DENSE LOCAL OPERATORS ON THE SPHERE

Anonymous authors
Paper under double-blind review

ABSTRACT

Spherical representations are ideal for wide-field-of-view (FoV) perception because they describe local visual geometry independently of the viewing direction. A spherical local operator should respect this geometric consistency by producing consistent responses to the same upright local pattern wherever it is relocated on the sphere—a property we call *translation consistency*. Achieving this property on a 2D grid is challenging, because no global 2D parameterization can preserve the spherical metric everywhere. On an equirectangular projection (ERP) grid, a common representation for spherical data, the same physical neighborhood has a latitude-dependent grid footprint. Prior spherical local operators recover this neighborhood through location-dependent sampling, but the resulting sparse memory access patterns limit GPU utilization. We propose **D**ense **L**ocal Operator on the **S**phere (*DLS*), a spherical local attention mechanism realizing *fully dense*, *approximately translation-consistent* operation on ERP grids. *DLS* combines distance-corrected longitude positional encoding (PE) with linear attention using separate row and column key–value (KV) aggregation. This separation cancels the cross-term involving global longitude, allowing the global PE to encode local spherical geometry, therefore enabling KV-first aggregation without explicitly constructing local KV fields through sparse operations. Using only dense operations, the formulation produces approximately translation-consistent responses to the same upright spherical patch across different locations. On the panoramic segmentation task, the *DLS*-based network achieves a $4.9\times$ speedup while improving mIoU by 2.9 points over the prior sparse-sampling-based network.

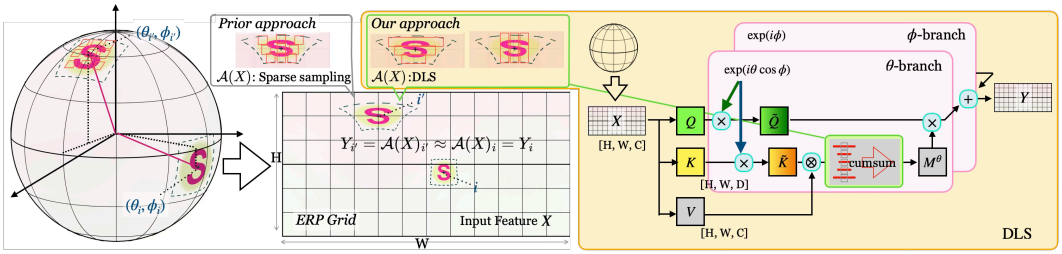


Figure 1: **Overview.** Left: On an equiangular ERP grid, the same upright patch placed at different latitudes spans different extents along rows. To obtain translation-consistent responses, prior approaches use location-dependent sparse sampling. Right: *DLS* combines distance-corrected longitude positional encoding with linear attention using separate row and column key–value (KV) aggregation. This formulation enables approximately translation-consistent responses through fully dense operations without explicit neighbor materialization.

1 INTRODUCTION

Under the standard pinhole camera model, the same object at the same distance can appear substantially different depending on the viewing direction, especially in wide-field-of-view (FoV) images. Regular-grid-based neural modules, such as convolutions and attention, can therefore respond differently to these observations; consistency generally has to be learned from viewpoint variation. Spherical representations provide an ideal basis for omnidirectional perception because they avoid privileging a particular viewing direction: the same camera-facing local pattern has identical local geometry wherever it appears on the viewing sphere (Figure 2). A spherical local operator should respect this geometric consistency: the same upright local pattern elicit a consistent response when relocated to different spherical locations—a property we call *translation consistency*.

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

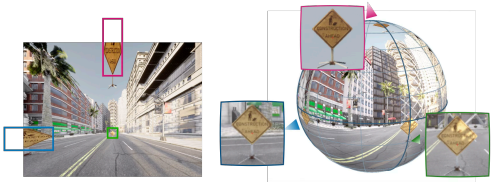


Figure 2: **Pinhole vs Sphere**. Identical signs are placed at equal distances from the camera. The fixed pinhole model (left) projects them differently, whereas the spherical model (right) projects them identically, providing a natural basis for zero-shot view transfer.

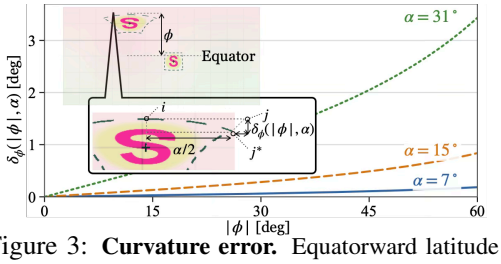


Figure 3: **Curvature error.** Equatorward latitude displacement of the east–west geodesic endpoint relative to the constant-latitude row endpoint. *DLS* is exactly translation consistent when $\delta_\phi(|\phi|, \alpha)$ is zero.

Achieving this property on a 2D grid is challenging, because no global 2D parameterization can preserve the spherical metric everywhere. On an equirectangular projection (ERP) grid, a common representation for spherical data, the same physical patch spans more row samples toward the poles and therefore has a latitude-dependent grid footprint. A fixed rectangular kernel changes its physical receptive field with latitude; maintaining a consistent spherical neighborhood therefore requires special care (Figure 1). Many existing methods recover a fixed physical neighborhood explicitly, using either sparse data representations (Benny and Wolf, 2025; Sekikawa et al., 2026) or sparse, irregular sampling on a regular 2D grid (Ocampo et al., 2023; Griffiths and Dansereau, 2024). Among them, Bonev et al. (2025) introduced an ERP-compatible formulation for local spherical attention. Their formulation sparsely samples geodesic neighborhoods directly on the ERP grid, respecting spherical geometry and achieving strong perception accuracy. However, the sampling relies on sparse gathers and scatters, which introduce irregular memory access and limit GPU utilization.

We ask whether the same geometric goal can be achieved without sparse spatial operations. To this end, we propose *Dense Local Operator on the Sphere* (*DLS*, Figure 1, right). *DLS* combines distance-corrected longitude positional encoding with linear attention using separate row and column key–value (KV) aggregation. The row–column decomposition cancels the dependence on global longitude, encoding relative spherical geometry and enabling KV-first linear attention. This KV-first aggregation avoids explicitly materializing local KV neighborhoods and can instead be evaluated using GPU-friendly dense reductions. The formulation is *approximately translation-consistent* and consists entirely of dense operations. On the Stanford2D3D semantic segmentation task, the *DLS* network improves mIoU by 2.9 points over the prior method while achieving a $4.9\times$ speedup.

2 Related Work

2.1 Distortion-aware operators

Two-dimensional projections, such as equirectangular and perspective projections, make spherical data compatible with mature planar vision architectures. However, a fixed rectangular kernel does not represent the same physical neighborhood at different locations; therefore, several studies seek to reduce projection-induced distortion. Distortion-aware operators adapt planar kernels to panoramic geometry (Tateno et al., 2018). Flat2Sphere and SphereNet transfer filters or features between planar and spherical domains (Su and Grauman, 2017; Coors et al., 2018). BiFuse, UniFuse, and HRDFuse combine ERP and perspective or cubemap features (Wang et al., 2020; Jiang et al., 2021; Ai et al., 2023). PanoFormer, EGFormer, SFSS, and Elite360D incorporate distortion-aware attention or cross-projection fusion (Shen et al., 2022; Yun et al., 2023; Guttikonda and Rambach, 2024; Ai et al., 2024). Trans4PASS (Zhang et al., 2022a;b) uses deformable patch embedding and deformable MLPs. These methods primarily rely on position-dependent kernel construction, sparse input feature sampling, or multiple projections combined with grid remapping to handle distortion, thereby reducing GPU utilization. Our goal is to achieve translation consistency on ERP solely through dense operations.

2.2 Spherical convolutions

Spherical CNNs use harmonic, group, gauge, graph, or polyhedral constructions to respect the geometry of the sphere $\mathbb{S}^2$ (Cohen et al., 2018; Esteves et al., 2018; Cohen et al., 2019; Defferrard et al., 2020; Jiang et al., 2019; Lee et al., 2019). Ocampo et al. (2023) introduced DISCO, a discrete–continuous convolution whose continuous kernel is sampled on a spherical grid. These approaches are particularly suitable when no direction is privileged and full $\mathrm{SO}(3)$ equivariance is desired. Our

setting deliberately assumes known gravity (e.g., via an IMU); therefore, we do not seek equivariance to global rotations. Instead, we ask a local operator to preserve its response when an upright local patch is relocated over the sphere. This target is narrower than full $\mathrm{SO}(3)$ equivariance but allows the model to exploit angles relative to the known privileged axis as a strong cue for perception.

2.3 Attention for spherical signals

Equivariant self-attention has been studied in broader geometric settings, including attention defined on Lie groups and on 3D roto-translation symmetry (Hutchinson et al., 2021; Fuchs et al., 2020). Recent work adapts spherical attention to omnidirectional perception by incorporating spherical geometry. HEAL-SWIN places the signal on the equal-area, hierarchical HEALPix grid and adapts Swin's patch hierarchy and shifted windows, enabling sphere processing without harmonic transforms (Carlsson et al., 2024). SpRePE injects absolute spherical position and pairwise spherical relations into query–key scores through reflection-matrix position encodings (Jia et al., 2026). SphereUFormer adopts an icosphere and applies local self-attention over graph neighborhoods with a learned spherical relative-position bias (Benny and Wolf, 2025). SoLA also uses an icosphere and introduces positional modulation using normals in the corresponding spherical coordinates, which induces local geodesic distance, thereby realizing full $\mathrm{SO}(3)$ equivariance (Sekikawa et al., 2026).

Most closely related to our work, Bonev et al. (2025) introduced S^2-Attention for a parameterized 2D spherical grid. They discretize a continuous attention integral over local geodesic disks on the sphere into an ERP-grid formulation. In conjunction with directional DISCO, their network is translation-consistent. It surpasses previous spherical methods in accuracy and also demonstrates a substantial gain over the non-spherical $\mathbb{R}^2$-variant in the matched setting. However, computing its position-dependent support relies on sparse operations with nonuniform memory access; therefore, it uses custom CUDA kernels to achieve practical latency. To our knowledge, no prior method realizes a *translation-consistent* local operator on an ERP grid using *fully dense* operations. In this work, we aim to achieve both simultaneously.

3 Method

We formalize our geometric and computational targets (Section 3.1) and derive *DLS* as a coupled solution (Section 3.2). We then analyze approximation error and first-order translation consistency (Section 3.3). Finally, we describe resolution transfer (Section 3.4) and the E2E network (Section 3.5).

3.1 Problem formulation: a dense translation-consistent local operator

Let $\mathbf{s}$ be a C-channel local patch expressed in an east–north tangent frame. Let $X := \mathrm{P}(\mathbf{s}, \theta, \phi) \in \mathbb{R}^{H\times W\times C}$ denote the ERP tensor obtained by mapping the patch onto the sphere with its center at (θ, ϕ) and sampling the resulting field on the ERP grid. Indices i denote ERP locations with spherical coordinates (θ_i, ϕ_i). We define *spherical patch translation* as changing the patch center while preserving its local east–north orientation (Figure 1). In general, this is not a uniform ERP-grid shift: the patch's footprint varies with the position-dependent tangent-plane parameterization. We aim to construct a local operator $\mathcal{A}_\alpha$ with physical support diameter α that satisfies the following two requirements for patch centers in the latitude band $|\phi| \le \pi/2 - \eta$, where η is a fixed angular margin.
Geometric target. The local operator $\mathcal{A}_\alpha$ should be approximately *translation-consistent*, i.e.,

$$\mathcal{A}_\alpha(\mathrm{P}(\mathbf{s}, \theta_i, \phi_i))_i \approx \mathcal{A}_\alpha(\mathrm{P}(\mathbf{s}, \theta_{i'}, \phi_{i'}))_{i'} \qquad \forall (i, i') \text{ in latitude band } |\phi| \le \pi/2 - \eta. \tag{1}$$

Computational target. Realize this response using only dense operations, avoiding sparse neighborhood gathers and spherical resampling, which recover location-dependent spherical neighborhoods but are difficult to execute efficiently on GPUs.

