LOOPE: Learnable Optimal Patch Order for Positional Encoders in Vision Transformers (Supplementary Material)

Supplementary Material

Algorithm 1 `gilbert2d(width, height)`

function GILBERT2D($width,\ height$)
  if $width \geq height$ **then**
    return GENERATE2D($0,\ 0,\ width,\ 0,\ 0,\ height$)
  else
    return GENERATE2D($0,\ 0,\ 0,\ height,\ width,\ 0$)

A. Methodology

A.1. Gilbert Curve Generation (Generalized Hilbert Curve)

The following pseudocode represents a recursive algorithm for generating a generalized Hilbert (Gilbert) space-filling curve for arbitrary 2D rectangular grids. The algorithm outputs discrete 2D coordinates that fill a rectangle of given width and height.

A.1.1. Main Procedure: `gilbert2d`

A.1.2. Helper Function: `sgn`

Algorithm 2 `sgn(x)`

function SGN(x)
  if $x < 0$ **then**
    return -1
  else if $x > 0$ **then**
    return 1
  else
    return 0

A.1.3. Recursive Procedure: `generate2d`

Notes:

- The function `gilbert2d` selects the major direction based on the aspect ratio.
- The helper function `sgn` returns the sign of its input.
- The `generate2d` procedure is recursive. It subdivides the rectangle until reaching trivial row or column fills, outputting coordinate points at each base case.
- In this pseudocode, "**Output** (x, y)" corresponds to yielding the coordinate in the Python implementation.

A.2. Context-Aware Index Generator Architecture

The architecture of the context-aware bias generator X_C is summarized in Table 1. The module takes as input the RGB image ($I_0 \in \mathbf{R}^{3\times H\times W}$) concatenated with coordinate maps ($x, y \in \mathbf{R}^{1\times H\times W}$), forming a 5-channel tensor. Through a series of convolutional layers followed by batch normalization, flattening, and an MLP, the network outputs continuous

Algorithm 3 `generate2d(x, y, ax, ay, bx, by)`

procedure GENERATE2D(x, y, ax, ay, bx, by)
  $w \leftarrow |ax + ay|$ ▷ Effective width
  $h \leftarrow |bx + by|$ ▷ Effective height
  $(dax,\ day) \leftarrow (\text{sgn}(ax),\ \text{sgn}(ay))$ ▷ Unit vector in major direction
  $(dbx,\ dby) \leftarrow (\text{sgn}(bx),\ \text{sgn}(by))$ ▷ Unit vector in orthogonal direction
  if $h = 1$ **then**
    for $i \leftarrow 0$ **to** $w - 1$ **do**
      Output (x, y)
      $(x, y) \leftarrow (x + dax,\ y + day)$
    return
  if $w = 1$ **then**
    for $i \leftarrow 0$ **to** $h - 1$ **do**
      Output (x, y)
      $(x, y) \leftarrow (x + dbx,\ y + dby)$
    return
  $(ax2,\ ay2) \leftarrow (\lfloor ax/2 \rfloor,\ \lfloor ay/2 \rfloor)$
  $(bx2,\ by2) \leftarrow (\lfloor bx/2 \rfloor,\ \lfloor by/2 \rfloor)$
  $w2 \leftarrow |ax2 + ay2|$
  $h2 \leftarrow |bx2 + by2|$
  if $2w > 3h$ **then** ▷ Long case: split into two parts
    if $(w2 \bmod 2 = 1)$ **and** $(w > 2)$ **then**
      $(ax2,\ ay2) \leftarrow (ax2 + dax,\ ay2 + day)$ ▷ Prefer even steps
    GENERATE2D($x,\ y,\ ax2,\ ay2,\ bx,\ by$)
    GENERATE2D($x + ax2,\ y + ay2,\ ax - ax2,\ ay - ay2,\ bx,\ by$)
  else
    if $(h2 \bmod 2 = 1)$ **and** $(h > 2)$ **then**
      $(bx2,\ by2) \leftarrow (bx2 + dbx,\ by2 + dby)$ ▷ Prefer even steps
    GENERATE2D($x,\ y,\ bx2,\ by2,\ ax2,\ ay2$)
    GENERATE2D($x + bx2,\ y + by2,\ ax,\ ay,\ bx - bx2,\ by - by2$)
    GENERATE2D($x + (ax - dax) + (bx2 - dbx),\ y + (ay - day) + (by2 - dby),\ -bx2,\ -by2,\ -(ax - ax2),\ -(ay - ay2)$)

