SPECUNIFY: LEARNING UNIFIED SPECTRAL REPRESENTATIONS FOR RGB, MULTISPECTRAL, AND HYPERSPECTRAL VISION

Anonymous authors
Paper under double-blind review

ABSTRACT

RGB, multispectral, and hyperspectral images provide different spectral observations of the same visual world, yet modern vision models inherit the sensor's channel configuration as the basis for downstream representation. This couples how a scene is measured with how it is represented for a task, encouraging modality-specific projections, fixed spectral reductions, or separate models and limiting the sharing of representations and supervision across spectral modalities. We ask whether these roles can instead be separated: can heterogeneous spectral observations be mapped into a common task representation whose cardinality is learned from downstream supervision rather than prescribed by the sensor? We present SpecUnify, a unified framework for learning spectral representations across RGB, multispectral, and hyperspectral vision. SpecUnify learns a common spectral representation across modalities by lifting RGB observations into the shared space while directly incorporating multispectral and hyperspectral observations. Within this shared representation, it learns the cardinality of the task representation jointly with the downstream objective, rather than fixing it according to the sensor's channel configuration or a manually chosen spectral reduction. The resulting variable-cardinality representation is converted into fixed-dimensional tokens compatible with pretrained vision models, without reconstructing physical hyperspectral measurements or introducing separate cross-modal pretraining. We evaluate SpecUnify on semantic segmentation and visual tracking under within-modality, cross-modal, and joint RGB-hyperspectral training. The results show that different spectral observations can be processed through a common task representation while maintaining competitive performance across modalities and tasks, and that joint training can exploit complementary supervision from heterogeneous spectral observations. These findings suggest a broader principle for spectral vision: sensor dimensionality need not determine task representation, and heterogeneous spectral observations can instead be unified through a task-learned representation whose cardinality is determined by the downstream objective.

1 INTRODUCTION

An image is not intrinsically a three-channel tensor (Bhargava et al., 2024; Hong et al., 2026). Its channel configuration is determined by how a sensor samples the visual world: RGB cameras provide a small number of broad spectral measurements, multispectral sensors capture selected spectral bands, and hyperspectral sensors acquire much denser spectral observations. These modalities therefore provide different spectral observations of visual scenes, while the downstream task itself does not fundamentally change with the sensor. A road still needs to be segmented and an object still needs to be tracked whether the scene is observed through RGB, multispectral, or hyperspectral measurements. This raises a fundamental question: *should a task representation have to inherit the dimensionality and organization of the sensor that produced the input?*

Modern vision models typically couple the two (Oquab et al., 2023; Siméoni et al., 2025; Li et al., 2025a). Their input interfaces are designed around a particular channel configuration (Hong et al., 2024; Ji et al., 2026), making the sensor's representation dimensionality part of the downstream model. For heterogeneous spectral inputs, one approach is to project or reduce observations to a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

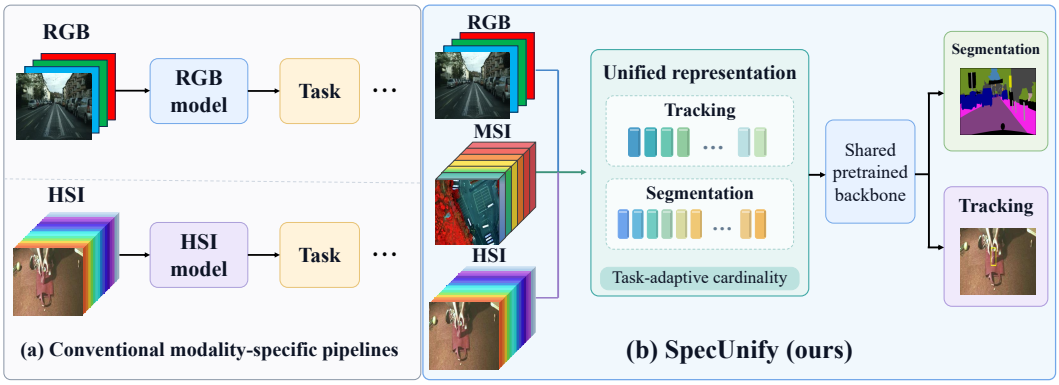

Figure 1: From modality-specific models to unified spectral representations. (a) Separate pipelines process RGB and hyperspectral observations. (b) SpecUnify unifies RGB, multispectral, and hyperspectral inputs through a representation whose content and cardinality are learned from downstream supervision, decoupling task representation from sensor channel count. Within each task, the representation learner and pretrained backbone are shared across modalities.

prescribed representation, such as an RGB-like input. This provides compatibility with existing backbones, but can impose a restrictive intermediate bottleneck. Another approach is to develop modality-specific input adapters (Li et al., 2025b) or separate models(Li et al., 2025a; Chen et al., 2024; Li et al., 2024), which preserves modality-specific observations but limits the sharing of representations and supervision across sensors. More broadly, these approaches treat the sensor's channel configuration as a property that the task model should accommodate, rather than an observation interface that can be separated from the representation required for prediction.

We take a different view: *spectral sensing and task representation are distinct roles*. The sensor determines what spectral observations are available, whereas the downstream task should determine how those observations are organized into a useful representation. Under this view, RGB, multispectral, and hyperspectral inputs need not retain their native channel dimensionalities throughout the network, and hyperspectral observations need not be reduced to a manually prescribed number of components. Instead, heterogeneous spectral observations can be transformed into a common learned representation, with its content and cardinality determined by downstream task supervision. This separation provides a principled way to share representations and supervision across spectral modalities while allowing each sensor to contribute through its native observations. We instantiate this idea in SpecUnify (Fig. 1), a unified framework for spectral representation learning across RGB, multispectral, and hyperspectral vision. SpecUnify learns a common spectral representation across modalities by lifting RGB observations into the shared space while directly incorporating multispectral and hyperspectral observations. A shared representation module then generates a set of learned task-representation components. Crucially, its cardinality is learned jointly with the downstream objective rather than prescribed by the sensor's native channel count or a manually chosen spectral reduction. Thus, the number of measured channels and the number of learned representation components are explicitly separated: $C_{\text{in}} \neq K$, where C_{in} denotes the sensor's input channel dimensionality and K denotes the cardinality of the learned task representation. The generated components are learned task features, not physical spectral bands: they need not correspond to particular wavelengths, selected input bands, or reconstructions of unobserved hyperspectral measurements.

This formulation introduces two technical challenges. First, K is a discrete integer that determines the shape of the intermediate representation, preventing ordinary backpropagation through the cardinality decision. SpecUnify therefore retains a hard integer cardinality in the forward pass and uses a local surrogate derived from neighboring representation widths to propagate task supervision to the cardinality predictor. Second, a variable number of representation components should interface with pretrained vision transformers that require fixed-dimensional patch tokens. SpecUnify addresses this mismatch by directly mapping the variable-cardinality representation into the backbone token space through a shared patch projection and a dynamically generated low-rank residual, rather than collapsing the representation back to a fixed three-channel image. The resulting token sequence has

a fixed spatial length and embedding dimension independent of K, allowing the same representation mechanism to interface with pretrained vision models across spectral modalities.

The distinction from existing heterogeneous-channel methods is therefore important. Supporting arbitrary input channel counts addresses *how a model can accept different sensors*; SpecUnify additionally addresses *how those observations should be represented for a downstream task*. Likewise, variable cardinality alone is not the central contribution. For a requested cardinality K, SpecUnify generates the corresponding representation as a whole, conditioned jointly on the input and K, rather than selecting a prefix or subset from a fixed bank of output components. Changing K can therefore change the organization of the entire representation. The learned cardinality is part of the task representation rather than merely a computational truncation. This shared representation enables a common learning framework across spectral modalities. Within-modality training tests whether the unified representation can support each spectral source on its native task. Cross-modal transfer tests whether a model trained from one spectral observation can be applied to another without target-domain task supervision. Joint training allows heterogeneous spectral observations to contribute supervision to the same task representation without requiring paired or synchronized multimodal inputs at inference. We study these settings in semantic segmentation and visual tracking, testing whether the proposed separation between spectral sensing and task representation transfers across dense prediction and object-level temporal prediction.

SpecUnify advances a simple principle for spectral vision: *the dimensionality of what a sensor measures need not determine the dimensionality of what a task learns to represent*. RGB, multispectral, and hyperspectral observations can instead be unified through a common task representation whose cardinality is learned from downstream supervision rather than inherited from the sensing modality. This decoupling provides a general route for sharing task representations and supervision across heterogeneous spectral observations while retaining compatibility with pretrained vision models. Our contributions are summarized as follows:

i. We formulate spectral representation learning as a separation between sensor observations and task representation, enabling RGB, multispectral, and hyperspectral inputs to share a common learned representation without requiring their native channel dimensionalities to determine the downstream representation.

ii. We introduce a variable-cardinality representation whose number of learned task components is determined jointly with the downstream objective rather than fixed by the sensor channel count or a manually chosen spectral reduction. The corresponding representation is generated as a complete set conditioned on both the input and its cardinality.

iii. We develop a discrete-cardinality learning mechanism based on neighboring-width task-gradient surrogates and a variable-to-fixed token interface that maps the learned representation directly into pretrained transformer tokens without physical hyperspectral reconstruction or separate cross-modal pretraining.

iv. We evaluate SpecUnify under within-modality, cross-modal, and joint RGB-hyperspectral training for semantic segmentation and visual tracking, with controlled comparisons designed to distinguish the effects of learned representation cardinality and shared spectral supervision.

A discussion of related work and its positioning relative to existing spectral representation methods is provided in Appendix A. Below, we introduce SpecUnify and its key components.