Throughout the paper, capital letters denote dense spatial tensors: $X, V, Y \in \mathbb{R}^{H\times W\times C}$, $Q, K \in \mathbb{R}^{H\times W\times D}$, and $Z, M \in \mathbb{R}^{H\times W\times D\times C}$. The corresponding lowercase symbols denote vectors or matrices at ERP location i: $x_i, v_i, y_i \in \mathbb{R}^C$, $q_i, k_i \in \mathbb{R}^D$, and $z_i, m_i \in \mathbb{R}^{D\times C}$. The learned projections, including the QKV projections, are *pointwise* and shared across spherical locations. Within a *DLS* layer, the only spatial aggregation occurs in the key–value fields described in Section 3.2.

3.2 DLS

To meet both the **geometric** and the **computational** requirements, we propose *DLS* (**D**ense **L**ocal Operator on the **S**phere; Figure 4, left). The construction couples distance-corrected longitude positional encoding (PE), row–column factorization, key–value-first linear attention, and latitude-dependent row support. The distance correction in the PE and the factorization remove global-longitude dependence from the induced geometric term. Combined with the KV-first linear attention formulation, they expose row and column key–value fields that can be evaluated through dense reductions without materializing a separate neighborhood for every query. Together, these components yield an approximately translation-consistent local operator composed entirely of dense operations.

Longitudinal distance correction for positional encoding. Consider increments $\Delta\phi$ and $\Delta\theta$ in ERP coordinates for latitude and longitude ϕ, θ. The physical distance corresponding to a longitude increment $\Delta\theta$ decreases with latitude magnitude $|\phi|$. The respective first-order displacements are

$$\text{North–south displacement} : \Delta\phi, \qquad \text{East–west displacement} : \Delta\theta\cos\phi. \tag{2}$$

This relation between ERP coordinates and physical distances is the starting point of our formulation, but it is not sufficient on its own, as we discuss next.

Why naive distance-corrected attention fails. One might consider jointly encoding latitude and the distance-corrected longitude $p_i^\theta = \theta_i \cos\phi_i$ using standard two-dimensional rotary positional embedding (2D-RoPE) (Heo et al., 2024). For clarity, consider only the longitudinal phase and omit its learned frequency. For a two-dimensional rotary pair, the attention score has the form

$$\big(R(p_i^\theta)q_i\big)^\top \big(R(p_j^\theta)k_j\big) = q_i^\top R\big(p_j^\theta - p_i^\theta\big)\, k_j, \tag{3}$$

where $R(\cdot) \in \mathbb{R}^{2\times 2}$ denotes a 2D rotation matrix. Writing $\Delta\theta = \theta_j - \theta_i$, the rotation angle is

$$p_j^\theta - p_i^\theta = \Delta\theta\cos\phi_j + (\cos\phi_j - \cos\phi_i)\theta_i. \tag{4}$$

We expect the response to be determined solely by relative spherical geometry rather than global longitude. In general, however, $(\cos\phi_j - \cos\phi_i)$ is nonzero; thus, the response is not solely a function of the local displacement $\Delta\theta$, but also depends on the global longitude θ_i. Consequently, this 2D-RoPE-based approach using distance-corrected longitude in local attention cannot consistently extract local spherical features. This failure motivates the row–column factorization below.

Row–column factorization removes global-longitude dependence. When query and key tokens x_i, x_j lie in the same row, so that $\phi_i = \phi_j$, the global-position term in Equation (4) vanishes, yielding

$$p_j^\theta - p_i^\theta = \Delta\theta\cos\phi_j. \tag{5}$$

This is precisely the first-order east–west displacement prescribed by Equation (2). We therefore factor attention into two branches: the θ-branch restricts corrected-longitude interactions to the same row, and the ϕ-branch restricts latitude interactions to the same column, as illustrated in Figure 4. Let $p_i^\theta = \theta_i\cos\phi_i$ and $p_i^\phi = \phi_i$, with learned frequencies ω^θ and ω^ϕ for each branch. The two branches share queries and keys but apply branch-specific rotary modulation:[1]

$$\widetilde{q}_i^b = R(\omega^b p_i^b)q_i, \quad \widetilde{k}_j^b = R(\omega^b p_j^b)k_j, \quad b \in \{\theta, \phi\}. \tag{6}$$

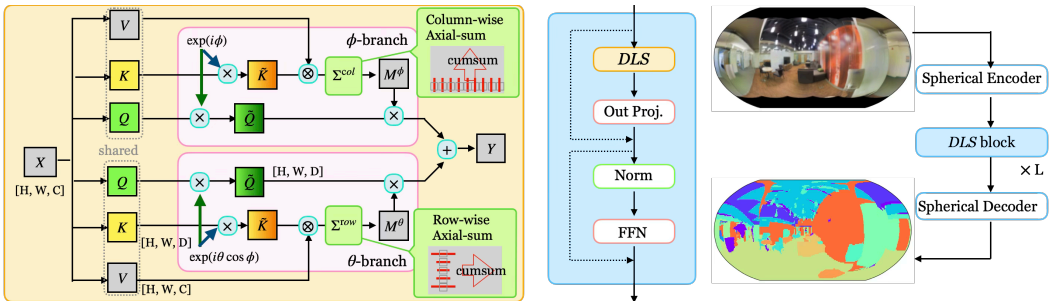
Here, θ and ϕ index the row and column branches, respectively. Their rotary scores are

$$(\widetilde{q}_i^\theta)^\top \widetilde{k}_j^\theta = q_i^\top R\big(\omega^\theta \Delta\theta\cos\phi_j\big)\, k_j, \tag{7}$$

$$(\widetilde{q}_i^\phi)^\top \widetilde{k}_j^\phi = q_i^\top R\big(\omega^\phi \Delta\phi\big)\, k_j. \tag{8}$$

The offsets $\Delta\theta\cos\phi_j$ and $\Delta\phi$ represent first-order geodesic displacements in the east–west and north–south directions, respectively. Thus, the positional dependence of each score is determined by the corresponding local physical displacement. Crucially, the geometry-aware score remains a dot product between separately modulated query and key vectors, $\widetilde{q}, \widetilde{k}$. Linear attention therefore allows us to first sum the key–value outer products over the local support and then contract the resulting moment with the query. This key–value-first form avoids creating query-specific attention maps through explicit neighborhood materialization. It is the key to realizing fully dense computation using prefix sums of contiguous key–value fields which we detail next.

[1] For brevity, the following equations show a single two-dimensional rotation block; the same operation is applied blockwise to the full D-dimensional query and key vectors.



DLS *DLS*-Block *DLS*-Transformer

Figure 4: ***DLS* networks.** *DLS*-Transformer replaces local $\mathcal{S}^2$-Attention modules and DISCO-based learned spatial mixers with *DLS*. *DLS*-SegFormer is constructed analogously.

Key–value aggregation. Translation consistency requires that each branch aggregate over the same physical span at every spherical center; otherwise, the corrected positional score would still be applied to different neighborhoods. We aggregate the pointwise key–value field defined as

$$z_i^b = \widetilde{k}_i^b(v_i)^\top, \qquad b \in \{\theta, \phi\}, \tag{9}$$

where v_i represents the value vector at i. We now define the averaging radius r^b for each branch $b \in \{\theta, \phi\}$ so that the support of the key–value aggregation respects spherical geometry.

Because an ERP grid has uniform latitude spacing $\Delta\phi$, a fixed-width column average preserves the intended north–south span.

The θ-branch requires special care because the physical spacing between adjacent longitude samples decreases with latitude. Let $\Delta\theta$ be the uniform longitude spacing. In the first-order chart in Equation (2), the physical distance between adjacent samples along a row is $\Delta\theta \cos\phi$. Thus, the row support must expand toward the poles. The required radii are

$$r_i^\theta = \alpha/(2\Delta\theta \cos\phi_i), \qquad r^\phi = \alpha/(2\Delta\phi). \tag{10}$$

Let $\mathbf{r}^\theta = (r_i^\theta)$ and $\mathbf{r}^\phi = r^\phi \mathbf{1}$ collect the row and column radii, respectively, and let $\Sigma_{\mathbf{r}}^{\mathrm{row}}$ and $\Sigma_{\mathbf{r}}^{\mathrm{col}}$ denote normalized local averaging operators with radius vector $\mathbf{r}$. The key–value moment tensors are

$$M^\theta = \Sigma_{\mathbf{r}^\theta}^{\mathrm{row}}(Z^\theta), \qquad M^\phi = \Sigma_{\mathbf{r}^\phi}^{\mathrm{col}}(Z^\phi), \tag{11}$$

Within each row, the support and normalization are shared by every longitude query, so the row average is a dense one-dimensional box filter. We evaluate all row and column moments simultaneously via dense one-dimensional reductions using prefix sums, thereby avoiding query-specific neighborhood gathers. Fractional-radius interpolation and ERP-seam handling are detailed in Appendix C. Finally, *DLS* contracts each moment with its modulated query and fuses the branch outputs:

$$y_i^b = (m_i^b)^\top \widetilde{q}_i^b, \quad b \in \{\theta, \phi\}, \qquad y_i = W_O[\, y_i^\theta \,\|\, y_i^\phi \,]. \tag{12}$$

3.3 Approximation error and first-order translation consistency

The first-order construction in *DLS* represents the east–west tangent direction by a constant-latitude row. In exact spherical geometry, however, a geodesic initially pointing east–west bends slightly toward the equator (Figure 3). Let (θ_i, ϕ_i) be a patch center on the sphere and $X = \mathrm{P}(\mathbf{s}, \theta_i, \phi_i)$ be the corresponding ERP tensor. Let j^* denote the exact point in ERP coordinates reached by traveling a distance $\alpha/2$ along the east–west geodesic from this center. This point generally lies slightly off the row through i, rather than at the constant-latitude point j that *DLS* samples. For centers in a compact latitude band $|\phi| \le \pi/2 - \eta$, the equatorward latitude displacement between j and j^* is

$$\delta_\phi(|\phi|, \alpha) = |\phi| - \arcsin\Big(\sin|\phi| \cos\frac{\alpha}{2}\Big) = \frac{1}{8}|\tan\phi|\alpha^2 + O_\eta(\alpha^4). \tag{13}$$

Thus, the discarded latitude displacement is $O_\eta(\alpha^2)$ and increases toward the poles, while the longitude remainder begins at $O_\eta(\alpha^3)$. For another placement $X' = \mathrm{P}(\mathbf{s}, \theta_{i'}, \phi_{i'})$, standard Lipschitz assumptions on the patch and layer consequently give the response bound

$$\|\mathcal{A}_\alpha(X)_i - \mathcal{A}_\alpha(X')_{i'}\| \le C_\eta \alpha^2 + \varepsilon_{\mathrm{grid}}, \tag{14}$$

where $\varepsilon_{\text{grid}}$ collects finite-grid sampling, interpolation, and quadrature error. Therefore, for a small support diameter α, we can approximate the relocated arc by the constant-latitude row through i. The complete expansion and response bounds are provided in Appendix A.