Table 1. Layer-wise specification of the context-aware index generator for X_C. Conv_k denotes a convolution with kernel size $k \times k$. All convolutions use ReLU activations unless stated otherwise.

Layer	Input Shape	Operation	Output Shape	Notes
Input	$[5 \times H \times W]$	—	$[5 \times H \times W]$	RGB + coordinate maps
Conv1	$[5 \times H \times W]$	32 filters, kernel $P \times P$, stride 16	$[32 \times h \times w]$	Downsampling to patch scale
Conv2	$[32 \times h \times w]$	16 filters, kernel 5×5, stride 1	$[16 \times h \times w]$	Local refinement
Conv3	$[16 \times h \times w]$	8 filters, kernel 5×5, stride 1	$[8 \times h \times w]$	—
Conv4	$[8 \times h \times w]$	4 filters, kernel 5×5, stride 1	$[4 \times h \times w]$	—
Conv5	$[4 \times h \times w]$	1 filter, kernel 5×5, stride 1	$[1 \times h \times w]$	Channel squeeze
BN + Flatten	$[1 \times h \times w]$	BatchNorm + reshape	$[1 \times N]$	$N = h \times w$ patches
MLP	$[1 \times N]$	Linear layer + bias	$[1 \times N]$	Fully connected refinement
Activation	$[1 \times N]$	$2\sigma(x) - 1$	$X_C \in [-1, 1]^{1 \times N}$	Fractional offsets

offsets $X_C \in \mathbf{R}^{1 \times N}$, which refine the static patch order X_G.

The stride of 16 in Conv1 ensures that each feature corresponds to a patch region of the image, aligning X_C directly with the patch grid used by the transformer. Subsequent layers refine this representation without further spatial downsampling, keeping the module lightweight and efficient.

A.3. Theoretical Effect of Context-Aware Adaptation on Frequency Sensitivity

To analyze why the context-aware refinement X_C reduces sensitivity to the choice of frequency set, consider a simplified form of the positional encoding:

$$\mathrm{PE}(f) = \sin\left(\frac{(x+\delta)}{N} f\right), \tag{1}$$

where x is the static patch index, δ is the adaptive offset produced by the context-aware generator, N is the normalization factor (e.g., $N = 196$ for a 14×14 patch grid), and f is the frequency. The offset δ itself depends on both the frequency and local image context:

$$\delta = g(f, \text{context}).$$

Frequency sensitivity without adaptation. When $\delta = 0$, the derivative of PE with respect to f is

$$\frac{\partial}{\partial f}\mathrm{PE}(f) = \frac{x}{N}\cos\left(\frac{x}{N} f\right). \tag{2}$$

Thus, sensitivity is strictly proportional to the normalized index x/N. In this case, different choices of frequency sets directly translate into large oscillations in the embedding, making performance highly dependent on design choices.