2 METHOD

We propose SpecUnify, a unified framework (see Fig. 2) that separates spectral sensing from task representation. Given heterogeneous spectral observations, SpecUnify preserves the native channel configuration of multispectral and hyperspectral inputs while lifting RGB observations into a learned channel space. All inputs are then processed by a shared channel encoder to construct a task representation whose cardinality is learned jointly with the downstream objective, rather than inherited from the sensor's channel configuration. The resulting variable-cardinality representation is mapped to fixed-dimensional tokens compatible with a pretrained vision backbone. Thus, SpecUnify separates three roles that are typically coupled in spectral vision: *how a scene is measured*, *how it is represented for a task*, and *how that representation is interfaced with a pretrained model*.

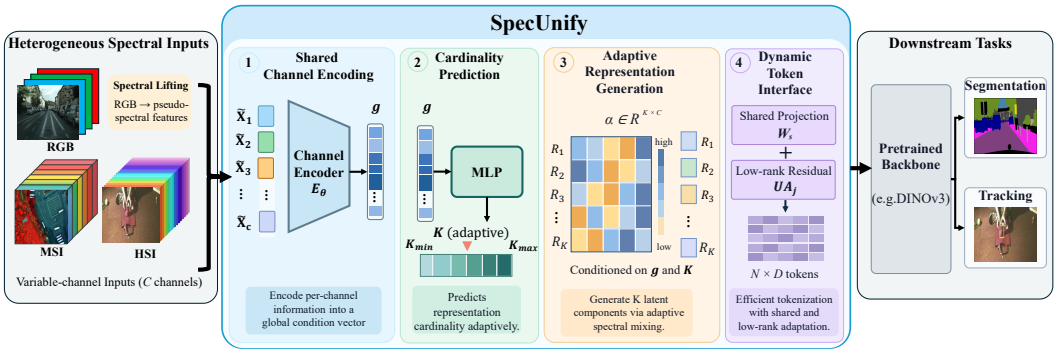

Figure 2: Overview of SpecUnify. A shared channel encoder summarizes heterogeneous RGB, MSI (Multispectral image), and HSI(hyperspectral image) inputs into a global descriptor $\mathbf{g}$. Downstream supervision learns an input-dependent cardinality K, and the generator constructs the complete set of K representations conditioned on both $\mathbf{g}$ and K. A dynamic token interface combines a shared projection with a component-conditioned low-rank residual to produce an $N \times D$ token sequence for pretrained backbones. This decouples sensor channel count, task-representation cardinality, and backbone embedding dimension.

2.1 Shared Channel Representation

SpecUnify first establishes a shared channel representation for heterogeneous spectral observations without requiring them to have the same native number of channels. Let $\mathbf{X} \in \mathbb{R}^{C_{\text{in}} \times H \times W}$ denote an input image, where H and W denote the spatial dimensions and C_{in} is the native number of sensor channels. RGB observations require a learned lifting step, whereas multispectral and hyperspectral observations retain their native channels and are processed directly. The resulting channel representations are then encoded by a shared channel encoder, providing a common conditioning representation for the subsequent task-representation generator.

RGB spectral lifting. RGB provides only three broad spectral measurements. We therefore use a lightweight point-wise mapping $L_\psi : \mathbb{R}^3 \to \mathbb{R}^C$ to lift each RGB observation into a learned C-channel space: $\widetilde{\mathbf{X}} = L_\psi(\mathbf{X}) \in \mathbb{R}^{C \times H \times W}$. Here, C is set to the average number of spectral bands across the hyperspectral datasets used in our study. The lifted channels are learned features rather than reconstructed physical spectral bands. Thus, the lifting step provides RGB with a richer learned channel representation without claiming to recover unobserved hyperspectral measurements. For multispectral and hyperspectral inputs, no analogous lifting is applied: $\widetilde{\mathbf{X}} = \mathbf{X}$. Their native spectral observations are therefore preserved and passed directly to the shared channel encoder. SpecUnify consequently does not require different sensors to be converted to a common number of raw channels; their commonality is established through the shared channel-level representation.

Shared channel encoding. We regard the channels of $\widetilde{\mathbf{X}}$ as a set of channel-wise observations, $\widetilde{\mathbf{X}} = \left\{\widetilde{\mathbf{X}}_i\right\}_{i \in \mathcal{V}}$, where $\mathcal{V}$ indexes the valid channels of the current input. For RGB, $\mathcal{V}$ indexes the C lifted channels, whereas for multispectral and hyperspectral inputs it indexes their native channels. Each channel is processed independently by the same encoder E_θ:

$$\boldsymbol{e}_i = E_\theta(\widetilde{\mathbf{X}}_i) \in \mathbb{R}^d, \qquad i \in \mathcal{V}. \tag{1}$$

The resulting channel descriptors are aggregated into a fixed-dimensional image-level descriptor:

$$\boldsymbol{g} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \boldsymbol{e}_i \in \mathbb{R}^d. \tag{2}$$

Because the same encoder is applied to every channel and mean aggregation removes the channel dimension, $\boldsymbol{g}$ has a fixed dimension across sensing modalities. This descriptor provides a shared conditioning signal for the subsequent task-representation generator, while the channel-wise descriptors $\{\boldsymbol{e}_i\}$ and input maps $\{\widetilde{\mathbf{X}}_i\}$ remain available for spectral aggregation.

2.2 VARIABLE-CARDINALITY REPRESENTATION

The shared channel representation decouples the task representation from the sensor's native channel count, but it does not determine how many components the representation should contain. SpecUnify therefore learns the number of task-representation components from downstream supervision. Given the image descriptor $\boldsymbol{g}$, a scalar predictor $a = f_\theta(\boldsymbol{g})$ determines the representation cardinality:

$$s_K = K_{\min} + (K_{\max} - K_{\min})\sigma(a), \qquad K = \mathrm{round}(s_K), \tag{3}$$

where σ denotes the sigmoid function. The continuous estimate s_K is used for optimization, while the integer K determines how many components are generated in the forward pass.

Whole-set representation generation. We generate the complete representation conditioned on both the input and its cardinality. For each component $j \in \{1, \dots, K\}$, we define

$$u_j = \frac{j - \frac{1}{2}}{K}, \qquad \boldsymbol{c}_j = [\boldsymbol{g}; \phi(u_j); E_K(\log K)], \tag{4}$$

where ϕ is a Fourier positional encoding and E_K is a learned cardinality embedding. Changing K updates both the component coordinates and the cardinality embedding, reconditioning the entire set rather than truncating a fixed representation.

Content-adaptive spectral aggregation. A shared condition-to-parameter network H_θ generates a query and modulation parameters for each component:

$$[\boldsymbol{q}_j, \boldsymbol{\gamma}_j, \boldsymbol{\beta}_j, \boldsymbol{w}_j, b_j] = H_\theta(\boldsymbol{c}_j), \tag{5}$$

where $\boldsymbol{q}_j \in \mathbb{R}^d$, $\boldsymbol{\gamma}_j, \boldsymbol{\beta}_j, \boldsymbol{w}_j \in \mathbb{R}^s$, and $b_j \in \mathbb{R}$. A shared learned projection maps channel descriptors to keys $\boldsymbol{k}_i = \boldsymbol{W}_e \boldsymbol{e}_i + \boldsymbol{b}_e \in \mathbb{R}^d$. Query-key similarities determine the contribution of each valid input channel:

$$\alpha_{ji} = \mathrm{softmax}_{i \in \mathcal{V}} \left(\frac{\boldsymbol{q}_j^\top \boldsymbol{k}_i}{\sqrt{d}} \right). \tag{6}$$

These weights form a component-specific mixture of the input maps:

$$\mathbf{M}_j = \sum_{i \in \mathcal{V}} \alpha_{ji} \widetilde{\mathbf{X}}_i \in \mathbb{R}^{H \times W}. \tag{7}$$

Spatial refinement and component modulation. The component-specific mixture $\mathbf{M}_j$ is processed by a shared spatial encoder:

$$\mathbf{V}_j = S_\theta(\mathbf{M}_j) \in \mathbb{R}^{s \times H \times W}. \tag{8}$$

The modulation parameters generated from the same condition $\boldsymbol{c}_j$ then transform and combine these spatial features:

$$\mathbf{R}_j(p) = \frac{1}{\sqrt{s}} \sum_{a=1}^{s} w_{j,a} \,\mathrm{GELU}\left((1 + \gamma_{j,a}) V_{j,a}(p) + \beta_{j,a}\right) + b_j. \tag{9}$$

Here, $\gamma_{j,a}$, $\beta_{j,a}$, and $w_{j,a}$ denote the scalar entries of the corresponding vectors $\boldsymbol{\gamma}_j$, $\boldsymbol{\beta}_j$, and $\boldsymbol{w}_j$. Thus, the same condition $\boldsymbol{c}_j$ controls both the spectral aggregation of component j and its subsequent spatial transformation. Generating all K components yields the complete task representation

$$\mathbf{R}_K = [\mathbf{R}_1, \dots, \mathbf{R}_K] \in \mathbb{R}^{K \times H \times W}. \tag{10}$$

2.3 DYNAMIC TOKEN INTERFACE AND CARDINALITY LEARNING

To use the generated variable-cardinality representation $\mathbf{R}_K$ for downstream prediction, we introduce a dynamic token interface to pretrained vision backbones. It combines a shared patch projection with component-conditioned low-rank residuals to aggregate all K components into fixed-dimensional patch tokens. The residuals use the same conditions as representation generation, adapting each component's contribution while keeping the backbone interface independent of K.

Dynamic token interface. Let $\mathbf{R}_{j,n} \in \mathbb{R}^{P^2}$ denote the flattened $P \times P$ patch of component j at spatial location n, and let D be the backbone embedding dimension. We use a trainable shared

Table 1: Semantic segmentation results on HSI-City v2 (%).

