAlign

8.1. Code Space Construction

Table 12. Performance of different strategy in forming code space. Training methods include e2e training(E2E) and fix encoder and backbone training(Fix E&B). TP is training parameter.

Training	Adapter	m1				m6			
		AP30	AP50	AP70	TP/M	AP30	AP50	AP70	TP/M
E2E	✗	95.07	94.64	**91.05**	5.50	55.08	45.51	25.19	7.04
Fix E&B	✗	93.69	93.05	85.54	0.02	58.71	49.89	29.51	0.02
Fix E&B	✓	**95.2**	**94.65**	89.37	0.30	**58.85**	**50.30**	**31.10**	0.30

Table 12 shows the effect of different code space setups. Freezing the encoder and backend (instead of end-to-end training) reduces trainable parameters and even improves performance. For LiDAR (m_1), CodeAlign boosts AP30 and AP50 but slightly drops AP70 by 1.68%, likely because the fixed codebook doesn't align perfectly with the fusion module, hurting spatial accuracy. For camera (m_6), it largely improves AP70 by 5.91%, as the codebook captures cleaner and more compact visual features.

Table 13. More ablation studies on code space construction of a single modality.

Method		m1			m6		
Strategy	Adapter	AP30	AP50	AP70	AP30	AP50	AP70
E2E	✗	95.07	94.64	91.05	55.08	45.51	25.19
Fix Encoder	✗	95.36	94.57	85.34	59.39	50.85	31.34
	4×ResBlock	94.78	94.52	**91.17**	59.13	51.09	**33.04**
	1×ConvNeXt block	**95.62**	**95.14**	90.4	59.37	51.25	31.48
	2-layer MLP	95.45	94.91	89.67	**59.4**	**51.29**	32.23
Fix Encoder &Backend	✗	93.69	93.05	85.54	58.71	49.89	29.51
	4×ResBlock	**95.2**	**94.65**	**89.37**	58.85	50.3	**31.1**
	1×ConvNeXt block	95.13	94.63	88.59	**59.4**	**50.46**	30.94
	2-layer MLP	94.40	93.65	86.03	59.11	50.2	29.71

Table 13 presents a more detailed ablation study, including variants that only fix the encoder and comparisons of different adapter designs. The experiments show that fixing only the encoder while allowing the backend to be trained yields slightly better performance, as the backend can adapt to the relatively discrete features produced by the codebook; notably, AP70 even surpasses that of full end-to-end training in some cases, likely because the discretized representations suppress certain noise. In contrast, fixing the backend incurs a modest drop of nearly 2% in AP70 but dramatically reduces training cost, making the approach more scalable. Regarding adapter architectures—ResBlock, ConvNeXt, and MLP—their parameter counts decrease progressively. As shown in the table, lightweight adapters already achieve satisfactory AP30 and AP50 scores, but for fine-grained localization (AP70), a larger ResBlock-based adapter remains necessary to properly align cross-modal representations.

To comprehensively evaluate the code space construction across all modalities, we conduct the experiments shown in Table 15. As demonstrated, for every modality, our proposed plug-in manner code space construction method achieves comparable perception performance to the end-to-end trained Pyramid Fusion, while explicitly extracting a codebook-based feature representation. This approach introduces only minimal additional training overhead and reduces communication cost by orders of magnitude.

8.2. Feature-Code-Feature Translation

Table 16 shows the performance of various modalities aligned to m1 via feature-code-feature translation. Despite significant heterogeneity in sensor types and encoder architectures, all agents consistently enhance perception performance over the m1-single baseline. Notably, collaborations with LiDAR-based neighbors (m2–m5) yield substantial gains (e.g., +14.31 AP30 with m2), while even camera-based modalities (m6, m7) provide meaningful improvements (+7.0 AP30). These results demonstrate that feature-code-feature translation enables diverse agents to

Table 14. Ablation on input source and output type of translator.

Source	Type	m1 + m2			m1 + m6			Comm Load
		AP30	AP50	AP70	AP30	AP50	AP70	
Encoded Feature	D2D	**95.65**	**95.06**	**90.72**	**88.93**	**86.96**	**75.04**	32MB
Encoded Feature	D2C	95.44	94.88	89.83	88.13	86.41	73.47	0.03MB
Adapted Feature	D2C	95.49	94.9	90.11	88.28	86.42	73.56	0.03MB
Reconstructed Feature	D2C	93.96	93.23	85.87	76.13	73.42	53.89	0.03MB
Code Map	C2C	94.58	93.87	87.4	76.93	74.26	55.16	0.03MB

Table 15. Single-modality performance comparison between end-to-end Pyramid Fusion and code space construction.