The network used in our experiments has a support diameter of approximately 5.6° at input resolution, for which the displacement at 60° latitude is only 0.12°. The first-order correspondence is therefore accurate at this operating scale, while its error grows for wider supports and toward the poles.

Under the first-order approximation that neglects this $O_\eta(\alpha^2)$ and higher-order terms, ***DLS*** is exactly translation-consistent in the continuous regime (Lemma 1). See Appendix B for the complete proof.

Lemma 1: First-order translation consistency

Under the first-order relocation approximation, ***DLS*** satisfies the translation-consistency condition in Equation (1) with equality in the continuous domain.

Proof sketch. Consider the same patch relocated from center i to i'. For the θ-branch, our *first-order relocation* replaces the exact relocated point j'^* with a point j' on the row through i', preserving the first-order physical offset of j from i (Figure 3). Under this approximation, corresponding locations contain identical features; hence, the shared pointwise projections give $q_i = q_{i'}$, $k_j = k_{j'}$, and $v_j = v_{j'}$, and the relative positional coordinates satisfy

$$p_j^\theta - p_i^\theta = (\theta_j - \theta_i)\cos\phi_i = (\theta_{j'} - \theta_{i'})\cos\phi_{i'} = p_{j'}^\theta - p_{i'}^\theta, \tag{15}$$

so the shared rotary frequency yields identical phase differences. Thus, for the θ-branch,

$$(\widetilde{q}_i^b)^\top \widetilde{k}_j^b = q_i^\top R\left(\omega^b(p_j^b - p_i^b)\right) k_j = q_{i'}^\top R\left(\omega^b(p_{j'}^b - p_{i'}^b)\right) k_{j'} = (\widetilde{q}_{i'}^b)^\top \widetilde{k}_{j'}^b. \tag{16}$$

For the ϕ-branch, a north–south meridional arc remains meridional after relocation, preserving its arc length exactly: $p_j^\phi - p_i^\phi = p_{j'}^\phi - p_{i'}^\phi$. The corresponding features are identical, so Equation (16) holds for the ϕ-branch as well. Finally, the radii in Equation (10) ensure that each branch covers the same physical interval $[-\alpha/2, \alpha/2]$ at both centers. In the continuous domain, the corresponding value-weighted scores therefore have identical normalized integrals, yielding $y_i^b = y_{i'}^b$ in both branches. The fusion in Equation (12) preserves this equality. □

3.4 Resolution-agnostic neural operator

Unlike a convolutional stencil tied to a particular sampling interval, ***DLS*** has no learned parameter tied to grid resolution. Its local pattern is parameterized by the learned *frequencies* ω^θ and ω^ϕ applied to the distance-corrected longitude and latitude coordinates. The same learned ***DLS*** operator can therefore act directly on inputs at resolutions not observed during training, without parameter interpolation or fine-tuning. In the experiments (Section 4.4), we demonstrate this through zero-shot generalization to higher-resolution inputs.

3.5 End-to-end dense network

We construct ***DLS***-Transformer and ***DLS***-SegFormer from the $\mathcal{S}^2$-Transformer and $\mathcal{S}^2$-SegFormer baseline architectures of Bonev et al. (2025) (Figure 4, right). We replace every $\mathcal{S}^2$-attention module and DISCO-based spatial mixer with ***DLS***. We otherwise retain the baseline topologies and channel dimensions, along with the task-specific heads, to enable an apples-to-apples comparison with the baselines in Section 4. The resulting ***DLS*** networks contain no sparse spatial operations and are composed entirely of dense operations. An individual ***DLS*** layer operates over an axial cross-shaped support rather than a dense 2D disk; stacking layers recovers full 2D spatial context.

4 Experiments

We first test zero-shot response consistency under latitude shifts in a controlled setting (Section 4.2). We then evaluate shallow-water forecasting (Section 4.3), omnidirectional depth estimation and semantic segmentation (Section 4.4), and local feature matching (Section 4.5). Finally, we report latency and memory (Section 4.6) and ablate the components of ***DLS*** (Section 4.7). Detailed architectures, training configurations, and additional results are provided in Appendices C–F.

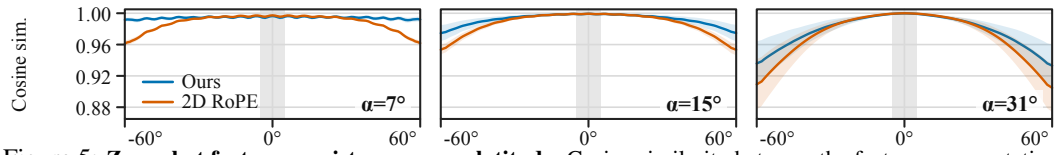


Figure 5: **Zero-shot feature consistency across latitude.** Cosine similarity between the feature representation of each tangent-pasted texture and its equatorial counterpart as the texture is swept over latitude ϕ (x-axis). Each operator is trained only near the equator, $|\phi| \leq 5^\circ$ (gray band). Curves show the mean over five seeds; envelopes indicate one standard deviation. *DLS* maintains higher similarity than 2D RoPE with a fixed ERP box.

4.1 Implementation

We build on `torch-harmonics` (Bonev et al., 2023) and use the Attention on the Sphere architectures of Bonev et al. (2025), denoted $\mathcal{S}^2$-Transformer and $\mathcal{S}^2$-SegFormer, as task-matched spherical baselines. We use the optimized official CUDA implementation[2] for these baselines and a fused Triton kernel (Tillet et al., 2019) for efficient *DLS* inference. Forecasting (Section 4.3) uses the spherical Transformer architecture, whereas depth estimation and segmentation (Section 4.4) and matching (Section 4.5) use spherical SegFormer architecture. Our *DLS* network retains the corresponding baseline topology, channel widths, and task heads while replacing neighborhood $\mathcal{S}^2$ attention and DISCO-based spatial operations with *DLS* to construct a fully dense end-to-end network. We also follow the baseline training protocol such as training epochs, optimizer, scheduler. All tasks use ERP grids for input, intermediate feature, and output.

4.2 Zero-shot feature consistency at novel latitudes

A translation-consistent operator should produce the same response to a patch wherever it is placed, while preserving its orientation relative to the local east–north frame. Using a single *DLS* layer, we train a center detector on tangent-pasted, upright S-shaped textures (Figure 1) centered within $[-5^\circ, 5^\circ]$ and then place the same test textures at unseen latitudes. For each placement, we compute the cosine similarity between the feature at its center and that of the corresponding texture at the equator. The 2D RoPE baseline follows the same training protocol but replaces the coupled row–column formulation with joint longitude–latitude RoPE and a fixed ERP-aligned 2D pooling box matched to *DLS* at the equator.

Figure 5 presents the results. The response of the 2D RoPE baseline changes more rapidly with latitude because its modulated query–key relationship differs from that at the equator. In contrast, *DLS* maintains higher similarity to the equatorial response at unseen latitudes, as expected. Its remaining deviation is consistent with the curvature mismatch quantified in Section 3.3.

4.3 Forecasting partial differential equations on the sphere

Forecasting shallow-water dynamics tests whether local spherical representations remain accurate under autoregressive rollout. The training and testing data are generated by using PDE solver using Earth's radius and angular velocity; initial conditions are sampled from Gaussian random fields, and reference trajectories are computed online with a spectral solver using a 150-second integration step. Each network learns one 30-minute state transition on a 256×512 ERP grid. We use $\mathcal{S}^2$-Transformer as the baseline, replacing its attention and DISCO-based spatial operations with *DLS*. Following Bonev et al. (2025), both models optimize the squared spherical L_2 objective with Adam and a ReduceLROnPlateau schedule. During inference, each model is rolled out autoregressively from the initial state for 10 steps (5 hours) using the same 50 physical initial conditions. We evaluate L_1, L_2 and $W^{1,1}$ errors at the final state.

As shown in Figure 6, *DLS*-Transformer predicts the multi-step future state more accurately than the baseline while halving latency (Appendix Table F).

[2]We use v0.9.1, the latest version at the time of our experiments. Its CUDA implementations of $\mathcal{S}^2$ attention and DISCO were further optimized and are therefore faster than those in v0.8.0, the version originally released with the paper. We use this optimized baseline in the computational-efficiency analysis in Section 4.6.

[3]The released $\mathcal{S}^2$ implementation uses fixed neighborhood cutoffs regardless of resolution; it therefore does not support zero-shot resolution transfer. We scale these cutoffs, which produces $4\times$ more two-dimensional support entries in DISCO while the pixel count also increases by $4\times$, resulting in a $16\times$ increase in FLOPs.

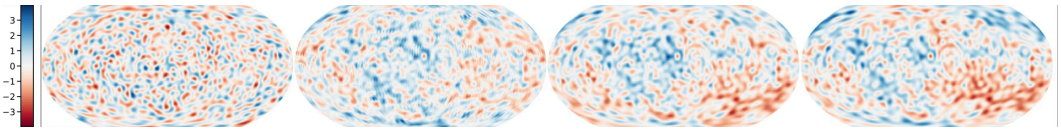


Figure 6: **PDE forecasting.** Predicted water height after a ten-step autoregressive rollout. The $W^{1,1}$ errors are 14.9 and 8.0 for S^2-Transformer and *DLS*-Transformer, respectively (Appendix Table B).

Table 1: **Omnidirectional perception.** Depth and segmentation results on Stanford2D3D. S^2-SegFormer and *DLS*-SegFormer use matched network configurations. The $1024 \times 2048^{\dagger}$ results use weights trained at 512×1024. Shaded rows show scores reported in SphereUFormer and SoLA.

Method	Input resolution	Segmentation		Depth		
		Acc.↑	mIoU↑	MAE↓	MRE↓	δ_1 ↑
S^2-SegFormer	128×256	85.3	65.5	**.175**	.087	91.4
DLS-SegFormer		**86.4**	**67.8**	.176	**.086**	**91.7**
PanoFormer (Shen et al., 2022)	256×512	83.1	60.6	.174	.078	92.5
EGFormer (Yun et al., 2023)		86.5	66.4	.170	.075	93.1
SFSS (Guttikonda and Rambach, 2024)		86.9	68.2	.179	.081	92.2
HexRUnet (Zhang et al., 2019)		81.7	56.1	.201	.090	90.1
HEAL-SWIN (Carlsson et al., 2024)		85.5	63.2	.189	.084	92.2
Elite360D (Ai et al., 2024)		87.4	71.4	.169	.069	93.5
SphereUFormer (Benny and Wolf, 2025)		88.6	72.2	.165	.071	94.0
SoLA (Sekikawa et al., 2026)		88.4	72.2	.166	.071	94.0
S^2-SegFormer		88.1	70.6	.146	.073	93.1
DLS-SegFormer		**89.1**	**73.5**	**.136**	**.063**	**94.1**
S^2-SegFormer	512×1024	89.3	73.3	**.123**	.065	94.4
DLS-SegFormer		**90.3**	**74.2**	.124	**.063**	**94.9**
S^2-SegFormer[3]	$1024 \times 2048^{\dagger}$	88.7	72.1	.184	.081	90.7
DLS-SegFormer		**90.1**	**75.0**	**.133**	**.067**	**94.9**

4.4 Omnidirectional scene understanding

Omnidirectional depth estimation and semantic segmentation test dense prediction under strong latitude-dependent ERP distortion. We evaluate both tasks on the Stanford2D3D dataset of Armeni et al. (2017). For training, we follow the S^2-SegFormer protocol, including its optimizer and loss. For evaluation, we follow SphereUFormer's controlled benchmark protocol, including its data splits, foreground-class segmentation metrics, and depth-reduction convention, to compare our models with the other methods reported in that paper. For segmentation, we report pixel accuracy and mean intersection over union (mIoU) over the 13 semantic classes. For depth, we report the mean absolute error (MAE), the mean relative error (MRE), and δ_1 accuracy. We also evaluate zero-shot resolution transfer by applying weights learned at 512×1024 directly at 1024×2048 without fine-tuning.[3]

Table 1 reports the main results. *DLS*-SegFormer improves over $\mathcal{S}^2$-SegFormer across almost all metrics at every resolution. At 256×512, it raises segmentation mIoU by 2.9 points while achieving a $4.9\times$ speedup (Section 4.6). The transfer results show that *DLS* retains strong performance at a new resolution without fine-tuning.