Method	Source	Trainable params (M)	Acc_{μ}	$\mathrm{Acc}_{\mathrm{M}}$	$\mathrm{F1}_{\mathrm{M}}$	J_{M}
CGRSeg (2024) (false-color)	R	19.1	89.56	56.31	63.46	47.07
U-Net (2015)	H	17.3	88.95	54.25	56.28	45.63
DeepLabV3+ (MobileNet) (2018; 2018)	H	5.2	91.49	56.56	60.13	49.71
DeepLabV3+ (ResNet101) (2018; 2016)	H	58.8	91.76	58.57	62.23	51.85
RU-Net (2024)	H	21.2	93.30	63.13	66.45	56.11
HyperFree (2025a)	H	15.9	90.14	58.88	60.56	49.07
HyperVision (2026)	H	15.9	93.41	66.05	67.87	57.04
CARL frontend + LiDeRe (2026)‡	H	15.8	96.08	68.48	71.07	62.70
SpecUnify + LiDeRe (Ours)	R	**1.56**	94.45	67.29	68.35	59.21
	H	**1.56**	99.09	88.22	89.11	85.26
	R+H	**1.56**	**99.13**	**89.26**	**90.18**	**86.88**

‡CARL spectral-encoder adaptation with the same HSI-only training setting as SpecUnify.

projection $\boldsymbol{W}_{\mathrm{s}} \in \mathbb{R}^{D\times P^2}$, initialized by averaging the three input-channel kernels of the pretrained RGB patch embedding. To make the projection component-specific, a hypernetwork generates a low-rank residual from the same condition $\boldsymbol{c}_j$ used for representation generation:

$$\boldsymbol{A}_j = \mathrm{reshape}(H_A(\boldsymbol{c}_j)) \in \mathbb{R}^{r\times P^2}, \qquad \boldsymbol{U} \in \mathbb{R}^{D\times r}, \tag{11}$$

where r is the residual rank and $\boldsymbol{U}$ is a shared trainable matrix. The token at location n is

$$\boldsymbol{t}_n = \boldsymbol{b} + \frac{3}{K}\boldsymbol{W}_{\mathrm{s}}\sum_{j=1}^{K}\mathbf{R}_{j,n} + \frac{\alpha}{rK}\boldsymbol{U}\sum_{j=1}^{K}\boldsymbol{A}_j\mathbf{R}_{j,n}, \tag{12}$$

where $\boldsymbol{b} \in \mathbb{R}^D$ is the patch bias and α controls the residual scale. Each component therefore contributes through a shared projection and a condition-dependent correction. The resulting tokens form $\boldsymbol{T} \in \mathbb{R}^{N\times D}$, with $N = (H/P)(W/P)$. At a fixed spatial resolution, both N and D are independent of K.

Learning discrete cardinality. To learn K from downstream supervision despite hard rounding, we introduce a neighboring-cardinality surrogate. The token interface maps representations of different cardinalities to the same shape for a given input, allowing us to estimate how changing K affects the backbone input. Let $\boldsymbol{T}^{(k)}$ denote the tokens generated for the same input at cardinality k. Using the admissible neighbors $k_- = \max(K_{\min}, K-1)$ and $k_+ = \min(K_{\max}, K+1)$, we estimate

$$\Delta_K = \frac{\boldsymbol{T}^{(k_+)} - \boldsymbol{T}^{(k_-)}}{k_+ - k_-}. \tag{13}$$

With $\mathrm{sg}(\cdot)$ denoting stop-gradient, the training-time token sequence is

$$\widehat{\boldsymbol{T}} = \boldsymbol{T}^{(K)} + \big(s_K - \mathrm{sg}(s_K)\big)\,\mathrm{sg}(\Delta_K). \tag{14}$$

The forward value remains exactly $\boldsymbol{T}^{(K)}$, while the backward pass propagates task supervision to s_K through the estimated direction Δ_K. Neighboring-width evaluations involve only the frontend and token interface, requiring no additional backbone forward passes. Initialization and gradient details are provided in Appendix B.2.

Width exploration. During early training, we sample a cardinality uniformly from $\{K_{\min}, \ldots, K_{\max}\}$ with probability p_t and otherwise use the predicted K. This trains the generator at alternative cardinalities, complementing the detached neighboring-width probes. The probability p_t decays to zero, and evaluation uses only the predicted cardinality.

Optimization. SpecUnify is optimized directly with downstream task losses. Its parameters are shared across modalities and datasets within each task, while segmentation and tracking are trained separately. We use LoRAT (Lin et al., 2024) for tracking and DINOv3 (Siméoni et al., 2025) with LiDeRe (Lüddecke et al., 2026) for segmentation, keeping the pretrained backbone base weights frozen. Task-specific integration is detailed in Appendix B.3.

Table 2: Object tracking results on HOTC2023 (%).

Method	Trainable Params(M)	AUC ↑	DP_{20} ↑
MHT 2020	–	45.3	71.7
BAE-Net (2020)	17.7	50.7	77.7
TSCFW (2022)	–	46.9	70.3
SiamF (2022)	54.8	51.1	76.7
SiamBAG (2023a)	90.44	51.2	73.9
SEE-Net (2023b)	53.9	51.9	76.3
SENSE (2024)	55.7	55.9	76.6
SSTtrack (2025)	55.7	62.5	81.9
HyperFree (2025a)	25.9	56.2	76.9
HyperVision (2026)	25.9	57.1	78.5
SpecUnify+LoRAT	**13.35**	**67.7**	**86.7**

Table 3: Object tracking results on multi-spectral dataset MUST (%).

Method	AUC ↑	DP_{20} ↑
BAE-Net (2020)	38.0	53.5
ViPT (2023)	58.5	72.6
SEE-Net (2023b)	41.8	59.2
SiamBAG (2023a)	40.3	54.6
MMF-Net (2024)	46.9	60.4
DaSSP-Net (2025b)	56.2	75.6
HotMoE (2025)	60.0	74.8
DRSST-Net (2026)	61.9	75.8
SpecUnify+LoRAT	**63.2**	**80.0**

3 Experiments

3.1 Experimental setup

Datasets. We evaluate RGB segmentation on the Cityscapes validation set (Cordts et al., 2016) and HSI segmentation on HSI-City v2 (Huang et al., 2021), using the HS3-Bench test split (Theisen et al., 2024) as in HyperVision (Fu et al., 2026). Tracking is evaluated on LaSOT (RGB) (Fan et al., 2019), HOTC2023 (HSI) (Xiong et al., 2023), and MUST(MSI) (Qin et al., 2025) .

Baselines and compared methods. We compare with RGB and spectral-domain methods listed in Tables 1–3. Controlled RGB baselines include DINOv2/DINOv3 linear probes and LiDeRe (Lüddecke et al., 2026) for segmentation, and a DINOv3 (Siméoni et al., 2025) adaptation of LoRAT (Lin et al., 2024) for tracking. A CARL(Baumann et al., 2026) spectral-encoder control retains our segmentation backend and HSI-only training budget. Baseline implementations and reproduction checks are documented in Appendix D.

Tasks and protocols. For segmentation, R, H, and R+H denote RGB-only, HSI-only, and joint supervision, respectively. We report mIoU, together with micro accuracy, macro accuracy, and macro F1 on HSI-City v2. For tracking, SpecUnify uses joint RGB-MSI-HSI supervision unless otherwise specified. We report one-pass evaluation AUC and distance precision at 20 pixels (DP_{20}), together with normalized precision on LaSOT. HOTC2023 and MUST follow the standard HOTC evaluation protocol (Xiong et al., 2020), consistent with the compared methods. Detailed evaluation settings are provided in Appendix C.

Implementation. Segmentation uses DINOv3-L/16 with LiDeRe at 512 resolution; tracking uses DINOv3-B/16 with LoRAT and template/search sizes of 192/384. The pretrained backbone base weights are frozen; LoRA, task heads, and enabled frontend modules are trained. RGB lifting widths are 128 for segmentation and 16 for tracking, matching the mean native spectral channel counts: $128/1 = 128$ and $(8 + 15 + 16 + 25)/4 = 16$, respectively. The exploration ranges are $[3, 128]$ and $[3, 25]$. Sampling and optimization details are provided in Appendix C, and frontend architectures in Appendix B.1.

3.2 Comparison to state-of-the-art

Semantic segmentation. On HSI-City v2 (Table 1), joint SpecUnify reaches 86.88% mIoU, compared with 85.26% for HSI-only and 59.21% for RGB-only training. The joint model exceeds HyperVision by 29.84 points, while the 1.62-point gain over HSI-only training is consistent with complementary supervision but also involves a longer training budget. Under the same DINOv3-L/LiDeRe backend, LoRA configuration, and HSI-only training setting, SpecUnify exceeds our CARL frontend adaptation by 22.56 mIoU points and 18.04 macro-F1 points, using 1.56M rather than 15.82M trainable parameters. This controlled comparison supports more effective frontend adaptation without a larger trainable parameter budget.

Object tracking. On HOTC2023 (Table 2), SpecUnify achieves 67.7% AUC and 86.7% DP_{20}, exceeding SSTtrack by 5.2 and 4.8 points, respectively. The gains demonstrate that the shared

Table 4: Segmentation ablations using mIoU (%). Extra parameters are relative to the shared backend. **A** replaces the entire frontend with a parameter-matched input MLP projection to three channels, using a comparable parameter budget. **B/C** retain RGB lifting and representation generation but fix $K = 19$ to match the learned joint operating width. B uses a nonlinear projection to three channels; C retains the dynamic token interface. A and B use the original pretrained patch embedding. **D** removes only the low-rank residual, retaining the shared patch projection; **E** removes only RGB lifting. Both retain learned cardinality. **F** disables random width exploration while retaining task-gradient-based cardinality learning. **G** is the complete model.