Frequency sensitivity with adaptation. With context-aware refinement ($\delta \neq 0$), we compute

$$\begin{aligned}\frac{\partial}{\partial f}\mathrm{PE}(f) &= \frac{\partial}{\partial f}\sin\left(\frac{(x+\delta)}{N} f\right) \\ &= \cos\left(\frac{(x+\delta)}{N} f\right) \cdot \frac{\partial}{\partial f}\left(\frac{(x+\delta)}{N} f\right).\end{aligned} \tag{3}$$

Expanding the inner derivative:

$$\frac{\partial}{\partial f}\left(\frac{(x+\delta)}{N} f\right) = \frac{x+\delta}{N} + \frac{f}{N}\frac{\partial \delta}{\partial f}. \tag{4}$$

Substituting back into Eq. 3 gives the simplified expression:

$$\frac{\partial}{\partial f}\mathrm{PE}(f) = \left(\frac{x+\delta}{N} + \frac{f}{N}\frac{\partial \delta}{\partial f}\right)\cos\left(\frac{(x+\delta)}{N} f\right). \tag{5}$$

Interpretation. Equation 5 reveals two contributions to frequency sensitivity:

- $\frac{x+\delta}{N}$: a shifted static term, analogous to the non-adaptive case but now adjusted by the learned offset δ.
- $\frac{f}{N}\frac{\partial \delta}{\partial f}$: an adaptive correction term, which enables the model to compensate for changes in the frequency set.

Without adaptation, only the first term exists, making sensitivity rigidly determined by x/N. With adaptation, the second term actively counterbalances frequency variations, thereby reducing sensitivity while preserving the sinusoidal form of the encoding.

Conclusion. Context-aware adaptation therefore moderates the dependency on frequency selection: the encoding remains sinusoidal in structure, ensuring consistency, but the added corrective term $\frac{\partial \delta}{\partial f}$ provides robustness. This theoretical view aligns with the empirical results in Table. 7, where LOOPE maintains stable accuracy even under non-optimal frequency choices.

A.4. Three Cell experiment

Overview

This dataset is designed to facilitate research in spatial reasoning, geometric transformations, and pattern recognition using synthetic images. It consists of structured grid-based images where three distinct color markers (red, green, and blue) are positioned according to predefined spatial constraints.

Purpose & Applications

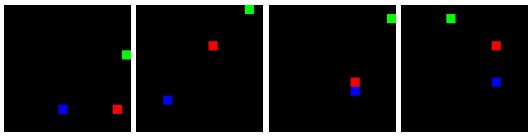
The dataset supports tasks such as:

- **Machine Learning & Deep Learning:** Training models to understand spatial relationships.
- **Computer Vision:** Evaluating geometric transformations and positional reasoning.
- **Representation Learning:** Analyzing how models interpret structured spatial layouts.

Dataset Statistics and Description

- The dataset consists of 10,000 synthetic images.
- The cell coordinates are uniformly sampled on a 14×14 grid.
- The relative cases are designed to be equiprobable, ensuring balanced distribution.
- Among the 196 patches, 193 are colored black; the three colored patches are non-colinear and non-overlapping, however they can share boundaries.

Three-Cell Experiment Dataset Visualization



(a) (b) (c) (d)

Figure 1. Three-Cell Experiment Sample Dataset

Structure & Content Each sample includes:

1. **Image:** A 224×224 pixel RGB image that represents a 14×14 grid. In this grid:
   - A red cell serves as the reference point.
   - A green cell and a blue cell are positioned based on mathematical constraints.
   - The background is black, with each cell having a distinct color.
2. **Binary Mask:** A 6-element vector encoding specific spatial properties, such as:
   - Distance comparisons.
   - Geometric orientation.
   - Area relationships.

Key Characteristics

- **Controlled Complexity:** Images are generated based on strict mathematical rules.
- **Diversity:** Multiple spatial configurations ensure a wide range of positional relationships.
- **Explainability:** The structured nature of the dataset makes it ideal for interpretable AI research.

Potential Use Cases

- Training AI models for spatial awareness and geometric reasoning.
- Benchmarking representation learning techniques in structured visual tasks.
- Investigating how neural networks learn spatial relationships in a controlled setting.