4.5 Local feature matching

Local feature extraction tests whether the translation-consistent representations produced by *DLS* benefit visual correspondence. We adopt the head architecture and training protocol of XFeat (Potje et al., 2024), a widely used local feature extractor and descriptor for perspective images. We combine the spherical SegFormer backbone with lightweight pointwise descriptor, keypoint, and reliability heads. Training objectives are also adopted from XFeat. Following the previous experiments, *DLS*-SegFormer replaces attention and DISCO-based spatial mixing with *DLS* while keeping the rest of the architecture unchanged. We evaluate both models on Stanford2D3D at 512×1024.

Figure 7 presents a comparison. Under this matched setup, *DLS*-SegFormer produces more correct correspondences and more accurate poses, with a $3.2\times$ speedup (Section 4.6).

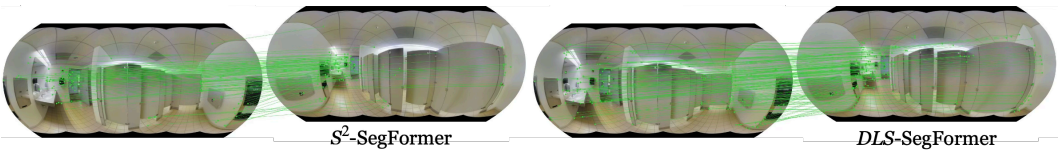


Figure 7: **Local feature matching.** Correct matches among the 512 highest-scoring pairs. Replacing the S^2-SegFormer backbone with *DLS*-SegFormer improves pose accuracy from 54% to 67% (Appendix Table D).

4.6 Computational complexity analysis

Table 2 profiles the segmentation models (Section 4.4), reporting forward FLOPs, batch-size-one latency, throughput, and peak allocated CUDA memory in BF16 on a single NVIDIA H200 with identical software settings. The profile also represents depth and matching networks, which share the same spherical SegFormer backbone and differ only in lightweight pointwise task heads.

The $\mathcal{S}^2$-SegFormer baseline uses its latest optimized CUDA implementation[2], whereas *DLS*-SegFormer uses our fused Triton kernel. *DLS* reduces latency at every resolution; at 256×512, it is $4.9\times$ faster and realizes $3.2\times$ higher throughput while also improving task accuracy (Sections 4.3 to 4.5). The *DLS* variant trades higher peak GPU memory for lower latency. These results demonstrate the GPU efficiency of the proposed dense formulation, which avoids materializing query-specific neighborhoods through irregular, sparse accesses.

Table 2: **Computational complexity.** Spherical SegFormer inference on an NVIDIA H200 at batch size 1. Throughput is FLOPs/latency in TFLOP/s; peak memory is in GB. Latency excludes dataset I/O.

Resolution	S^2-**SegFormer**(Official optimized CUDA)				*DLS*-**SegFormer** (Our Triton)			
	FLOPs↓	Latency ↓	Throughput ↑	Memory ↓	FLOPs↓	Latency ↓	Throughput ↑	Memory↓
128×256	63.1G	26.4ms	2.39	**0.14**	**48.2G**	**8.0ms**	**6.03**	0.29
256×512	302.3G	64.3ms	4.70	**0.39**	**196.8G**	**13.1ms**	**15.05**	1.05
512×1024	1.4T	177.8ms	7.90	**1.30**	**819.8G**	**55.3ms**	**14.83**	4.08
1024×2048[3]	20.7T	1856.6ms	11.14	**5.56**	**3.54T**	**315.6ms**	**11.22**	11.92

4.7 Component ablation

Table 3 evaluates cumulative removals: replacing latitude-adaptive row support by its per-resolution mean, removing distance correction with latitude-independent longitude RoPE, and finally replacing row–column separation by joint 2D RoPE with fixed ERP box pooling. The accuracy drops monotonically from the full *DLS* across all reported metrics, supporting our geometric design.

Table 3: **DLS component ablation.** Successive rows cumulatively replace adaptive support by fixed support, remove distance correction, and replace row–column separation by joint 2D RoPE with fixed ERP box pooling.

Method	Segmentation (256×512)		Depth (256×512)		
	Acc.↑	mIoU↑	MAE↓	MRE↓	δ_1 ↑
DLS (complete)	**89.1**	**73.5**	**.136**	**.063**	**94.1**
− Adaptive support	88.0	70.7	.162	.080	92.8
− Distance correction	87.9	68.1	.163	.081	92.1
− Row–column separation	83.8	61.7	.234	.118	86.3

5 Conclusion, Limitations, and Future Work

We introduced *DLS*, a formulation that realizes an approximately translation-consistent local operator using fully dense computation. Across tasks, *DLS* improves accuracy over the corresponding baselines. Its regular, dense formulation enables up to a $5.9\times$ end-to-end speedup and $3.2\times$ higher throughput. The curvature mismatch neglected by *DLS* grows with support size and toward the poles. Moreover, some low-level features may appear in arbitrary orientations. Higher-order corrections and integration with an $\mathrm{SO}(3)$-equivariant PE would be promising directions for future work.

AI USE STATEMENT

We used large language models (including Google PAT provided by ICLR) to help refine mathematical statements and proofs, improve the clarity of the writing, and correct typographical and grammatical errors. We also used them to assist with experimental design, add implementation options, and improve the efficiency of the Triton-based implementation. The authors implemented the core DLS mechanism and take responsibility for the manuscript, code, and reported results, including all AI-assisted content.

REPRODUCIBILITY STATEMENT

The geometric derivations and proof of first-order translation consistency are provided in Appendices A and B. Appendices C–F provide implementation details, training and evaluation protocols, computational-cost measurements, and ablation settings omitted from the main text for space. We also provide `dls_demo.py`, a minimal, self-contained PyTorch example illustrating the core *DLS* operator, including fractional supports and ERP-seam handling. This illustrative example does not reproduce the experimental results or include the optimized Triton kernels. We will publicly release the complete codebase upon acceptance.

REFERENCES

Hao Ai, Zidong Cao, Yan-Pei Cao, Ying Shan, and Lin Wang. HRDFuse: Monocular 360deg depth estimation by collaboratively learning holistic-with-regional depth distributions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Hao Ai, Zidong Cao, Yan-Pei Cao, Ying Shan, and Lin Wang. Elite360D: Towards efficient 360deg depth estimation via semantic- and distance-aware bi-projection fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Iro Armeni, Alexander Sax, Amir R. Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

Yaniv Benny and Lior Wolf. Sphereuformer: A u-shaped transformer for spherical 360deg perception. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical fourier neural operators: Learning stable dynamics on the sphere, 2023.

Boris Bonev, Max Rietmann, Andrea Paris, Alberto Carpentieri, and Thorsten Kurth. Attention on the sphere. In *Advances in Neural Information Processing Systems*, 2025.

Oscar Carlsson, Jan E. Gerken, Hampus Linander, Heiner Spieß, Fredrik Ohlsson, Christoffer Petersson, and Daniel Persson. HEAL-SWIN: A vision transformer on the sphere. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6067–6077, 2024.

Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *International Conference on Learning Representations (ICLR)*, 2018.

Taco S. Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *International Conference on Machine Learning (ICML)*, 2019.

Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *European Conference on Computer Vision (ECCV)*, 2018.

Michaël Defferrard, Roberto Milani, Frédéric Gusset, and Nathanaël Perraudin. DeepSphere: Efficient spherical convolutional neural network with HEALPix sampling for cosmological applications. *Astronomy and Computing*, 2020.

Johan Edstedt, Georg Bökman, and Zhenjun Zhao. DeDoDe v2: Analyzing and Improving the DeDoDe Keypoint Detector . In *IEEE/CVF Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024.

Carlos Esteves, Christine Allen-Blanchette, Xiaowei Zhou, and Kostas Daniilidis. Learning SO(3)-equivariant representations with spherical CNNs. In *European Conference on Computer Vision (ECCV)*, 2018.

Fabian Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3D roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems*, 2020.

Ryan Griffiths and Donald G. Dansereau. Adapting cnns for fisheye cameras without retraining. *arXiv preprint arXiv:2404.08187*, 2024.

Suresh Guttikonda and Jason Rambach. Single frame semantic segmentation using multi-modal spherical images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3222–3231, 2024.

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision (ECCV)*, 2024.

Michael J Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. Lietransformer: Equivariant self-attention for lie groups. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4533–4543. PMLR, 2021.

Qilong Jia, Yi Xiao, and Wei Xue. SprePE: A spherical geometry-aware position embedding for vision transformers. *OpenReview preprint*, 2026.

Chiyu Max Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Nießner. Spherical CNNs on unstructured grids. In *International Conference on Learning Representations (ICLR)*, 2019.

Hualie Jiang, Zhe Sheng, Siyu Zhu, Zilong Dong, and Rui Huang. UniFuse: Unidirectional fusion for 360deg panorama depth estimation. *IEEE Robotics and Automation Letters*, 2021.

Yeonkun Lee, Jaeseok Jeong, Jongseob Yun, Wonjune Cho, and Kuk-Jin Yoon. SpherePHD: Applying CNNs on a spherical polyhedron representation of 360deg images. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Jeremy Ocampo, Matthew A. Price, and Jason D. McEwen. Scalable and equivariant spherical CNNs by discrete–continuous (DISCO) convolutions. In *International Conference on Learning Representations*, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Guilherme Potje, Felipe Cadar, André Araujo, Renato Martins, and Erickson R Nascimento. Xfeat: Accelerated features for lightweight image matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2682–2691, 2024.

Yusuke Sekikawa, Jun Nagata, Itsumi Araki, and Ruka Eto. Spherical SO(3)-equivariant local attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

Zhijie Shen, Chunyu Lin, Kang Liao, Lang Nie, Zishuo Zheng, and Yao Zhao. Panoformer: Panorama transformer for indoor 360deg depth estimation. In *European Conference on Computer Vision (ECCV)*, 2022.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), 2024.

Yu-Chuan Su and Kristen Grauman. Learning spherical convolution for fast features from 360deg imagery. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Keisuke Tateno, Nassir Navab, and Federico Tombari. Distortion-aware convolutional filters for dense prediction in panoramic images. In *European Conference on Computer Vision (ECCV)*, 2018.

Philippe Tillet, H. T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, page 10–19, New York, NY, USA, 2019. Association for Computing Machinery.

Fu-En Wang, Yu-Hsuan Yeh, Min Sun, Wei-Chen Chiu, and Yi-Hsuan Tsai. BiFuse: Monocular 360deg depth estimation via bi-projection fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Ilwi Yun, Chanyong Shin, Hyunku Lee, Hyuk-Jae Lee, and Chae Eun Rhee. EGFormer: Equirectangular geometry-biased transformer for 360deg depth estimation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Chao Zhang, Stephan Liwicki, William Smith, and Roberto Cipolla. Orientation-aware semantic segmentation on icosahedron spheres. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Jiaming Zhang, Kailun Yang, Chaoxiang Ma, Simon Reiß, Kunyu Peng, and Rainer Stiefelhagen. Bending reality: Distortion-aware transformers for adapting to panoramic semantic segmentation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16917–16927, 2022a.