Variant	Training source	Extra params (M)	RGB FLOPs (G)	HSI FLOPs (G)	Cityscapes	HSI-City v2
A: Input MLP projection	R	0.4282	967.27	–	79.04	–
	H	0.4602	–	984.05	–	80.98
B: Fixed K + MLP projection	R+H	0.4416	971.38	966.99	78.37	81.31
C: Fixed K + dynamic interface	R+H	0.4372	753.71	749.31	78.68	85.44
D: w/o dynamic residual	R+H	0.2945	753.61	749.21	78.55	81.43
E: w/o RGB lifting	R+H	0.4330	743.35	749.31	78.21	79.59
F: w/o width exploration	R+H	0.4416	753.71	749.31	78.82	77.12
G: Full SpecUnify	R	0.4416	753.71	749.31	**79.26**	59.21
	H				59.08†	85.26
	R+H				79.01	**86.88**

†Only RGB lifting is transferred from a joint-trained donor. Architecture and FLOP conventions appear in Appendix C.3.

frontend can support strong hyperspectral tracking with a general-purpose LoRAT backend. On MUST (Table 3), SpecUnify reaches 63.2% AUC and 80.0% DP_{20}, improving over DRSST-Net by 1.3 and 4.2 points. Together with HOTC2023, this extends the spectral tracking results from HSI to eight-band MSI within the same trained model. Attribute results in Appendix E.4 show the largest margins under Similar Color, Background Clutter, Full Occlusion, and Low Resolution, while Scale Variation remains a weakness.

Additional RGB segmentation and tracking results are provided in Appendix E.1.

3.3 Ablation and analysis

Table 4 evaluates the frontend components and width-learning strategy through controlled segmentation ablations. The larger gains on HSI than RGB highlight the importance of representation construction and token interfacing for spectral adaptation.

Representation and token interface. Under matched single-source training, G improves over the parameter-matched input MLP (A) by 0.22/4.28 RGB/HSI mIoU points. The larger HSI gain supports constructing task representations beyond a prescribed three-channel input projection, particularly when richer spectral observations are available. At fixed $K = 19$, C exceeds B by 0.31/4.13 points: even without learning the width, directly mapping the generated components into backbone tokens is more effective than compressing them through a three-channel projection. Removing the low-rank residual (D) reduces accuracy by 0.46/5.45 points relative to joint G. Whereas the shared projection applies the same mapping to every component, the content-conditioned residual provides component-specific mappings into token space. These results support its role in translating the generated spectral representation into features useful for downstream prediction, especially on HSI.

Learned width and RGB lifting. Joint G exceeds C by 0.33/1.44 RGB/HSI points, although C uses the same final operating width. This suggests that the width-learning procedure contributes through the optimization trajectory, not merely the final channel count: the generator and token interface are trained alongside the evolving representation width. Removing RGB lifting (E) reduces RGB/HSI accuracy by 0.80/7.29 points. Lifting supplies learned channel-level features for RGB inputs before shared processing. Its substantial effect on HSI, whose inputs bypass this module, indicates that this RGB parameterization also shapes the representation learned through joint supervision, rather than serving only the RGB prediction branch.

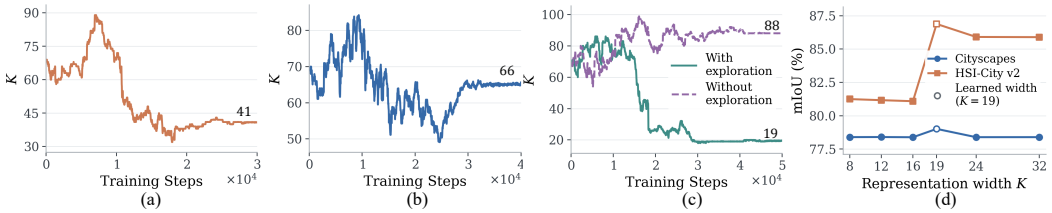

Figure 3: Representation-width analysis. (a)–(c) Predicted width during HSI-only, RGB-only, and joint RGB-HSI segmentation training, respectively. Panel (c) compares training with exploration (solid) and without exploration (dashed). Widths are recorded before random exploration; training steps are shown in units of 10^4. Endpoint annotations in (a)–(c) report the final mean predicted widths, rounded to the nearest integer. (d) Inference-time width sensitivity of the joint model; hollow markers denote the learned-width reference at $K = 19$.

Width learning and exploration. Removing random width exploration reduces Cityscapes mIoU only slightly, from 79.01 to 78.82, but lowers HSI-City v2 mIoU from 86.88 to 77.12. This asymmetric degradation suggests that exploration is important for learning an effective representation of richer spectral inputs. Exploration allows alternative widths, whereas detached neighboring-width probes only estimate the width gradient. The results therefore support its role in coordinating cardinality learning with representation optimization, beyond enabling width changes alone.

Task-learned representation width. Fig. 3(a)–(c) shows width adjustments followed by regime-specific stabilization. With exploration, joint training reaches the smallest operating width among the three training regimes. Without exploration, the joint width still adapts but settles near 88 rather than 19, indicating that exploration influences the operating point reached, rather than merely enabling width changes. These trajectories illustrate that representation cardinality need not be prescribed by the sensor channel count.

The fixed-checkpoint interventions in Fig. 3(d) reveal stronger width sensitivity on HSI-City v2 than on Cityscapes: narrow representations substantially reduce HSI accuracy, whereas RGB performance changes little across the tested widths. Increasing K from 24 to 32 provides little additional benefit. These observations support interpreting K as an operating width learned jointly with the representation, rather than a capacity measure whose increase uniformly improves accuracy.

4 Discussion

SpecUnify shows that spectral unification is a representation-design problem, not merely an input-compatibility problem. Heterogeneous observations can share a task-representation mechanism without being standardized to the same raw channel count. At fixed K, the dynamic token interface improves HSI segmentation over a three-channel MLP readout under comparable parameter budgets, showing that how representations enter a pretrained backbone matters as much as how they are constructed. Variable-width training also outperforms fixing the final operating width from initialization, while removing cardinality exploration leads to a wider but less accurate representation. These results show that effective representation learning depends not only on the final cardinality, but also on how that cardinality is learned and explored. Since K conditions whole-set generation, changing it can reorganize the representation rather than simply add or remove components.

The gains are asymmetric across modalities, with strong spectral performance accompanied by modest losses relative to native RGB baselines. This reflects the different adaptation demands: RGB already matches the pretrained interface, whereas spectral inputs require a new mapping into the token space. Thus, the main benefit is broader spectral applicability through a shared task-specific model, rather than uniform improvement over modality-specific models.

Limitations. SpecUnify does not yet improve all modalities simultaneously, with a modest trade-off relative to native RGB baselines. Better preserving RGB-pretrained priors while learning shared representations is an important direction. Moreover, supporting different channel counts does not establish generalization to unseen sensors. Evaluating transfer across different wavelength coverage and sensor responses would further test the generality of the shared representation.

REFERENCES

Guillaume Astruc, Nicolas Gonthier, Clement Mallet, and Loic Landrieu. Anysat: One earth observation model for many resolutions, scales, and modalities. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19530–19540. IEEE, 2025.

Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. In *Forty-second International Conference on Machine Learning*, 2025.

Alexander Baumann, Leonardo Ayala, Silvia Seidlitz, Jan Sellner, Alexander Studier-Fischer, Berkin Özdemir, Lena Maier-Hein, and Slobodan Ilic. CARL: Camera-agnostic representation learning for spectral image analysis. In *International Conference on Learning Representations*, volume 2026, pp. 23335–23357, 2026.

Anuja Bhargava, Ashish Sachdeva, Kulbhushan Sharma, Mohammed H Alsharif, Peerapong Uthansakul, and Monthippa Uthansakul. Hyperspectral imaging and its applications: A review. *Heliyon*, 10(12), 2024.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

Wenrui Cai, Qingjie Liu, and Yunhong Wang. Hiptrack: Visual tracking with historical prompts. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19258–19267. IEEE, 2024.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European conference on computer vision*, pp. 833–851. Springer, 2018.

Xin Chen, Houwen Peng, Dong Wang, Huchuan Lu, and Han Hu. SeqTrack: Sequence to sequence learning for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14572–14581, 2023.

Yuzeng Chen, Qiangqiang Yuan, Yuqi Tang, Yi Xiao, Jiang He, and Zhenqi Liu. SENSE: Hyperspectral video object tracker via fusing material and motion cues. *Information Fusion*, 109: 102395, 2024.

Yuzeng Chen, Qiangqiang Yuan, Yuqi Tang, Yi Xiao, Jiang He, Te Han, Zhenqi Liu, and Liangpei Zhang. SSTtrack: A unified hyperspectral video tracking framework via modeling spectral-spatial-temporal conditions. *Information Fusion*, 114:102658, 2025.

Yezhen Cong, Samar Khanna, Chenlin Meng, Patrick Liu, Erik Rozi, Yutong He, Marshall Burke, David Lobell, and Stefano Ermon. Satmae: Pre-training transformers for temporal and multi-spectral satellite imagery. *Advances in Neural Information Processing Systems*, 35:197–211, 2022.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5369–5378. IEEE, 2019.

Ke Fan, Zechen Bai, Tianjun Xiao, Tong He, Max Horn, Yanwei Fu, Francesco Locatello, and Zheng Zhang. Adaptive slot attention: Object discovery with dynamic slot number. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23062–23071. IEEE, 2024.

Guanyiman Fu, Jingtao Li, Zihang Cheng, Zhuanfeng Li, Diqi Chen, Yan Xu, Xiangyu Liu, Fengchao Xiong, Jianfeng Lu, Chengrong Chen, et al. Hypervision: A channel-adaptive ground-based hyperspectral vision pre-trained backbone. *arXiv preprint arXiv:2605.17286*, 2026.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Danfeng Hong, Bing Zhang, Xuyang Li, Yuxuan Li, Chenyu Li, Jing Yao, Naoto Yokoya, Hao Li, Pedram Ghamisi, Xiuping Jia, et al. Spectralgpt: Spectral remote sensing foundation model. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5227–5244, 2024.

