Unified Spherical Frontend: Learning Rotation-Equivariant Representations of Spherical Images from Any Camera

Mukai Yu Mosam Dabhi Liuyue Xie Sebastian Scherer László A. Jeni
Carnegie Mellon University, Robotics Institute
{mukaiy,mdabhi,liuyuex,basti,laszloaj}@andrew.cmu.edu

Abstract

Modern perception increasingly relies on fisheye, panoramic, and other wide field-of-view (FoV) cameras, yet most pipelines still apply planar CNNs designed for pinhole imagery on 2D grids, where pixel-space neighborhoods misrepresent physical adjacency and models are sensitive to global rotations. Traditional spherical CNNs partially address this mismatch but require costly spherical harmonic transform that constrains resolution and efficiency. We present Unified Spherical Frontend (USF), a distortion-free lens-agnostic framework that transforms images from any calibrated camera onto the unit sphere via ray-direction correspondences, and performs spherical resampling, convolution, and pooling canonically in the spatial domain. USF is modular: projection, location sampling, value interpolation, and resolution control are fully decoupled. Its configurable distance-only convolution kernels offer rotation-equivariance, mirroring translation-equivariance in planar CNNs while avoiding harmonic transforms entirely. We compare multiple standard planar backbones with their spherical counterparts across classification, detection, and segmentation tasks on synthetic (Spherical MNIST) and real-world (PANDORA, Stanford 2D-3D-S) datasets, and stress-test robustness to extreme lens distortions, varying FoV, and arbitrary rotations. USF scales efficiently to high-resolution spherical imagery and maintains less than 1% *performance drop under random test-time rotations without training-time rotational augmentation, and enables* zero-shot *generalization to any unseen (wide-FoV) lenses with minimal performance degradation.* [1]

1. Introduction

Spherical signals arise naturally in many domains - from astrophysics and global climate modeling to omnidirectional perception in robotics and virtual reality. However, modern computer vision pipelines overwhelmingly rely on planar convolutional neural networks (CNNs), which assume a standard pinhole camera model and operate strictly in the 2D image domain. This design becomes problematic as real-world perception systems increasingly adopt fisheye, panoramic, and other wide field-of-view (FoV) lenses. Directly applying planar CNNs to such distorted imagery often leads to suboptimal performance, as image-space neighborhoods do not reflect true physical adjacency.

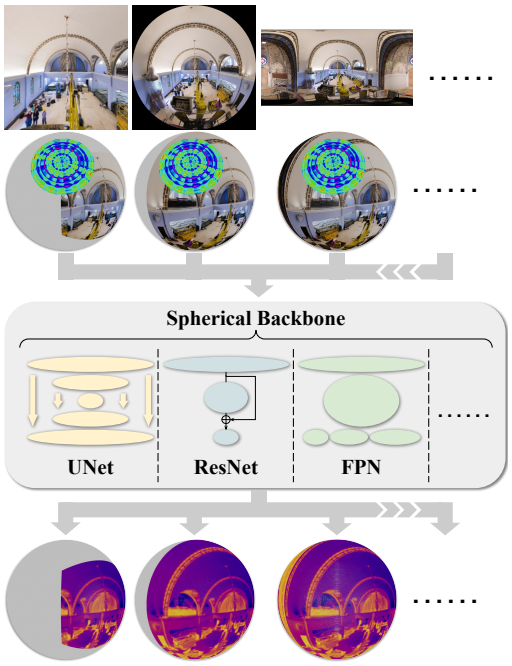


Figure 1. **Unified Spherical Representation.** From any camera to any architecture: a unified spherical pipeline for modern vision.

The core issue is a mismatch between the domain of *processing* and the domain of *geometry*. In a planar CNN, convolution aggregates information from adjacent pixels on a 2D plane, whose proximity is defined by the image grid instead of the physical configuration of light rays. This discrepancy becomes especially pronounced in wide-FoV images, where physically adjacent pixels (e.g., near the poles of an equirectangular panorama) may appear far apart after projection.

[1] Code available on our project website: tomnotch.com/USF

This distortion is not incidental, but a fundamental limitation. According to Gauss's Theorema Egregium [10, 15], no 2D projection can preserve the intrinsic curvature of the sphere. As a result, any attempt to represent spherical signals on a flat image inevitably introduces distortion, undermining the spatial assumptions that planar CNNs rely on. Moreover, since the convolution kernel is fixed in image coordinates, it inherently encodes the coordinate frame of the image, leading to models that are dependent on global rotations.

To address these limitations, we propose *Unified Spherical Frontend (USF)*, a generic, modular, and lens-agnostic framework that lifts vision pipelines from the image plane to the sphere. As illustrated in Fig. 3, USF transforms input planar images from cameras with known intrinsics into spherical signal. It then applies resampling, convolution, and pooling operations entirely on the sphere, abstracting away lens distortions and preserving physical geometry. This representation enables a new class of spherical CNNs that exhibit built-in rotation-equivariance and cross-lens adaptability without requiring task-specific designs or heavy augmentation.

USF is built to be flexible and composable. Every stage, including projection, resampling, convolution, and pooling, is decoupled, allowing users to swap in different location samplers, value interpolators, or output resolutions. Moreover, we show that distance-only weighting functions guarantee rotation-equivariance by construction, enabling models to be robust to arbitrary SO(3) transformations via architectural bias rather than data augmentation.

We evaluate USF on a wide range of vision tasks: MNIST digit classification, object detection on panoramic images, and semantic segmentation across lenses. Our results show that USF maintains competitive performance while demonstrating superior robustness to rotation and zero-shot generalization across unseen wide-FoV lenses. In particular, our spherical CNN layers support plug-and-play replacement of planar layers, enabling general-purpose spherical vision.

Our contributions are summarized as follows:

- We propose a unified and lens-agnostic spherical vision pipeline that processes arbitrary camera inputs in a geometry-aware and rotation-equivariant manner.
- We design a modular spherical resampling module composed of decoupled and configurable location sampling and value interpolation stages.
- We introduce an expressive and efficient spherical convolution kernel with decomposable distance and direction weighting, and demonstrate how architectural design ensures equivariance.
- We validate our approach on three representative tasks, demonstrating zero-shot lens generalization, robustness against random rotation, and competitive performance compared to standard planar models.

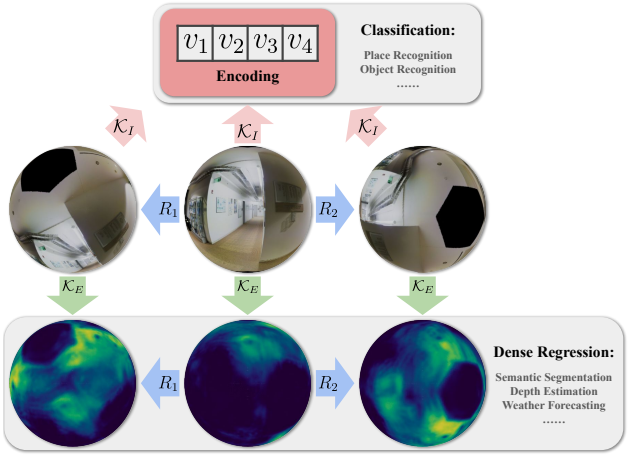


Figure 2. **Rotation Equivariance and Invariance.** A function $\mathcal{K}$ is rotation-equivariant if $\mathcal{K}_E(R \cdot \mathbf{x}) = R \cdot \mathcal{K}_E(\mathbf{x})$, and rotation-invariant if $\mathcal{K}_I(R \cdot \mathbf{x}) = \mathcal{K}_I(\mathbf{x})$, for all $R \in \mathrm{SO}(3)$.

2. Related Work

Lens Distortion and Panoramic Perception. Handling wide-FoV and heavily distorted imagery from fisheye and panoramic lenses challenges planar CNNs, which assume a pinhole model and fixed grid sampling. Directly applying planar convolutions introduces spatial bias, as image-domain adjacency does not necessarily reflect physical proximity. Several methods address this by adapting to spherical geometry: SphereNet [8] samples features on tangent planes, while [9, 28, 29, 39, 40] define specialized kernels on polyhedral, mesh, or graph. These techniques, while geometrically informed, depend on handcrafted grids, predefined connectivity, or structured sampling schemes tailored to full-sphere panoramic cameras. In contrast, our approach treats spherical data as an *unstructured*, *unordered* set of points, enabling flexible processing across arbitrary lens types without assuming any mesh, grid, or sequential order.

Spherical CNNs and Rotation Equivariance. Spectral methods [7, 12] achieve exact $\mathrm{SO}(3)$-equivariance (bottom-half of Fig. 2) by performing convolution in the spherical harmonics domain. Spatial-domain methods [22, 28] define convolution using local graph neighborhoods or predefined meshes, but often relax equivariance or assume specific structures. DISCO [30] advances spatial filtering via learned radial-directional kernels but focuses on dense full-sphere signals and fixed discretizations, thereby lacking flexibility for partial views. Our method remains entirely in the spatial domain, supports partial-sphere coverage, scales efficiently to dense signals, and retains built-in rotation-equivariance, making it well-suited for high-resolution perception tasks with any lens.

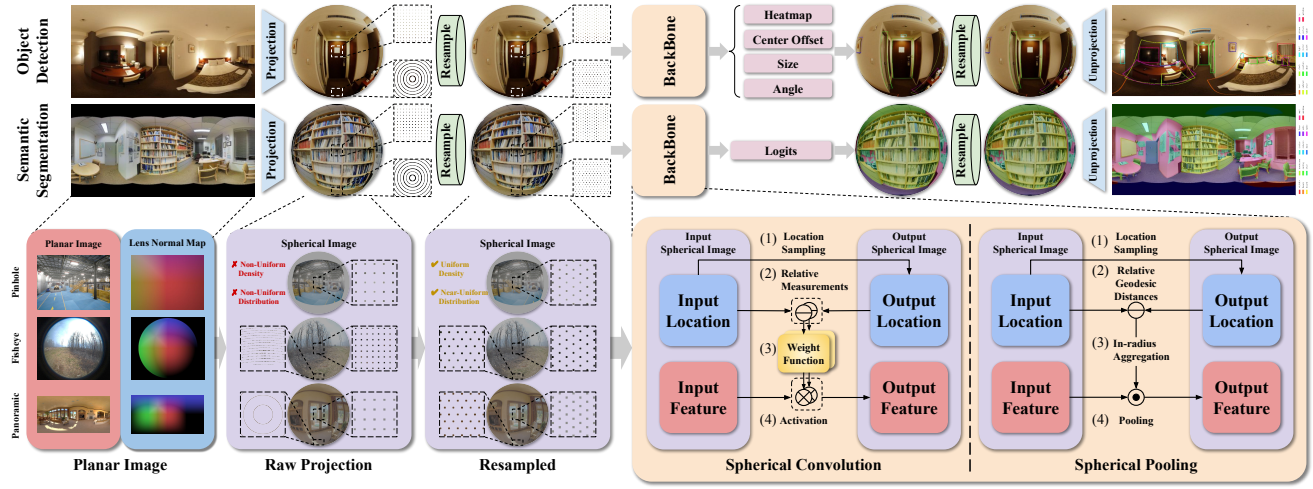


Figure 3. **Unified Spherical Frontend.** (i) A planar image and its lens normal map can be combined to form a (ii) spherical image. Cameras with different lenses produce spatially varying densities and distributions of pixels when projected onto the sphere. Thus, it is crucial to perform (iii) resampling before (iv) feeding into the backbone composed of spherical convolution and pooling layer. Optionally, the results can be (v) resampled back into the raw projected spherical image pixel locations, and (vi) unproject back to the planar image for downstream integration.