Algorithm 4 Synthetic Image Dataset Generation

procedure GENERATESYNTHETICDATASET
  Input: Grid size $G \leftarrow 14$, Sum-of-squares limit $L \leftarrow 338$, $max\_attempts$
  Output: Expanded grid image $grid_{224}$ and binary mask $mask[6]$
  $max\_val \leftarrow \lfloor \sqrt{L} \rfloor$
  Initialize dictionary $D \leftarrow \emptyset$
  for $a = -max\_val$ **to** max_val **do**
    for $b = -max\_val$ **to** max_val **do**
      $n \leftarrow a^2 + b^2$
      if $n \leq L$ **then**
        Append pair (a, b) to $D[n]$
  Remove keys from D that have only one representation
  Randomly choose mode from $\{1, 2, 3\}$
  $valid \leftarrow$ **false**
  while not $valid$ **do**
    Randomly select red cell $X_r \in [0, G-1]^2$
    if mode $= 1$ **then**
      Select random key k from keys of D
      for $attempt = 1$ to $max\_attempts$ **do**
        Randomly choose two distinct pairs (a_1, b_1) and (a_2, b_2) from $D[k]$
        $X_g \leftarrow X_r + (a_1, b_1)$
        $X_b \leftarrow X_r + (a_2, b_2)$
        if X_g and X_b are within the grid and distinct from X_r and each other **then**
          $valid \leftarrow$ **true**
          break
    else if mode $\in \{2, 3\}$ **then**
      for $attempt = 1$ to $max\_attempts$ **do**
        Randomly select two keys k_1, k_2 from D
        if mode $= 2$ **then**
          $k_g \leftarrow \min(k_1, k_2), k_b \leftarrow \max(k_1, k_2)$
        else
          $k_g \leftarrow \max(k_1, k_2), k_b \leftarrow \min(k_1, k_2)$
        Randomly select pair (a_g, b_g) from $D[k_g]$ and (a_b, b_b) from $D[k_b]$
        $X_g \leftarrow X_r + (a_g, b_g)$
        $X_b \leftarrow X_r + (a_b, b_b)$
        if X_g and X_b are within the grid and distinct **then**
          $valid \leftarrow$ **true**
          break
  Create a $G \times G$ grid $grid_{14}$ with black background
  Set $grid_{14}[X_r] \leftarrow$ red, $grid_{14}[X_g] \leftarrow$ green, $grid_{14}[X_b] \leftarrow$ blue
  Expand $grid_{14}$ to $grid_{224}$ by replicating each cell into a 16×16 block
  Initialize mask $mask[6] \leftarrow (0, 0, 0, 0, 0, 0)$
  Call: DISTANCECOMPARE, ORIENTATION, AREACOMPARE, VECTORSUM on $(X_r, X_g, X_b, mask)$
  return $grid_{224}$ and $mask$

A.5. Positional Embeddings Structural Integrity (PESI) Metrics

A.5.1. Undirected Monotonicity

Algorithm 5 Compute Global Undirected Monotonicity Score M_u

Input: Positional embedding tensor $P \in \mathbb{R}^{H\times W\times D}$
Output: Global undirected monotonicity score M_u

Initialize an empty list $S \leftarrow [\,]$
for $x = 1, 2, \ldots, H$ **do**
  for $y = 1, 2, \ldots, W$ **do**
    $v_{\text{center}} \leftarrow P(x, y, :)$
    Compute norm: $\|v_{\text{center}}\| \leftarrow \text{norm}(v_{\text{center}}) + \epsilon$
    Compute cosine similarity for all (i, j): $A_{(x,y)}(i, j) \leftarrow \frac{v_{\text{center}} \cdot P(i,j,:)}{\|v_{\text{center}}\| \cdot \|P(i,j,:)\| + \epsilon}$
    Compute Euclidean distances: $d(i, j) \leftarrow \sqrt{(i-x)^2 + (j-y)^2}$
    Form radial bins: $r(i, j) \leftarrow \text{round}\big(d(i, j)\big)$
    for each unique radial bin r **do**
      $S_{(x,y)}(r) \leftarrow$ average of $A_{(x,y)}(i, j)$ for all (i, j) with $r(i, j) = r$
    Compute Spearman's rank correlation $\rho_{(x,y)}$ between $\{r\}$ and $\{S_{(x,y)}(r)\}$
    Append $\rho_{(x,y)}$ to S
Compute global score: $M_u \leftarrow \frac{1}{HW} \sum_{(x,y)} \big(1 - \rho_{(x,y)}\big)$
return M_u

