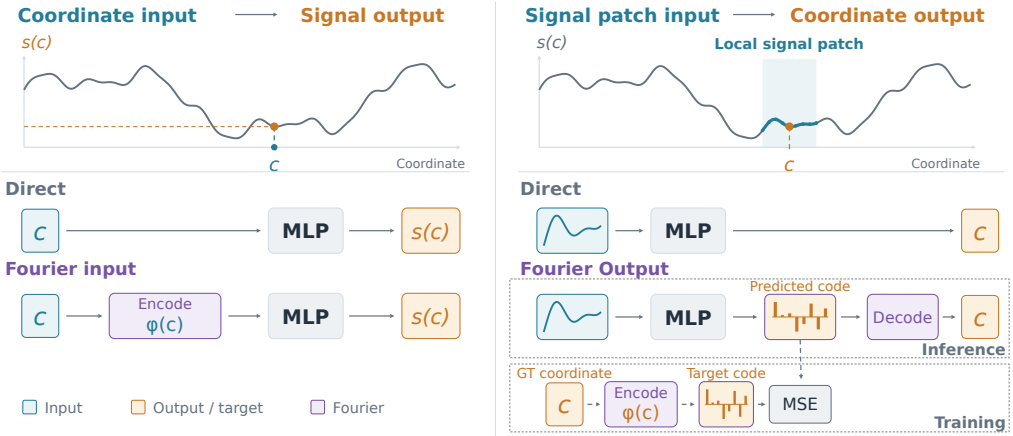
FOURIER OUTPUT FOR VISUAL COORDINATE REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Coordinate regression is a fundamental component of many computer vision tasks, yet most existing methods rely on direct scalar regression, coordinate discretization and classification, or task-specific output representations. We investigate whether a common representation can improve coordinate prediction across diverse visual tasks. While Fourier encodings are widely used to represent coordinate inputs, their use as representations for coordinate outputs remains underexplored. We introduce Fourier Output, which represents each scalar coordinate using multi-frequency sine–cosine features and directly supervises predictions in the Fourier encoding space, while leaving the underlying backbone unchanged. To efficiently recover coordinates at inference time, we develop a hierarchical decoding algorithm that progressively resolves phase ambiguity from low to high frequencies without candidate search. Through a controlled patch localization study, we systematically analyze the representation, supervision, frequency configuration, and decoding strategy, establishing a practical design for Fourier Output. We then evaluate it on human pose estimation, homography estimation, optical flow, and vision-language model (VLM)-based localization. Across these diverse tasks, Fourier Output consistently improves prediction accuracy over direct coordinate regression with minimal architectural changes, while training curves show that it can reach comparable accuracy with fewer training updates. These results demonstrate that Fourier features, widely used for representing coordinate inputs, can also provide a simple and general representation for visual coordinate outputs.



(a) 1D signal fitting (INR) (b) 1D signal localization

Figure 1: **Fourier representations from inputs to outputs.** (a) An implicit neural representation (INR) maps a coordinate to a signal value, using either direct coordinates or Fourier-encoded inputs (Tancik et al., 2020). (b) Our reverse-direction task localizes a signal patch by predicting its center coordinate. We input a patch rather than a value to avoid ambiguity when multiple coordinates correspond to the same signal value. Fourier Output supervises the predicted code against $\phi(c)$ and decodes it at inference. On a fixed signal, matched Direct and Fourier models achieve test MAE of 3.25×10^{-4} and 4.47×10^{-5} in $[0, 1]$ coordinate units. Curves are illustrative.

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.

1 INTRODUCTION

Coordinate regression is an essential component of computer vision, with applications including pose estimation (Newell et al., 2016; Sun et al., 2018b; Xiao et al., 2018), optical flow estimation (Dosovitskiy et al., 2015; Sun et al., 2018a; Teed & Deng, 2020), and homography estimation (DeTone et al., 2016). Although these tasks differ in their inputs, coordinate dimensions, and prediction density, they share the need to predict numerical coordinates accurately.

Existing approaches to coordinate prediction often rely on either direct regression or task-specific designs. For example, optical flow methods use iterative refinement to improve prediction accuracy (Sun et al., 2018a; Teed & Deng, 2020), while human pose estimation methods commonly use heatmaps to represent joint locations (Newell et al., 2016; Xiao et al., 2018). In scene coordinate regression, HSCNet (Li et al., 2020) decomposes coordinate prediction into hierarchical classification of coarse locations and fine-grained coordinate regression. These approaches illustrate different ways to organize coordinate prediction, from refining estimates to changing the output representation. Alongside these task-specific advances, we seek a common output representation that can improve coordinate prediction across tasks with minimal architectural changes.

Interestingly, a closely related problem has been extensively studied on the input side. Fourier and sinusoidal encodings are widely used to represent coordinates, including spatial positions in neural rendering (Mildenhall et al., 2020) and token positions in Transformer-based models (Vaswani et al., 2017). Despite their widespread use for coordinate inputs, Fourier encodings have received much less attention as representations for coordinate outputs. This observation leads us to ask: can Fourier encodings provide a similarly general and easy-to-use representation for output coordinates?

To investigate this question, we present *Fourier Output*, inspired by Fourier feature mappings for input coordinates (Tancik et al., 2020). We replace each direct scalar coordinate output c with a series of sine–cosine pairs at different frequencies:

$$\phi(c) = \left[\sin(\omega_0 c),\, \cos(\omega_0 c),\, \ldots,\, \sin(\omega_{K-1} c),\, \cos(\omega_{K-1} c)\right]^{\mathsf{T}} \in \mathbb{R}^{2K},$$

where K is the number of frequencies and ω_k is the angular frequency of the k-th pair. We supervise the predicted Fourier code directly in the encoding space. At inference time, our hierarchical decoding algorithm progressively recovers the coordinate from low to high frequencies, avoiding candidate search. Because this representation applies separately to each scalar coordinate, it can accommodate different output dimensions and structures. Integrating it into existing models only requires adapting the output head and supervision and adding a decoding step, while retaining the existing backbone.

To illustrate Fourier Output, we compare a 1D implicit neural representation (INR) task (Tancik et al., 2020) in Figure 1(a) with a 1D signal localization task in Figure 1(b). A 1D INR maps a coordinate or its Fourier code to a signal value. As a reverse-direction counterpart, the 1D signal localization task predicts the center coordinate of a local patch from a fixed signal. The output representation can be either a scalar or a Fourier code.

We systematically examine the design of Fourier Output through controlled image patch localization experiments. This study supports predicting and directly supervising Fourier codes. It also shows that adding higher frequencies does not necessarily improve accuracy, motivating a moderate frequency configuration. We then evaluate Fourier Output on pose estimation, homography estimation, optical flow estimation, and vision-language model (VLM)-based localization. To assess its applicability across tasks, we select published methods with minimal task-specific design and replace their direct coordinate output heads with Fourier Output. Results show that Fourier Output consistently improves the precision of the direct output baseline with negligible overhead, validating the effectiveness and generality of Fourier Output. In particular, the successful application in VLM-based localization demonstrates the potential of Fourier Output in large vision-language models for language-conditioned coordinate prediction.

Our contributions can be summarized as follows:

- We introduce Fourier Output, a simple and general representation for visual coordinate regression that predicts multi-frequency Fourier encodings instead of scalar coordinates and can be integrated into existing models with minimal architectural changes.

- We systematically study its representation, supervision, frequency configuration, and decoding strategy, and develop an efficient hierarchical decoding algorithm that recovers coordinates without candidate search, making Fourier Output suitable for both sparse and dense prediction.

- We demonstrate the effectiveness of Fourier Output across diverse coordinate regression tasks, including pose estimation, homography estimation, optical flow, and VLM-based localization, where it improves prediction accuracy and can reach comparable accuracy with fewer training updates.

2 Related Work

Input Coordinate Representations. Input coordinate representations influence how neural networks learn spatial information. Work on spectral bias shows that neural networks tend to learn low-frequency components more readily than high-frequency ones (Rahaman et al., 2019). Fourier feature mappings address this difficulty in coordinate-based networks by embedding input coordinates into sinusoidal features with a suitable frequency bandwidth (Tancik et al., 2020). Related sinusoidal encodings represent token positions in Transformers (Vaswani et al., 2017) and spatial coordinates and viewing directions in NeRF (Mildenhall et al., 2020). Input positional encodings demonstrate the importance of how coordinate information enters a network. Fourier Output instead studies the representation of coordinates that the network predicts.