Danfeng Hong, Chenyu Li, Naoto Yokoya, Bing Zhang, Xiuping Jia, Antonio Plaza, Paolo Gamba, Jon Atli Benediktsson, and Jocelyn Chanussot. Hyperspectral imaging. *Nature Reviews Methods Primers*, 6(1):19, 2026.

Zengfu Hou, Wei Li, Jun Zhou, and Ran Tao. Spatial–spectral weighted and regularized tensor sparse correlation filter for object tracking in hyperspectral videos. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–12, 2022.

Yuxing Huang, Tianqi Ren, Qiu Shen, Ying Fu, and Shaodi You. Hsicityv2: Urban scene understanding via hyperspectral images. 2021.

Renjie Ji, Xue Wang, Chao Niu, Wen Zhang, Yong Mei, and Kun Tan. Specaware: a spectral-content aware foundation model for unifying multi-sensor learning in hyperspectral remote sensing mapping. *ISPRS Journal of Photogrammetry and Remote Sensing*, 234:242–260, 2026.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

Jingtao Li, Yingyi Liu, Xinyu Wang, Yunning Peng, Chen Sun, Shaoyu Wang, Zhendong Sun, Tian Ke, Xiao Jiang, Tangwei Lu, et al. Hyperfree: A channel-adaptive and tuning-free foundation model for hyperspectral remote sensing imagery. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23048–23058. IEEE, 2025a.

Wei Li, Zengfu Hou, Jun Zhou, and Ran Tao. SiamBAG: Band attention grouping-based siamese object tracking network for hyperspectral videos. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–12, 2023a.

Zhuanfeng Li, Fengchao Xiong, Jun Zhou, Jing Wang, Jianfeng Lu, and Yuntao Qian. BAE-Net: A band attention aware ensemble network for hyperspectral object tracking. In *IEEE International Conference on Image Processing*, pp. 2106–2110, 2020.

Zhuanfeng Li, Fengchao Xiong, Jianfeng Lu, Jun Zhou, and Yuntao Qian. Material-guided siamese fusion network for hyperspectral object tracking. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2809–2813, 2022.

Zhuanfeng Li, Fengchao Xiong, Jun Zhou, Jianfeng Lu, and Yuntao Qian. Learning a deep ensemble network with band importance for hyperspectral object tracking. *IEEE Transactions on Image Processing*, 32:2901–2914, 2023b.

Zhuanfeng Li, Fengchao Xiong, Jun Zhou, Jianfeng Lu, Zhuang Zhao, and Yuntao Qian. Material-guided multiview fusion network for hyperspectral object tracking. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–15, 2024.

Zhuanfeng Li, Fengchao Xiong, Jianfeng Lu, Jing Wang, Diqi Chen, Jun Zhou, and Yuntao Qian. Multi-domain universal representation learning for hyperspectral object tracking. *Pattern Recognition*, 162:111389, 2025b.

Zhuanfeng Li, Jing Wang, Jue Zhang, Diqi Chen, Guanyiman Fu, Jiangtao Wang, and Jianfeng Lu. Hyperspectral object tracking with dimensionality reduction and spatial–spectral–temporal modeling. *ISPRS Journal of Photogrammetry and Remote Sensing*, 238:14–32, 2026.

Liting Lin, Heng Fan, Zhipeng Zhang, Yaowei Wang, Yong Xu, and Haibin Ling. Tracking meets lora: Faster training, larger model, stronger performance. In *European Conference on Computer Vision*, pp. 300–318. Springer, 2024.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Timo Lüddecke, Jan Frederik Meier, Jan van Delden, and Alexander Ecker. LiDeRe: A lightweight readout for fast and data-efficient dense prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2959–2971, 2026.

Zhenliang Ni, Xinghao Chen, Yingjie Zhai, Yehui Tang, and Yunhe Wang. Context-guided spatial feature reconstruction for efficient semantic segmentation. In *European conference on computer vision*, pp. 239–255. Springer, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Haolin Qin, Tingfa Xu, Tianhao Li, Zhenxiang Chen, Tao Feng, and Jianan Li. Must: The first dataset and unified framework for multispectral uav single object tracking. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16882–16891. IEEE, 2025.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 4510–4520. Ieee, 2018.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Wenfang Sun, Yuedong Tan, Jingyuan Li, Shuwei Hou, Xiaobo Li, Yingzhao Shao, Zhe Wang, and Beibei Song. Hotmoe: Exploring sparse mixture-of-experts for hyperspectral object tracking. *IEEE Transactions on Multimedia*, 27:4072–4083, 2025.

Nick Theisen, Robin Bartsch, Dietrich Paulus, and Peer Neubert. Hs3-bench: A benchmark and strong baseline for hyperspectral semantic segmentation in driving scenarios. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5895–5901. IEEE, 2024.

Leonard Waldmann, Ando Shah, Yi Wang, Nils Lehmann, Adam Stewart, Zhitong Xiong, Xiao Xiang Zhu, Stefan Bauer, and John Chuang. Panopticon: Advancing any-sensor foundation models for earth observation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2229–2239, 2025.

F. Xiong, J. Zhou, W. Charle, Y. Zhong, P. Ghamisi, and J. Chanussot. The hyperspectral object tracking challenge hot2023. `https://www.hsitracking.com/`, 2023.

Fengchao Xiong, Jun Zhou, and Yuntao Qian. Material based object tracking in hyperspectral videos. *IEEE Transactions on Image Processing*, 29:3719–3733, 2020.

Zhitong Xiong, Yi Wang, Fahong Zhang, Adam J Stewart, Joëlle Hanna, Damian Borth, Ioannis Papoutsis, Bertrand Le Saux, Gustau Camps-Valls, and Xiao Xiang Zhu. Neural plasticity-inspired multimodal foundation model for earth observation. *arXiv preprint arXiv:2403.15356*, 2024.

Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint feature learning and relation modeling for tracking: A one-stream framework. In *European Conference on Computer Vision*, pp. 341–357, 2022.

Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10799–10808. IEEE, 2022.

Yaozong Zheng, Bineng Zhong, Qihua Liang, Zhiyi Mo, Shengping Zhang, and Xianxian Li. Odtrack: Online dense temporal token learning for visual tracking. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 7588–7596, 2024.

Jiawen Zhu, Simiao Lai, Xin Chen, Dong Wang, and Huchuan Lu. Visual prompt multi-modal tracking. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9516–9526. IEEE, 2023.

APPENDIX CONTENTS

A Related Work **15**

B Additional method details **16**

B.1 Frontend architecture 16

B.2 Token initialization and cardinality optimization 17

B.3 Task integration 17

C Experimental protocols **18**

C.1 Data splits and evaluation 18

C.2 Optimization and supervision 18

C.3 Ablation architectures and computation 19

D Baseline implementations and validation **20**

D.1 RGB backbone integrations 20

D.2 CARL spectral-encoder adaptation 20

E Additional quantitative analyses **21**

E.1 RGB benchmark comparisons 21

E.2 RGB input-module transfer 21

E.3 Interpreting learned width 21

E.4 MUST attributes 21

F Qualitative results **22**

F.1 Semantic segmentation 22

F.2 Object tracking 23

AI USE STATEMENT

In this work, we used generative AI tools for the writing and polishing of the manuscript, including improving grammar, clarity, conciseness, and organization. We have not used generative AI tools for the ideation, research methodology, or experimental design. All research concepts, ideas, and analyses were developed and conducted by the authors. LLM only involved in improving the linguistic quality of the paper, with no involvement in the methodology or experimental analysis. All text generated or revised with the assistance of generative AI was carefully reviewed and verified by the authors to ensure accuracy, consistency, and faithful representation of the intended scientific meaning. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses existing, publicly available benchmarks for segmentation and tracking. It does not involve new data collection or human-subject studies.

REPRODUCIBILITY STATEMENT

Appendices B and C specify the frontend architectures, cardinality optimization, data partitions, label mapping, evaluation metrics, and training schedules. Appendix D documents the backbone validation experiments and CARL adaptation, while Appendix C.3 defines the controls and computation-counting conventions. Published references, our implementations, and donor-assisted transfer are identified separately. We will publicly release our code and experimental configurations to facilitate reproduction of the reported results.

A RELATED WORK

Camera-agnostic spectral representation learning. Recent work has increasingly sought representations that are less dependent on the spectral configuration of the sensing device. CARL (Baumann et al., 2026) is the closest to our setting, introducing camera-agnostic representation learning across RGB, multispectral, and hyperspectral images. It uses wavelength-aware spectral encoding and attention-based aggregation to distill variable-channel observations into learned spectral representations. DOFA (Xiong et al., 2024) instead generates wavelength-conditioned input weights with a hypernetwork, enabling a shared Transformer to process observations from different sensors. Panopticon (Waldmann et al., 2025) further develops an any-sensor foundation model using wavelength-aware channel cross-attention, while SpecAware conditions spectral embeddings on both image content and sensor metadata for cross-sensor hyperspectral learning (Ji et al., 2026).

These approaches establish that spectral observations with different channel configurations can be mapped to a common representation. SpecUnify addresses a further degree of freedom: *the cardinality of the task representation*. We explicitly separate sensor dimensionality from representation cardinality, $C_{\text{in}} \neq K$, where C_{in} is determined by the sensor and K is learned jointly with the downstream objective. Thus, the goal is not only to make the input interface independent of the sensor, but also to let the task determine how many learned representation components should be constructed. Moreover, for a given K, SpecUnify generates the representation as a whole conditioned jointly on the input and its cardinality, rather than selecting a subset or prefix from a fixed bank of output components.