A.5.2. Directed Monotonicity

Algorithm 6 Compute Global Directed Monotonicity Measure M_D

Input: Positional embedding tensor $P \in \mathbb{R}^{H\times W\times D}$, quantization angle δ
Output: Global directed monotonicity measure M_D

$N \leftarrow \lceil 2\pi/\delta \rceil$ ▷ Total number of directional buckets
Initialize list $S \leftarrow [\,]$
for $x = 1, 2, \ldots, H$ **do**
  for $y = 1, 2, \ldots, W$ **do**
    $v_{\text{center}} \leftarrow P(x, y, :)$
    Cosine similarity: $A_{(x,y)}(i, j) \leftarrow \frac{v_{\text{center}} \cdot P(i,j,:)}{\|v_{\text{center}}\| \, \|P(i,j,:)\| + \epsilon}$
    Radial distances: $d(i, j) \leftarrow \sqrt{(i-x)^2 + (j-y)^2}$
    Angles: $\theta_{(x,y)}(i, j) \leftarrow \text{atan2}(j - y, i - x)$
    Quantize: $k(i, j) \leftarrow \lfloor \theta_{(x,y)}(i, j)/\delta \rfloor \bmod N$
    Initialize list $\rho^k \leftarrow [\,]$
    for $k = 0, 1, \ldots, N - 1$ **do**
      Let $B_k \leftarrow \{(i, j) \mid k(i, j) = k\}$
      if $|B_k| < 2$ **then**
        Append 0 to ρ^k
      else
        Order B_k by increasing $d(i, j)$
        Compute similarity profile $S^k_{(x,y)}(r)$ for ordered cells
        Spearman: $\rho^k_{(x,y)} \leftarrow 1 - \frac{6 \sum_r d_r^2}{|B_k|(|B_k|^2-1)}$
        Append $\rho^k_{(x,y)}$ to ρ^k
    Mean correlation: $\bar{\rho}_{(x,y)} \leftarrow \frac{1}{N} \sum_{k=0}^{N-1} \rho^k$
    Append $1 - \bar{\rho}_{(x,y)}$ to S
Global measure: $M_D \leftarrow \frac{1}{HW} \sum_{(x,y)} \big(1 - \bar{\rho}_{(x,y)}\big)$
return M_D

A.5.3. Undirected Asymmetry

Algorithm 7 Compute Global Undirected Asymmetry A_{SU}

Input: Positional embedding tensor $P \in \mathbb{R}^{H\times W\times D}$
Output: Global undirected asymmetry measure A_{SU}

for each center (x, y) in $\{1, \ldots, H\} \times \{1, \ldots, W\}$ **do**
  Cosine similarity: $A_{(x,y)}(i, j) = \frac{P(x,y,:) \cdot P(i,j,:)}{\|P(x,y,:)\| \, \|P(i,j,:)\|}$
  Euclidean distance: $d(i, j) = \sqrt{(i-x)^2 + (j-y)^2}$
  Radial bins: $B_r = \{(i, j) \mid r = \text{round}(d(i, j))\}$
  for each radial bin $r \in R$ **do**
    Mean similarity: $\mu_{(x,y)}(r) = \frac{1}{|B_r|} \sum_{(i,j)\in B_r} A_{(x,y)}(i, j)$
    Std. deviation: $\sigma_{(x,y)}(r) = \sqrt{\frac{1}{|B_r|} \sum_{(i,j)\in B_r} \big(A_{(x,y)}(i, j) - \mu_{(x,y)}(r)\big)^2}$
    Coeff. of variation: $\text{CV}_{(x,y)}(r) = \sigma_{(x,y)}(r)/\mu_{(x,y)}(r)$
  Undirected asymmetry: $A'_{SU}(x, y) = \frac{1}{|R|} \sum_{r\in R} \text{CV}_{(x,y)}(r)$
Global measure: $A_{SU} = \frac{1}{HW} \sum_{x=1}^{H} \sum_{y=1}^{W} A'_{SU}(x, y)$
return A_{SU}