Output Coordinate Representations. Alternatives to direct coordinate regression include spatial heatmaps (Newell et al., 2016; Xiao et al., 2018), from which Integral regression (Sun et al., 2018b) and DSNT (Nibali et al., 2018) recover continuous coordinates through differentiable expectations. SimCC (Li et al., 2022) instead classifies coordinates along each axis separately, while HSCNet (Li et al., 2020) combines progressively finer location classes with coordinate regression. These methods represent coordinates through spatial distributions or discrete classes.

Sinusoidal representations have also been used as prediction targets. Biternion Nets (Beyer et al., 2015) represent angles with sine–cosine pairs, while Phase-Shifting Coder (PSC) and its dual-frequency variant (Yu & Da, 2023) encode orientations as phase-shifted cosine values. These angle representations address intrinsic periodicity and rotational symmetry. Fourier Head (Gillman et al., 2025) uses Fourier series to parameterize output probability distributions over ordered tokens. Our work instead investigates how Fourier encodings can improve general coordinate regression, by directly predicting multi-frequency codes and decoding them into scalar coordinates.

SACReg (Revaud et al., 2024) is the closest prior work, predicting multi-frequency sinusoidal encodings of 3D scene coordinates to generalize across coordinate ranges. Its representation design is studied within scene coordinate regression, and its decoding algorithm uses candidate search for nonlinear least-squares inversion. We systematically examine representation, supervision, frequency, and decoding choices in a controlled setting and evaluate Fourier Output across different tasks. Our hierarchical decoding algorithm avoids candidate search, reducing the decoding cost for dense predictions.

Visual Coordinate Regression. Coordinate prediction spans tasks with different geometric structures and sources of information. Human pose estimation predicts joint locations from images (Toshev & Szegedy, 2014) or lifts sequences of 2D joints to 3D using spatial and temporal structure (Mehraban et al., 2024). Homography estimation can regress the displacements of four image corners (DeTone et al., 2016). For optical flow, PWC-Net (Sun et al., 2018a) combines pyramids, warping, and cost volumes, while RAFT (Teed & Deng, 2020) iteratively updates flow estimates using correlation features. CroCo v2 (Weinzaepfel et al., 2023) transfers representations learned through cross-view completion to dense correspondence tasks. More recently, Talking Points (Rusanovsky et al., 2026) uses a VLM to localize pixels from natural-language descriptions. These methods develop task-specific architectures, representations, or training procedures. Our focus is a general coordinate output interface across these tasks that can be easily integrated into backbones.

3 FOURIER OUTPUT

3.1 COORDINATE REGRESSION WITH OUTPUT REPRESENTATIONS

We first describe a general formulation of coordinate regression with output representations. Consider predicting a scalar coordinate $c \in \mathbb{R}$ from a feature vector $\mathbf{h} \in \mathbb{R}^d$. Given a dataset $\mathcal{D} = \{(\mathbf{h}_i, c_i)\}_{i=1}^N$ and a coordinate encoding function ϕ, the network f_θ with learnable parameters θ predicts the encoded target. We denote the target and predicted codes by

$$\mathbf{z}_i = \phi(c_i), \qquad \hat{\mathbf{z}}_i = f_\theta(\mathbf{h}_i). \tag{1}$$

We train the network by minimizing a per-sample loss ℓ in the encoding space:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(\mathbf{h}_i), \phi(c_i)) \,. \tag{2}$$

At inference time, a decoding algorithm Dec maps the predicted code to a scalar coordinate:

$$\hat{c}_i = \mathrm{Dec}(\hat{\mathbf{z}}_i) = \mathrm{Dec}(f_\theta(\mathbf{h}_i)) \,. \tag{3}$$

This formulation extends to multidimensional coordinate regression by encoding and decoding each coordinate component separately.

3.2 FOURIER ENCODING AND SUPERVISION

Following sinusoidal and Fourier feature encodings (Vaswani et al., 2017; Tancik et al., 2020), we define the coordinate encoding as

$$\phi(c) = \big[\sin(\omega_0 c), \cos(\omega_0 c), \ldots, \sin(\omega_{K-1} c), \cos(\omega_{K-1} c)\big]^\mathsf{T} \in \mathbb{R}^{2K}, \tag{4}$$

where K is the number of frequencies and ω_k is the angular frequency of the k-th sine–cosine pair. We use geometrically spaced frequencies:

$$\omega_k = \omega_{\min} b^k, \qquad k = 0, 1, \ldots, K-1, \tag{5}$$

where $\omega_{\min} > 0$ is the lowest frequency and $b > 1$ is the ratio between consecutive frequencies. The highest frequency is $\omega_{\max} = \omega_{\min} b^{K-1}$.

The network f_θ predicts the $2K$ code components for each scalar coordinate. We directly supervise the predicted code. By default, we use mean squared error (MSE), with equal weight assigned to all frequencies:

$$\ell_{\mathrm{MSE}}(\hat{\mathbf{z}}_i, \mathbf{z}_i) = \frac{1}{2K} \|\hat{\mathbf{z}}_i - \mathbf{z}_i\|_2^2 \,. \tag{6}$$

3.3 HIERARCHICAL DECODING

We design a decoding algorithm for the Fourier representation based on the properties of sinusoidal functions. For comparable phase errors, higher frequencies yield smaller coordinate errors. However, because sinusoidal functions are periodic, a sine–cosine pair at a high frequency may correspond to multiple candidate coordinates. We resolve this ambiguity using a coarse coordinate estimate from lower frequencies, a mechanism similar to multi-frequency phase unwrapping (Zhang et al., 2017). The resulting hierarchical decoding algorithm is summarized in Algorithm 1.

First, we initialize the estimate using the lowest frequency and ensure that its period exceeds the interval width $2R$ for coordinates $c \in [-R, R]$. Then, we progressively refine the coordinate estimate using higher frequencies. At each level, we compute the phase from the predicted sine–cosine pair using $\mathrm{atan2}$ and choose the candidate coordinate closest to the current estimate. We then clamp the phase correction to $[-\tau, \tau]$, where $\tau \in [0, \pi]$ is measured in radians, limiting how much each new frequency can change the estimate.

Algorithm 1 Hierarchical decoding algorithm for Fourier Output

Input: Fourier code $\hat{\mathbf{z}} \in \mathbb{R}^{2K}$, coordinate bound $R > 0$, and phase clamp parameter $\tau \in [0, \pi]$ (in radians).
Output: Scalar coordinate $\hat{c} \in [-R, R]$.
Given: Encoding frequencies $\omega_k = \omega_{\min} b^k$, with $\omega_{\min} R < \pi$.

$\alpha \leftarrow \operatorname{atan2}(\hat{z}_0, \hat{z}_1)$ ▷ Initial phase from the lowest frequency
$\omega \leftarrow \omega_{\min}$
for $k = 1, \ldots, K-1$ **do**
  $\beta \leftarrow \operatorname{atan2}(\hat{z}_{2k}, \hat{z}_{2k+1})$ ▷ Phase at the current frequency
  $\tilde{\alpha} \leftarrow b\alpha$ ▷ Express the previous estimate at this frequency
  $m \leftarrow \operatorname{round}((\tilde{\alpha} - \beta)/(2\pi))$
  $\beta \leftarrow \beta + 2\pi m$ ▷ Choose the closest phase candidate
  $\alpha \leftarrow \tilde{\alpha} + \operatorname{clamp}(\beta - \tilde{\alpha}, -\tau, \tau)$ ▷ Limit the phase correction
  $\omega \leftarrow b\omega$
$\hat{c} \leftarrow \operatorname{clamp}(\alpha/\omega, -R, R)$
return $\hat{c}$

4 Empirical Design Study

4.1 Patch Localization Setup

To evaluate Fourier Output and guide its design choices, we use a patch localization task on a subset of ImageNet (Deng et al., 2009). Given an image and a square patch randomly cropped from it (the query patch), a ViT (Dosovitskiy et al., 2021) predicts the query patch's center coordinates within the image. The image and query patch are resized to 128×128 and 64×64, respectively.
We keep the backbone architecture fixed and use matched training protocols within each comparison, varying the output head, representation, supervision, or decoding procedure according to the design choice under study.