Modality	E2E Pyramid Fusion			Code Space Construction		
	AP30	AP50	AP70	AP30	AP50	AP70
m1	94.69	94.14	90.29	95.2	94.65	89.37
m2	94.93	94.54	91.72	94.74	94.31	90.23
m3	95.86	95.46	91.51	95.00	94.47	88.89
m4	89.26	88.66	83.68	86.79	85.82	77.47
m5	95.02	94.60	89.94	94.71	93.57	87.96
m6	58.11	50.79	33.72	58.85	50.30	31.10
m7	61.93	54.27	37.17	61.93	54.22	35.57

Table 16. Performance of more modalities aligned to m1 when m1 is ego with feature-code-feature translation.

Scenario	AP30	AP50	AP70
m1 single	81.18	79.44	68.26
m1 + m2	95.49	94.90	90.11
m1 + m3	95.22	94.67	89.48
m1 + m4	94.63	93.93	87.61
m1 + m5	95.06	94.53	89.23
m1 + m6	88.13	86.41	73.47
m1 + m7	88.11	86.59	74.47

participate effectively in cooperative perception with minimal communication overhead, significantly boosting accuracy without requiring retraining of existing components.

Table 17. Feature-code-feature translation can be trained under single-agent data. We show an example of scene m1+m2.

Data Type	AP30	AP50	AP70
Collaborative Data	95.44	94.88	89.83
Single Data	94.83	94.24	88.09

A key advantage of our framework is that the code translator can be trained using only single-agent (non-collaborative) data, eliminating the need for synchronized multi-vehicle scenes during training. This greatly enhances practicality in real-world settings where collaborative data is scarce or unavailable. Table 17 validates this capability. When trained solely on single-agent data, the translator achieves 94.83 AP30 and 94.24 AP50. Compared to training with collaborative data, it leads to a performance drop of less than 1 in AP50. This minor degradation demonstrates that high-quality cross-modal alignment can be learned from local perception data alone, making our method highly adaptable to diverse deployment scenarios.

Table 14 analyzes the impact of input source and output type on translation performance. The D2D (dense-to-dense) variant achieves the highest accuracy by preserving full feature fidelity, but incurs a prohibitive communication cost of 32MB, failing to address the core bottleneck. In contrast, D2C (dense-to-code) and C2C (code-to-code) translation drastically reduce communication to just 0.03MB by transmitting compact code maps. However, C2C suffers severe performance degradation (AP70 drop by nearly 20 for m1+m6) due to excessive information loss in direct code-to-code mapping. Among D2C variants, using raw encoded features yields strong results, while reconstructed features suffer significant drops like C2C with similar reason. Notably, adapted features achieve better performance, slightly outperforming raw encoded features and confirming that adapter enhances cross-modal alignment.

Table 18 shows the effect of codebook size on perception performance. As the codebook size grows from 4 to 16, AP metrics consistently improve for all settings, indicating that a larger codebook captures richer and more discriminative semantic information. Beyond size 16, however, performance plateaus, suggesting that 16 entries are sufficient to represent the essential environmental semantics in this setting. Moreover, the mapping difficulty increases with codebook size growing, where bigger codebook even leads to lower translation performance. Thus, a codebook size of 16 achieves the optimal trade-off between representation capacity and communication efficiency.

8.3. Group Code Space Construction

Table 19 presents ablation studies on group code space construction. The end-to-end Pyramid Fusion baseline achieves the highest performance, as it incurs no information loss. Introducing a codebook alone preserves coarse-level accuracy but degrades fine-grained localization (AP50 drops by 0.29, AP70 by 2.57), indicating quantization harms spatial detail. By freezing the encoder, adding an adapter, and incorporating a similarity loss L_{sim}, performance across all metrics improves significantly, nearly recovering the baseline while enabling compact feature representation. We evaluate multiple similarity losses; all consistently enhance

Table 18. Ablation on different codebook sizes.

Codebook Size	m1			m6			m1 + m6		
	AP30	AP50	AP70	AP30	AP50	AP70	AP30	AP50	AP70
4	94.01	93.37	84.80	56.88	47.90	26.84	84.49	82.48	66.30
8	94.85	94.24	87.94	58.07	49.36	30.07	86.86	84.59	69.49
16	95.20	94.65	89.37	58.85	50.30	31.10	88.13	86.41	73.47
32	95.02	94.42	89.09	58.56	49.84	29.20	88.23	86.08	73.82
64	95.18	94.72	89.75	59.02	50.55	31.93	87.75	85.98	74.56

Table 19. More ablation studies on group code space construction, especially on similarity losses, in group (m1,m6).