Jiaming Zhang, Kailun Yang, Hao Shi, Simon Reiß, Kunyu Peng, Chaoxiang Ma, Haodong Fu, Kaiwei Wang, and Rainer Stiefelhagen. Behind every domain there is a shift: Adapting distortion-aware vision transformers for panoramic semantic segmentation. *arXiv preprint arXiv:2207.11860*, 2022b.

Appendix for ICLR 2027 8173

Dense Local Operators on the Sphere

This appendix provides a detailed approximation error analysis (Section A), the complete proof of Lemma 1 (Section B), dense prefix-sum algorithms (Section C), experimental details and additional results (Section D), a computational complexity analysis (Section E), and the component-ablation protocol (Section F).

A Detailed approximation error analysis

We first derive the curvature mismatch between an east–west geodesic and the constant-latitude row used by *DLS*, expanding on Section 3.3. We then describe the second-order correction evaluated in the supplementary ablation. Section B proves translation consistency when this mismatch and discretization error are neglected.

Curvature displacement. Consider a starting point (θ_0, ϕ_0) on the unit sphere and a geodesic launched in the east–west tangent direction. Let l denote the signed arc length from the starting point, distinct from the patch $\mathbf{s}$. Writing the great circle as $\mathbf{x}(l) = \mathbf{x}_0 \cos l + \mathbf{e}_\theta \sin l$, where $\mathbf{e}_\theta$ is the unit east–west tangent at $\mathbf{x}_0$, and taking its vertical component gives

$$\sin \phi(l) = \sin \phi_0 \cos l. \tag{S1}$$

Hence,

$$\phi(l) = \operatorname{sgn}(\phi_0) \arcsin(\sin |\phi_0| \cos l)\,. \tag{S2}$$

The first-order construction instead keeps the latitude fixed at ϕ_0. The resulting equatorward displacement is therefore

$$\delta_\phi(\phi_0, l) = |\phi_0| - \arcsin(\sin |\phi_0| \cos l)\,. \tag{S3}$$

Using $\cos l = 1 - l^2/2 + O(l^4)$ and Taylor expanding $\arcsin(\cdot)$ around $\sin |\phi_0|$ gives

$$\delta_\phi(\phi_0, l) = \frac{1}{2} |\tan \phi_0| l^2 + O_\eta(l^4), \tag{S4}$$

where $|\phi_0| \le \pi/2 - \eta$ for a fixed $\eta > 0$ keeps the center away from the poles. The notation O_η allows the remainder constant to depend on this latitude band.

For a support of physical diameter α, $|l| \le \alpha/2$, and the largest displacement occurs at the support endpoints. Substituting $l = \alpha/2$ gives

$$\delta(|\phi_0|, \alpha) = \delta_\phi(\phi_0, \alpha/2) = \frac{1}{8} |\tan \phi_0| \alpha^2 + O_\eta(\alpha^4), \tag{S5}$$

This endpoint displacement is denoted by $\delta_\phi(|\phi|, \alpha)$ in Equation (13), where the second argument is the support diameter rather than the signed arc length.

The longitude coordinate agrees with the first-order tangent construction to one higher order. The first-order longitude is

$$\theta^{(1)}(l) = \theta_0 + \frac{l}{\cos \phi_0}. \tag{S6}$$

Expanding the exact great-circle longitude around $l = 0$ gives

$$\theta(l) = \theta_0 + \frac{l}{\cos \phi_0} + O_\eta(l^3), \tag{S7}$$

and hence

$$\theta(l) - \theta^{(1)}(l) = O_\eta(l^3). \tag{S8}$$

Thus, over a support of diameter α, the dominant discrepancy between the exact geodesic and the first-order row approximation is the $O_\eta(\alpha^2)$ latitude displacement; the longitude error is only $O_\eta(\alpha^3)$.

Effect on the response. We next relate this geometric discrepancy to the operator response. Let $\zeta(l)$ and $\zeta^{(1)}(l)$ denote the local feature and positional representations evaluated at the exact and first-order locations, respectively, with positional offsets expressed relative to the query. Assume that the feature field and positional mappings are Lipschitz continuous and bounded on the local supports, which remain away from the poles. The geometric bounds above then imply

$$\sup_{|l|\le\alpha/2} \left\|\zeta(l) - \zeta^{(1)}(l)\right\| \le C_\eta \alpha^2, \tag{S9}$$

for a constant C_η independent of α within the considered support range.

The pointwise projections, rotary scores, and value-weighted products are Lipschitz on these bounded representations. Averaging over the support and applying the shared output projection therefore preserve the $O_\eta(\alpha^2)$ bound. Here normalization is by the support length, not by the sum of attention scores.

To compare two placements $X = \mathrm{P}(\mathbf{s}, \theta_i, \phi_i)$ and $X' = \mathrm{P}(\mathbf{s}, \theta_{i'}, \phi_{i'})$, consider a reference response that samples the exact east–west and north–south geodesics and uses their signed arc lengths in the rotary phases. This reference has the same features, phases, and integration interval at both centers, so its responses coincide. Each *DLS* row response differs from its reference by $O_\eta(\alpha^2)$; the column follows a meridian and introduces no such curvature error. The triangle inequality then bounds the difference between the two *DLS* responses by the sum of the two row errors. Including discretization gives

$$\|\mathcal{A}_\alpha(X)_i - \mathcal{A}_\alpha(X')_{i'}\| \le C'_\eta \alpha^2 + \varepsilon_{\mathrm{grid}}, \tag{S10}$$

where C'_η also depends on the feature regularity and layer parameters, and $\varepsilon_{\mathrm{grid}}$ collects errors due to finite-grid sampling, interpolation, and quadrature at both placements. This yields Equation (14).

Thus, the constant-latitude row agrees with the east–west geodesic to first order, while the leading curvature-induced departure from translation consistency is second order in the physical support diameter α. The error increases toward the poles through its dependence on $\tan\phi$, motivating the restriction to a compact latitude band.

Second-order approximation. An east–west geodesic bends toward the equator, whereas the first-order baseline remains on a constant-latitude row. For a small longitude offset $\xi = \theta(l) - \theta_0$ at starting latitude ϕ, Equations (S4) and (S7) give

$$\phi_{\mathrm{geo}}(\xi) - \phi = -\tfrac{1}{2}\sin\phi\cos\phi\,\xi^2 + O_\eta(\xi^4). \tag{S11}$$

We use this quadratic drift to correct the aggregated row moment without explicitly sampling a curved neighborhood.

Let $Z^\theta = \widetilde{k}^\theta v^\top$ denote the row branch's rotary-modulated key–value field. Using the baseline's normalized fractional-box weights $w_{i,t}$ and longitude spacing $\Delta\theta$, we define

$$M_{k,i}(\theta) = \sum_t w_{i,t}(t\Delta\theta)^k Z^\theta(\phi_i, \theta + t\Delta\theta), \qquad k \in \{0, 2\}. \tag{S12}$$

Here $M_{0,i}$ is the baseline row average, while $M_{2,i}$ weights each contribution by its squared longitude offset. Both use the baseline's seam-unwrapped longitude coordinates.

Let $j(i)$ denote the adjacent row closer to the equator. At the same query longitude, the corrected moment is

$$M_i^{\mathrm{corr}} = M_{0,i} + \frac{|\sin\phi_i|\cos\phi_i}{2|\phi_{j(i)} - \phi_i|}\left(M_{2,j(i)} - M_{2,i}\right). \tag{S13}$$

Omitting the second term recovers the first-order baseline; we also omit it when no adjacent row lies closer to the equator. The corrected moment replaces the baseline row moment before query contraction, leaving the column aggregation and support diameter α unchanged.

Each row moment uses its own fractional support, normalization, and rotary phase. The finite difference therefore includes their latitude dependence, rather than isolating a derivative of the feature field at fixed support. This is a discrete, geometry-motivated correction, not exact geodesic sampling or a guarantee of second-order accuracy for the full operator. We evaluate its accuracy and computational cost in the second-order ablation in Section D.

B PROOF OF FIRST-ORDER TRANSLATION CONSISTENCY

We prove Lemma 1 by comparing the same upright spherical patch at two admissible centers i and i', with $X = \mathrm{P}(\mathbf{s}, \theta_i, \phi_i)$ and $X' = \mathrm{P}(\mathbf{s}, \theta_{i'}, \phi_{i'})$. We use the continuous formulation, so there is no discretization error. The first-order relocation approximation identifies each curved row trace with the constant-latitude row at the same signed physical offsets. In other words, we set the curvature displacement δ_ϕ in Equation (13) to zero in this correspondence and omit the higher-order longitude remainder derived in Section A.

Proof. Consider any signed arc-length offset $l \in [-\alpha/2, \alpha/2]$ in either branch. For the θ-branch, the corresponding keys j and j' satisfy

$$(\theta_j - \theta_i)\cos\phi_i = l = (\theta_{j'} - \theta_{i'})\cos\phi_{i'}. \tag{S14}$$

Thus, the distance-corrected longitude difference, and hence the rotary phase difference, is the same at both centers. For the ϕ-branch, the same statement holds exactly because meridians preserve signed arc length: $\phi_j - \phi_i = l = \phi_{j'} - \phi_{i'}$.

Under the stated correspondence, the query and key locations contain the same pixel features at both placements. Shared pointwise projections therefore give identical queries, keys, and values. Together with the equal relative phases, this gives identical rotary scores $a_i^b(l)$ and values $v_i^b(l)$ for each branch $b \in \{\theta, \phi\}$.

Finally, the adaptive radii in Equation (10) select the same physical interval at both centers. The normalized box average in Equation (11) therefore yields

$$y_i^b = \frac{1}{\alpha}\int_{-\alpha/2}^{\alpha/2} a_i^b(l) v_i^b(l)\,\mathrm{d}l = y_{i'}^b. \tag{S15}$$

The conversion from longitude to l has a constant Jacobian within each row, which cancels in the normalized average. The shared fusion in Equation (12) preserves the branch equalities, giving $\mathcal{A}_\alpha(X)_i = \mathcal{A}_\alpha(X')_{i'}$. □

For finite spherical supports and a sampled ERP grid, the omitted terms give the departure from equality bounded in Equation (S10).

C PREFIX SUMS ON A DISCRETE GRID

The continuous operator averages over a physical interval of diameter α, whereas the corresponding grid radius r is generally noninteger. We use a normalized fractional box operator. For $n = \lfloor r \rfloor$ and $\gamma = r - n \in [0, 1)$, samples at offsets $-n, \ldots, n$ receive unit weight, and the samples at offsets $\pm(n+1)$ receive weight γ:

$$\mathcal{B}_r[Z]_j = \frac{\sum_{t=-n}^{n} Z_{j-t} + \gamma\big(Z_{j-(n+1)} + Z_{j+(n+1)}\big)}{2n + 1 + 2\gamma}. \tag{S16}$$

The denominator is the sum of the weights, so constant fields are preserved for every fractional radius. At integer r, the endpoint weights vanish and Equation (S16) reduces to the usual centered box average. The weights vary continuously as r crosses an integer, avoiding abrupt changes in the row response as its latitude-dependent radius changes.