Detection and Segmentation on the Sphere. Modern detection and segmentation architectures, such as YOLOv11 [25], DeepLab v3 [6], and UNet [32], are designed primarily for distortion-free pinhole images and lack built-in rotation-equivariance. Extensions like R-CenterNet [38] adapt detection heads to panoramic data but still depend on planar backbones, which degrade under arbitrary rotations. While effective within their target domains, these models are limited in generalizability across lens types and robustness against test-time rotations. In our experiments, we empower these architectures with spherical vision by replacing their planar layers with rotation-equivariant spherical counterparts, enhancing their robustness to unseen camera lenses and random rotations without altering their macro design.

3. Methodology

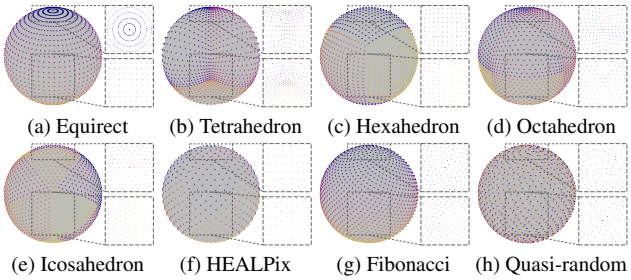
We propose *Unified Spherical Frontend*, a lens-agnostic and task-agnostic strategy to approach generic computer vision applications. We begin by introducing its core components: spherical projection and resampling, spherical convolution, and pooling. These components are then integrated into a complete, modular pipeline, which we instantiate on representative applications in Sec. 4.

3.1. Spherical Projection and Resampling

Given an input image with calibrated intrinsics under any camera model (e.g., pinhole, fisheye, or panoramic), we derive the per-pixel geometry as unit-norm $\mathbb{R}^3$ vectors. Collectively, they form a *lens normal map*[2], which defines a *bijective* mapping between image coordinates and spherical coordinates. Formally, each image coordinate $\mathbf{u} \in \mathbb{R}^2$ maps to a ray direction $\mathbf{p_u} \in \mathbb{S}^2$, with subpixel rays obtained via planar interpolation. Traditional camera models approximate the mapping $\mathbf{p_u}$ using a low-dimensional parametric form (e.g., 4-9 camera intrinsic parameters [21, 24, 36]). In contrast, dense ray maps represent a full-rank mapping $\mathbb{R}^2 \rightarrow \mathbb{S}^2$ over all pixel coordinates.

Projecting all pixels onto the unit sphere yields a spherical image, in which geometry (unit norm $\mathbb{R}^3$ vectors and polar coordinates) and scalar values (e.g., RGB colors, features) are explicitly separated and associated. Unlike prior works [8, 22, 39], we do not assume any structured grid, mesh connectivity, or predefined sequential ordering over spherical points. Instead, our formulation treats the input as an unordered set of sample values located on the sphere. However, since the spherical points originate from the direct projection of planar images, they collectively exhibit non-uniform density and distribution, which hinders learning efficiency and creates undesired bias towards densely sampled regions, such as poles in panoramic images. To mitigate this, it is crucial to resample the spherical image into a near-uniform distribution on the sphere with matching density, and without information loss. This involves two decoupled steps: (1) location sampling to select new points with improved spatial distribution, and (2) value interpolation to assign feature values to those points based on inputs.

[2]Sometimes called *ray map* in relevant literature [41]



(a) Equirect (b) Tetrahedron (c) Hexahedron (d) Octahedron

(e) Icosahedron (f) HEALPix (g) Fibonacci (h) Quasi-random

Figure 4. **Spherical Sampling Methods.** Various location sampling strategies produce different levels of uniformity across the sphere. The bottom row displays point distributions with higher uniformity compared to coarser Goldberg polyhedron discretizations.

3.1.1. Location Sampling

This step selects new pixel locations with near-uniform distribution, matching the density and FoV coverage of the input spherical image from raw projection.

It is well known [1, 4, 7] that no perfectly uniform discretization of the sphere exists under all uniformity metrics. Therefore, as illustrated in Fig. 4, we provide several sampling schemes, including the Goldberg polyhedron [16], HEALPix [18, 19], Fibonacci lattice [17, 35], and quasi-random sampling [20, 34] and systematically evaluate their effects on rotation-equivariance.[3]

To match the pixel density of an input spherical image, we estimate the average area per input pixel and adjust the number of output points accordingly. The average pixel area is computed as the mean of the lower 75% quantile of spherical-Voronoi cell areas, which is robust to outliers near the image boundary.

The raw output points of a location sampler distribute across the entire sphere. Thus, to determine whether a newly sampled point falls within the input spherical image's FoV, we compare its geodesic distance to nearby input points. Specifically, let the minimum radius be twice the average nearest-1-neighbor distance among all input spherical points; an output point is considered outside the FoV if its distance to the closest input point exceeds this threshold.

3.1.2. Value Interpolation

This step assigns scalar values (pixel intensities) to the newly sampled spherical locations. Since the input pixels do not have regular grid or mesh connectivity, standard bilinear interpolation is not applicable. We therefore adopt a two-stage procedure consisting of neighborhood aggregation and local weighting.

Neighborhood aggregation. For each output location $\mathbf{p}_o$, its neighborhood $\mathcal{N}(\mathbf{p}_o)$ is defined either as the N nearest input points or as all input points within a circular cap of radius r centered at the output. Nearest-neighbor aggregation is useful for discrete label interpolation, similar to *nearest_exact* interpolation in planar images. However, it can introduce discontinuities and weaken rotation equivariance, especially when points are sparse, because it selects the closest point based solely on distance, ignoring rotational symmetry.

Local weighting. Given an output point and its local neighborhood, interpolation weights can be computed using either radial or spectral methods. A common approach is to use a *radial basis function (RBF)*[4] with respect to the geodesic distance, which ensures rotational symmetry. This yields interpolation of the form $x_o = \sum \omega_k x_k$, where each weight ω_k depends only on the distance between input and output locations. Alternatively, one may fit a bandlimited spherical harmonic model to the neighborhood, analogous to *moving-least-squares (MLS)* regression.

Both stages of the resampling pipeline are deterministic with respect to geometry. For location sampling, the set of output locations is fully determined by the input pixel locations and the chosen sampling method. For value interpolation, the neighborhood structure and interpolation weights are fixed given two sets of input and output locations. Although the pixel intensities or feature values may vary arbitrarily across frames, the geometric relations remain constant for a given camera. Hence, the entire resampling pipeline is geometry-cacheable: once the geometric mappings are computed, subsequent inference reuses them with negligible overhead.

3.2. Spherical Convolution and Pooling

We begin by revisiting planar CNNs from the perspective of location-based feature aggregation, then generalize to the spherical domain.

3.2.1. Planar CNN

In a standard planar CNN, each output feature is computed by aggregating values from a regular neighborhood around an output location using a square kernel:

$$x_o = \mathcal{K}_{\text{conv}}(\mathcal{X}_i, \mathbf{p}_o) = \sum_{k \in \mathcal{N}(\mathbf{p}_o)} x_k \, \omega_k, \tag{1}$$

$$\mathcal{N}(\mathbf{p}_o) = \left\{ k : \mathbf{p}_k \in \mathcal{P}_i, \ \mathbf{p}_k = \mathbf{p}_o \pm \lfloor \frac{\text{kernel size}}{2} \rfloor \right\}. \tag{2}$$

Planar CNN achieves the same effect as frequency-domain convolution by applying linear aggregation directly in the spatial domain, due to the convolution theorem. Translation-equivariance arises from applying the same kernel at all positions, which also reduces the number of parameters

[3] Sampler implementation details in the supplementary material

[4] RBF weighting scheme details in the supplementary material

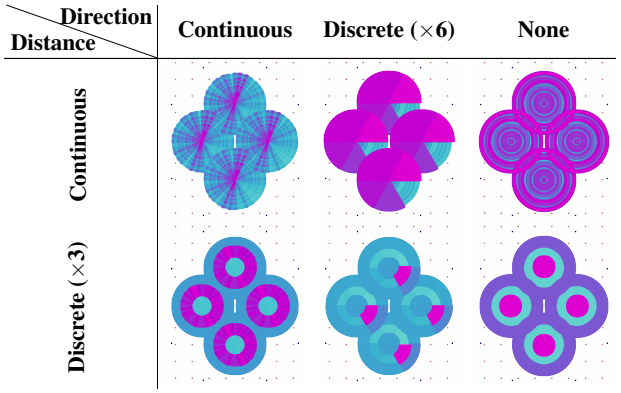


Table 1. **Generic Spherical CNN.** Brown and blue dots denote input and output locations. Colors visualize the activation weights between a given input-output channel pair.

compared to dense MLPs. However, the relative distance and direction of neighbor pixels are inherently coupled in the kernel, and rotating the input image changes how the kernel aligns with features, since the kernel's orientation is fixed with the coordinate frame of the image plane, i.e., tied to a *global gauge*. As a result, planar convolutions are not rotation-equivariant and typically rely on random rotation data augmentation to stay robust against arbitrary unknown rotations at test time.

3.2.2. Generic Spherical CNN

Similar to planar CNNs, spherical CNNs can achieve the same filtering function as convolution in the spherical harmonics domain like [7, 12, 13], by simply performing linear activation or correlation in the spatial domain.[5] This avoids the high computational cost of spherical harmonic transforms, which require a large bandlimit ℓ for lossless reconstruction on dense signals, as dictated by the Nyquist–Shannon sampling theorem [11].

However, unlike the planar counterpart, there is no structured grid or predefined mesh on the sphere. An image projected onto the unit sphere becomes an unordered set of samples on $\mathbb{S}^2$, often non-uniform depending on the lens or projection, so each input pixel can lie at an arbitrary location relative to a given output point, which happens even after resampling. Thus, we define spherical convolution as a local aggregation over input points within a circular cap centered at each output location. In other words, the neighborhood $N(\mathbf{p}_o)$ of an output point $\mathbf{p}_o$ is the set of all input points $\mathbf{p}_k$ whose geodesic distance $d(\mathbf{p}_k, \mathbf{p}_o) \leq r$. This respects the sphere's geometry and does not impose a rigid grid on the data. On the contrary, irregular-shaped neighborhood aggregation, such as square [8], hexagonal, or pentagonal kernels, would compromise rotation-equivariance because they are not radial.