4.2 Design Analysis

Can alternative representations match Fourier Output? In Table 1, we compare Fourier Output with direct coordinate regression (Direct) and two alternative encodings. All representations follow the general framework introduced in Section 3.1, using their respective encoding functions ϕ.

Fourier Output reduces test MAE by approximately 41% relative to Direct. Polynomial encoding expands a scalar coordinate into its powers and matches Fourier's output dimensionality, but does not improve on Direct. To examine the role of periodicity, we also evaluate a multiscale sawtooth encoding. Each channel produces a periodic linear ramp at a different scale. This encoding yields a higher test MAE than Direct. These comparisons support Fourier's smooth, multiscale periodic encoding over the tested alternatives.

Table 1: Output representations on patch localization, with $c \in [0, 1]$ and $\omega_k = 2.5 \cdot 2^k$. All three encodings use unnormalized squared code distance over a 4,001-point uniform grid for decoding. Fourier reuses the checkpoint selected with hierarchical decoding.

Representation	Encoding function	Channels / scalar	Test MAE ↓
Direct	c	1	0.0628
Polynomial	$(c^j)_{j=0}^{7}$	8	0.0652
Multiscale sawtooth	$\left(\operatorname{frac}(2^{j-1}c)\right)_{j=0}^{7}$	8	0.0906
Fourier	$\left(\sin(\omega_k c), \cos(\omega_k c)\right)_{k=0}^{3}$	8	**0.0373**

Should we directly supervise the predicted Fourier code? Table 2 compares direct supervision of the predicted Fourier code with two alternatives. Encode(Direct) predicts scalar coordinates and applies supervision after Fourier encoding, whereas Decode(Fourier) predicts Fourier codes and applies supervision after decoding them into coordinates. Both alternatives yield higher test MAE,

Table 2: Output and supervision spaces on patch localization. All variants use MSE; Fourier codes are decoded hierarchically with $\tau = 0.5$.

Method	Network output	Supervision space	Test MAE ↓
Direct	Coordinate	Coordinate	0.0628
Fourier	Fourier code	Fourier code	**0.0368**
Encode(Direct)	Coordinate	Fourier code	0.1769
Decode(Fourier)	Fourier code	Coordinate	0.0688

supporting direct code supervision. Intuitively, this gives each predicted sine–cosine component an explicit target, without requiring gradients to pass through an additional coordinate encoding or decoding step.

How should we set the frequency hyperparameters? Following the coordinate-range condition in Section 3.3, we set $\omega_{\min} = 2.5$, giving a period that covers the patch localization coordinate range $[0, 1]$ with a margin. We use $b = 2$ as an empirical default, leaving K to be selected on the validation set.

When the correct period branch is selected, a phase error $\Delta\theta_k$ at frequency ω_k corresponds to a coordinate error $|\Delta c_k| = |\Delta\theta_k|/\omega_k$. Thus, for comparable phase errors across frequencies, higher frequencies offer finer coordinate precision. However, higher frequencies may be harder to learn, and their supervision may also affect learning at lower frequencies. Increasing K therefore need not lead to continued improvements.

Figure 2(a) reports test MAE averaged over three initialization seeds for each value of K. Mean test MAE decreases as K increases from 1 to 4, but does not improve further and deteriorates at $K = 12$ and $K = 16$ under the same training budget. We select $K = 4$ using mean validation MAE and adopt it as the default for this patch localization setting.

Which decoding algorithm should we use? We compare the hierarchical decoding algorithm described in Section 3.3 with grid search and the SACReg solver (Revaud et al., 2024). As shown in Table 3, hierarchical decoding achieves slightly lower test MAE than both alternatives while substantially reducing decoding time. In our dense decoding benchmark, it is approximately $12\times$ faster than grid search and $40\times$ faster than the SACReg solver.

Table 3: Decoding accuracy and runtime ($K = 4$). Runtime covers decoding 320×384 synthetic 2D points on an H200 GPU (FP32, median of 35 trials), including sine–cosine pair normalization for grid search and SACReg.

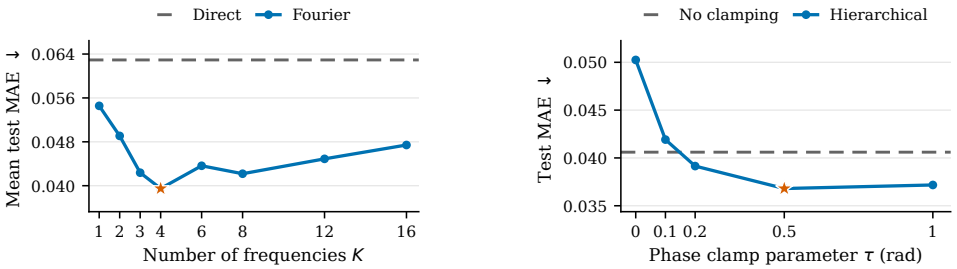
Decoding algorithm	Test MAE ↓	Dense decoding (ms) ↓
Grid search	0.0391	4.818
SACReg solver	0.0390	16.706
Hierarchical ($\tau = 0.5$)	**0.0368**	**0.415**

Figure 2(b) examines the phase clamp parameter τ introduced in Section 3.3, using the fixed $K = 4$ checkpoint evaluated in Table 3. Moderate clamping ($\tau = 0.5$ or 1.0) yields lower test MAE than unrestricted updates, whereas an overly small τ limits useful refinement. We use $\tau = 0.5$.

5 Applications and Evaluation

We evaluate Fourier Output in three settings: sparse coordinate regression, dense coordinate regression, and VLM-based localization. Together, they cover image inputs, sequences of keypoints, and language-conditioned prediction, with outputs ranging from individual points to dense displacement fields. This diversity allows us to assess whether Fourier Output can serve as a common output representation across coordinate regression tasks with different input and output structures.

Model selection. To study Fourier Output as a general coordinate regression interface, we select models that predict coordinates directly from learned features, without relying on specific modules



(a) Frequency count

(b) Phase clamp

Figure 2: Hyperparameter analysis on patch localization. (a) Mean test MAE over three initialization seeds, with $b = 2$, $\omega_{\min} = 2.5$, and $\tau = 0.5$. (b) Clamp sweep on the fixed $K = 4$ checkpoint from Table 3; $\tau = 0$ uses only the lowest frequency. Stars mark validation-selected settings.

related to coordinates, such as heatmap prediction, explicit cost-volume matching (Sun et al., 2018a; Teed & Deng, 2020), or iterative coordinate refinement (Teed & Deng, 2020). This distinguishes our experimental setting from approaches such as SimpleBaseline (Xiao et al., 2018), which uses a deconvolutional head to progressively upsample and produce joint heatmaps, and RAFT (Teed & Deng, 2020), which combines correlation volumes with recurrent flow updates. Our selected models allow us to replace the coordinate output head and its supervision while retaining the feature extraction backbone, minimizing the need for task-specific architectural adaptation.

For each task, Direct and Fourier Output use the same backbone architecture, data split, and training budget. We follow the formulation and decoding algorithm in Section 3, using $b = 2$, $K = 4$, and $\tau = 0.5$ unless otherwise specified.

5.1 Sparse Coordinate Regression

2D pose estimation. Given a person-centered image crop, the task aims to predict 16 two-dimensional joint locations. We adopt a single-stage DeepPose-style regressor (Toshev & Szegedy, 2014) with a ResNet-50 (He et al., 2016) pretrained on ImageNet. We train for 140 epochs on MPII (Andriluka et al., 2014), replacing each scalar coordinate output with a Fourier code using the default settings. We report head-normalized keypoint accuracy PCKh@0.5.

Homography estimation. Given two grayscale image patches, the model predicts the horizontal and vertical displacements of their four corresponding corners. We follow the regression formulation of HomographyNet (DeTone et al., 2016), using 128×128 patches and corner perturbations in $[-32, 32]$ pixels. Training pairs are generated from the COCO training images (Lin et al., 2014) and evaluation uses 5,000 fixed pairs from the validation images. We compare a HomographyNet-style CNN with an MLP head and a joint ViT with a linear readout, training each for 48 epochs. Fourier Output replaces the eight scalar displacements with their codes, using $K = 4$ for the CNN and $K = 2$ for the ViT. Both Direct and Fourier use MSE. We report mean average corner error (MACE) in pixels.