A.6. Sensitivity Calculation Documentation

We consider four input signals of length $L = 768$:

$$f_1(i) = 0.978^i, \tag{6}$$

$$f_2(i) = 1 - i\frac{1 - 0.0001}{L - 1}, \tag{7}$$

$$f_3(i) = 0.9^i, \tag{8}$$

$$f_4(i) \sim \text{Uniform}(0.0001, 1), \text{sorted descending.} \tag{9}$$

The first signal f_1 is chosen as the baseline reference.

A.6.1. Relative Change of Input Signals

For each signal f_j ($j = 2, 3, 4$), its relative change compared to the baseline f_1 is computed as the normalized root mean squared difference:

$$\Delta f_j = \frac{\|f_j - f_1\|_2}{\|f_1\|_2}, \tag{10}$$

where $\|\cdot\|_2$ denotes the Euclidean norm.

A.6.2. Output Values

Each black-box algorithm produces four output values corresponding to the four input signals. Denote the outputs for algorithm k as

$$O^{(k)} = [O_1^{(k)}, O_2^{(k)}, O_3^{(k)}, O_4^{(k)}], \tag{11}$$

where $O_1^{(k)}$ is the output corresponding to the baseline f_1.

A.6.3. Sensitivity per Input

For each algorithm k and each signal $j = 2, 3, 4$, the sensitivity is defined as the ratio of the absolute output change to the relative input change:

$$S_j^{(k)} = \frac{|O_j^{(k)} - O_1^{(k)}|}{\Delta f_j}. \tag{12}$$

A.6.4. Normalized Average Sensitivity

Since the relative changes Δf_j may differ significantly across signals, we compute a normalized weighted average sensitivity for each algorithm:

$$\bar{S}^{(k)} = \frac{\sum_{j=2}^{4} S_j^{(k)} \Delta f_j}{\sum_{j=2}^{4} \Delta f_j}. \tag{13}$$

This measure reflects the overall sensitivity of the algorithm to input variations, properly weighted by the magnitude of the input changes.

B. Additional Experiments and Visualization

B.1. Training from Scratch on ImageNet-100

A natural question is whether LOOPE's gains are tied to pretrained weight initialization or reflect a genuine structural advantage of the positional encoding. To answer this, we train ViT-Base and Cross-ViT entirely from scratch on ImageNet-100 using an identical training protocol (learning rate schedule, augmentation, optimizer) across all positional encoding baselines. No pretrained weights are used for any method. Table 2 reports the best top-1 accuracy achieved and the epoch at which it occurred. The results confirm that the performance ranking observed in the main paper (Table 1) is preserved under from-scratch training: LOOPE consistently achieves the highest accuracy on both architectures while converging within a comparable number of epochs to the other methods. This demonstrates that the benefit of LOOPE stems from its encoding structure rather than any interaction with pretrained representations.

Model	No PE	Learnable	1D Sin	Static	LOOPE
ViT-Base	87.1%	87.9%	88.8%	89.0%	**89.2%**
(Best Epochs)	258	262	280	265	267
Cross-ViT	86.6%	87.7%	87.9%	88.8%	**89.4%**
(Best Epochs)	268	271	293	275	277

Table 2. From-scratch training on ImageNet-100 shows a similar trend as Table 1 in the main paper. LOOPE achieves the highest accuracy on both ViT-Base and Cross-ViT.