[5] Relevant proofs in the supplementary material

This motivates our design of a generic spherical convolution kernel that computes weights from relative geometric measurements between each input neighbor $\mathbf{p}_k$ and the output $\mathbf{p}_o$. Each output feature is computed as the average of input neighbor values, weighted by a learned function of their relative geometry measurement $\mathcal{M}_m$. We adopt average reduction instead of summation because different output points may aggregate varying numbers of inputs due to non-uniform sampling. Averaging ensures consistent scaling and better reflects the contribution of each neighbor, regardless of local density.

As illustrated in the bottom-right of Fig. 3, formally:

$$x_o = \mathcal{K}_{\text{conv}}(\mathcal{X}_i, \mathbf{p}_o) \tag{3}$$

$$= \frac{1}{|\mathcal{N}(\mathbf{p}_o)|} \sum_{k \in \mathcal{N}(\mathbf{p}_o)} x_k \prod_m f^{(m)}_{\text{weight}} \left(\mathcal{M}_m(\mathbf{p}_k, \mathbf{p}_o)\right), \tag{4}$$

$$\mathcal{N}(\mathbf{p}_o) = \{k : \mathbf{p}_k \in \mathcal{P}_i,\ d\left(\mathbf{p}_k, \mathbf{p}_o\right) \leq r\}. \tag{5}$$

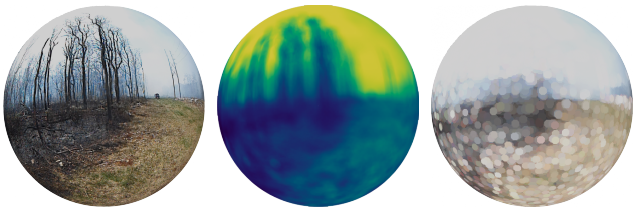
Where x_k is the input feature at neighbor $\mathbf{p}_k$ and $\mathcal{M}_m(\mathbf{p}_k, \mathbf{p}_o)$ denotes a relative measurement between $\mathbf{p}_k$ and $\mathbf{p}_o$. In this work, we use two such measurements: (i) the geodesic distance $d(\mathbf{p}_k, \mathbf{p}_o)$ and (ii) the local 1D direction of $\mathbf{p}_k$ on the tangent plane centered at $\mathbf{p}_o$. We explicitly decouple these two measurements and assign each its own weighting function $f^{(m)}_{\text{weight}}$, whose outputs are multiplied to produce the final weight. This factorization allows the kernel to modulate radial and angular sensitivity independently.

The kernel can take different forms depending on the weighting scheme (see Tab. 1 for variations). The choice of weighting function is up to design, which can be a discrete piecewise-constant (PWC) function, a continuous MLP, or a grid-sampled interpolant as in [30].

In the case of MLP, the direction is represented as a 1D bearing angle in $[-\pi, \pi]$, the relative azimuth from the tangent-projected north at $\mathbf{p}_o$. The geodesic distance is embedded with cosines at integer multiples of a support-adapted base frequency $\lfloor \frac{\pi}{r} \rfloor$, which makes the MLP output both even and 2π-periodic. The bearing angle is embedded with sines and cosines at integer spaced frequencies excluding the raw angle to preserve strict 2π-periodicity and ensure smooth weight transitions as the angle wraps across $-\pi$ and π.

By construction, if the direction branch is absent, the convolution reduces to a zonal/radial filter that depends only on relative distance, a measurement invariant to rotation, thus making the kernel trivially rotation-equivariant.[6] However, once a directional component is introduced, the kernel becomes gauge-dependent, i.e., it relies on a locally defined *up vector* in the tangent plane that rotates with the signal. As a result, the kernel's response varies under global rotation, breaking rotation-equivariance.

[6] Proofs in [12] and the supplementary material



(a) Spherical Image (b) Convolution (c) Max Pooling

Figure 5. **Spherical Convolution and Pooling.** The output locations are set to be identical to the input locations. (b) visualizes a channel of convolution output with weight = 1 and bias = 0, effectively a summation operator.

3.2.3. Spherical Pooling

Spherical pooling is defined analogously over the same geodesic neighborhood as Eq. (5).

As illustrated in the bottom-right of Fig. 3, formally:

$$x_o = \mathcal{K}_{\text{pool}}(\mathcal{X}_i, \mathbf{p}_o) = f_{\text{pool}}\left(x_k : k \in \mathcal{N}(\mathbf{p}_o)\right). \quad (6)$$

f_{pool} can be a simple reducer like min, max, or average, or more complex local statistics such as the mean of the upper quartile.

The output sample locations for spherical convolution or pooling layers are chosen via a location sampler with a resolution factor that controls the density of output points for upsampling or downsampling. Since these coordinates are fixed per layer, all geometric measurements between input and output can be cached and reused efficiently after the first forward pass, while preserving the dynamic configurability of the spherical layers.

Figure 5 visualizes spherical convolution and pooling on a sample input spherical image captured and projected from a fisheye camera.

Putting all components together, the whole pipeline is shown in Fig. 3.

4. Experiments

In this section, we demonstrate the effectiveness and versatility of the proposed *Unified Spherical Frontend* on several representative vision tasks, instantiating our generic pipeline for each. In each case, we replace standard planar layers with our spherical layers to construct the spherical variant, while keeping other aspects of the models and training protocol the same for a fair comparison. All benchmarks are performed in the planar domain to ensure extra consistency. These tasks demonstrate that our approach matches the accuracy of conventional models while preserving robustness under arbitrary image rotations and varying fields of view.

Prior spherical CNNs operate in the harmonics domain and incur prohibitively high computational and memory costs that scale with spatial resolution, we therefore compare with such methods only in the low-resolution MNIST experiment.

Model	NR ↑	RR ↑
Planar	98.45%	41.08%
S^2CNN [7]	96%	94%
SO(3) CNN [12]	98.7%	98.1%
(1) Spherical Dis PWC $\times 3$	87.18%	85.43%
(2) Spherical Dis MLP [8, 8], $L = 0$	67.01%	65.74%
(3) Spherical Dis MLP [8, 8], $L = 6$	92.13%	91.50%
(4) Spherical Dis$\times$Dir MLP [16, 16], $L = 8$	98.28%	43.54%

Table 2. **MNIST Classification Results.** All models are trained without random rotation. L denotes embedding levels.

In the following tables, *NR* and *RR* denote training or testing under *non-rotated* and *randomly rotated* settings.[7]

4.1. MNIST Classification

4.1.1. Experiment Setup

We first evaluate our pipeline on the Spherical MNIST benchmark by stereographically projecting MNIST digits onto the sphere following [7]. We apply global average pooling over feature values before fully-connected layers to ensure rotation-invariance. Training is performed on 6,000 spherical digits for 100 epochs (batch size 1024) without augmentation. To avoid projection distortion for the planar model, random rotations are applied only around the central axis.

In this experiment, we also ablate 4 weighting function parameterizations: (1) Radial Discrete: PWC radial function with 3 segments on distance; (2) Radial Continuous: MLP with [8, 8] hidden channels on unembedded distance; (3) High-Frequency Radial Continuous: similar to (2) but embed distance with 6 fourier levels; and (4) Distance $\times$ Direction: MLP with [16, 16] hidden channels on both distance and direction embedded with 8 fourier levels. Variants (1) and (4) correspond to the bottom-right and top-left cells of Tab. 1.

4.1.2. Results and Discussion

As shown in Tab. 2, spherical CNNs with radial kernel maintain strong performance under rotation, while planar CNNs degrade sharply when test digits are randomly rotated.

Radial-only variants offer built-in rotation-equivariance and perform best under random rotations. In contrast, the distance $\times$ direction kernel captures orientation-sensitive cues (e.g., distinguishing "6" vs. "9") and matches the planar CNN on upright digits, but sacrifices equivariance.

Finally, expressivity depends heavily on kernel parameterization: a low-frequency MLP underperforms a simple PWC design, highlighting the importance of appropriate embeddings.

[7] Training details and additional results in the supplementary material

Train \ Test		NR		RR	
		mAP_{10} ↑	mAP_{50} ↑	mAP_{10} ↑	mAP_{50} ↑
R-CenterNet[38]	NR	35.73%	22.7%	N/A	N/A
Planar YOLOv11[25]	NR	39.65%	24.41%	12.71%	4.66%
	RR	27.76%	9.99%	28.01%	10.24%
Spherical YOLOv11	NR	29.54%	11.41%	29.59%	7.90%

Table 3. **Object Detection Results on PANDORA Dataset.**

4.2. Object Detection

4.2.1. Experiment Setup

Next, we evaluate our pipeline on object detection in 360° panoramic images on the PANDORA dataset [38]. In this experiment, we focus on demonstrating rotation robustness purely from architectural inductive bias. PANDORA contains 3,000 panorama images of resolution 1920×960, annotated with 94,353 oriented bounding boxes of 47 categories. Each box is represented as Rotated Bounding Field-of-View (RBFoV) with 6 parameters: $(\theta, \phi, \alpha, \beta, \gamma, \text{category})$, as articulated in [38]. Raw images are downsampled to 960×480 before training. We adapt the recent YOLOv11 [25] as the backbone and R-CenterNet [38]'s detection head for both planar and spherical models. Detection performance is evaluated with mean Average Precision (mAP) at Intersection over Union (IoU) thresholds of 10% and 50%.[8]

All spherical convolutions adopt the 3-segment discrete PWC weighting function on geodesic distance, identical to scheme (1) from the MNIST experiments.

To apply random rotation on a spherical image, we rotate the spherical image's vectors and then resample back to the canonical unrotated vectors. This ensures that the pixel values reflect a globally rotated view, while maintaining consistent geometry across batches and samples.

4.2.2. Results and Discussion

Table 3 shows that our spherical pipeline offers improved robustness to random rotations compared to the planar baseline. Notably, the planar YOLOv11 model achieves strong performance when trained and tested without rotation, even surpassing the original R-CenterNet [38] due to a more powerful backbone. However, its performance collapses when evaluated under rotations unless trained with explicit rotation augmentation.

In contrast, our spherical model demonstrates rotation-equivariance without augmentation, maintaining stable performance across rotation conditions. This robustness aligns with the MNIST results in Tab. 2, where distance-only spherical kernels preserved rotation consistency, though with some reduction in raw accuracy due to limited expressiveness. The same trade-off recurs here: using discrete radial weights ensures equivariance but restricts directional sensitivity, which is often important for capturing orientation-specific patterns. It is also worth noting that certain prediction targets, such as angular offsets or bounding box orientation, are inherently gauge-dependent and cannot be preserved under global rotation simply by a rotation-equivariant model. Capturing such directional cues may require either directional kernels with explicit data-driven learning through augmentation, or gauge-equivariant architectures that estimate local frame direction.