3D pose lifting. This task predicts 3D joint positions from sequences of detected 2D joints. We use MotionAGFormer-Base (Mehraban et al., 2024) on Human3.6M (Ionescu et al., 2014), with 243-frame inputs and 17 joints. We retain the spatial and temporal backbone and expand the final layer from three outputs per joint to $3 \times 2K$ Fourier components. Both variants are trained from scratch for 60 epochs using plain MSE, omitting the original scale and velocity auxiliary losses. Fourier decoding uses $\tau = 0.2$. We report mean per-joint position error before and after Procrustes alignment, denoted P1 and P2, respectively.

Results. Table 4 shows improvements across these sparse prediction settings. For 2D pose estimation, Fourier Output increases PCKh@0.5 from 81.91% to 85.24%. For homography estimation, it reduces MACE by 4.1% with the CNN and 37.2% with the ViT, showing that the magnitude of the gain depends on the backbone. In 3D pose lifting, P1 decreases from 44.489 to 43.419 mm, with a smaller improvement in P2. In addition, Figure 3(a) shows that Fourier Output reaches comparable

Table 4: Sparse coordinate regression with matched backbones and training budgets. PCKh is reported as a percentage.

Task / backbone	Evaluation	Epochs	Metric	Direct	Fourier
DeepPose-style / ResNet-50	MPII val	140	PCKh@0.5 ↑	81.91	85.24
Homography / CNN+MLP	COCO val	48	MACE (px) ↓	4.05	3.88
Homography / ViT	COCO val	48	MACE (px) ↓	2.54	1.60
MotionAGFormer-Base	H36M test	60	P1 (mm) ↓	44.48	43.41
MotionAGFormer-Base	H36M test	60	P2 (mm) ↓	36.52	36.11

Table 5: Optical flow EPE in pixels (↓) with CroCo v2 ViT-L/Base and sliding-window overlap 0.9. Sintel is used only for evaluation.

	Things test subset		Sintel train transfer	
Output / loss	Clean	Final	Clean	Final
Direct L1	3.3165	3.1328	1.2576	2.5131
Fourier cosine	3.1758	2.9697	1.2117	2.4372

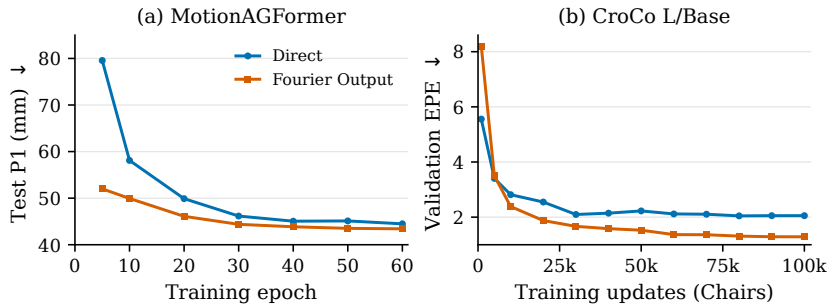


Figure 3: Training curves for sparse and dense coordinate regression, with one run per variant and no smoothing. (a) MotionAGFormer-Base H36M test P1 with flip augmentation, shown from epoch 5, then sampled every 10 epochs starting at epoch 10. (b) CroCo v2 ViT-L/Base FlyingChairs validation EPE, sampled at 1,000 and 5,000 updates, then every 10,000 updates during the first training stage (Direct L1, Fourier cosine).

prediction accuracy in fewer training epochs, with a larger advantage early in training. These results support applying the same output representation to both image-based regression and prediction from structured keypoint sequences.

5.2 Dense Coordinate Regression

Optical flow predicts a two-dimensional displacement for each pixel between two frames. We use CroCo v2 (Weinzaepfel et al., 2023) with the ViT-L/Base encoder/decoder configuration. Starting from the released self-supervised initialization, each variant is trained for 100,000 updates on FlyingChairs (Dosovitskiy et al., 2015), followed by 100,000 updates on FlyingThings3D (Mayer et al., 2016). Fourier Output predicts $K = 8$ sine–cosine pairs for each displacement component and uses cosine loss in the encoding space; Direct uses an L1 coordinate loss. The comparison therefore evaluates each representation together with its supervision choice.

We evaluate the final checkpoints on a fixed subset of 512 FlyingThings3D test pairs per rendering pass and all 1,041 pairs of the Sintel training split (Butler et al., 2012). Both models use the same sliding-window inference with uniform averaging of predicted displacements. Table 5 reports endpoint error (EPE) for the clean and final rendering passes.

Fourier Output lowers EPE in all four evaluations in Table 5. Figure 3(b) also shows lower validation EPE from 10,000 updates onward during FlyingChairs training, following an initial lag. Decoding efficiency is examined separately in Section 4.

Table 6: Talking Points test results after 32,385 training updates for both models. PCK values are percentages.

Model / output	mPCK ↑	PCK@0.05 ↑	Norm. L2 ↓
Direct + sigmoid	86.020	38.939	0.0762
Fourier (affine init.)	86.831	42.669	0.0743

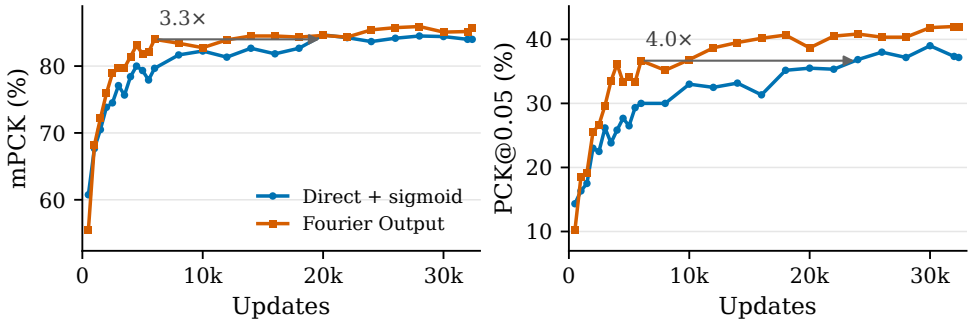


Figure 4: Point Localizer validation accuracy. Arrows mark when Direct first matches Fourier's accuracy at 6,000 updates, using the best value observed so far. Time per update on the same GPU is 4.94 s for Direct and 4.92 s for Fourier.

5.3 VLM-based Localization

Given an image and a natural-language description of a point, this task predicts its two-dimensional location. We adopt the Point Localizer from Talking Points (Rusanovsky et al., 2026), built on a 7B vision–language model. We retain the VLM backbone and replace the final two-coordinate output layer with a Fourier head. Direct uses a sigmoid output and coordinate MSE; Fourier Output uses a linear output and code-space MSE. For Fourier, we initialize the output layer with a reduced weight scale and a bias set to the mean training code. We report mPCK, the mean of PCK@0.1 and PCK@0.2, along with PCK@0.05 and mean normalized Euclidean error.

Table 6 reports the final test results. Fourier Output increases mPCK from 86.020% to 86.831% and improves PCK@0.05, which measures precise localization, by 3.7 percentage points. Normalized coordinate error also decreases from 0.0762 to 0.0743.

Figure 4 shows that Direct requires $3.3\times$ and $4.0\times$ as many updates to match the validation mPCK and PCK@0.05 attained by Fourier Output at 6,000 updates, respectively. These results demonstrate the applicability of Fourier Output to language-conditioned localization in a VLM.

Summary across tasks. The benefits of Fourier Output are reflected in both final prediction accuracy and the number of training updates needed to reach a given accuracy, with their relative importance varying across tasks and backbones.

6 Conclusion

We study Fourier Output, a general representation for coordinate regression that replaces each scalar coordinate output with Fourier features. Our controlled design study supports direct supervision in the encoding space and hierarchical decoding for accurate and efficient coordinate recovery. Across the evaluated tasks, Fourier Output improves prediction accuracy over direct coordinate regression. Training curves on representative tasks also show that it can reach comparable accuracy with fewer training updates. These findings suggest that Fourier features, widely used for input positional encoding, can also serve as a common representation for coordinate outputs.

AI USE DISCLOSURE

We used generative AI tools to help plan experiments, review methodological choices, implement our methods, translate text, and interpret experimental findings. We did not use these tools to generate synthetic datasets, develop theoretical or conceptual models, formulate mathematical propositions, supply essential steps in mathematical proofs, write proofs, propose or revise hypotheses, clean or restructure datasets, or conduct qualitative or thematic analyses.