B.2. 3-Cell Input Ablation: RGB vs. Coordinate Channels

The context-aware index generator G receives both patch coordinates (x, y) and RGB pixel values as input (Sec. 3.2 in the main paper). A key design question is whether the learned refinement X_C relies on semantic (appearance) cues from the RGB channels or primarily on geometric (positional) information from the coordinate maps. To disentangle these contributions, we evaluate LOOPE on the 3-Cell benchmark under three input configurations: RGB channels only, coordinate maps only, and the full RGB+Coordinate input. Table 3 reports the sub-task accuracies. The coordinate-only variant captures the majority of the gain over the static baseline across all four sub-tasks, while the RGB-only variant provides only a marginal improvement over the static ordering. The full RGB+Coordinate configuration yields the best results, indicating that the two signal types are complementary but that coordinate geometry is the dominant driver. This confirms that LOOPE's context-aware refinement is primarily geometry-driven, which is consistent with the theoretical motivation in Sec. 3.

PE	Dist.	Orient.	Area	Vec-Sum
Static (X_G)	88.8%	95.0%	89.9%	93.8%
LOOPE (RGB only)	89.9%	95.0%	91.6%	92.4%
LOOPE (Coordinates only)	91.9%	95.5%	92.8%	94.4%
LOOPE (RGB+Coord.)	**93.4%**	**95.8%**	**93.3%**	**94.6%**

Table 3. 3-Cell ablation: performance is largely explained by coordinate (geometry) input, not RGB semantics. Coordinate-only captures most of the gain over the Static baseline.

B.3. PESI Metrics and 3-Cell Task Alignment

The PESI metrics (M_U, M_D, A_{SU}) are intended as diagnostic probes of the similarity geometry induced by different positional encodings. To validate that these metrics capture meaningful structural properties, Table 4 presents the PESI scores alongside the corresponding 3-Cell sub-task accuracies for each PE. The expected alignment is clearly visible: PEs with higher undirected monotonicity M_U yield stronger distance reasoning accuracy, while higher directed monotonicity M_D corresponds to stronger orientation accuracy. Notably, Learnable PE has the lowest scores on both M_U and M_D, and correspondingly achieves the weakest distance and orientation accuracy in the 3-Cell benchmark. At the other end, LFF achieves the highest M_D and orientation accuracy, while LOOPE achieves the highest M_U and distance accuracy. The asymmetry metric A_{SU} captures a different axis: the degree of directional bias in the similarity field. These results confirm that PESI metrics serve as compact, interpretable proxies for the geometric behaviors that matter in downstream positional reasoning.

PE	M_U	Dist.	M_D	Orient.	A_{SU}
Learnable	1.7493	85.6%	1.2003	89.3%	−0.7272
1D Sin	1.9567	90.9%	1.4905	94.9%	0.1243
LFF	1.9623	93.1%	**1.5230**	**96.7%**	0.2683
Static (X_G)	1.9670	88.8%	1.2897	95.0%	0.0945
LOOPE ($X_G + X_c$)	**1.9674**	**93.4%**	1.2900	95.8%	0.0939

Table 4. Combined view of PESI metrics and 3-Cell sub-task accuracies. Underlined: lowest; **Bold**: highest. Higher M_U aligns with stronger distance accuracy; higher M_D aligns with stronger orientation accuracy.

B.4. Trends in PESI metrics

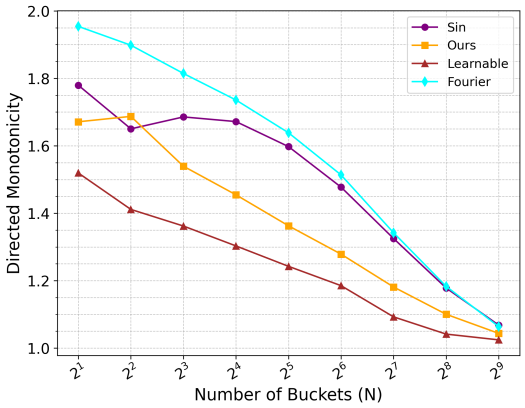


Figure 2. Trend of directed Monotonicity, M_D with increasing angle precision, $\delta = 2\pi/N$

Figure 2, shows the interesting facts about directed monotonicity, with this tool one can investigate how precisely the positional embeddings can maintain monotonicity. Clearly, increasing precision all positional encoders struggles to provide a monotone trend in cosine similarity. As $N \to 1, M_D \to M_U$ which is exactly what we expected. At $N = 4, \delta = \pi/2$, we can see that, LOOPE outperforms zigzag and learnable as it highly depends on hilbert order which propagates in square pattern.