Spectral foundation models and channel-adaptive backbones. Spectral foundation models have further explored large-scale pretraining and flexible input interfaces for heterogeneous remote-sensing observations. SatMAE (Cong et al., 2022) and SpectralGPT (Hong et al., 2024) learn spatial-spectral representations through masked or generative pretraining, while HyperFree develops a channel-adaptive embedding mechanism for hyperspectral inputs with varying channel numbers. HyperVision (Fu et al., 2026) further develops a channel-adaptive hyperspectral backbone with dynamic input embedding and cross-modal knowledge transfer from RGB models. AnySat (Astruc et al., 2025) takes a broader multimodal perspective, learning a unified representation across heterogeneous Earth-observation resolutions, scales, and modalities.

These methods primarily improve the universality of the *model or input interface* across sensing configurations. SpecUnify instead focuses on the task representation formed between the sensor and the pretrained backbone. Given a pretrained vision model, RGB observations are first lifted into a common spectral space, while multispectral and hyperspectral observations enter the shared representation directly. The resulting representation is learned from the downstream objective rather than from physical spectral reconstruction or a manually prescribed spectral reduction. This separates the representation required by the task from both the sensor's native channel dimensionality and the fixed embedding dimension of the pretrained backbone.

Adaptive representation cardinality. Adaptive computation (Yin et al., 2022; Bolya et al., 2022), dynamic token selection(Rao et al., 2021), slot-based representations (Locatello et al., 2020; Fan et al., 2024), and related conditional architectures have explored representations whose size or structure varies with the input (Kusupati et al., 2022; Bachmann et al., 2025). These methods demonstrate that intermediate representations need not be fixed, but typically target object-centric decomposition, token allocation, or computational efficiency rather than heterogeneous spectral representation learning.

SpecUnify introduces adaptive cardinality in a different setting, where the representation is learned directly for a downstream task and must remain compatible with a fixed pretrained vision backbone. Its components are learned task features rather than selected input bands, retained tokens, or predefined physical factors. For a given K, the complete set is generated jointly conditioned on the input and K, allowing the representation to be reorganized when its cardinality changes rather than merely truncated. Learning K therefore requires both a discrete cardinality mechanism and a compatible variable-to-fixed token interface. SpecUnify couples these through a neighboring-width task-gradient surrogate and a dynamic token interface, enabling the cardinality decision to be learned directly from the original downstream objective.

B Additional method details

This section specifies the implementation of the shared frontend. The same design is used for both tasks, with separate learned parameters for segmentation and tracking. Within each task, the parameters are shared across sensing modalities.

B.1 Frontend architecture

RGB lifting and channel encoding. RGB lifting uses two pointwise convolutions, $3 \to 64 \to C_{\text{lift}}$, with GELU between them. We set $C_{\text{lift}} = 128$ for segmentation and $C_{\text{lift}} = 16$ for tracking, following the mean channel count across the spectral input configurations in each task: $128/1 = 128$ and $(8 + 15 + 16 + 25)/4 = 16$. These mappings contain 8,576 and 1,296 parameters, respectively. They produce learned features, not reconstructed spectral measurements; MSI and HSI inputs bypass lifting.

Each resulting channel is encoded independently by a shared 3×3 convolution ($1 \to 8$), GELU, adaptive average pooling to 2×2, a $32 \to 64$ linear layer, and LayerNorm. Averaging the descriptors over valid channels gives $\boldsymbol{g} \in \mathbb{R}^{64}$. Padded channels are excluded from both this average and spectral attention. The shared encoder uses channel content, without wavelength metadata, input-channel indices, or modality identifiers.

Cardinality and component conditions. The cardinality predictor consists of LayerNorm, a $64 \to 64$ linear layer, GELU, and a $64 \to 1$ linear layer, followed by the bounded sigmoid mapping and integer rounding defined in the main text. The admissible ranges are $K \in [3, 128]$ for segmentation and $K \in [3, 25]$ for tracking; these bounds are independent of RGB lifting. For component $j \in \{1, \dots, K\}$, the coordinate $u_j = (j - \frac{1}{2})/K$ is encoded as

$$\phi(u_j) = \left[u_j;\, \{\sin(2\pi f u_j)\}_{f\in\{1,2,4,8\}};\, \{\cos(2\pi f u_j)\}_{f\in\{1,2,4,8\}}\right]. \tag{15}$$

A $1 \to 16 \to 16$ MLP with an intermediate GELU embeds $\log K$. Concatenating $\boldsymbol{g}$, $\phi(u_j)$, and this width embedding yields the 89-dimensional condition $\boldsymbol{c}_j$. It is shared by the representation generator and dynamic token interface. Changing K therefore changes the conditions of the complete set, rather than selecting additional entries from a fixed component bank.

Spectral mixing and spatial modulation. An $89 \to 64 \to 89$ MLP with an intermediate GELU produces the 64-dimensional query $\boldsymbol{q}_j$, three eight-dimensional vectors $\boldsymbol{\gamma}_j, \boldsymbol{\beta}_j, \boldsymbol{w}_j$, and scalar b_j. Queries attend to channel descriptors through a shared $64 \to 64$ key projection and scaled dot-product softmax over valid channels. For the resulting mixture $\mathbf{M}_j$, a shared 3×3 convolution $S_\theta : 1 \to 8$ produces spatial features $\mathbf{V}_j$. The component is then computed as

$$\mathbf{R}_j(p) = \frac{1}{\sqrt{8}} \sum_{a=1}^{8} w_{j,a} \, \mathrm{GELU}((1 + \gamma_{j,a}) V_{j,a}(p) + \beta_{j,a}) + b_j. \tag{16}$$

Thus, the same content-cardinality condition controls which input channels are mixed and how their spatial evidence is transformed. The mixing and refinement path preserves image resolution; only the descriptor branch is pooled.

B.2 Token initialization and cardinality optimization

Shared and component-conditioned patch projections. For patch size $P = 16$, the shared projection is initialized as $\boldsymbol{W}_{\mathrm{s}} = \frac{1}{3} \sum_{c=1}^{3} \boldsymbol{W}_{\mathrm{RGB},:,\mathrm{c},:}$, and its bias is copied from the pretrained patch embedding. Both are subsequently trained. The factor $3/K$ scales the component sum relative to this initialization; it does not imply equivalence to RGB patch embedding for arbitrary generated features. The residual uses rank $r = 8$ and scale $\alpha = 8$. An $89 \to 64 \to (rP^2)$ MLP generates each $\boldsymbol{A}_j$, while the shared $\boldsymbol{U} \in \mathbb{R}^{D \times r}$ is initialized to zero. The residual therefore vanishes initially without zero-initializing both factors. The patch bias is added once.

The shared branch projects the summed components with a common kernel, whereas the residual applies a distinct, condition-dependent mapping to each component before summation. This gives the downstream task a way to distinguish component contributions while keeping the token shape $N \times D$ independent of K. The dynamic route produces tokens directly, without an intermediate sigmoid, RGB normalization, or three-channel image.

Neighboring-width task-gradient surrogate. For a predicted integer K, the neighboring admissible widths are $k_- = \max(K_{\min}, K - 1)$ and $k_+ = \min(K_{\max}, K + 1)$. Their complete representations and token projections are regenerated with gradients disabled, reusing the channel descriptors. The token difference $\Delta_K = (\boldsymbol{T}^{(k_+)} - \boldsymbol{T}^{(k_-)})/(k_+ - k_-)$ supplies

$$\widehat{g}_K = \left\langle \frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{T}}}, \Delta_K \right\rangle \tag{17}$$

to the continuous cardinality estimate through the stop-gradient construction in the main text. This is a local linearization of the task loss in token space, not a direct finite difference of the downstream losses at k_- and k_+. At the bounds, the difference becomes one-sided. Only the executed representation passes through the pretrained backbone; the probes add frontend computation but no additional backbone passes.

Width exploration. During training, each image independently replaces its predicted width with a uniformly sampled admissible integer with probability p. Segmentation uses $p(t) = 0.5 \max(1 - t/10000, 0)$ for zero-based optimizer update t. Tracking uses $p(e) = 0.5 \max(1 - e/49, 0)$ for zero-based epoch e, reaching zero in the 50th epoch. Template and search images receive separate draws. Every executed width trains the frontend through the ordinary task loss. The cardinality surrogate is applied only when the executed width equals the predicted width, including coincidentally equal random draws; otherwise its contribution is zero. Exploration and neighboring probes are both disabled at evaluation.

Exploration and the surrogate consequently play different roles: exploration trains representations at alternative widths, whereas the detached probes estimate a direction for updating the width predictor. The probes do not themselves train those alternative representations. Grouping examples by executed width and chunking channel or component computations are implementation choices that preserve these operations.

B.3 Task integration

Semantic segmentation. SpecUnify supplies patch tokens to DINOv3-L/16. LiDeRe (Lüddecke et al., 2026) decodes features from zero-based blocks 7, 15, 21, and 23, using a readout dimension

and query grid of 64. The decoder predicts the shared 19-class label space. Training uses cross-entropy averaged over valid pixels per image, then over the batch. The pretrained backbone base is frozen; the frontend, backbone LoRA, and LiDeRe are optimized jointly.

Visual tracking. The same frontend processes template and search crops, predicting their widths independently. Its tokens replace the original patch-embedding outputs of the DINOv3-B/16 LoRAT adaptation. The template-search interaction, tracking head, and LoRAT objective (Lin et al., 2024) are retained. The frontend, LoRA adapters, and tracking-specific trainable components are optimized jointly, with the pretrained backbone base frozen. Inference requires only the observed modality, not paired RGB-spectral inputs.