The discrete normalization includes the center sample. For physical grid spacing h and $r = \alpha/(2h)$, its total weight is $2r + 1$, corresponding to an effective integration width $(2r + 1)h = \alpha + h$. This finite-grid convention differs from the continuous diameter-α interval by $O(h)$ and is included in $\varepsilon_{\mathrm{grid}}$. The difference vanishes as $h \to 0$ at fixed α. Here $h = \Delta\theta\cos\phi$ for a row and $h = \Delta\phi$ for a column.

A dense prefix-sum backend can evaluate this box operator *without explicitly materializing* its windows. Let $C_j = \sum_{t \le j} Z_t$ be the cumulative sum along a row or column. The unit-weight interior in Equation (S16) is obtained by one difference,

$$\sum_{t=-n}^{n} Z_{j-t} = C_{j+n} - C_{j-n-1}, \tag{S17}$$

and the two fractional endpoint terms are added by shifted indexing. For the θ-branch, $r = r^{\theta}(\phi)$ is shared by all longitude positions in a given row; for the ϕ-branch, $r = r^{\phi}$ is shared globally. Consequently, all windows can be evaluated as batched dense tensor operations, even though the row radius changes with latitude. This prefix-sum formulation evaluates the normalized averaging operators in Equation (11) without materializing a query-specific neighborhood list.

ERP seam and rotary phase. Periodic values alone are insufficient at the ERP seam because the rotary phase must represent the short, unwrapped longitudinal displacement. We therefore extend each row with a periodic halo wide enough to include both of its fractional endpoints. Samples copied across the left or right boundary are assigned the unwrapped longitudes $\theta - 2\pi$ or $\theta + 2\pi$, respectively, before rotary modulation and construction of the key–value field. A key across the seam then has the same relative phase as it would in a continuous unwrapped chart.

Specifically, samples near $+\pi$ copied into the left halo use $\theta - 2\pi$, while samples near $-\pi$ copied into the right halo use $\theta + 2\pi$. A halo of $\lceil r \rceil$ samples covers all nonzero weights. The feature values are periodic, but the modulated key–value field is not generally periodic: RoPE must be evaluated at the unwrapped coordinates.

In the prefix-sum backend, we form the cumulative sum on the seam-extended field, extract each interior window by the difference above, add its two fractional endpoint contributions, normalize it, and retain the central W outputs. Each supported offset is assumed to have magnitude less than π in longitude, which makes this unwrapping unique. Column windows use replicated latitude boundaries: out-of-range indices are clamped to the nearest latitude row of the modulated key–value field. They do not wrap between the poles.

Triton implementation. The optimized inference path applies a shared 1×1 QKV projection, a fused DLS kernel, and a 1×1 output projection. The fused kernel is specialized for per-head key/value dimensions $(4, 2)$ and $(2, 1)$. Each program processes a contiguous longitude tile for one batch element, head, and latitude row, enabling coalesced reads from channel-first tensors. Q/K normalization, axial rotary modulation, weighted aggregation, and box normalization are fused; low-precision inputs are accumulated in FP32 and the responses are cast back to the input dtype.

The direct kernel traverses the row and column supports in registers, including the fractional row endpoints in Equation (S16). Across the ERP seam, feature indices wrap periodically while angle-addition rotations account for the $\pm 2\pi$ phase offset. Latitude indices are clamped as above. The kernel writes the concatenated branch responses without allocating neighborhood tensors, explicit halos, global prefix tensors, or spatial $K \otimes V$ fields. Its work per query depends on the row and column support lengths.

For wider row supports, an alternative rolling-prefix kernel reuses overlapping windows within each longitude tile. It forms the first key–value moment sum once and expresses subsequent windows through weighted differences of entering and leaving endpoints. A local `tl.cumsum` scan accumulates these differences before contraction with the queries, avoiding a global prefix tensor. The direct, rolling-prefix, and dense reference paths implement the same fractional operator up to floating-point summation order. Training uses the autograd-compatible reference path; inference uses the same learned parameters with the specialized kernels.

D Experimental details

Architecture and implementation details

Spherical SegFormer. We use the four-stage S^2-SegFormer topology of Bonev et al. (2025) as the spherical SegFormer baseline. Each stage downsamples by a factor of 2, with channel widths $[32, 64, 128, 256]$ and block depths $[3, 4, 6, 3]$. The S^2 baseline uses $[2, 4, 8, 16]$ neighborhood-attention heads and DISCO operators for patch merging, feed-forward spatial mixing, and decoder upsampling. It adopts GELU and stochastic depth that increases linearly to 0.1.

Our model retains the same hierarchy and task head, but replaces neighborhood attention and the DISCO spatial operators with DLS. Repeated attention blocks use $[16, 32, 64, 128]$ heads, so that $h = C/2$, with QKV projections shared between the row and column branches. Their per-head query,

key, and value dimensions are 4, 4, and 2, respectively; the auxiliary DLS mixers use the smaller 2, 2, and 1 configuration. We adopt this many-head architecture to reduce the cost of linear attention. For fixed total Q/K and value projection dimensions D_k and D_v, the per-location key–value moments contain $h(D_k/h)(D_v/h) = D_k D_v/h$ entries per branch. Increasing the number of heads therefore reduces the moment-aggregation FLOPs and intermediate storage.

We use this spherical SegFormer configuration for segmentation, depth, and matching; the task-specific sections describe only their prediction heads and data protocols.

Spherical Transformer. For shallow-water forecasting, we use the encoder–processor–decoder S^2-Transformer topology of Bonev et al. (2025). The input contains three physical fields: geopotential, vorticity, and divergence. The encoder maps the 256×512 input to a 128×256 grid with 128 channels, followed by four residual processor blocks. Each block uses affine instance normalization before attention and before a pointwise two-layer MLP, with GELU activation. Dropout and stochastic depth are disabled. The model predicts a residual increment for each normalized physical field at the input resolution, and we apply this update in an autoregressive manner during inference. The baseline uses single-head spherical neighborhood attention and an MLP expansion ratio of two. Its encoder and decoder are DISCO operators with a 5×4 piecewise-linear basis and nodal normalization. A spectral positional embedding is added after the encoder, before the processor. The decoder maps back to the three physical channels.

Our variant removed the spectral positional embedding and replaces these spatial operators with six shared-QKV DLS operators: one in the encoder, four in the processor, and one in the decoder. All use 64 heads with per-head query, key, and value dimensions 4, 4, and 2. The encoder resamples to the half-resolution spherical grid, projects to 128 channels, and applies DLS. The decoder resamples to the input grid and applies a pointwise projection and DLS while retaining 128 channels; a separate 1×1 head then predicts the three residual increments for the next time step. As in the spherical SegFormer, QKV projections are shared between branches.

Grid and angular support. All experiments use equiangular latitude–longitude grids for both input and intermediate representations. We denote *DLS*'s equatorial row and column half-radii by $(r^{\theta}_{\mathrm{eq}}, r^{\phi})$, both *measured in pixels* of each operator's grid.

At a fixed input resolution, all operators in a model use the same pixel half-radii $(r^{\theta}_{\mathrm{eq}}, r^{\phi})$ on their respective grids. Therefore, within each hierarchy, coarser feature grids cover larger angular regions. For example, within the same network, an operator on a 128×256 grid has approximately twice the angular support of one on a 256×512 grid.

Resolution transfer. During zero-shot transfer to a higher resolution, learned weights remain fixed, and we retain the same support radius on the sphere, so each layer's support covers more ERP pixels. The pretrained RoPE frequencies are held fixed to preserve their angular scale.

The default S^2-attention cutoff is proportional to $1/H_\ell$ on an operator grid of height H_ℓ; rebuilding it at higher resolution therefore shrinks its angular support and substantially degrades accuracy, as reported in Table A. For resolution transfer of the S^2 network, we therefore updated their implementation to use the angular support to allow the resolution transfer. For the S^2 network, native-resolution rows use the default cutoffs; the transfer rows ($512 \times 1024 \to 1024 \times 2048$) of Tables 1 to 2, E and F use these adapted cutoffs.

Table A: **Naive S^2-SegFormer transfer**. S^2-SegFormer* is the transfer using the original implementation in `torch-harmonics` without support scaling. The second and the third row is the same as the Table 1.

Method	Resolution		Segmentation		Depth		
	Source	Target	Acc.↑	mIoU↑	MAE↓	MRE↓	δ_1 ↑
S^2-SegFormer*	512×1024	1024×2048	68.5	41.8	.931	.400	27.1
S^2-SegFormer			88.7	72.1	.184	.081	90.7
DLS-SegFormer			**90.1**	**75.0**	**.133**	**.067**	**94.9**

ZERO-SHOT FEATURE-CONSISTENCY PROTOCOL

The controlled experiment in Figure 5 uses a single-layer soft-S center-detector. We use 512×1024 equiangular ERP grid for this experiment. Upright soft-S textures are pasted via a local tangent-plane projection with nominal angular diameters $\alpha \in \{7^\circ, 15^\circ, 31^\circ\}$, where α is the tangent-patch width. Training centers lie between -5° and 5° latitude. Models are optimized for center detection over 300 steps in five independent runs. At test time, the same textures are swept from -60° to 60° latitude. We optimize binary cross-entropy with AdamW, learning rate 2×10^{-3}, weight decay 10^{-4}, and batch size eight. Evaluation uses a 1° latitude spacing and five test longitudes. For each run, we report the mean cosine similarity to the corresponding equatorial representation,

The 2D RoPE baseline (Heo et al., 2024) uses the same data and training protocol, but jointly encodes longitude and latitude and aggregates over a latitude-independent ERP-aligned box matched at the equator. 2D RoPE extends RoPE (Su et al., 2024), originally proposed for one-dimensional token positions, to a two-dimensional grid. Thus, the experiment compares the approximately translation-consistent *DLS* formulation with attention using 2D RoPE and a fixed ERP box.

COMMON CONVENTIONS AND SPHERICAL REDUCTIONS

In this subsection, we define common notation and spherical reductions for the task-specific losses and evaluation metrics described below. These reductions account for the unequal spherical areas represented by grid samples and allow invalid samples to be excluded.

Predictions and targets are denoted by x and x^*, respectively. For samples i with nonnegative spherical quadrature weights ω_i and binary validity masks m_i, we define the normalized masked spherical average as

$$\langle f \rangle_{S^2,m} = \frac{\sum_i \omega_i m_i f_i}{\sum_i \omega_i m_i}. \tag{S18}$$

This reduction excludes invalid samples from both the numerator and denominator; the distinct convention used for depth evaluation is specified below. The quadrature weights ω_i should not be confused with the RoPE frequencies ω^θ and ω^ϕ. On the equiangular grids, Clenshaw–Curtis quadrature accounts for the latitude-dependent spherical area. The exact evaluation grid and weights used for each scene-understanding and matching reduction are specified in the corresponding protocols.

For a K-channel error field $e = x - x^*$, we define the channel-averaged spherical norm and mean-squared error as

$$L_p(e)_{S^2,m} = \frac{1}{K} \sum_{c=1}^{K} \langle |e_{i,c}|^p \rangle_{S^2,m}^{1/p}. \tag{S19}$$

For an error field $e = x - x^*$ in latitude ϕ and longitude θ, the spherical Sobolev seminorm is

$$W^{1,1}(e)_{S^2,m} = \left\langle \left| \frac{1}{\cos\phi} \partial_\theta e \right| + |\partial_\phi e| \right\rangle_{S^2,m}. \tag{S20}$$

For multichannel fields, the seminorm is evaluated componentwise and averaged over channels. Depth training and shallow-water evaluation use the discrete proxy

$$W^{1,1}_{\mathrm{FFT}}(e)_{S^2,m} = \langle |D_\theta e| + |D_\phi e| \rangle_{S^2,m}, \tag{S21}$$

where D_θ and D_ϕ are two-dimensional FFT derivatives with longitude and latitude periods 2π and π, respectively. Each derivative is computed on the full error field before the validity mask is applied in the reduction. This proxy treats latitude periodically and omits the $1/\cos\phi$ factor in Equation (S20); it is not a direct discretization of that continuous spherical seminorm.