B.5. Visualization

Since, showing exactly how the properties in Table 1, retains in real-life vision tasks is quite hard and the lone contribution of PEs to the performance of ViTs are not huge (generally 4-6%). That's why generating an edge case, where any property may fail/pass, is barely possible. Instead, we can visually verify the implication of PESI metrics in the plotted correlation map of Positional encoding. In figure 3, it has shown the correlation map with different location of patch, on different embedding policy.

From instance, according to Table 6, LFF(fourier) have highest Directed Monotonicity, M_D, as any viewer can see,

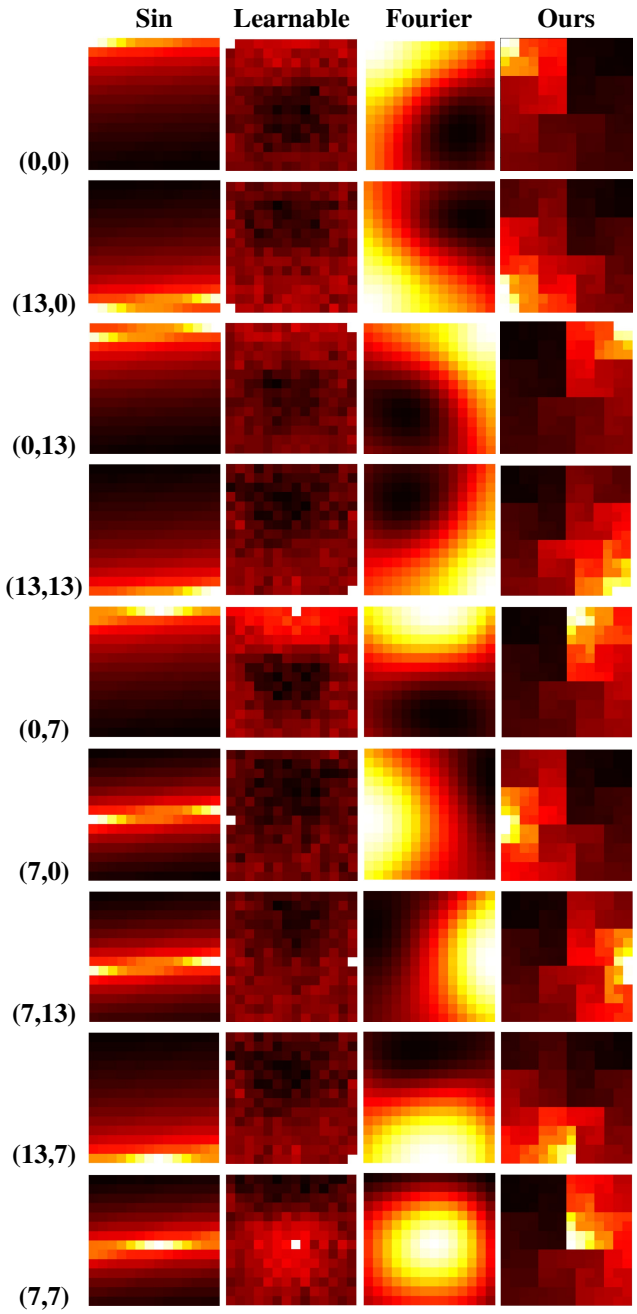


Figure 3. Cosine Similarity Map Comparison among varios PEs on different positions

LFF has the smoothest descending correlation in any particular direction. But, LOOPE has the highest score in Undirected Monotonicity.