[8] Additional implementation details in the supplementary material

Train \ Test		NR		RR	
		mIoU ↑	mAcc ↑	mIoU ↑	mAcc ↑
Planar DeepLab v3[6]	NR	35.01%	58.30%	12.11%	22.50%
	RR	32.29%	52.89%	38.30%	53.99%
Planar UNet[32]	NR	33.33%	55.48%	12.91%	23.40%
	RR	33.75%	51.13%	35.91%	51.52%
Planar YOLOv11[25]	NR	28.32%	48.09%	8.17%	16.43%
	RR	28.53%	44.39%	30.62%	45.13%
Spherical DeepLab v3	NR	28.78%	45.27%	28.09%	41.18%
	RR	30.55%	44.58%	32.59%	45.38%
Spherical UNet	NR	25.72%	42.20%	22.99%	35.29%
	RR	25.07%	40.85%	27.83%	41.81%
Spherical YOLOv11	NR	24.29%	40.59%	15.61%	28.08%
	RR	21.52%	38.88%	24.05%	38.98%

Table 4. **Semantic Segmentation Results on Stanford 2D-3D-S.**

4.3. Semantic Segmentation

4.3.1. Experiment Setup

Finally, we evaluate our framework on semantic segmentation and assess its ability to generalize across different camera lenses on Stanford 2D-3D-S [2], which contains 1,413 equirectangular RGB-D panoramas of resolution 4096×2048 with per-pixel semantic labels for 13 classes. To isolate the effect of geometry-aware processing, we use only RGB inputs, omitting available depth data. Input image downsampling and random rotation are performed similarly to object detection.

We experiment with three different backbones to show the plug-and-play nature of our spherical pipeline: DeepLab v3 [6], UNet [32], and YOLOv11 [25]. All spherical models adopt the same distance-only 3-segment PWC kernel.

To simulate images captured by other lens types, we generate lens normal maps for two other camera models: (1) a 90°

Train \ Test		Pinhole		Fisheye		Panoramic	
		mIoU ↑	mAcc ↑	mIoU ↑	mAcc ↑	mIoU ↑	mAcc ↑
Planar DeepLab v3[6]	Pinhole	53.75%	59.70%	33.47%	45.73%	19.57%	36.40%
	Fisheye	67.95%	81.58%	68.54%	82.70%	57.46%	77.46%
	Panoramic	51.56%	62.24%	55.57%	67.91%	71.20%	92.12%
Spherical DeepLab v3	Pinhole	48.71%	62.21%	36.51%	62.07%	35.62%	61.05%
	Fisheye	40.27%	45.45%	54.65%	66.21%	48.04%	63.85%
	Panoramic	36.54%	42.38%	58.52%	69.75%	65.71%	90.44%

Table 5. **Zero-shot Lens Generalizability Test.** Overfitted and tested on the same batch. Random rotation is disabled.

horizontal and vertical FoV pinhole camera with 280×280 resolution, and (2) a 180° FoV fisheye camera with 560×560 resolution (yielding a valid pixel ratio of $\frac{\pi}{4}$) The resolutions are chosen to ensure that the number of pixels is proportional to the FoV coverage area on the sphere. To ensure broad spatial coverage from the original equirectangular image, each lens normal map is randomly oriented toward one of the six cube face directions prior to value interpolation.

To evaluate cross-lens generalization, we perform a single-batch overfitting experiment without applying random rotation. Specifically, we overfit each model on a batch of samples from one lens type and evaluate its zero-shot performance on the same batch resampled into other lens types. This setting explicitly isolates cross-lens adaptability from conventional train-to-test generalization.[9]

4.3.2. Results and Discussion

The results in Tab. 4 confirm that our spherical models exhibit significantly greater robustness to random rotations compared to their planar counterparts. This pattern mirrors the results in MNIST and object detection, where radial-only spherical kernels preserved rotation performance but incurred a small drop in peak accuracy.

The equivariance diagram in Fig. 2 provides more than just intuition. It is a visualization of a logit channel from a spherical segmentation model *trained without random rotation*. This demonstrates the model's true equivariant behavior under global $\mathrm{SO}(3)$ transformations, providing compelling visual evidence of the theory in practice.

As shown in Tab. 5, planar models show clear performance drops when evaluated on different lenses from training, especially in off-diagonal entries. Spherical models perform more consistently across lenses, especially when the source and target lens share similar FoV coverage. Degradation is more noticeable when moving between views with drastically different FoV, such as from pinhole to panoramic, due to mismatched pixel counts. While neither model achieves perfect performance across all lens combinations, the spherical model limits the drop to fewer and less severe cases. Finally, since the 2D-3D-S dataset excludes polar regions from evaluation and no rotation is applied here, a setting that favors planar models. Despite this, the spherical model still shows better generalization across lenses.

[9] Full-scale dataset experiment in the supplementary material

4.4. Ablation Study

We ablate two critical design factors: location samplers, which affect rotation-equivariance through spatial uniformity, and the number of distance segments in PWC function, which controls kernel expressivity.

Location Sampler	Distance Bins	NR		RR	
		mIoU ↑	mAcc ↑	mIoU ↑	mAcc ↑
Icosahedron	3	28.78%	45.27%	28.09%	41.18%
Icosahedron	4	27.99%	45.52%	23.50%	35.57%
Icosahedron	5	29.66%	45.36%	22.82%	34.69%
Icosahedron	6	29.02%	45.95%	21.39%	33.40%
Fibonacci	3	31.69%	47.63%	12.60%	22.79%
HEALPix	3	29.59%	46.98%	13.87%	25.20%
Quasi-random	3	29.85%	48.06%	8.70%	17.73%
Octahedron	3	28.96%	44.12%	14.05%	24.57%
Hexahedron	3	29.25%	45.41%	18.06%	29.27%
Equirectangular	3	30.25%	46.69%	12.87%	23.48%

Table 6. **Ablation Study on Hyperparameters.** Random rotation is disabled during training.

Table 6 shows that icosahedron with 3-segment discretization consistently delivers the best trade-off between accuracy and rotational stability. Using more bins may lead to overfitting, as each segment contains fewer samples. Non-uniform sampling schemes, on the other hand, introduce spatial bias that degrades rotation-equivariance, reinforcing the importance of uniformity in spherical sampling.

5. Conclusion

We presented the *Unified Spherical Frontend (USF)*, a modular and lens-agnostic framework for generic vision tasks. USF separates projection, resampling, convolution, and pooling into independent components, and supports configurable location sampling, value interpolation, and per-layer output resolutions. This modularity allows task-specific architectures to be composed with minimal changes to existing pipelines while scaling efficiently to high-resolution inputs.

Crucially, using distance-only weighting functions guarantees rotation-equivariance by construction. Instead of relying on heavy augmentation to approximate symmetry, USF builds equivariance directly into the geometric formulation.

By representing all camera types within a single spherical domain, USF enables a unified processing space where lens-specific distortions are eliminated. This provides a practical foundation for spherical vision as a general-purpose framework across diverse perception systems in robotics, AR/VR/MR/XR, and beyond.

6. Acknowledgement

We thank Yuheng Qiu and Yuchen Zhang for helpful discussions. This work was supported by computational resources from AirLab Cloud, CUBE Lab clusters, and Pittsburgh Supercomputing Center (PSC) Bridges-2 [5]. We also gratefully acknowledge NVIDIA for providing GPUs through academic hardware grants.

References

[1] Kasra Alishahi and Mohammadsadegh Zamani. The spherical ensemble and uniform distribution of points on the sphere. *Electronic Journal of Probability*, 20:1–27, 2015. 4

[2] Iro Armeni, Sasha Sax, Amir R. Zamir, and Silvio Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding, 2017. 7

[3] Marco Evangelos Biancolini, Andrea Chiappa, Ubaldo Cella, Emiliano Costa, Corrado Groth, and Stefano Porziani. Radial Basis Functions Mesh Morphing. In *Computational Science – ICCS 2020*, pages 294–308, Cham, 2020. Springer International Publishing. 12

[4] Luca Maria Del Bono, Flavio Nicoletti, and Federico Ricci-Tersenghi. The most uniform distribution of points on the sphere. *PLOS ONE*, 19(12):e0313863, 2024. 4

[5] Shawn T. Brown, Paola Buitrago, Edward Hanna, Sergiu Sanielevici, Robin Scibek, and Nicholas A. Nystrom. Bridges-2: A Platform for Rapidly-Evolving and Data Intensive Research. In *Practice and Experience in Advanced Research Computing 2021: Evolution Across All Dimensions*, pages 1–4, New York, NY, USA, 2021. Association for Computing Machinery. 9

[6] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation, 2017. 3, 7, 8, 16, 19, 20

[7] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *International Conference on Learning Representations*, 2018. 2, 4, 5, 6

[8] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. SphereNet: Learning Spherical Representations for Detection and Classification in Omnidirectional Images. In *Computer Vision – ECCV 2018*, pages 525–541, Cham, 2018. Springer International Publishing. 2, 3, 5

[9] Michaël Defferrard, Martino Milani, Frédérick Gusset, and Nathanaël Perraudin. DeepSphere: A graph-based spherical CNN. In *International Conference on Learning Representations*, 2020. 2

[10] Manfredo Perdigão do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, N.J, 1976. 2

[11] J. R. Driscoll and D. M. Healy. Computing Fourier Transforms and Convolutions on the 2-Sphere. *Advances in Applied Mathematics*, 15(2):202–250, 1994. 5, 13

[12] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) Equivariant Representations with Spherical CNNs. *International Journal of Computer Vision*, 128(3):588–600, 2020. 2, 5, 6, 13

[13] Carlos Esteves, Jean-Jacques Slotine, and Ameesh Makadia. Scaling Spherical CNNs. In *Proceedings of the 40th International Conference on Machine Learning*, pages 9396–9411. PMLR, 2023. 5

[14] Matthias Fey. torch_scatter: PyTorch extension library of optimized scatter operations. GitHub Repository: https://github.com/rusty1s/pytorch_scatter, 2023. Version 2.1.2. 13

[15] Carl Friedrich Gauss. *Disquisitiones generales circa superficies curvas*. Typis Dieterichianis, Göttingen, 1828. 2

[16] Michael Goldberg. A Class of Multi-Symmetric Polyhedra. *Tohoku Mathematical Journal, First Series*, 43:104–108, 1937. 4

[17] Álvaro González. Measurement of Areas on a Sphere Using Fibonacci and Latitude–Longitude Lattices. *Mathematical Geosciences*, 42(1):49–64, 2010. 4

[18] Krzysztof M. Gorski, Benjamin D. Wandelt, Frode K. Hansen, Eric Hivon, and Anthony J. Banday. The HEALPix Primer, 1999. arXiv:astro-ph/9905275. 4, 11

[19] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal*, 622(2):759, 2005. 4

[20] J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2(1):84–90, 1960. 4

[21] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, Cambridge, 2. edition, 17.printing edition, 2018. 3

[22] Chiyu Max Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Niessner. Spherical CNNs on Unstructured Grids. In *International Conference on Learning Representations*, 2019. 2, 3

[23] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. 13

[24] J. Kannala and S.S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006. 3

[25] Rahima Khanam and Muhammad Hussain. YOLOv11: An Overview of the Key Architectural Enhancements, 2024. 3, 7, 18