C Experimental protocols

C.1 Data splits and evaluation

Segmentation splits and labels. Cityscapes uses its standard training and validation sets. For HSI-City v2, we following (Fu et al., 2026) use the three CSV partitions publicly released with HS3-Bench (Theisen et al., 2024)[1]: 527 training, 105 validation, and 422 test images. We retain these sample assignments rather than constructing a new random split. Both datasets use the standard Cityscapes 19-class taxonomy. Cityscapes raw IDs $\{7, 8, 11, 12, 13, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 31, 32, 33\}$ map, in order, to training IDs $0, \ldots, 18$. The HSI labels already use these training IDs and are not remapped. Void pixels have ID 255 and are excluded from loss and evaluation. Test labels are not used for training or checkpoint selection.

Segmentation inference. Cityscapes images are resized to short side 512 and evaluated with overlapping 512×512 windows at stride 341. Window logits are averaged in overlapping regions and restored to the original annotation resolution before taking the argmax. For HSI-City v2, spatial operations are applied to the compact coefficient maps in the supplied HSD representation, retaining all 128 decoded bands. The loader first resizes these maps to short side 512; the resize evaluator then maps them to 512×512 before spectral decoding. One full-image forward is performed, followed by logit interpolation to the original annotation resolution and argmax. Bilinear interpolation is used for coefficients and logits, with antialiasing for coefficient resizing. No multi-scale or flip test-time augmentation is used.

Segmentation metrics. Metrics are computed from one confusion matrix accumulated over all valid pixels in the evaluation split, not by averaging image-wise mIoUs. Let t_c, g_c, and p_c denote true positives, ground-truth support, and predicted support for class c. Cityscapes mIoU and HSI-City v2 macro IoU average $t_c/(g_c + p_c - t_c)$ over classes with nonzero union. For HSI-City v2, micro accuracy is $\sum_c t_c / \sum_c g_c$; macro accuracy and macro F1 average $t_c / \max(g_c, 1)$ and $2t_c/(g_c + p_c)$ over the same active classes. A class absent from the targets but predicted by the model contributes zero; classes absent from both are excluded. Published comparator scores and our implementations are identified separately in the results tables.

Tracking. HOTC2023 and MUST follow the standard HOTC one-pass evaluation protocol (Xiong et al., 2020), consistent with the compared methods. The tracker is initialized with the first-frame ground-truth box and follows the sequence without reinitialization. Success AUC and distance precision at 20 pixels (DP_{20}) are computed per sequence and then averaged. LaSOT follows its benchmark protocol and additionally reports normalized precision. HOTC2023 evaluation covers VIS, NIR, and RedNIR; MUST uses its test split. MUST training uses only the released training list.

C.2 Optimization and supervision

Both tasks use AdamW, BF16, and cosine learning-rate decay after warmup. Table 5 summarizes the training schedules. Segmentation LoRA adapts the fused QKV projections with rank 8 and scale 16; tracking uses rank 64 and scale 64.

Joint segmentation samples RGB and HSI with probabilities 0.85 and 0.15. All joint segmentation variants (B–F and the R+H instance of G) are trained for 50,000 updates with the same global batch

[1] https://github.com/nickstheisen/hyperseg

Table 5: Training settings. Tracking samples 98,304 image pairs per epoch; batch sizes are global.

Setting	Segmentation	Tracking
Backbone	DINOv3-L/16	DINOv3-B/16
Task head	LiDeRe	LoRAT
Crop size	512×512	$192/384$ (template/search)
Batch size	16	96
Head learning rate	10^{-3}	10^{-4}
Frontend / LoRA learning rate	10^{-4}	10^{-4}
Weight decay	10^{-3}	0.1
Warmup	1,500 updates	2 epochs
Training budget	R: 40k; H: 30k updates R+H: 50k updates	200 epochs

size and RGB/HSI sampling probabilities, ensuring matched training budgets. The single-source A and G runs use 40,000 RGB-only or 30,000 HSI-only updates, respectively. Training uses a random scale in $[0.5, 2.0]$, random crops, and horizontal flips; photometric distortion is applied only to RGB. Joint tracking assigns sampling probabilities 0.24 each to LaSOT, GOT-10k, COCO, and TrackingNet, 0.01 to HOTC2023, and 0.03 to MUST. The RGB-only tracking control samples the four RGB sources equally. Segmentation validation is performed every 5,000 updates. The best checkpoint is tracked by validation mIoU, averaging the two domain scores in joint training; held-out test scores are excluded.

Training exposure. The full joint segmentation run has more optimizer updates but fewer HSI presentations than HSI-only training: the expected RGB/HSI counts are 680k/120k for joint training, compared with 640k RGB-only and 480k HSI-only presentations. These are expectations from the sampling schedule, not distinct images. The joint result is therefore not explained by increased HSI exposure alone, but the unequal schedules do not isolate a causal gain from cross-modal supervision.

C.3 Ablation architectures and computation

Input projection (A). A replaces the entire frontend with a pointwise MLP having three hidden layers, $C \to 256 \to 536 \to 536 \to 3$. All four affine layers are biased 1×1 convolutions, with GELU after each hidden layer. Separate RGB and HSI models use $C = 3$ and $C = 128$, giving 428,219 and 460,219 projection parameters. The output undergoes ImageNet RGB normalization, without sigmoid clamping, before the original frozen patch embedding.

Fixed-width interfaces (B/C). Both retain RGB lifting and the complete representation generator at $K = 19$. B replaces the dynamic interface with a three-layer pointwise MLP, $19 \to 632 \to 632 \to 3$, with biases and GELU after the first two layers. Sigmoid and ImageNet RGB normalization precede the original frozen patch embedding. The MLP has 414,595 parameters; including the retained modules, B adds 441,580 trainable parameters, versus 441,578 for the full frontend. C retains the shared-plus-low-rank token interface. For both controls, the width predictor is frozen and bypassed, and neither exploration nor neighboring-width probes are used. B versus C therefore compares the complete readout designs, including their output constraints and patch projections.

Component removals (D/E/F). D removes the generated low-rank residual while retaining the trainable shared patch projection. E removes only RGB lifting. Both retain width learning and exploration. F retains the full architecture and task-gradient cardinality learning but disables random exploration throughout training. G is the complete model; each training-source row is a separately trained instance.

Parameters and FLOPs. Extra trainable parameters are relative to the shared DINOv3-L/16, LiDeRe, and LoRA backend. FLOPs cover a complete single-crop inference forward at batch size one, with RGB/HSI inputs of $3/128 \times 512 \times 512$. B–G are profiled at controlled $K = 19$ to compare architectures; G's merged entries give this same reference across training sources. They are not measured costs at each checkpoint's predicted widths, nor the cost of complete sliding-window Cityscapes inference. One multiply-add counts as two FLOPs. Convolutions, matrix multiplications, and attention products are included; elementwise operations, normalization, and interpolation are excluded.

Table 6: DINOv3-L/16 linear-head validation on Cityscapes (mIoU, %).

Resolution	Published	Our implementation
1024	79.4	79.4
512	–	73.56

Table 7: GOT-10k-only RGB tracking validation (%).

Tracker	Backbone	AO	$SR_{0.5}$	$SR_{0.75}$
LoRAT-B-378 (Lin et al., 2024)	DINOv2	73.7	82.6	72.9
LoRAT (our adaptation)	DINOv3	73.8	83.7	72.5

Training cost depends on the executed widths and additionally includes eligible neighboring probes, backward computation, and activation recomputation.

D Baseline implementations and validation

D.1 RGB backbone integrations

Segmentation. Our frozen DINOv3-L/16 linear-head pipeline obtains 79.4% Cityscapes mIoU at the original 1024-resolution setting, matching the published value in Table 37 of (Siméoni et al., 2025). At the 512 setting used in this study, it obtains 73.56% (Table 6). Recovering the reference-resolution result supports the implementation's validity and distinguishes the lower-resolution comparison from a failure to reproduce the original pipeline. Linear-head baselines do not use LoRA; the LiDeRe comparisons use its LoRA setting (Lüddecke et al., 2026).

Tracking. Under GOT-10k-only downstream training following LoRAT's recipe (Lin et al., 2024), the DINOv3 adaptation obtains 73.8/83.7/72.5 in AO/$SR_{0.5}$/$SR_{0.75}$. These are close to the published DINOv2-based LoRAT reference (Table 7), supporting the reliability of the backbone integration before adding SpecUnify. This is validation of an adaptation, not an exact reproduction with the original backbone.

D.2 CARL spectral-encoder adaptation

We adapt CARL's spectral encoder (Baumann et al., 2026) to the same DINOv3-L/16 and LiDeRe backend for HSI-only training. It uses dimension 384, eight attention blocks (four spectral self-attention and four query cross-attention blocks), six heads, and eight learned queries, with wavelength Fourier scale 3. A shared 16×16 per-band convolution produces the spectral tokens at each spatial patch. Query outputs are summed, normalized with a parameter-free LayerNorm, and mapped from 384 to 1024 dimensions by a learned linear connector.

The transformer blocks are initialized from RGB-pretrained DINOv2-S; the per-band patch embedder and connector are independently initialized. No CARL-SSL checkpoint is used. The per-band patch embedder, spectral encoder, connector, spatial-backbone LoRA, and LiDeRe are trained, totaling 15.82M trainable parameters. The adaptation uses 30k HSI-only updates and the segmentation recipe above, with HSI validation for checkpoint selection. Its wavelength inputs are a nominal, uniformly spaced 128-point grid spanning 449–955 nm, not calibrated per-band centers; no additional per-band standardization is applied. Thus, the 62.70% mIoU result characterizes this specified CARL-frontend adaptation rather than the complete pretrained CARL system. The MLP control, matched to SpecUnify's frontend parameter count, provides a complementary comparison without this encoder-specific initialization and wavelength assumption.

Table 8: Cityscapes segmentation (%).