SHALLOW-WATER FORECASTING DETAILS

We follow the shallow-water protocol of Bonev et al. (2025), with the geographic orientation fixed relative to Earth's rotation axis. Reference states contain geopotential, vorticity, and divergence on a

256×512 equiangular grid. They are generated online from spectrally band-limited Gaussian initial conditions with Earth's radius and angular velocity, using a 150-second `torch-harmonics` solver step (Bonev et al., 2023). Twelve solver steps form one 30-minute supervised transition. The solver, loss, and evaluation use this grid, and the model uses equiangular grids at both input and hidden resolutions. Spatial reductions use the Clenshaw–Curtis weights in Equation (S18).

Training. The online stream supplies 256 training pairs and eight validation pairs per epoch. We train for 200 epochs with batch size four and Adam, reducing the learning rate by a factor of ten after ten epochs without improvement in validation loss. To expose the model to evolved states, the input time is sampled uniformly from rollout steps 0–10; normalized inputs receive Gaussian noise with standard deviation 0.02. Both methods use the Transformer backbone described in the architecture and implementation details.

For *DLS*, the equatorial row and column half-radii are both four pixels on each operator grid, and the frequency multiplier for $\omega^\theta, \omega^\phi$ is initialized to 18. The initial learning rates are 6×10^{-4} for *DLS* and 10^{-4} for the S^2 baseline.

Loss and evaluation metrics. The training objective is

$$\mathcal{L}_{\mathrm{pde}} = L_{2,\mathrm{sq}}(x - x^*)_{S^2,m}. \tag{S22}$$

The final one-step training and validation losses use the squared L_2 objective, obtained from Equation (S19) with $p = 2$ by omitting the per-channel square root before averaging over channels. In contrast, the reported rollout L_2 error retains this square root. At the final rollout step, we report spherical L_1 and L_2 errors and the discrete $W^{1,1}$ proxy.

Table B reports one trained model per method, using the final epoch-200 checkpoint. Both models are rolled out autoregressively for ten 30-minute steps (five hours) on the same 50 physical initial conditions. The reported $W^{1,1}$ error uses the discrete proxy in Equation (S21).

Omnidirectional scene-understanding details

We adopt the Stanford2D3D split and evaluation conventions of SphereUFormer (Benny and Wolf, 2025). The fixed lists contain 1,000 training and 40 validation panoramas from Areas 1, 2, 3, 4, and 6. For each training run, native-resolution entries in Table 1 report scores from the final checkpoint: segmentation accuracy and mIoU, and depth MAE, MRE, and δ_1 accuracy.

Semantic and depth target panoramas are decoded at native resolution. Semantic colors are mapped to 14 IDs, with label 0 denoting unknown or invalid content. The dataset-provided validity mask excludes invalid semantic and depth targets. Depth is valid when its raw value is finite and satisfies $0 < d_{\mathrm{raw}} \leq 5120$, and training targets are normalized as $d_{\mathrm{raw}}/5120$. Depth uses area interpolation; invalid native depth values are set to infinity before resizing and invalid resized values to zero, so mixed invalid regions remain masked. Semantic labels use nearest-neighbor interpolation. Training applies random yaw shifts and horizontal flips before resizing; RGB values in $[0, 1]$ are normalized as $(x - 0.5)/0.225$.

Both tasks use the SegFormer described in the architecture and implementation details. We use AdamW with learning rate 5×10^{-4}, weight decay 0.1, cosine decay to 10^{-6}, global batch size eight, BF16, and gradient-norm clipping at four. Training lasts 49,600 optimizer steps (approximately 400 epochs).

Training losses. For class probabilities $p_{i,c}$ and labels y_i, segmentation uses the masked spherical average in Equation (S18) to compute cross-entropy loss,

$$\mathcal{L}_{\mathrm{seg}} := \mathrm{CE}(p, y)_{S^2,m} = -\left\langle \log p_{i,y_i} \right\rangle_{S^2,m}. \tag{S23}$$

We minimize $\mathcal{L}_{\mathrm{seg}}$ with label 0 ignored. Cross-entropy is pooled over the valid spherical area of the global training batch.

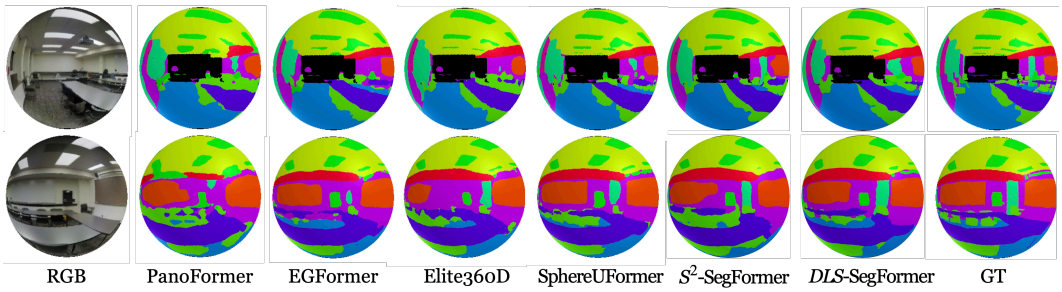
For depth, we use the spherical L_1 loss with the FFT derivative penalty in Equations (S19) and (S21),

$$\mathcal{L}_{\mathrm{depth}} = L_1(x - x^*)_{S^2,m} + 0.1\, W^{1,1}_{\mathrm{FFT}}(x - x^*)_{S^2,m}. \tag{S24}$$

Each depth loss is reduced spatially per image and then averaged over images; the FFT derivatives are computed in FP32 before masking, as described in Equation (S21).

Table B: **Ten-step shallow-water forecasting.** Models learn a 30-minute transition for 200 epochs on a 256×512 equiangular grid and are rolled out autoregressively for ten steps (five hours). Training and validation columns give the final one-step squared spherical L_2 loss; test L_1, L_2, and $W^{1,1}$ are measured at step 10 on the same 50 initial conditions. $\pm$ sample standard deviation across the 50 evaluation initial conditions.

Method	Training loss↓	Validation loss↓	L_1 error↓	L_2 error↓	$W^{1,1}$ error↓
$\mathcal{S}^2$-Transformer	0.00301	0.00327	0.3020 ± 0.0275	0.3793 ± 0.0331	14.91 ± 0.56
DLS-Transformer	**0.00136**	**0.00146**	$\mathbf{0.1986 \pm 0.0238}$	$\mathbf{0.2475 \pm 0.0277}$	$\mathbf{8.00 \pm 0.20}$



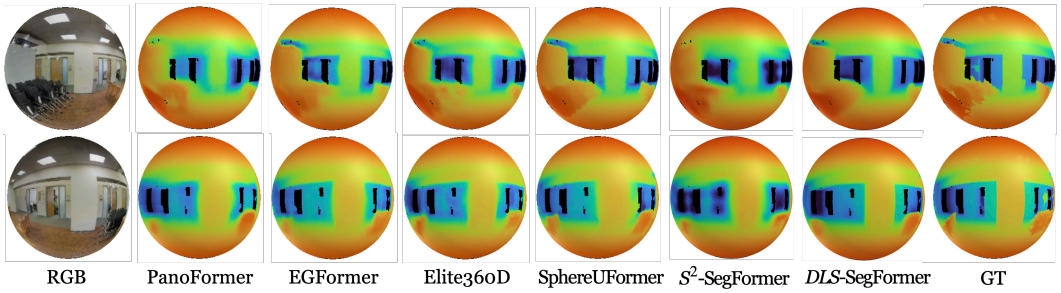


Figure A: **Qualitative comparison in omnidirectional perception.** Top: semantic segmentation. Bottom: depth estimation at 256×512 (or equivalent resolution for the sampling based methods such as SphereUFormer.).

Table C: **Second-order approximation at** 256×512**.** All variants retain the task-specific support and training protocol. Accuracy is from the the final-checkpoint as in the main ablation.

Approximation	Segmentation			Depth			
	Train↓	Acc.↑	mIoU↑	Train↓	MAE↓	MRE↓	δ_1 ↑
First order	0.0776	89.1	73.5	0.1228	**.136**	**.063**	94.1
Second order	**0.0766**	**89.4**	**74.1**	**0.1211**	.140	.069	**94.3**

Evaluation protocol. Segmentation excludes targets with label 0, suppresses output channel 0 when selecting foreground predictions, accumulates a single dataset-level confusion matrix, and reports standard foreground pixel accuracy and mIoU over classes 1–13. For each class c, IoU is $T_p/(T_p + F_p + F_n)$; mIoU averages these ratios over foreground classes with nonzero union. Here, T_p, F_p, and F_n denote true-positive, false-positive, and false-negative node counts. Pixel accuracy is the fraction of correct foreground predictions.

Ignoring label 0 is semantically appropriate because it denotes unknown or masked content. Foreground pixel accuracy also avoids the true-negative dominance of one-versus-rest accuracy, while mIoU gives each foreground class equal weight. We adopt this convention to compare with the prior methods rebenchmarked by SphereUFormer, while retaining the spherical backbone implementation from Attention on the Sphere (Bonev et al., 2025). The latter's evaluation used a random 1,343/35/35 split across all areas, retained class 0, averaged 14 one-versus-rest accuracies including true negatives, and reported per-image micro IoU. Consequently, our S^2-SegFormer results under the common SphereUFormer protocol are not directly comparable to the scores originally reported by the paper.

Depth follows SphereUFormer's released reduction: normalized predictions and targets are multiplied by 10 and clipped to $[0.01, 10]$ meters. Invalid predictions and targets are first zeroed, so both become

0.01 m after clipping and count as perfect predictions. We retain this depth reduction for consistency with SphereUFormer; unlike the training loss, it does not exclude invalid pixels from the denominator.

Ablation on the second-order approximation. We test whether correcting the leading curvature error of the constant-latitude row improves the accuracy–efficiency trade-off, using the second-order approximation in Section A. Table C compares this variant against the first-order baseline at 256×512 for segmentation and depth. Each variant is trained from scratch, otherwise retaining the corresponding first-order configuration (Table 1). The second-order correction introduces no additional learned parameters, but requires extra moment aggregation.

The accuracy changes are mixed. For both tasks, training loss improved. In Table C, segmentation mIoU increases from 73.5 to 74.1, a gain of 0.6 percentage points. For depth, MAE and MRE worsen, whereas the training loss decreases and δ_1 accuracy increases from 94.1 to 94.3. For the latency and memory, Batch-one BF16 inference on an H200 takes 10.9–11.3% longer and uses 76–84% more peak allocated GPU memory with the correction. Each paired measurement uses the same reference backend and precision, 20 warmups, and 100 timed forward passes. Given the mixed changes in accuracy and additional resource costs, we retain the first-order approximation as the default.

LOCAL FEATURE-MATCHING DETAILS

We evaluate sparse local feature matching on Stanford2D3D at 512×1024.

Pair selection. We voxelize depth-derived point clouds at 0.25 m and retain same-room pairs with a maximum directional overlap of at least 0.1. The fixed, room-disjoint split contains 2,778 pairs from 248 rooms for training, 548 pairs from three rooms for checkpoint validation, and 87 pairs from eight rooms for final evaluation.