[26] Andi Kivinukk and Gert Tamberg. On Sampling Operators Defined By The Hann Window And Some Of Their Extensions. *Sampling Theory in Signal and Image Processing*, 2 (3):235–257, 2003. 12

[27] Peter J. Kostelec and Daniel N. Rockmore. FFTs on the Rotation Group. *Journal of Fourier Analysis and Applications*, 14(2):145–179, 2008. 13

[28] Yeonkun Lee, Jaeseok Jeong, Jongseob Yun, Wonjune Cho, and Kuk-Jin Yoon. SpherePHD: Applying CNNs on a Spherical PolyHeDron Representation of 360° Images. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9173–9181, 2019. ISSN: 2575-7075. 2

[29] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. Geodesic Convolutional Neural Networks on Riemannian Manifolds. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 832–840, 2015. 2

[30] Jeremy Ocampo, Matthew Alexander Price, and Jason McEwen. Scalable and Equivariant Spherical CNNs by Discrete-Continuous (DISCO) Convolutions. In *International Conference on Learning Representations*, 2023. 2, 5

[31] Ofir Press, Noah Smith, and Mike Lewis. Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. In *International Conference on Learning Representations*, 2022. 15

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. 3, 7

[33] Daniel G. A. Smith and Johnnie Gray. Opt_einsum - A Python package for optimizing contraction order for einsum-like expressions. *Journal of Open Source Software*, 3(26):753, 2018. 13

[34] I. M Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967. 4

[35] Richard Swinbank and R. James Purser. Fibonacci grids: A novel approach to global modelling. *Quarterly Journal of the Royal Meteorological Society*, 132(619):1769–1793, 2006. 4

[36] Jianhua Wang, Fanhuai Shi, Jing Zhang, and Yuncai Liu. A New Calibration Model and Method of Camera Lens Distortion. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5713–5718, 2006. 3

[37] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOLOv2: Dynamic and Fast Instance Segmentation. In *Advances in Neural Information Processing Systems*, pages 17721–17732. Curran Associates, Inc., 2020. 16

[38] Hang Xu, Qiang Zhao, Yike Ma, Xiaodong Li, Peng Yuan, Bailan Feng, Chenggang Yan, and Feng Dai. PANDORA: A Panoramic Detection Dataset for Object with Orientation. In *Computer Vision – ECCV 2022*, pages 237–252. Springer Nature Switzerland, Cham, 2022. 3, 7

[39] Yusheng Yang, Zhiyuan Gao, Jinghan Zhang, Wenbo Hui, Hang Shi, and Yangmin Xie. UVS-CNNs: Constructing general convolutional neural networks on quasi-uniform spherical images. *Computers & Graphics*, 122:103973, 2024. 2, 3

[40] Chao Zhang, Stephan Liwicki, Sen He, William Smith, and Roberto Cipolla. HexNet: An Orientation-Aware Deep Learning Framework for Omni-Directional Input. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12): 14665–14681, 2023. 2

[41] Jason Y. Zhang, Amy Lin, Moneish Kumar, Tzu-Hsuan Yang, Deva Ramanan, and Shubham Tulsiani. Cameras as Rays: Pose Estimation via Ray Diffusion. In *The Twelfth International Conference on Learning Representations*, 2024. 3

Unified Spherical Frontend: Learning Rotation-Equivariant Representations of Spherical Images from Any Camera

Supplementary Material

A. Conventions

In this paper, *location*, *coordinate*, or *geometry* refers to a point on the unit sphere, represented either by Cartesian coordinates in a right-handed system:

$$\mathbf{p} = (x, y, z) \in \mathbb{R}^3, \quad \|\mathbf{p}\| = 1 \tag{7}$$

or by polar coordinates:

$$(\theta, \phi) \in \mathbb{S}^2, \quad \theta \in [-\frac{\pi}{2}, \frac{\pi}{2}], \quad \phi \in [-\pi, \pi] \tag{8}$$

where θ and ϕ are latitudes and longitudes. $(x, y, z) = (1, 0, 0)$ corresponds to $(\theta, \phi) = (0, 0)$, and the bidirectional mapping between Cartesian and polar coordinates is given below:

$$\begin{cases} x = \cos\theta\cos\phi \\ y = \cos\theta\sin\phi \\ z = \sin\theta \end{cases}, \begin{cases} \theta = \arctan\left(\frac{z}{\sqrt{x^2+y^2}}\right) \\ \phi = \arctan\left(\frac{y}{x}\right) \end{cases} \tag{9}$$

Both Cartesian and polar coordinates are *2D* parameterizations of the same manifold. However, the spherical parameterization exhibits singularities at the poles: all $(\frac{\pi}{2}, \phi)$ map to the north pole, and all $(-\frac{\pi}{2}, \phi)$ to the south pole, regardless of ϕ.

B. Location Sampling Method

This section details point generation methods.

B.1. Goldberg Polyhedron

Polyhedral sampling generates spherical points by subdividing each polygon face of a base convex polyhedron (e.g., tetrahedron, hexahedron, dodecahedron) with generalized barycentric coordinates. Let a polygon face have m vertices $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{S}^2$. For a subdivision level n_{side}, we enumerate all integer tuples:

$$(i_1, \ldots, i_m),\ i_k \geq 0,\ \sum_{k=1}^{m} i_k = n_{\text{side}}, \tag{10}$$

Each tuple defines a generalized barycentric combination inside the face:

$$\mathbf{p} = \sum_{k=1}^{m} \frac{i_k}{n_{\text{side}}} \mathbf{v}_k. \tag{11}$$

After normalization to unit norm, this yields a point on the sphere. Applying this process to all faces yields a near-uniform spherical point set. Different polyhedra use the same subdivision rule but differ in how N_{side} relates to the target average pixel area.

In this work, "Goldberg polyhedron" refers only to the *spherical Voronoi* tessellation induced by the sampled points, which typically manifests as a mixture of pentagons and hexagons. The sampling itself is purely barycentric.

B.2. HEALPix

Refer to the HEALPix Primer[18].

B.3. Fibonacci Lattice

$$\mathbf{p}_i = (\sin\phi_i\cos\theta_i,\ \sin\phi_i\sin\theta_i,\ \cos\phi_i), \tag{12}$$

$$\phi_i = \arccos\left(1 - \frac{2i}{N}\right), \tag{13}$$

$$\theta_i = \frac{2\pi \cdot i}{\varphi},\ \varphi = \frac{1+\sqrt{5}}{2}. \tag{14}$$

Where φ is the golden ratio, N is the total number of points. Note that ϕ and θ here are different from the definition in Sec. A.

B.4. Quasi-Random Sampling

Quasi-random sampling generates low-discrepancy sequences similar to Sobol or Halton sequences.

First, generate evenly distributed points in $[0, 1]^2$ with irrational-ratio recurrence. Specifically, the plastic constant $\psi \approx 1.32471795724474602596$, the unique real root of $\psi^3 - \psi - 1 = 0$, is adopted as the irrational number. Define two incommensurate step sizes $\begin{cases} \alpha_u &= \psi^{-1} \approx 0.7549, \\ \alpha_v &= \psi^{-2} \approx 0.5698 \end{cases}$ together with starting offsets $s_{0,u} = s_{0,v} = 0.5$, 2D quasi-random sequence can be generated by:

$$u_i = (s_{0,u} + i \cdot \alpha_u) \bmod 1, \tag{15}$$

$$v_i = (s_{0,v} + i \cdot \alpha_v) \bmod 1, \tag{16}$$

for $i = 0, \ldots, N-1$.

Then, each point $(u_i, v_i) \in [0, 1]^2$ is mapped to the unit sphere with the Lambert equal-area projection:

$$\mathbf{p}_i = (r_i\cos\phi_i,\ r_i\sin\phi_i, z_i)\,, \tag{17}$$

$$z_i = 1 - 2u_i, \tag{18}$$

$$r_i = \sqrt{1 - z_i^2}, \tag{19}$$

$$\phi_i = 2\pi \cdot v_i. \tag{20}$$

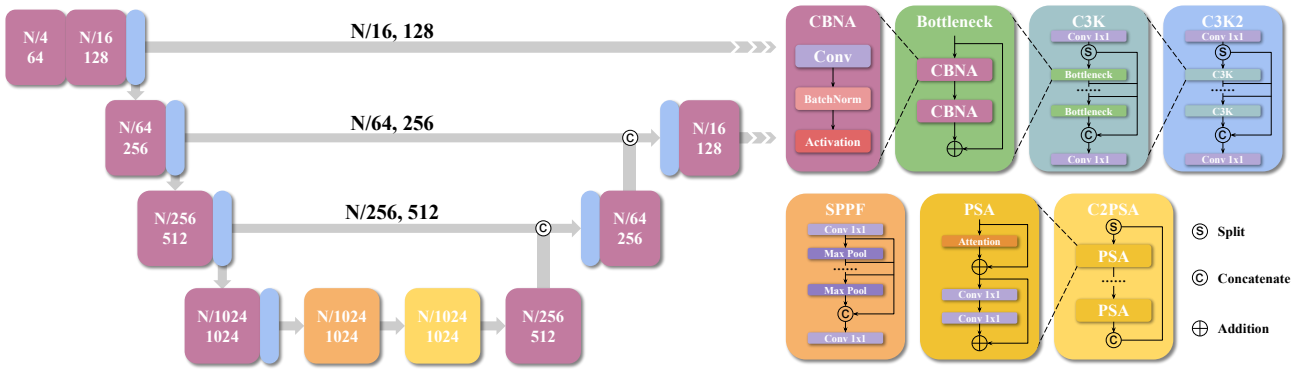


Figure 6. **YOLOv11.** Core components include: **CBNA**: convolution, batch normalization, activation; **SPPF**: spatial pyramid pooling-fast; **PSA**: Partial Spatial Attention.

C. Value Interpolation RBF Kernel

For an output location $\mathbf{p}_o$, let $\mathcal{N}(p_o)$ be the collection of neighbor input points and ω_k be the weight applied to input value x_k at input location $\mathbf{p}_k$. $d(\mathbf{p}_i, \mathbf{p}_j)$ is the geodesic distance between point $\mathbf{p}_i$ and $\mathbf{p}_j$.
The interpolated value x_o at a new output location $\mathbf{p}_o$ is expressed as:

$$x_o = \sum_{k \in \mathcal{N}(\mathbf{p}_o)} \omega_k \, x_k \tag{21}$$

Where weights ω_k may come from different RBF kernels. Weights are normalized so that their sum equals 1 for an output value.

C.1. Softmax

$$\omega_k = \frac{e^{-\frac{d(\mathbf{p}_k, \mathbf{p}_o)}{\mathcal{T}}}}{\sum_{i \in \mathcal{N}(\mathbf{p}_o)} e^{-\frac{d(\mathbf{p}_i, \mathbf{p}_o)}{\mathcal{T}}}} \tag{22}$$

$\mathcal{T}$ is the temperature hyperparameter that controls sharpness.