Method	Source	mIoU
DINOv2 + linear (2023)*	R	69.35
DINOv3 + linear (2025)*	R	73.56
LiDeRe (2026)*	R	**80.13**
SpecUnify + LiDeRe	R	79.26
	H†	59.08
	R+H	79.01

*Our implementation. †Transferred RGB lifting; see Appendix E.2.

Table 9: LaSOT tracking (%).

Method	AUC	P_{Norm}	DP_{20}
$\mathrm{OSTrack}_{384}$ (2022)	71.1	81.1	77.6
$\mathrm{SeqTrack\text{-}B}_{384}$ (2023)	71.5	81.1	77.8
$\mathrm{ODTrack\text{-}B}_{384}$ (2024)	**73.2**	**83.2**	**80.6**
$\mathrm{HIPTrack}_{384}$ (2024)	72.7	82.9	79.5
$\mathrm{LoRAT\text{-}B}_{378}$ (2024)	72.9	81.9	79.1
$\mathrm{LoRAT\text{-}B}^{*}_{384}$	72.4	81.0	78.2
SpecUnify + $\mathrm{LoRAT\text{-}B}_{384}$	72.0	80.8	77.8

$\mathrm{LoRAT\text{-}B}_{378}$ uses DINOv2; * denotes our DINOv3-B/16 adaptation with template/search sizes 192/384.

E Additional quantitative analyses

E.1 RGB benchmark comparisons

Semantic segmentation. RGB-only SpecUnify reaches 79.26% mIoU, 0.87 points below LiDeRe; joint training reaches 79.01%, a further 0.25-point decrease. The gap is therefore not solely a consequence of mixing training sources: it is already present under RGB-only supervision. The benefit of the shared frontend lies primarily in accommodating spectral observations, while retaining much of the RGB backend's accuracy.

Object tracking. On LaSOT, SpecUnify is 0.4/0.2/0.4 points below our DINOv3 LoRAT adaptation in AUC/normalized precision/DP_{20}. Together with HOTC2023 and MUST, this result characterizes a useful trade-off: a single jointly trained tracker accepts RGB, MSI, and HSI while remaining close to its RGB-specialized reference. It does not establish an improvement in RGB tracking accuracy.

E.2 RGB input-module transfer

For the H-trained model evaluated on Cityscapes, only RGB lifting is transferred from a joint-trained donor. The representation generator, token interface, backbone LoRA, and decoder retain HSI-trained weights. The resulting 59.08% mIoU shows that a learned input module can connect RGB observations to a task representation trained on HSI, although it does not remove the domain gap. Since the donor received RGB supervision, this is donor-assisted module transfer, not HSI-only zero-shot evaluation.

E.3 Interpreting learned width

Without exploration, the width still changes during training and stabilizes near 88–89, yet HSI mIoU falls from 86.88% to 77.12%; Cityscapes changes only from 79.01% to 78.82%. Width movement therefore persists without exploration; in this comparison, removing exploration leads to a different operating width and lower HSI accuracy. Because K reconditions the complete representation, increasing it does not simply append useful components to an otherwise unchanged representation. The outcome is consistent with exploration helping jointly optimize width and representation quality, rather than larger cardinality alone ensuring better accuracy.

The fixed-width controls disable both exploration and width learning, whereas F disables exploration alone. These comparisons support the combined training procedure but do not separately establish the benefit of the learned predictor over exploration-only training. The stabilized trajectories are interpreted as task-supervised operating widths, not evidence of beneficial per-image capacity allocation.

E.4 MUST attributes

SpecUnify ranks first on eight attributes and ties on Similar Object. Its largest AUC gains over the strongest competitor occur under Similar Color (+4.8 points), Background Clutter (+3.5), Full Occlusion (+3.2), and Low Resolution (+2.8). This pattern is consistent with improved target discrimination in visually confusing scenes, while Scale Variation remains the largest deficit (-8.9 points).

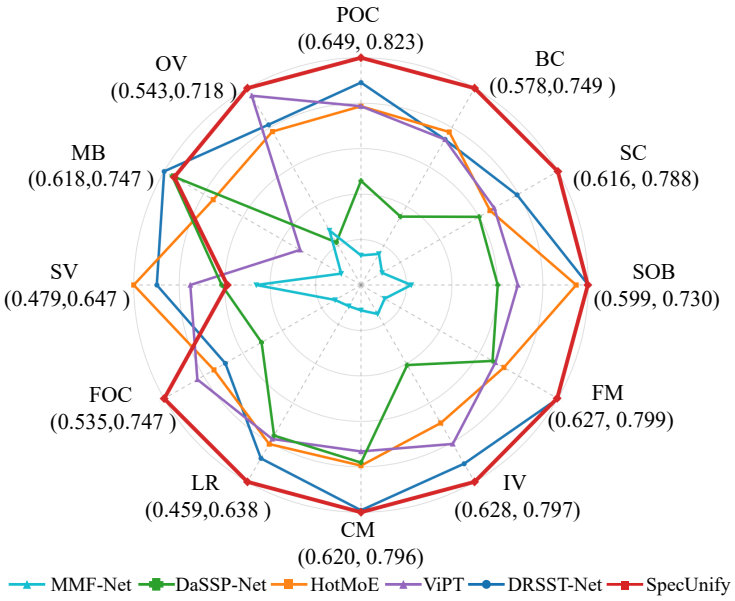

Figure 4: MUST attribute comparison with reference scores from (Li et al., 2026). Parentheses give raw (AUC, DP_{20}) in $[0, 1]$. Each axis ranges from its minimum plotted AUC minus 0.025 (0.10 for SV) to its maximum. Radial distances are not comparable across attributes.

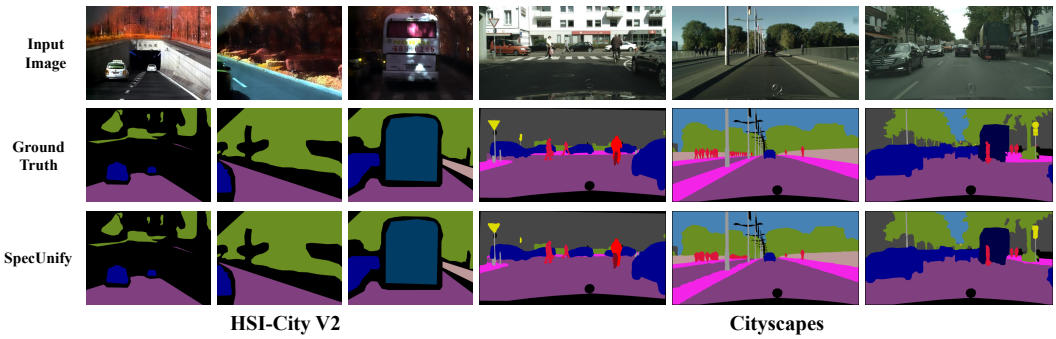

Figure 5: Qualitative segmentation examples on HSI-City v2 (left three columns) and Cityscapes (right three columns). Rows show inputs, ground truth, and SpecUnify predictions. Colors denote semantic classes.

The attributes describe overlapping sequence subsets, not only frames exhibiting the named challenge.

F Qualitative Results

F.1 Semantic Segmentation

Figure 5 places HSI-City v2 and Cityscapes predictions alongside their inputs and annotations. The same jointly trained model produces both sets of predictions: only RGB inputs pass through lifting, while the representation generator, token interface, and segmentation backend are shared. The comparison makes the spatial correspondence between predictions and annotated regions visible across the two input modalities.

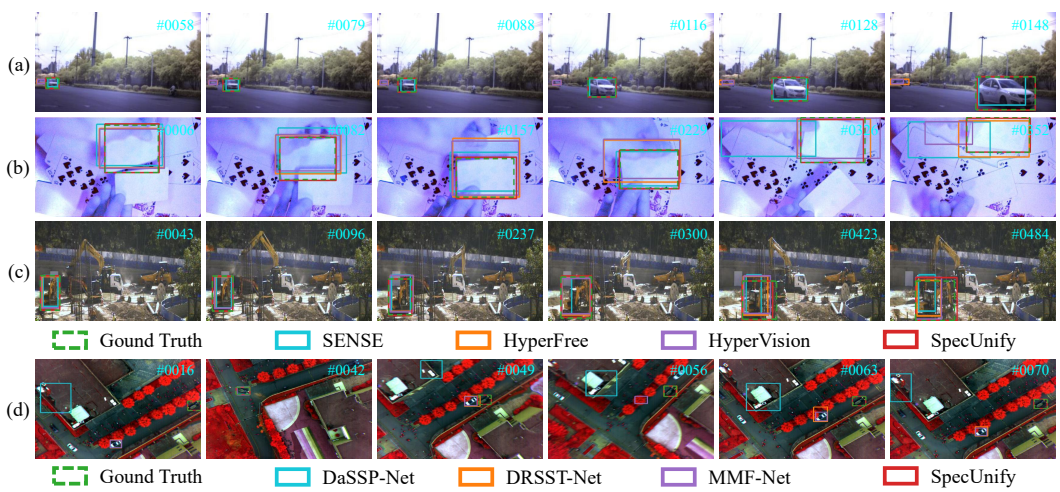

Figure 6: Tracking examples on HOTC2023: (a) *car50*, NIR; (b) *cards16*, RedNIR; (c) *excavator*, VIS; and (d) *MB-Car-001* on MUST. Green dashed boxes denote ground truth and red boxes SpecUnify; other methods follow the legends. Cyan numbers are frame indices. HOTC2023 uses the supplied false-color images; MUST uses a fixed mapping of its multispectral bands.

F.2 Object Tracking

In the displayed frames, SpecUnify follows the changing vehicle extent in *car50* and more closely aligns with the card and excavator in *cards16* and *excavator*. The MUST example highlights localization amid nearby vehicles and background structures. These examples illustrate how the shared tracker handles localization across distinct spectral configurations, including scenes where competing boxes cover background or drift from the target.