Strategy	L_{sim}	AP30	AP50	AP70
E2E	\	89.3	**88.03**	**80.44**
+ Codebook(16)	\	**89.54**	87.74	77.87
++ Fix Enc ++ Add Adapter	\	88.37	86.83	78.09
	L2	88.82	87.31	78.88
	Instance	88.99	87.01	76.08
	Reconstruction	88.89	87.36	79.06
	Cosine	88.66	87.29	79.54
	Smooth L1	**89.04**	**87.54**	79.63
	MMD	88.76	87.28	**79.75**
	Coral	88.73	87.23	79.03
	JS	89.01	87.50	79.14

alignment over the no-loss variant. Smooth L1 Loss yields the best overall results, facilitating stable and precise alignment of heterogeneous feature distributions.

8.4. Group Feature-Code-Feature Translation

Table 20. Alignment performance between group (m1,m6) and group (m2,m7) in order [m1,m7,m2,m6].

	AP30	AP50	AP70	Comm Load
Late Fusion	89.61	86.47	69.97	0.5KB
HEAL	92.50	91.74	85.64	32MB
CodeAlign	92.27	91.48	85.42	0.03MB

Table 20 demonstrates CodeAlign's effectiveness in cross-group alignment. Here, modalities m1 and m6 share a common codebook, and the translator aligns m2, m7 into this common code space. Despite the heterogeneity across groups, CodeAlign achieves performance on par with HEAL, while reducing communication overhead. This validates that inter-group alignment via a shared discrete code space is both feasible and efficient.

Table 21 highlights a critical limitation of HEAL. Specifically, HEAL requires retraining encoders of non-base groups to align them with a single base group, disrupting the original intra-group alignment. This process degrades the internal consistency and performance within those groups. For instance, when m2 and m7 are realigned to the base group (m1, m6), their mutual alignment drop in AP70 by 2.82, compared to the original pyramid fusion performance.

Table 21. HEAL degrades the alignment between the original group after being aligned to other groups. In order [m2,m7]

	AP30	AP50	AP70
Late Fusion	75.25	69.00	50.20
HEAL(pyramid fusion)	**85.73**	**84.21**	**76.41**
HEAL(backward alignment)	83.76	82.55	73.59

9. Comparison to Other Methods

The comparison in Table 22 highlights fundamental limitations of existing heterogeneous collaborative perception methods in handling modality isolation and practical deployment. V2X-ViT, HM-ViT, and MPDA all rely on end-to-end attention-based fusion, which fundamentally assumes that input features from different agents are spatially aligned, which is a condition only satisfied when modalities co-occur in the same scene during training. Similarly, PnPDA employs contrastive loss to align heterogeneous features, but this alignment also requires paired, spatially corresponding data from multiple modalities in identical scenarios. PolyInter leverages channel selection and spatial attention modules, yet these components still depend on co-occurring modalities to learn meaningful cross-agent interactions.

STAMP introduces a protocol network to unify representations, but it mandates that the protocol modality be present in every scene so that other modalities can align through feature similarity, making it inapplicable when the protocol agent is absent. CodeFilling attempts to learn a shared codebook across all modalities, but this requires simultaneous access to all modalities during training; consequently, its performance degrades significantly under modality isolation, and it lacks extensibility to new sensors without retraining the entire system.

HEAL addresses modality isolation via backward alignment, retraining each encoder locally to map into a common space. While effective, this approach incurs high computational cost and operational inconvenience where every new modality or update necessitates full encoder retraining. In contrast, our method, CodeAlign, decouples representation learning by constructing modality-specific code spaces and

Table 22. Comparison of heterogeneous collaborative perception methods in terms of encoder/sensor diversity, extensibility, and adaptability to Modality Isolation. Remarks detail limitations regarding isolation handling and key drawbacks.

Method	Dif Enc	Dif Sensor	Extensible	M.I. Adapt	Remarks
V2X-ViT [24]	✓				E2E attention fusion; needs same-scene features for spatial attention.
HM-ViT [19]		✓			E2E attention fusion; needs same-scene features for spatial attention.
MPDA [22]	✓				E2E attention fusion; needs same-scene features for spatial attention.
PnPDA [13]	✓				Contrastive loss for hetero features requires co-occurring modalities.
PolyInter [17]	✓		✓		Channel selection and spatial attention module need co-occurring modalities.
STAMP [4]	✓	✓	✓		Aligns to protocol net; protocol modality must be present in every scene.
CodeFilling [8]	✓	✓			All modalities needed to train shared codebook; degrades with isolated modalities.
HEAL [11]	✓	✓	✓	✓	Supports modality isolation; but needs encoder retraining with high training cost.
CodeAlign	✓	✓	✓	✓	Supports modality isolation; plug-in design; training-efficient, communication-efficient.

establishing a lightweight feature-code-feature translation pipeline. This design eliminates the need for co-occurring data during training, supports plug-and-play integration of new modalities, and enables both training-efficient adaptation and ultra-low communication overhead. These advantages make it uniquely suited for real-world heterogeneous multi-agent systems under dynamic and incomplete observation conditions.