C.2. Gaussian

$$\omega_k = e^{-\frac{d(\mathbf{p}_k, \mathbf{p}_o)^2}{2\sigma^2}} \tag{23}$$

C.3. Hann

[26]:

$$\omega_k = \frac{1}{2}\left(1 + \cos\left(\pi \cdot d(\mathbf{p}_k, \mathbf{p}_o)\right)\right) \tag{24}$$

C.4. Wendland-C2

[3]:

$$\omega_k = \left(1 - d(\mathbf{p}_k, \mathbf{p}_o)\right)^4 \cdot \left(1 + 4 \cdot d(\mathbf{p}_k, \mathbf{p}_o)\right) \tag{25}$$

D. Continuous Spherical Spatial Correlation

Analogous to a corollary in Fourier analysis for planar CNNs, we prove that spherical correlation in the spatial domain corresponds to multiplication in the spherical harmonic (frequency) domain.

Let $f, \mathcal{K} : \mathbb{S}^2 \to \mathbb{C}$ denote signals on $\mathbb{S}^2$, we define their spherical correlation $f \star \mathcal{K}$ by:

$$(f \star \mathcal{K})(\omega) = \int_{\mathbb{S}^2} f(\omega') \, \overline{\mathcal{K}(R_\omega^{-1}\omega')} \, d\Omega(\omega'), \tag{26}$$

Where $\omega, \omega' \in \mathbb{S}^2$, R_ω is any fixed rotation in $\mathrm{SO}(3)$ such that $R_\omega(\mathbf{n}) = \omega$ with $\mathbf{n} = (0, 0, 1)$ denoting the north pole. $d\Omega(\omega')$ is the surface measure on $\mathbb{S}^2$, and $\bar{\cdot}$ denotes complex conjugation.

Any $f \in L^2(\mathbb{S}^2) : \mathbb{S}^2 \to \mathbb{C}$ admits a spherical harmonic expansion:

$$f(\omega) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \hat{f}_{\ell,m} \, Y_\ell^m(\omega), \tag{27}$$

$$\hat{f}_{\ell,m} = \int_{\mathbb{S}^2} f(\omega) \, \overline{Y_\ell^m(\omega)} \, d\Omega(\omega), \tag{28}$$

Where $\{Y_\ell^m\}$ are harmonic functions that form an orthonormal basis of $L^2(\mathbb{S}^2)$, likewise for $\mathcal{K}(\omega)$.

We now derive the spherical harmonic coefficients of $f \star \mathcal{K}$

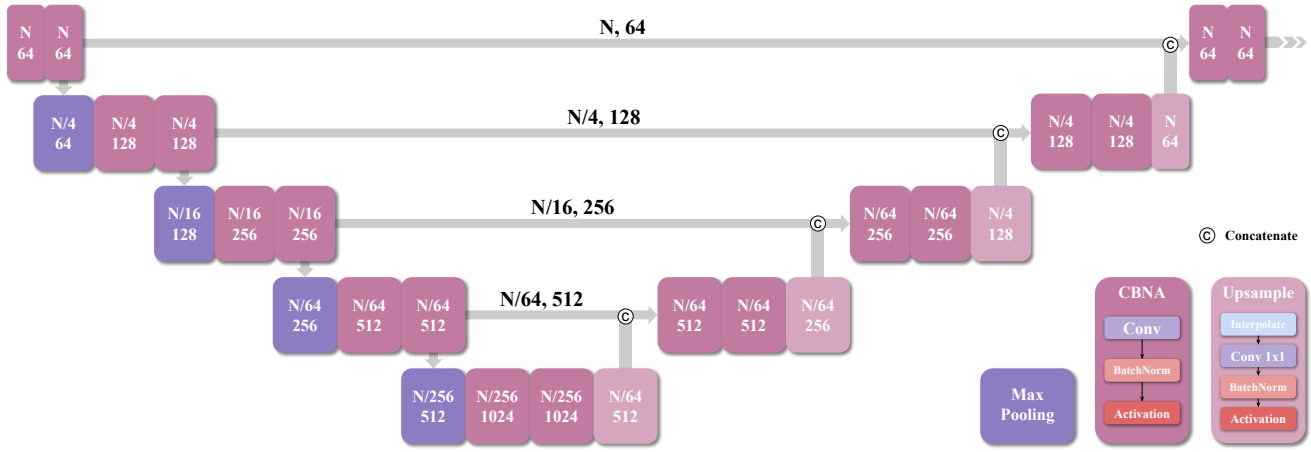


Figure 7. **UNet.** Core components include: **CBNA**: convolution, batch normalization, activation; **Upsample**: value interpolation, 1x1 channel-wise convolution, batch normalization, activation.

as:

$$\widehat{(f \star \mathcal{K})}_{\ell,m}$$

$$= \int_{\mathbb{S}^2} (f \star \mathcal{K})(\omega)\, \overline{Y_\ell^m(\omega)}\, d\Omega(\omega) \tag{29}$$

$$= \int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} f(\omega')\, \overline{\mathcal{K}(R_\omega^{-1}\omega')}\, d\Omega(\omega') \right] \overline{Y_\ell^m(\omega)}\, d\Omega(\omega) \tag{30}$$

$$= \int_{\mathbb{S}^2} f(\omega') \left[\int_{\mathbb{S}^2} \overline{\mathcal{K}(R_\omega^{-1}\omega')}\, \overline{Y_\ell^m(\omega)}\, d\Omega(\omega) \right] d\Omega(\omega') \tag{31}$$

For general anisotropic kernel $\mathcal{K}$, $\widehat{(f \star \mathcal{K})}_{\ell,m}$ becomes a finite sum of products $\hat{f}_{\ell,m}\ \overline{\hat{\mathcal{K}}_{\ell,m}}$ multiplied by algebraic factors[11, 27]:

$$\widehat{(f \star \mathcal{K})}_{\ell,m} = \sum_{m'=-\ell}^{\ell} \hat{f}_{\ell,m'}\, \overline{\hat{\mathcal{K}}_{\ell,m'}}\, \Lambda_\ell(m, m'). \tag{32}$$

Which corresponds to multiplication in the spherical harmonic domain.

E. Rotation Equivariant Correlation

As detailed in[12], when the kernel $\mathcal{K}$ is isotropic, meaning that it is zonal, radial, isotropic, and its value only depends on the relative geodesic distance, $\widehat{(f \star \mathcal{K})}_{\ell,m}$ simplifies to:

$$\widehat{(f \star \mathcal{K})}_{\ell,m} = \alpha_\ell\ \hat{f}_{\ell,m} \tag{33}$$

Which is purely diagonal pointwise multiplication in (ℓ, m), where α_ℓ can be regarded as the kernel's "*frequency response*", instead of "*phase response*" (m-mode coefficients) that changes with rotation, hence preserving rotation-equivariance.

Intuitively, if we rotate all the coordinates ω of a spherical image with any rotation $R \in \mathrm{SO}(3)$, $\mathcal{K}$ activates $R\omega$ with the same set of weights for local signals around $R\omega$ regardless of orientation because it's isotropic.

F. Computation Optimization

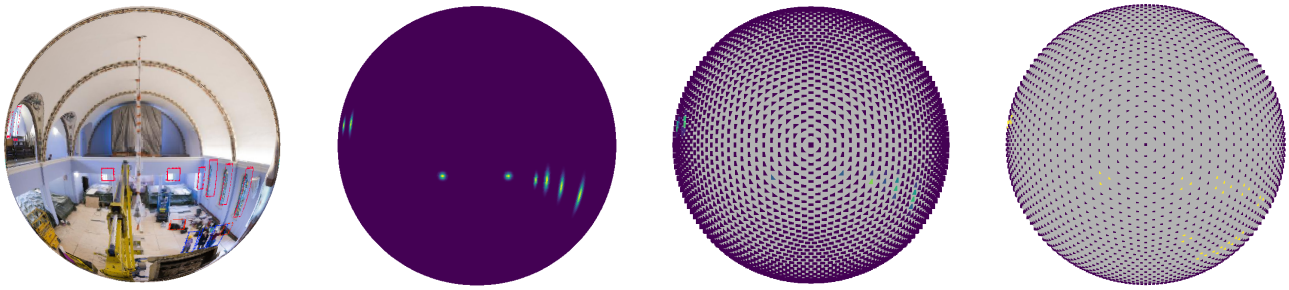
A key design consideration of USF is that operations contraction schemes changes only with respect to geometry and not feature values. This allows expensive geometric computations such as neighborhood construction, interpolation weights, and sparse aggregation structures to be precomputed once and reused across subsequent forward passes. This geometry caching mechanism is critical for making high-resolution spherical processing computationally feasible in practice.

Our implementation utilizes several optimized libraries for efficient spherical processing. We use *FAISS* [23] for fast nearest-neighbor search during neighborhood construction, *torch_scatter* [14] for efficient neighborhood aggregation and interpolation, and *opt_einsum* [33] for optimized tensor contraction ordering in dense operations. These components significantly reduce overhead compared to naive implementations and make large-scale spherical processing feasible within the PyTorch framework, even without custom CUDA kernels.

In all benchmarks, the input to both spherical and planar pipelines is a 960×480 panorama image with batch size 8 and RGB channels. All experiment results are averaged over 10 runs on an NVIDIA H200 GPU using PyTorch 2.8.0, CUDA 12.8, torch_scatter 2.1.2, and float32 precision.

F.1. Location Sampling and Value Interpolation

The location sampling resolution factor is set to 1.0, meaning the number of output spherical samples is matched to the



(a) **Spherical Image with RBFoVs.** Red boxes denote windows.

(b) **Continuous Heatmap.** Per-category gaussian heatmaps are generated through gnomonic projection regardless of the point distribution on the sphere.

(c) **Discrete Heatmap.** Used in actual supervision.

(d) **Regression Mask.** L1 regression loss only supervises the yellow points of regression head outputs, where RBFoVs are centered.

Figure 8. **Object Detection Supervision.**

number of input pixels with less than 1% difference. The sampling benchmark measures the runtime of location sampling and value interpolation separately under both cold-start (first run) and sustained (cached) settings. In the cold-start scenario, geometric structures such as spherical neighborhood structure and interpolation weights must be constructed. In the sustained setting, these structures are reused and only value aggregation via matrix multiplication is performed.

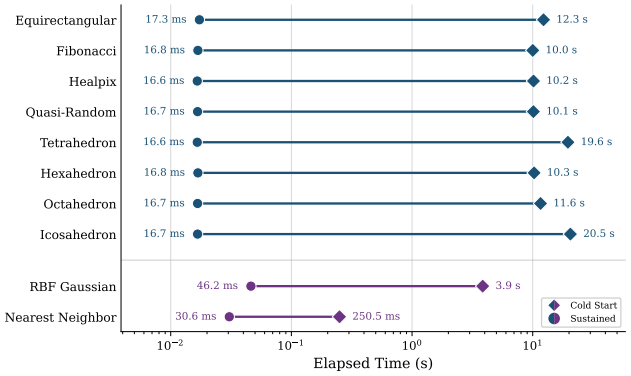


Figure 9. **Location Sampling and Value Interpolation Benchmarks.** Cold-start runtime includes geometric preprocessing such as neighborhood construction and interpolation weight computation. Sustained runtime reuses geometry and only performs value aggregation via matrix multiplication.