AI assistance also supported literature search and review, reference preparation, manuscript organization, drafting and language editing, and the preparation of figures and plotting code.

We have reviewed all AI-assisted work. Implementation and numerical checks included code tests and comparisons of reported values with experiment logs. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3686–3693, 2014. URL https://openaccess.thecvf.com/content_cvpr_2014/html/Andriluka_2D_Human_Pose_2014_CVPR_paper.html.

Lucas Beyer, Alexander Hermans, and Bastian Leibe. Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels. In *Pattern Recognition (GCPR)*, pp. 157–168, 2015. doi: 10.1007/978-3-319-24947-6_13. URL https://doi.org/10.1007/978-3-319-24947-6_13.

Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A Naturalistic Open Source Movie for Optical Flow Evaluation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 611–625, 2012. doi: 10.1007/978-3-642-33783-3_44. URL https://doi.org/10.1007/978-3-642-33783-3_44.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/cvpr.2009.5206848. URL https://doi.org/10.1109/CVPR.2009.5206848.

Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep Image Homography Estimation. *arXiv preprint arXiv:1606.03798*, 2016. URL https://arxiv.org/abs/1606.03798.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow With Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2758–2766, 2015. URL https://openaccess.thecvf.com/content_iccv_2015/html/Dosovitskiy_FlowNet_Learning_Optical_ICCV_2015_paper.html.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Nate Gillman, Daksh Aggarwal, Michael Freeman, and Chen Sun. Fourier Head: Helping Large Language Models Learn Complex Probability Distributions. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d3e38662377dcce6203f4bb91b2a9b03-Abstract-Conference.html.

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 (CVPR)*, pp. 770–778, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014. doi: 10.1109/tpami.2013.248. URL https://doi.org/10.1109/TPAMI.2013.248.

Xiaotian Li, Shuzhe Wang, Yi Zhao, Jakob Verbeek, and Juho Kannala. Hierarchical Scene Coordinate Classification and Regression for Visual Localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11983–11992, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Li_Hierarchical_Scene_Coordinate_Classification_and_Regression_for_Visual_Localization_CVPR_2020_paper.html.

Yanjie Li, Sen Yang, Peidong Liu, Shoukui Zhang, Yunxiao Wang, Zhicheng Wang, Wankou Yang, and Shu-Tao Xia. SimCC: A Simple Coordinate Classification Perspective for Human Pose Estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 89–106, 2022. doi: 10.1007/978-3-031-20068-7_6. URL https://doi.org/10.1007/978-3-031-20068-7_6.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014. doi: 10.1007/978-3-319-10602-1_48. URL https://doi.org/10.1007/978-3-319-10602-1_48.

Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4040–4048, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/Mayer_A_Large_Dataset_CVPR_2016_paper.html.

Soroush Mehraban, Vida Adeli, and Babak Taati. MotionAGFormer: Enhancing 3D Human Pose Estimation With a Transformer-GCNFormer Network. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6920–6930, 2024. URL https://openaccess.thecvf.com/content/WACV2024/html/Mehraban_MotionAGFormer_Enhancing_3D_Human_Pose_Estimation_With_a_Transformer-GCNFormer_Network_WACV_2024_paper.html.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 405–421, 2020. doi: 10.1007/978-3-030-58452-8_24. URL https://doi.org/10.1007/978-3-030-58452-8_24.

Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked Hourglass Networks for Human Pose Estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 483–499, 2016. doi: 10.1007/978-3-319-46484-8_29. URL https://doi.org/10.1007/978-3-319-46484-8_29.

Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast. Numerical Coordinate Regression with Convolutional Neural Networks. *arXiv preprint arXiv:1801.07372*, 2018. URL https://arxiv.org/abs/1801.07372.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the Spectral Bias of Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 5301–5310, 2019. URL https://proceedings.mlr.press/v97/rahaman19a.html.

Jerome Revaud, Yohann Cabon, Romain Brégier, Jongmin Lee, and Philippe Weinzaepfel. SACReg: Scene-Agnostic Coordinate Regression for Visual Localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 688–698, 2024. URL https://openaccess.thecvf.com/content/CVPR2024W/3DMV/html/Revaud_SACReg_Scene-Agnostic_Coordinate_Regression_for_Visual_Localization_CVPRW_2024_paper.html.

Matan Rusanovsky, Shimon Malnick, and Shai Avidan. Talking Points: Describing and Localizing Pixels. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/313f68091be2eca09f37abeb2c50a018-Abstract-Conference.html.

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8934–8943, 2018a. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Sun_PWC-Net_CNNs_for_CVPR_2018_paper.html.

Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral Human Pose Regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 529–545, 2018b. URL https://openaccess.thecvf.com/content_ECCV_2018/html/Xiao_Sun_Integral_Human_Pose_ECCV_2018_paper.html.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7537–7547, 2020. URL https://proceedings.nips.cc/paper_files/paper/2020/hash/55053683268957697aa39fba6f231c68-Abstract.html.

Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 402–419, 2020. doi: 10.1007/978-3-030-58536-5_24. URL https://doi.org/10.1007/978-3-030-58536-5_24.

Alexander Toshev and Christian Szegedy. DeepPose: Human Pose Estimation via Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1653–1660, 2014. URL https://openaccess.thecvf.com/content_cvpr_2014/html/Toshev_DeepPose_Human_Pose_2014_CVPR_paper.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Philippe Weinzaepfel, Thomas Lucas, Vincent Leroy, Yohann Cabon, Vaibhav Arora, Romain Brégier, Gabriela Csurka, Leonid Antsfeld, Boris Chidlovskii, and Jerome Revaud. CroCo v2: Improved Cross-view Completion Pre-training for Stereo Matching and Optical Flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17969–17980, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Weinzaepfel_CroCo_v2_Improved_Cross-view_Completion_Pre-training_for_Stereo_Matching_and_ICCV_2023_paper.html.

Bin Xiao, Haiping Wu, and Yichen Wei. Simple Baselines for Human Pose Estimation and Tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 466–481, 2018. URL https://openaccess.thecvf.com/content_ECCV_2018/html/Bin_Xiao_Simple_Baselines_for_ECCV_2018_paper.html.

Yi Yu and Feipeng Da. Phase-Shifting Coder: Predicting Accurate Orientation in Oriented Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13354–13363, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Yu_Phase-Shifting_Coder_Predicting_

Accurate_Orientation_in_Oriented_Object_Detection_CVPR_2023_paper.html.

Minliang Zhang, Qian Chen, Tianyang Tao, Shijie Feng, Yan Hu, Hui Li, and Chao Zuo. Robust and efficient multi-frequency temporal phase unwrapping: optimal fringe frequency and pattern sequence selection. *Optics Express*, 25(17):20381–20400, 2017. doi: 10.1364/OE.25.020381. URL https://doi.org/10.1364/OE.25.020381.

A Implementation and Experimental Settings

This section specifies the protocols used for the results in the main paper. Table 7 summarizes the coordinate conventions and Fourier configurations. Frequencies are angular frequencies in the corresponding coordinate units; $\omega_{\max} = \omega_{\min} b^{K-1}$. The coordinate transformations below are task-specific preprocessing choices, rather than requirements of Fourier Output. Each application result and the 1D comparison use one training run per variant. The patch localization frequency sweep uses three initialization seeds, as detailed below.

Table 7: Fourier configurations used in the main paper, with the default setting shown for patch localization. All use $b = 2$. The intervals specify the coordinate support or decoding bounds in the units used by the encoding. Evaluation uses hierarchical decoding except for the explicitly stated decoding comparisons and the grid readout in Table 1.

Experiment	Coordinate convention / interval	$\omega_{\min}$	K	τ
1D localization	Signal position, $[1/16, 15/16]$	2.5	4	0.5
Patch localization	Image-relative position, $[0, 1]$	2.5	4	0.5
2D pose	Crop pixels, $[0, 256]$	$\pi/256$	4	0.5
Homography / CNN	Corner offset (px), $[-32, 32]$	$\pi/64$	4	0.5
Homography / ViT	Corner offset (px), $[-32, 32]$	$\pi/64$	2	0.5
3D pose lifting	Normalized root-relative, $[-0.75, 0.75]$	$2\pi/3$	4	0.2
Optical flow	Displacement (px), $[-400, 400]$	0.0078	8	0.5
VLM localization	Padded-image position, $[0, 1]$	2.5	4	0.5

A.1 1D Signal Localization

Signal and samples. The measured comparison in Figure 1 uses one fixed signal on $[0, 1]$. We generate three groups of sinusoids containing 2, 3, and 3 terms, with frequencies sampled uniformly from $[0.5, 2]$, $[3, 8]$, and $[12, 24]$ cycles per unit coordinate, respectively. Initial amplitudes are sampled uniformly from $[0.5, 1.5]$ and rescaled within each group so that half their squared sum equals one; phases are uniform on $[0, 2\pi]$. The groups have weights 1, 0.4, and 0.2. We use signal seed 42 and standardize the resulting signal using its mean and standard deviation on an 8,193-point uniform grid. This normalization is global, not per patch.