Teacher supervision. We generate teacher keypoints using the DeDoDe-v2 L-C4-v2 detector (Edstedt et al., 2024). For each panorama, we exclude centers in the 10° polar caps from 160 Fibonacci candidate directions and render a gravity-upright 45° pinhole view at 272×272 for each remaining direction. We retain at most 2,048 detections per view, project them to the nearest ERP pixel, resolve collisions by keeping the highest-confidence detection, and select the global top 8,192. Source keypoints are unprojected using depth, transformed through the camera poses, and reprojected to exact subpixel target locations. Valid correspondences require valid depths in both images, forward and backward angular errors of at most 1°, and target-depth disagreement no greater than $\max(0.15\,\mathrm{m}, 0.1 d_{\mathrm{proj}})$, where d_{proj} is the reprojected depth. Pixels without a retained teacher peak, including excluded polar pixels, are background; there is no separate ignore class.

Model and loss. Both methods use the spherical SegFormer described in the architecture and implementation details. A shared pointwise fusion layer maps decoded native-resolution features to 64-dimensional descriptors, while pointwise heads predict one-channel keypoint and reliability logits. We adapt the sparse XFeat-style descriptor, reliability, and detector objectives (Potje et al., 2024) to spherical sampling and native-resolution prediction. For raw descriptors $\mathbf{d}_i$ and $\mathbf{d}'_j$, we use scores $A_{ij} = 0.2\,\mathbf{d}_i^\top \mathbf{d}'_j$. The descriptor loss $\mathcal{L}_{\mathrm{DS}}$ sums the two directional dual-softmax negative log-likelihoods over valid one-to-one teacher correspondences, using spherical-area weights. The reliability losses $\mathcal{L}_{\mathrm{rel}}^{(k)}$ regress the product of the two directional peak confidences using spherical L_1 loss.

Table D: **Detector-based local feature matching.** Canonical relative-pose evaluation on the Stanford2D3D (87 pairs). Pose error is the maximum of rotation and translation-direction errors; MIR is averaged over all pairs. The GT$\leq \alpha$ column reports the average number of correct matches comparing the estimation with ground truth.

Descriptor backbone	Pose AUC ↑ 5°	10°	20°	Acc.@10° ↑	MIR↑	GT$\leq 1^\circ$ ↑	GT$\leq 3^\circ$ ↑	GT$\leq 5^\circ$ ↑
$\mathcal{S}^2$-SegFormer	27.9	39.7	49.3	54.0	20.2	54.4	134.8	178.7
DLS-SegFormer	**35.7**	**50.3**	**61.2**	**66.7**	**21.5**	**54.7**	**146.9**	**197.6**

Source correspondences use the Clenshaw–Curtis weight of their ERP row; subpixel target coordinates linearly interpolate adjacent row weights. Each directional descriptor loss and reliability loss is normalized by its own sum of correspondence weights (Equations (S18) and (S19)). For the keypoint losses $\mathcal{L}_{\mathrm{kp}}^{(k)}$, every nonzero teacher peak is a binary positive, weighted by spherical area times teacher confidence. The positive and background binary cross-entropy terms are normalized separately and averaged, giving equal class contributions. The complete loss is

$$\mathcal{L}_{\mathrm{match}} = \frac{1}{3}\left[\mathcal{L}_{\mathrm{DS}} + 3\sum_{k=0}^{1}\mathcal{L}_{\mathrm{rel}}^{(k)} + 2\sum_{k=0}^{1}\mathcal{L}_{\mathrm{kp}}^{(k)}\right], \tag{S25}$$

where $k \in \{0, 1\}$ indexes the two images.

Training. We train for 50 epochs using AdamW with a learning rate of 5×10^{-4}, weight decay of 0.1, cosine decay, BF16, and an effective batch size of eight. Each epoch samples 256 training pairs, yielding 32 optimizer steps. Checkpoint validation uses 32 pairs from the reserved validation rooms, and we select the checkpoint with the highest mean top-1 accuracy.

Inference and pose estimation. The learned detector scores candidates by the product of the sigmoid keypoint and reliability predictions. We apply ERP-aware 5×5 non-maximum suppression, mask the 10° polar caps, and retain up to 8,192 highest-scoring candidates per image before mutual nearest-neighbor (MNN) matching. Matched ERP coordinates are converted to unit bearing vectors for spherical eight-point LO-RANSAC, using a 1° inlier threshold and 20–5,000 iterations.

Evaluation metrics. Relative-pose error is the larger of the rotation and translation-direction errors, with the latter minimized over both translation signs. We report normalized AUC of the pose-recall curve at 5, 10, and 20 degrees, computed by trapezoidal integration, and pose accuracy at 10 degrees, all as percentages. We also report the mean spherical-RANSAC inlier fraction (MIR) and the mean number of ground-truth-correct matches within 1, 3, and 5 degrees, averaged over test pairs (Table D).

E Computational complexity analysis details

We benchmark batch-size-one inference on an NVIDIA H200 with FP32 inputs and resident weights, using BF16 CUDA autocast while retaining native FP32 computation scopes. The environment uses Python 3.12, NVIDIA PyTorch 2.6 (25.01), CUDA 12.8, and Triton 3.1. For optimized inference, *DLS* uses the custom Triton kernel described in Section C, whereas S^2 uses its original optimized CUDA implementation. For S^2-Transformer, we additionally report latency with further pointwise/GELU and resampling fusion. Tables E and F report the spherical SegFormer and Transformer benchmarks, respectively; their lower blocks show the pure PyTorch implementations (Paszke et al., 2019).

Inputs are randomly generated. Latency excludes dataset I/O, model construction, checkpoint loading, compilation, and correctness checks. We measure latency using CUDA events and report the median of five trial means, each computed over 100 forward passes after 100 warm-up iterations. Peak allocated CUDA memory, in GB, is measured in a fresh process after warm-up and includes resident model tensors, one input, the output, activations, and workspaces.

We count one multiply–accumulate as two FLOPs. The analytical count includes pointwise projections, local query–key and value contractions, DISCO contractions, and spherical resampling, but excludes indexing. These counts describe the reference formulation, not the instructions executed by the optimized CUDA kernels. For example, reference transposed convolution inserts zeros and processes every output longitude, whereas the optimized CUDA implementation directly scatters contributions and avoids some of this work. Reported throughput is this analytical FLOP count divided by latency, expressed in TFLOP/s; it is not the hardware's executed FLOP rate.

Computational trade-offs. S^2-SegFormer has a higher analytical FLOP count than *DLS*-SegFormer. Transposed DISCO upsampling accounts for approximately 94% of its reported FLOPs, primarily through distributing and accumulating contributions across spherical neighborhoods during decoding. In contrast, *DLS*-SegFormer has more parameters, mainly because it replaces the DISCO spatial mixers with *DLS*.

The memory difference largely reflects where expanded intermediate features are constructed. S^2 projects into DISCO basis features on the coarse grid, then accumulates into the full-resolution output. *DLS* upsamples first, then constructs expanded Q/K/V features. Thus, S^2 uses less CUDA memory, while *DLS* favors GPU-efficient dense computation at the cost of larger intermediate tensors.

Table E: **Spherical SegFormer computational cost.** Batch-one BF16 inference on an NVIDIA H200. The upper block uses optimized CUDA for S^2 and our Triton kernels for *DLS* (same as Table 2).

Resolution	S^2-**SegFormer** (3.2 M parameters)				*DLS*-**SegFormer** (6.4M parameters)			
	FLOPs↓	Latency ↓	Throughput ↑	Memory ↓	FLOPs↓	Latency ↓	Throughput ↑	Memory↓
Optimized implementations (Same as Table 2)								
128×256	63.1G	26.4ms	2.39	**0.14**	**48.2G**	**8.0ms**	**6.03**	0.29
256×512	302.3G	64.3ms	4.70	**0.39**	**196.8G**	**13.1ms**	**15.05**	1.05
512×1024	1.4T	177.8ms	7.90	**1.30**	**819.8G**	**55.3ms**	**14.83**	4.08
1024×2048	20.7T	1856.6ms	11.14	**5.56**	**3.5T**	**315.6ms**	**11.22**	11.92
PyTorch references								
128×256	63.1G	560.5ms	0.11	**0.31**	**48.2G**	**84.6ms**	**0.57**	1.15
256×512	302.3G	2.03s	0.15	**1.02**	**196.8G**	**189.5ms**	**1.04**	4.42
512×1024	1.4T	8.30s	0.17	**3.78**	**819.8G**	**639.2ms**	**1.28**	17.63
1024×2048	20.7T	76.04s	0.27	**15.66**	**3.5T**	**2.5s**	**1.42**	70.65

Table F: **Spherical-Transformer computational cost.** Batch-one BF16-autocast inference on an NVIDIA H200. S2 entries give the original optimized CUDA and our further optimization with kernel fusion in parentheses.

Resolution	S^2-**Transformer** (0.54 M parameters)					*DLS*-**Transformer** (1.23M parameters)			
	FLOPs↓	Latency↓		Throughput↑	Memory↓	FLOPs↓	Latency ↓	Throughput↑	Memory↓
Optimized implementations									
128×256	**19.8G**	4.2 ms	(4.1 ms)	4.71	**0.22**	28.7G	**1.8 ms**	**16.29**	0.36
256×512	**85.0G**	11.2 ms	(10.9 ms)	7.59	**0.76**	115.8G	**5.6 ms**	**20.57**	1.32
512×1024	**362.0G**	43.4 ms	(42.0 ms)	8.34	**2.90**	466.3G	**22.4 ms**	**20.80**	5.17
1024×2048	**1.54T**	181.1 ms	(176.8 ms)	8.51	**11.44**	1.88T	**89.9 ms**	**20.88**	20.56
PyTorch references									
128×256	**19.8G**	239.1 ms		0.08	**0.52**	28.7G	**26.3 ms**	**1.09**	0.78
256×512	**85.0G**	1.1 s		0.08	**1.97**	115.8G	**89.1 ms**	**1.30**	3.02
512×1024	**362.0G**	5.5 s		0.07	**7.74**	466.3G	**343.0 ms**	**1.36**	11.94
1024×2048	**1.54T**	33.6 s		0.05	**30.76**	1.88T	**1.4 s**	**1.36**	47.63

F Component-ablation protocol

Table 3 evaluates cumulative changes at 256×512 for segmentation and depth with the same settings as in the main experiments (Section 4.4). Starting from the complete DLS formulation, the first ablation replaces the latitude-dependent support with its per-resolution mean. The next also removes the longitude distance correction, and the final variant additionally replaces row–column separation with joint 2D RoPE and fixed ERP box pooling. Adjacent rows therefore test a further change conditional on all preceding removals.

On each operator grid of height H_ℓ and width W_ℓ, the fixed row radius is the unweighted arithmetic mean over its latitude rows,

$$\bar{r}_\ell = \frac{1}{H_\ell} \sum_{i=1}^{H_\ell} \min\left(\frac{r_{\mathrm{eq}}^{\theta}}{\max(\cos\phi_i, 10^{-6})}, \left\lfloor \frac{W_\ell - 1}{2} \right\rfloor \right). \tag{S26}$$

Thus, each hierarchy level has its own mean, with polar radii capped before averaging. Column support is unchanged. Removing distance correction replaces metric longitude by raw longitude in the phase. The final variant uses joint 2D RoPE and one separable ERP box aggregation, so it also changes branch fusion and the parameter count.