As shown in Fig. 9, the cold-start cost is dominated by geometric preprocessing and can take several seconds depending on the sampling method and interpolation kernel. However, once geometry is cached, the runtime drops to the millisecond level across all sampling methods, representing orders-of-magnitude speedup. This demonstrates that the computational bottleneck lies in geometry construction rather than interpolation itself, and confirms that geometry caching is essential for practical spherical processing.

This result highlights a fundamental property of spherical pipelines: while geometry-aware processing introduces an initial preprocessing cost, this cost is amortized over all subsequent forward passes, making sustained runtime comparable to standard image processing pipelines.

F.2. Spherical Convolution and Pooling

We next benchmark individual spherical convolution and pooling operators and compare them with planar Conv2d and MaxPool2d layers. For fairness, the planar convolution uses a 5×5 kernel, since each spherical output location aggregates approximately $25 - 30$ neighboring samples on average, making the receptive field sizes comparable.

Three spherical convolution implementations are evaluated:

1. **Continuous Distance $\times$ Direction MLP:** Distance and direction weighting functions are parameterized by an MLP with hidden dims $[16, 16]$ and positional encoding $L = 8$. During training, the MLP must be evaluated for each neighbor pair, but during inference, weights can be fully cached once geometry is fixed.
2. **Discrete Distance Piecewise-Constant (PWC) (Dense):** The MLP is replaced by discrete distance bins implemented using torch embedding.
3. **Discrete Distance Piecewise-Constant (PWC) (Sparse):** Aggregation is performed by sparse matrix multiplication.

Figure 10 show that cold-start runtime is significantly higher due to kernel construction and sparse structure generation. However, sustained runtime drops substantially once geometry and weights are cached. Among spherical implementations, the discrete PWC (sparse) implementation achieves the fastest sustained runtime, while the continuous MLP variant benefits significantly from weight caching during inference.

Although spherical operators remain slower than highly optimized planar Conv2d kernels, the gap narrows significantly in sustained execution. This difference is largely due to the fact that planar convolutions are implemented as heav-

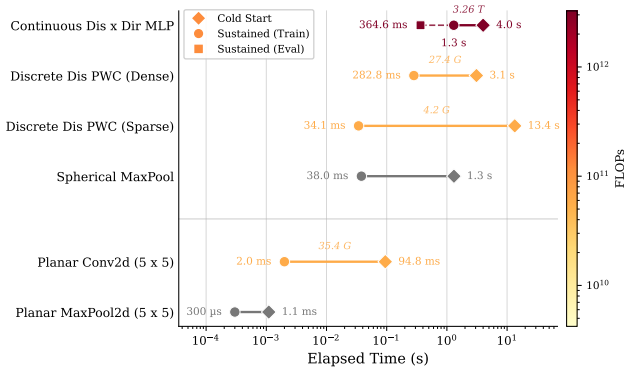


Figure 10. **Spherical Convolution and Pooling Benchmarks.** Sustained inference benefits from weight caching and reduces spherical convolution to matrix multiplication.

ily optimized CUDA kernels, whereas spherical operators are currently implemented using PyTorch scatter and sparse operations rather than custom fused CUDA kernels.

F.3. Network-Level Comparison

Finally, we compare full network runtime between planar and spherical versions of YOLOv11, DeepLab v3, and UNet using identical macro-architectures, input resolution, batch size, and channel configurations. The only difference between models is the replacement of planar convolution and pooling layers with spherical counterparts, making the comparison strictly architecture-level and apples-to-apples.

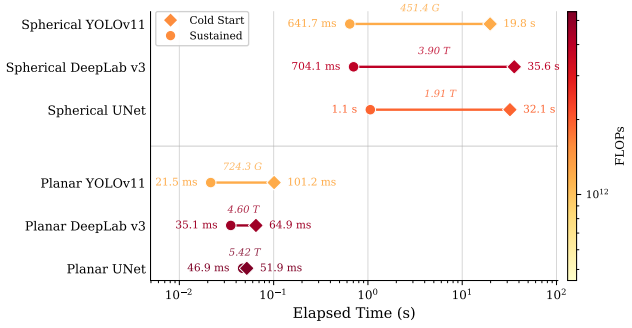


Figure 11. **End-to-End Network-Level Comparison Benchmarks.** Spherical networks achieve comparable runtime while requiring fewer FLOPs but currently lack fully optimized CUDA kernels.

Figure 11 shows that cold-start runtime for spherical networks is significantly higher due to geometry construction and kernel initialization. However, in sustained execution, spherical networks become much closer to planar networks in runtime. Across models, spherical networks take approximately **2× the overall time** of their planar counterparts in full-scale training, which we consider acceptable given the additional geometric processing and built-in rotation-equivariance.

Interestingly, the spherical networks often require **fewer FLOPs** than the planar counterparts while still taking longer in wall-clock time. This phenomenon is common in custom operator research: theoretical arithmetic complexity (FLOPs) does not directly translate to runtime when standard operators benefit from highly optimized low-level CUDA kernels, while custom operators rely on higher-level sparse and scatter operations. Further performance gains could be achieved by implementing dedicated CUDA kernels for spherical sampling and convolution.

Overall, these benchmarks demonstrate that **geometry caching is the key enabler** that makes high-resolution spherical processing practical. Without caching, the cost of geometric preprocessing would dominate runtime and make spherical pipelines infeasible for large-scale vision tasks. With caching, however, sustained runtime becomes comparable to planar networks while providing additional geometric consistency, rotation-equivariance, and lens-agnostic processing capabilities.

G. Common Training Setup

If not otherwise mentioned in the main paper, all the full-scale experiments share the same configurations as follows:

- Icosahedron location sampler for both the resampling stage and all the intermediate output locations of spherical convolution and pooling layers.
- *AdamW* optimizer with learning rate 1×10^{-3}, weight decay 1×10^{-2}, learning rate scheduler warms up at the first 40% steps and drops to 1×10^{-2} of the learning rate at the end.
- Batch size of 8 with 2 distributed data parallel (DDP), trained for 200 epochs on NVIDIA A100 and H200 GPUs.
- Data augmentation consists of 50% chance of random chroma, luma jitter, gaussian blur, 5% chance of grayscaling, and 50% chance of random horizontal reflection.

H. Backbone Architecture

We applied our framework to three representative backbone architectures: YOLOv11 (Fig. 6), UNet (Fig. 7), and DeepLab v3 (Fig. 12).

We incorporate attention bias as positional encoding in the C2PSA self-attention layers with pairwise Euclidean distance in the planar domain and geodesic distance on the sphere, analogous to the mechanism in ALiBi[31].

Note that atrous (dilated) convolution is not defined for our generic spherical CNN, so we use larger kernels instead in ASPP block of spherical DeepLab v3.

Each downsampling layer in the spherical model reduces the number of output points to $\frac{1}{4}$ of the input, analogous to

Train \ Test		Pinhole		Fisheye		Panoramic	
		mIoU ↑	mAcc ↑	mIoU ↑	mAcc ↑	mIoU ↑	mAcc ↑
Planar DeepLab v3[6]	Pinhole	42.53%	55.75%	33.53%	47.36%	36.07%	53.61%
	Fisheye	39.88%	53.46%	40.05%	55.86%	33.11%	56.53%
	Panoramic	29.66%	43.85%	24.91%	37.54%	35.01%	58.30%
Spherical DeepLab v3	Pinhole	34.76%	47.47%	22.36%	35.52%	19.70%	35.09%
	Fisheye	19.44%	31.52%	30.44%	44.21%	28.16%	43.99%
	Panoramic	12.57%	23.05%	28.35%	41.58%	28.78%	45.27%

Table 7. **Semantic Segmentation Full-dataset Lens Adaptability Test.** Random Rotation is disabled.

stride-2 downsampling in planar CNNs. Because location sampling is not invertible with respect to resolution factors, we align the output locations of spherical CNN upsampling layers with corresponding downsampling layers to ensure valid channel concatenations.

I. Object Detection

The loss function includes focal loss for category classification and L1 loss for bounding box center (θ, ϕ), size (α, β), and angle γ regression. Formally, focal loss:

$$L_{\text{focal}} = \frac{1}{N}\sum_{i=1}^{N}[L_{\text{positive}} + L_{\text{negative}}], \tag{34}$$

$$L_{\text{positive}} = -\ln(p_i)(1-p_i)^{\lambda_{\text{positive}}} y_i, \tag{35}$$

$$L_{\text{negative}} = -\ln(1-p_i)p_i^{\lambda_{\text{positive}}}(1-y_i)^{\lambda_{\text{negative}}}. \tag{36}$$

Where p_i is the predicted probability, y_i is the ground truth heatmap, and $\lambda_{\text{positive}}, \lambda_{\text{negative}}$ are hyperparameter exponents controlling the relative focus on positive versus negative detections.

Efficient Pairwise IoU. We propose a generic and vectorized method to compute pairwise IoU between arbitrary-shaped bounding boxes. Each bounding box is converted into a binary mask over the same set of dense and uniformly sampled spherical points. The intersection and union are then approximated by applying logical AND and OR operations on these masks, with IoU estimated as the ratio between the number of points inside the intersection and those in the union.

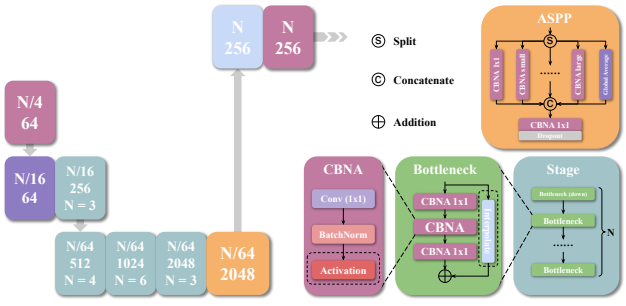


Figure 12. **DeepLab v3.** Core components include: **CBNA**: convolution, batch normalization, activation; **Stage**: N Bottleneck blocks; **ASPP**: atrous spatial pyramid pooling.

Finally, we apply Matrix Non-Maximum Suppression (NMS)[37] with a Gaussian decay factor $\sigma = 5.0$ and score threshold 0.3 to suppress overlapping but inconfident proposals.

As illustrated in Fig. 8, classification heatmap and bounding box regression are supervised with separate loss functions at different locations, and regression losses are only applied to points near the object's center.

Qualitative results are in Tab. 8 in addition to the quantitative results in the main paper. Observe the cyan boxes carefully for light patches on the ground in the NR and RR cases of Spherical YOLOv11, which exhibits the same orientation with or without random rotation. This behavior is expected because orientation is the raw output of one head, which should not change under rotation because of rotation-equivariance.