Each input contains 33 ordered signal values sampled uniformly within radius $1/16$ of its center, without observation noise. The center is uniform on $[1/16, 15/16]$. Training patches are sampled online; validation and test use fixed sets of 2,048 and 4,096 centers. Their sampling seeds are 44, 45, and 46 for training, validation, and test, respectively. The network receives only the patch values. The curves in Figure 1 are schematic; the caption reports measured localization errors, not an INR fitting comparison.

Training and evaluation. Both variants use three hidden layers of width 128 with GELU activations and a final linear layer. The trunk initialization is shared (seed 43). Output weights are Gaussian with standard deviation 10^{-3}; biases are 0.5 for Direct and $\phi(0.5)$ for Fourier. Both use mean MSE and AdamW with zero weight decay, batch size 256, and 32,000 updates (1,000 epochs of 32 updates). The learning rate follows cosine decay from 10^{-3} to 10^{-5} over the first 3,200 updates and remains at 10^{-5} thereafter. We evaluate validation MAE every 25 epochs and select each variant's checkpoint independently. The selected epochs are 950 for Direct and 1,000 for Fourier. Predictions are clamped to the valid center interval before computing scalar-coordinate MAE.

A.2 Patch Localization

Data and task generation. All main-paper patch localization experiments use 50,000 preprocessed 256×256 ImageNet images, sampled as 50 images per class with sampling seed 20260919. The fixed validation and test sets each contain 1,000 separate images. Exact file duplicates within training and between training and the held-out sets are excluded. These are experimental splits of the preprocessed image collection. For each image, we sample a square side length by rounding $256s$, where $s \sim \mathcal{U}(0.15, 0.4)$, then sample a valid integer top-left location uniformly. The full image

and cropped query are resized bilinearly to 128×128 and 64×64. RGB intensities are mapped to $[-1, 1]$. The target is the crop center divided by 256. Training crops change with each data traversal; validation and test crops remain fixed. Image order and crop geometry are shared across compared variants, using shuffle seed 44 and a crop generator determined by seed 42, traversal index, and image index.

Architecture. The two inputs have separate 16×16 patch embeddings, yielding 64 context tokens and 16 query tokens. We add fixed 2D sine–cosine positional embeddings and learned segment embeddings, concatenate the tokens with a learned CLS token, and apply six joint Transformer blocks with width 192, three attention heads, and MLP ratio 4. A final layer normalization is applied to the tokens. The 16 query tokens are flattened into a 3,072-dimensional vector and passed to one linear output layer. Direct predicts two scalars; each eight-channel encoding predicts 16 components. These models have 2,971,586 and 3,014,608 parameters, respectively.

Optimization and selection. Every run uses 2,400 optimizer updates and 300,000 image presentations, corresponding to six traversals of the 50,000-image training set. To retain the same update schedule across studies, each 5,000-image block comprises 39 batches of 128 and one batch of 8. Training uses FP32, AdamW with zero weight decay, and learning rate

$$\eta_t = 3 \times 10^{-4} \min\left(1, \frac{t+1}{120}\right) \frac{1 + \cos(\pi t/2400)}{2}, \qquad t = 0, \ldots, 2399. \tag{7}$$

Validation runs every 40 updates. Each checkpoint is selected by validation coordinate MAE, with the representation's designated decoding method. All reported patch MAEs average absolute error over both coordinates and all images after clamping predictions to $[0, 1]$.

Initialization and frequency sweep. The representation and supervision comparisons use initialization seed 43 and share the trunk weights. Output weights have standard deviation 2.5×10^{-4}; biases encode the center $c = 0.5$ in the corresponding representation. Polynomial, sawtooth, and Fourier share the same output-weight samples. Direct and Encode(Direct) share their head initialization, as do Fourier and Decode(Fourier). The frequency sweep evaluates $K \in \{1, 2, 3, 4, 6, 8, 12, 16\}$ and Direct with seeds 43, 44, and 45. It uses a separate head sampling scheme that matches each shared frequency's weight rows across values of K, with the same weight scale and center bias. Data and training schedules are fixed across seeds. We select a checkpoint per run, average the resulting metrics equally across seeds, and select K by mean validation MAE. This is an average of metrics, not predictions. The frequency sweep's $K = 4$ result therefore need not equal the single-run Fourier entry in the representation or supervision tables.

A.3 2D Pose Estimation

We use 22,246 training and 2,958 validation person instances from MPII. Person-centered affine crops are resized to 256×256 and normalized with ImageNet RGB statistics. Training augmentation includes horizontal flipping with probability 0.5, a scale multiplier sampled from a Gaussian with mean 1 and standard deviation 0.25 and clipped to $[0.75, 1.25]$, and rotation with probability 0.6 sampled from a zero-mean Gaussian with standard deviation 30° and clipped to $[-60^\circ, 60^\circ]$. Invisible joints and joints outside the crop are excluded from the training loss.

The ImageNet-pretrained ResNet-50 is followed by global average pooling and a single linear output layer. Its output weights are initialized with standard deviation 10^{-3} and zero bias. Direct regresses crop coordinates divided by 256; Fourier encodes coordinates in crop pixels using Table 7. Both minimize MSE averaged over valid joints, coordinate components, and output channels. We train all parameters for 140 epochs with Adam, batch size 128, initial learning rate 10^{-4}, zero weight decay, and seed 0. The learning rate is multiplied by 0.1 after epochs 84 and 119. Training and network evaluation use BF16 autocast, with decoding in FP32.

Validation is performed every two epochs, selecting the checkpoint with highest PCKh@0.5. Both selected checkpoints are from epoch 132 (one-based). The Fourier checkpoint was originally selected with a local refinement decoding implementation and subsequently evaluated with hierarchical decoding at $\tau = 0.5$, without checkpoint reselection. Evaluation uses no flip augmentation. Predictions are mapped back to the original image by the inverse crop transform. We follow the

MPII evaluator: distances are normalized by 0.6 times the annotated head-box diagonal, and the aggregate PCKh excludes pelvis and thorax.

A.4 Homography Estimation

We use all 118,287 COCO train2017 images and one fixed pair from each of the 5,000 val2017 images. Images are converted to grayscale and resized to 320×240. A 128×128 square is sampled with a 32-pixel boundary margin, and each corner's horizontal and vertical offsets are sampled independently and uniformly from the integers $-32, \ldots, 32$. The second patch is obtained by warping the full image with the inverse homography and cropping at the same location. Intensities are mapped to $[-1, 1]$. Training pairs are resampled each epoch; evaluation pairs use a fixed generator seeded by 2024 plus the image index.

The CNN uses eight convolutional layers and an MLP head with a 1,024-unit hidden layer and dropout 0.5. The ViT shares a 16×16 patch embedding between the two images, adds fixed 2D sine–cosine positional and learned segment embeddings, and jointly processes 128 tokens with six blocks of width 192, three heads, and MLP ratio 4. All final normalized tokens are flattened and fed to a single linear output layer; no CLS token is used. Direct predicts $(c + 32)/64$, while Fourier encodes pixel offsets c. Both use MSE.

Both backbones train from scratch for 48 epochs with batch size 64, learning rate 3×10^{-4}, zero weight decay, seed 0, and BF16 autocast. The CNN uses Adam with $0.1\times$ learning-rate drops after epochs 29 and 41. The ViT uses AdamW, a three-epoch linear warmup from 10^{-3} times the base learning rate, and cosine decay over the remaining 45 epochs. The CNN output weights have standard deviation 10^{-3} and zero bias. The ViT output weights use standard deviation $10^{-3}\sqrt{1024/24576}$ with head seed 43; its bias is 0.5 for Direct and $\phi(0)$ for Fourier, corresponding to zero corner displacement.

We select checkpoints by lowest validation MACE, evaluated every two epochs. The selected epochs are 48 for CNN Direct, 46 for CNN Fourier, and 48 for both ViT variants. As in 2D pose, the CNN Fourier checkpoint was selected with the earlier refinement implementation and then re-evaluated with hierarchical decoding without reselection. The ViT uses hierarchical decoding throughout. Decoding is in FP32. MACE averages Euclidean corner error over all four corners and then over the fixed validation pairs.

A.5 3D Pose Lifting

We use MotionAGFormer-Base with the released Human3.6M preprocessing based on Stacked Hourglass 2D detections (H36M-SH). Each joint input contains two normalized image coordinates and a confidence score. Clips contain 243 frames and 17 joints; training clips use stride 81 and evaluation clips use stride 243, giving 14,070 training clips and 2,228 evaluation clips. The released subject split and evaluation protocol are retained. The 3D training targets use the preprocessing's normalized image-space coordinates with the root joint subtracted in each frame. Direct regresses these values without an additional range normalization; Fourier uses the bounds in Table 7.

Both models train from scratch for 60 epochs with AdamW, batch size 16, initial learning rate 5×10^{-4}, weight decay 0.01, and a learning-rate multiplier of 0.99 after each epoch. We retain the model's standard linear-layer initialization and use plain elementwise MSE over frames, joints, coordinates, and code channels. Training includes horizontal flipping with probability 0.5. No scale, velocity, or other auxiliary loss is used in either variant.

We report the final epoch rather than selecting a checkpoint by test performance. At evaluation, original and horizontally flipped inputs are both processed. Direct predictions are unflipped and averaged in coordinate space. Fourier predictions are unflipped by swapping left/right joints and negating the sine components for the horizontal coordinate, then averaged in code space before decoding. The root prediction is set to zero. The released evaluator restores the original scale using image dimensions and the provided 2.5D factors, computes root-relative P1 and Procrustes-aligned P2, and averages errors within each action before averaging across actions. We retain its exclusions of sequences `s_09_act_05_subact_02`, `s_09_act_10_subact_02`, and `s_09_act_13_subact_01`. Results are in millimeters. Figure 3(a) uses these same evaluation rules at intermediate epochs, starting at epoch 5 and then showing epochs 10, 20, . . ., 60.

A.6 Optical Flow

We initialize both variants from the released CroCo v2 ViT-Large/Base-decoder self-supervised checkpoint. The encoder has 24 layers, width 1,024, and 16 attention heads; the decoder has 12 layers, width 768, and 12 heads. Patch size is 16. We retain the dense prediction head architecture and change its final output width to two displacement channels for Direct or 32 code channels for Fourier, without an uncertainty output. The two variants share the initial randomly initialized dense head except for its final projection, which uses the framework's default convolution initialization. All model parameters are trained. RGB inputs use ImageNet mean and standard-deviation normalization.

Training consists of 100,000 updates on the FlyingChairs training split, followed by 100,000 on FlyingThings3D. The latter uses both clean and final rendering passes, the left camera, and forward and backward temporal directions. Both stages use 320×384 crops, batch size 8, FP32, and seed 1234. AdamW uses learning rate 2×10^{-5}, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.05 (excluding biases and one-dimensional parameters), and gradient-norm clipping at 1. Each stage uses 500 warmup updates followed by cosine decay with a 100,000-update horizon. The second stage initializes model weights from the first stage's final checkpoint and starts a new optimizer and schedule.

The flow augmentation pipeline applies color jitter, random erasing in the second image, spatial scaling and stretching, random cropping, and horizontal/vertical flips. Brightness, contrast, and saturation jitter have strength 0.4, and hue jitter has strength $0.5/3.14$; the two frames receive independent color transforms with probability 0.2. Erasing is applied with probability 0.5, using one or two rectangles with side lengths sampled from 50 to 99 pixels. The log-base-2 scale intervals are $[-0.1, 1]$ on Chairs and $[-0.4, 0.8]$ on Things. Spatial scaling is applied with probability 0.8, with additional independent log-base-2 axis stretches in $[-0.2, 0.2]$ sampled with probability 0.8. Horizontal and vertical flip probabilities are 0.5 and 0.1. Training supervision masks nonfinite flow and pixels with ground-truth flow magnitude at least 400 pixels. Direct uses the mean absolute error of the two unnormalized displacement components. Fourier uses the per-coordinate cosine loss defined in Section B.3, with no auxiliary coordinate loss or sequence loss.

Main-table evaluation uses the final Things checkpoint, 512 fixed test pairs per rendering pass on Things, and all 1,041 training pairs per pass on Sintel. The Things subset uses rounded, evenly spaced indices over the 7,866-pair left-camera list, including both temporal directions; the same list is used for both models. Sintel is used only for evaluation. Sliding-window inference uses 320×384 tiles, overlap 0.9, and uniform averaging of decoded displacements in overlapping regions. Evaluation retains finite ground-truth flow without the training-time magnitude cutoff or an additional out-of-frame mask. EPE averages Euclidean displacement error over valid pixels. The curve in Figure 3(b) instead evaluates the FlyingChairs validation split during the first training stage, at 1,000 and 5,000 updates and every 10,000 updates thereafter. Chairs validation uses full-image forward passes without spatial augmentation or tiling, retaining the data loader's finite-flow and magnitude-below-400 mask. These are recorded evaluations without smoothing.

A.7 VLM-based Localization

We use the Talking Points Point Localizer with the released OMG-LLaVA initialization and InternLM2-7B language backbone. The input combines the image with the point description in the prompt `Please segment region1: {description}`, followed by the response template `<p> keypoint </p> [SEG]`. The final hidden state of `[SEG]` passes through the text-to-vision projector ($4096 \to 512 \to 512$) and a coordinate head ($512 \to 128$, ReLU, output layer). The final layer has two sigmoid outputs for Direct and 16 linear outputs for Fourier.

The language backbone uses 4-bit NF4 quantization with FP16 computation and LoRA rank 512, scaling parameter 256, and dropout 0.05. We train the decoder LoRA parameters, text-to-vision projector, and coordinate head, keeping the visual encoder and vision-to-text projector frozen. The coordinate head and trainable adapter parameters are stored in FP32. Both models share the initialization of the pretrained components and the hidden coordinate-head layer. For Fourier, the final layer's initial weights are multiplied by 0.25 and its bias is replaced by the mean target code over the training set; this scaling is applied only at initialization. Direct retains its standard sigmoid-head initialization.

From the official 17,265-example training set, we reserve 600 examples for internal validation and remove one additional image duplicate, leaving 16,664 training examples. The official 4,317-example test set is retained. Both variants use the same training order, head and data seeds 0, batch size 8, and 32,385 updates. This matches the update count associated with the original 15-epoch budget, rather than 15 exact traversals of our reduced training split. AdamW uses a constant learning rate of 10^{-5}, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and weight decay 0.01, without warmup or gradient clipping. Optimizer state persists across data traversals and resumed training segments. Internal validation is recorded every 500 updates through update 6,000 and every 2,000 updates thereafter; the test table uses the final checkpoint.

Images are padded to a square with the image processor's mean RGB color, resized bicubically to 1024×1024, and normalized using mean $(0.4814, 0.4578, 0.4082)$ and standard deviation $(0.2686, 0.2613, 0.2757)$. For an original image of width W and height H, let $L = \max(W, H)$ and (s_x, s_y) be the integer left/top padding offsets. The training target is $((x + s_x)/L, (y + s_y)/L)$. At evaluation, we invert this transformation, clamp horizontal and vertical predictions to $[0, W]$ and $[0, H]$, respectively, and retain floating-point coordinates. This corrects the swapped width/height clipping bounds and integer truncation in the released evaluation script; the same restoration is used for both variants. We compute normalized Euclidean error as

$$d = \sqrt{\left(\frac{\hat{x} - x}{W}\right)^2 + \left(\frac{\hat{y} - y}{H}\right)^2}. \tag{8}$$

PCK@a is the fraction of examples with $d \leq a$, mPCK averages PCK@0.1 and PCK@0.2, and normalized L2 is the mean of d. In Figure 4, the update ratios compare Fourier's validation accuracy at update 6,000 with the first recorded Direct evaluation reaching that level; no interpolation is used.