J. Semantic Segmentation

Training uses a composite loss of 70% cross-entropy and 30% Dice loss, with proper class weights and 0.05 label smoothing. Formally, Dice loss:

$$L_{\text{Dice}} = 1 - \frac{1}{|S|}\sum_{c\in S}\text{Dice}_c, \tag{37}$$

$$\text{Dice}_c = \frac{2\cdot\sum_{i\in\Omega} p_{i,c}\, y_{i,c}}{\sum_{i\in\Omega} p_{i,c} + \sum_{i\in\Omega} y_{i,c}}. \tag{38}$$

Where $p_{i,c}$ and $y_{i,c}$ denote the predicted probability and ground-truth label for class c at pixel i, Ω is the set of valid pixels, and S is the set of non-ignored classes in Ω.

We follow the official 3-fold cross-validation scheme as a benchmark on the Stanford-2D-3D-S dataset.

Random rotation qualitative results are visualized in Tab. 9 in addition to the quantitative results in the main paper. Zero-shot lens adaptability results trained on the full-scale dataset are shown in Tab. 7 and Tab. 10. Ideally, a perfect model that adapts to all lenses would have the same performance in all the entries of the square performance matrix. USF reduces off-diagonal performance degradations compared to planar models, but not yet completely eliminates them.

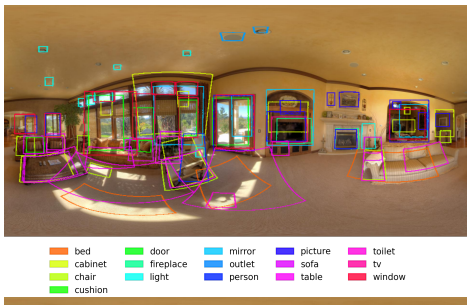
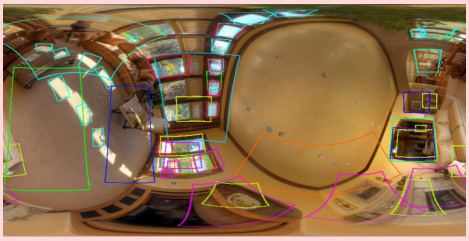
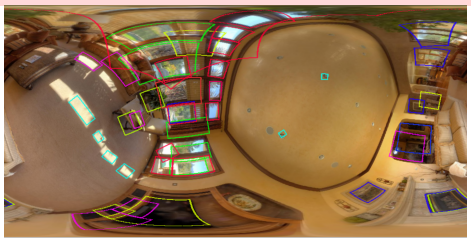
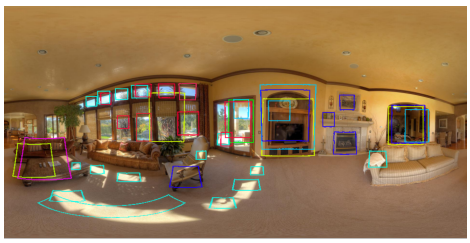
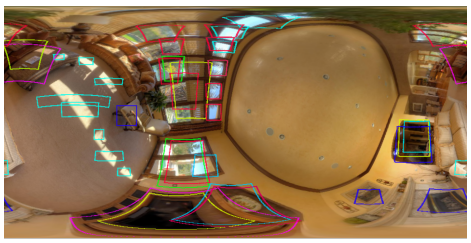
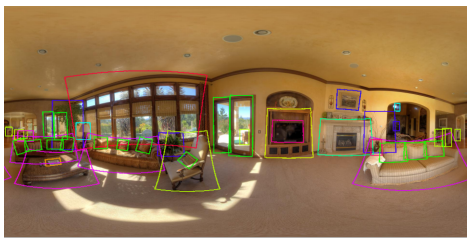
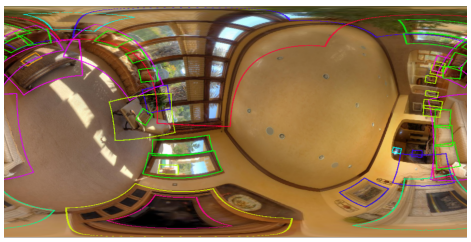
Train \ Test		NR	RR
Planar YOLOv11[25]	NR		
	RR		
Spherical YOLOv11	NR		
Ground Truth			

Table 8. **Object Detection Qualitative Result.**

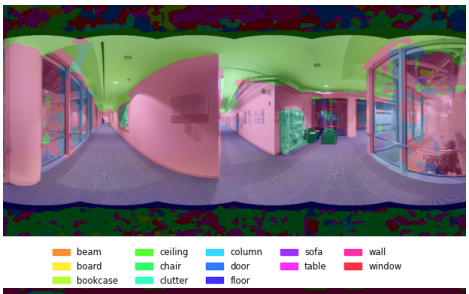
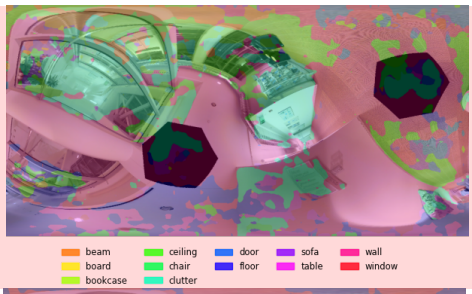
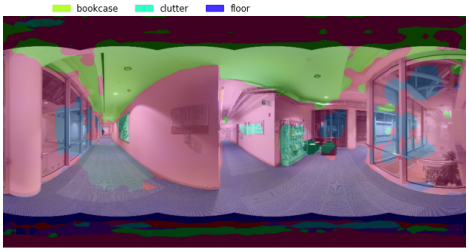
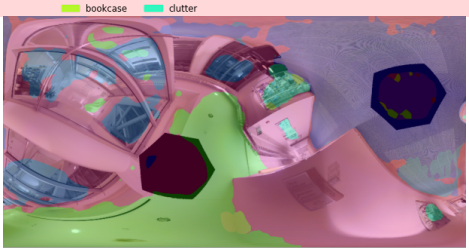
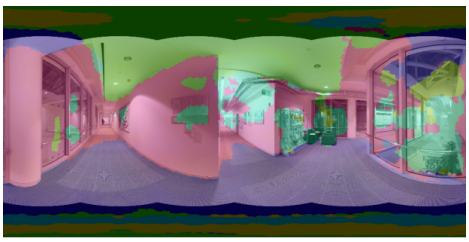
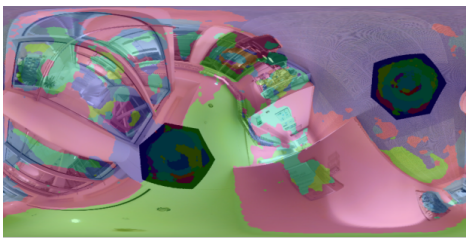
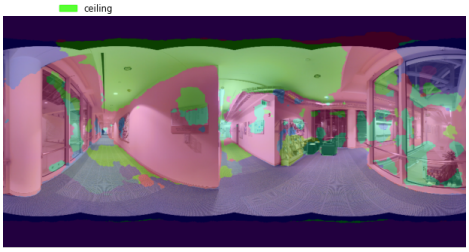
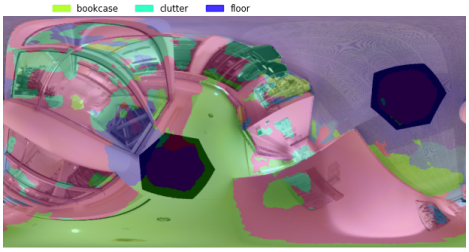
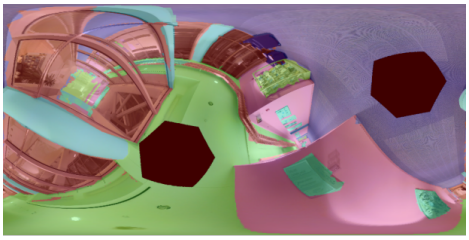
Train \ Test		NR	RR
Planar DeepLab v3[6]	NR		
	RR		
Spherical DeepLab v3	NR		
	RR		
Ground Truth			

Table 9. **Semantic Segmentation Qualitative Result.**

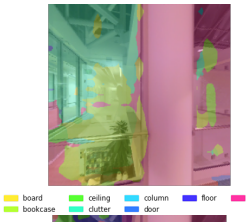
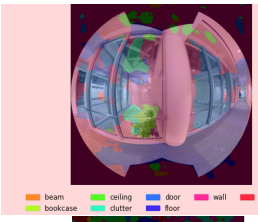
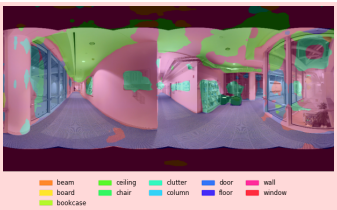
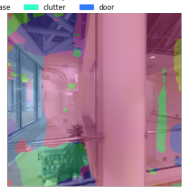
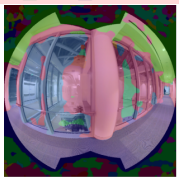
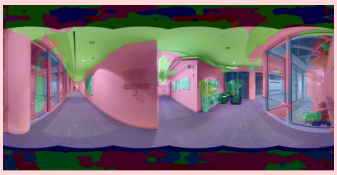
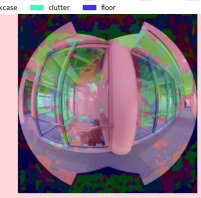
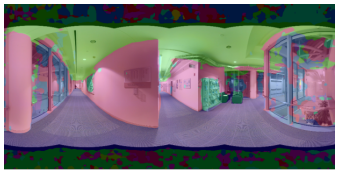
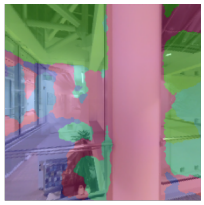
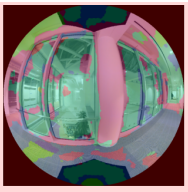
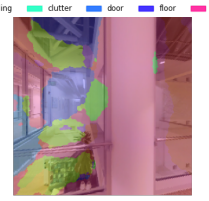
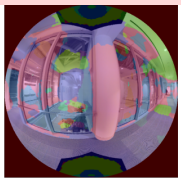
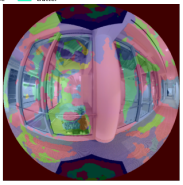
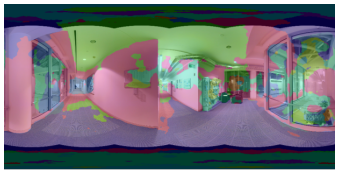
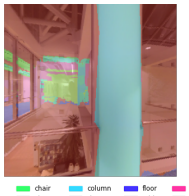
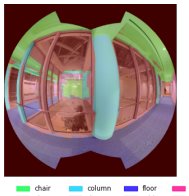
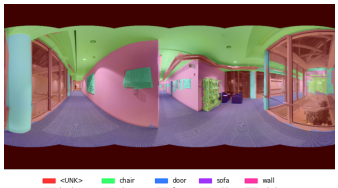
Train \ Test		Pinhole	Fisheye	Panoramic
Planar DeepLab v3[6]	Pinhole			
	Fisheye			
	Panoramic			
Spherical DeepLab v3	Pinhole			
	Fisheye			
	Panoramic			
Ground Truth				

Table 10. **Semantic Segmentation Full-dataset Lens Adaptability Test.** Random Rotation is disabled.