B Representation, Supervision, and Decoding Details

B.1 Alternative Representations

The representation comparison in Table 1 uses Direct and three eight-channel encodings per scalar coordinate $c \in [0, 1]$. Polynomial encoding is $(c^j)_{j=0}^{7}$, including the constant channel $c^0 = 1$. Multiscale sawtooth encoding is $(\mathrm{frac}(2^{j-1}c))_{j=0}^{7}$, where $\mathrm{frac}(x) = x - \lfloor x \rfloor$. Its channels have periods 2^{1-j} and linear ramps separated by discontinuities. It uses continuous-valued targets rather than quantized binary labels. Fourier uses the four frequencies $2.5, 5, 10, 20$. Each encoding is trained with MSE averaged over all output components; we do not normalize the predicted code during training.

For the three encodings, the table uses the same decoding criterion,

$$\hat{c} = \operatorname*{argmin}_{c \in \mathcal{G}} \|\hat{\mathbf{z}} - \phi(c)\|_2^2, \qquad \mathcal{G} = \{j/4000 : j = 0, \ldots, 4000\}, \tag{9}$$

without code normalization. Polynomial and sawtooth checkpoints are selected using this grid readout. The Fourier row reuses the checkpoint selected by hierarchical-decoding validation MAE and evaluates it with the grid readout, rather than selecting another checkpoint. This accounts for Fourier's different values in Table 1 (raw grid, 0.0373) and Table 2 (hierarchical, 0.0368). Direct predicts the scalar itself and is clamped only for coordinate evaluation.

B.2 Supervision Paths and Gradients

All four paths in Table 2 use the same patch localization protocol. For one scalar, Direct minimizes $(\hat{c} - c)^2$, while Fourier minimizes the component-averaged distance between $\hat{\mathbf{z}}$ and $\phi(c)$. Encode(Direct) outputs an unconstrained scalar $\hat{c}$ and minimizes

$$\ell_{\mathrm{enc}} = \frac{1}{2K} \|\phi(\hat{c}) - \phi(c)\|_2^2. \tag{10}$$

Its prediction is not clamped before encoding during training. Decode(Fourier) outputs a free $2K$-dimensional code and minimizes

$$\ell_{\mathrm{dec}} = \big(\mathrm{Dec}_{\mathrm{train}}(\hat{\mathbf{z}}) - c\big)^2, \tag{11}$$

where $\mathrm{Dec}_{\mathrm{train}}$ applies hierarchical decoding with $\tau = 0.5$ but omits the final coordinate-range clamp. The loss is averaged over samples and coordinate components.

Decode(Fourier) uses ordinary automatic differentiation through `atan2`, phase updates, and the phase clamp. The rounded branch index has zero derivative almost everywhere; no straight-through estimator or soft branch selection is used. A saturated phase correction blocks the gradient through that correction, although the preceding estimate still contributes through the phase update. Both training alternatives use the same final $[0, 1]$ clamp for evaluation as Direct and Fourier. Thus, equal use of mean MSE does not imply identical gradient scales or optimization landscapes across these objectives.

B.3 Cosine Supervision for Optical Flow

Let $\mathcal{V}$ denote the valid training pixels and $d \in \{x, y\}$ a displacement component. The optical-flow Fourier loss is

$$\mathcal{L}_{\cos} = \frac{1}{2|\mathcal{V}|} \sum_{p \in \mathcal{V}} \sum_{d \in \{x,y\}} \left(1 - \frac{\hat{\mathbf{z}}_{p,d}^{\mathsf{T}} \phi(c_{p,d})}{\max(\|\hat{\mathbf{z}}_{p,d}\|_2, \epsilon) \max(\|\phi(c_{p,d})\|_2, \epsilon)} \right), \qquad \epsilon = 10^{-8}. \tag{12}$$

Each norm spans all $2K$ channels of one scalar coordinate. We do not normalize each sine–cosine pair separately for this training loss, and neither spatial positions nor the two displacement components are combined into a single cosine vector. Direct instead uses component-averaged L1 on the same valid pixels. Invalid targets are masked before numerical operations.

B.4 Decoding Comparisons

Fixed predictions and normalization. Table 3 and Figure 2(b) reuse the same seed-43 Fourier model trained on 50,000 images. Its checkpoint is selected at update 2,320 by validation MAE using hierarchical decoding with $\tau = 0.5$. All decoding methods receive the same cached FP32 predictions; no network is retrained or reselected for a different decoding method. The phase-clamp candidates are $0, 0.1, 0.2, 0.5, 1.0$, and unrestricted updates, with the setting selected using validation MAE. This decoder comparison is therefore conditional on a checkpoint originally selected with $\tau = 0.5$.

Grid search and the SACReg solver in the decoding table first normalize each predicted pair:

$$\bar{\mathbf{z}}_k = \frac{(\hat{z}_{2k}, \hat{z}_{2k+1})^{\mathsf{T}}}{\max\left(\sqrt{\hat{z}_{2k}^2 + \hat{z}_{2k+1}^2}, 10^{-12} \right)}. \tag{13}$$

An exactly zero pair remains zero. This differs from the raw-code objective used in the representation table: pair normalization removes amplitude weighting from nonzero pairs above the numerical threshold. Hierarchical decoding depends on pair phase and does not require this normalization. Its inference implementation uses the framework's `atan2` convention, yielding zero phase for a zero pair, and clamps the final coordinate to its support. At $\tau = 0$, only the lowest-frequency estimate remains; unrestricted corrections are equivalent to $\tau = \pi$ after nearest-branch selection.

Grid search. We use 4,001 uniformly spaced candidates including the endpoints of $[0, 1]$. For Fourier codes, $\|\phi(c)\|_2^2 = K$ is constant, so minimizing squared code distance is equivalent to maximizing $\bar{\mathbf{z}}^{\mathsf{T}} \phi(c)$. We implement this correlation by matrix multiplication and process scalar coordinates in chunks, avoiding a tensor of all candidate-wise channel errors.

SACReg solver. Our implementation follows the approximate inverse in SACReg (Revaud et al., 2024), with channels reordered to our sine–cosine convention. For $E(c) = \|\bar{\mathbf{z}} - \phi(c)\|_2^2$, we evaluate

$$g(c) = \tfrac{1}{2} E'(c) = \sum_{k=0}^{K-1} \omega_k \left[\bar{z}_{2k+1} \sin(\omega_k c) - \bar{z}_{2k} \cos(\omega_k c) \right] \tag{14}$$

on a zero-anchored grid with spacing $\pi/(2\omega_{\max})$, restricted to the coordinate interval and augmented with its endpoints. For adjacent samples a, b with $g(a) < 0 < g(b)$, we form one linearly

interpolated candidate,

$$c^* = a - (b - a)\frac{g(a)}{g(b) - g(a)}. \tag{15}$$

We also include interval endpoints and sampled points with exactly zero derivative, then select the candidate with lowest squared code distance. These additions cover boundaries and degenerate inputs. The solver uses a single interpolation, not iterative exact root finding, and does not guarantee the global optimum.

Runtime protocol. The runtime column of Table 3 is a separate decoding-only benchmark on an NVIDIA H200. Each call decodes 320×384 synthetic two-dimensional points ($245{,}760$ scalar coordinates) from contiguous GPU-resident standard-normal codes, using $K = 4$, $\omega_{\min} = 2.5$, $b = 2$, and support $[0, 1]$. We use PyTorch 2.10.0 with CUDA 12.8, FP32, disabled TF32, and eager inference mode. Grid search and SACReg process chunks of 4,096 scalars. Each method receives at least 20 warmup calls, and each group is warmed up for at least one second. We collect 35 trials per method in five blocks of seven rounds, randomizing method order within each round and synchronizing the GPU before and after wall-clock timing. The reported value is the median.

Timing includes pair normalization for the search methods, construction of search bases, decoding both components, output allocation, and concatenation. Hierarchical decoding uses $\tau = 0.5$ without redundant pair normalization. Input generation, host-to-device transfer, network prediction, and sliding-window fusion are excluded. Consequently, the reported speed ratios describe the tested decoding implementations, not end-to-end optical-flow inference.

C MORE RESULTS

In this section, we provide additional qualitative results in Fig. 5.



Figure 5: Selected qualitative examples of improved localization with Fourier Output on the VLM task. Circles denote ground-truth points, crosses denote direct predictions, and triangles denote Fourier predictions. The text below each image is the original input description.