

EECE 576L

Advanced Topics in Deep Learning

Lecture 4: Group Equivariant Deep Learning

Part 4: Typed features and $SO(3)$ kernels

Renjie Liao

University of British Columbia

2026/27 Winter Session, Term 1 (Fall 2026)

Outline

1. Groups and actions
2. Group convolutions
3. Steerable kernels
4. Typed features and $SO(3)$ kernels
 - **Typed feature fields**
 - Kernel constraint and concrete kernels
 - Nonlinearities and trainable layers
 - $SO(3)$ feature types and angular kernels
5. Implementation and verification (optional reading)

Steerable Kernels and Typed Feature Fields

Part 3: a fixed basis expresses the rotated filter.

$$k_{\alpha}(x) = k(R_{-\alpha}x) = \sum_q w_q(\alpha) \psi_q(x)$$

Part 4: feature channels follow a prescribed geometric rule.

$$s \mapsto s, \quad v \mapsto hv, \quad A \mapsto hAh^{\top}$$

A layer must respect the transformation rules of both input and output.

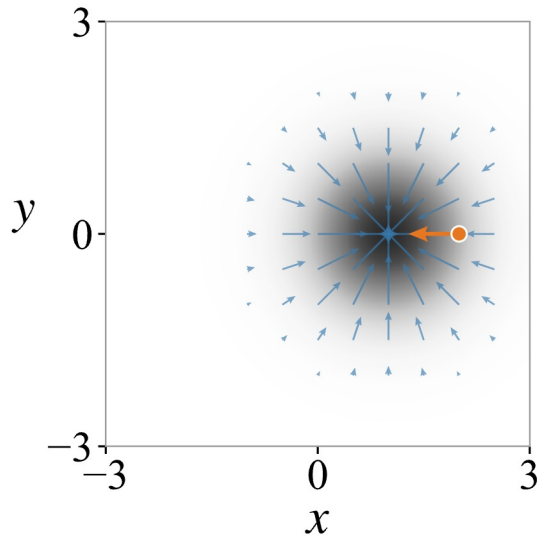
This constrains its kernels, nonlinearities, and pooling.

Directional Features Rotate Together

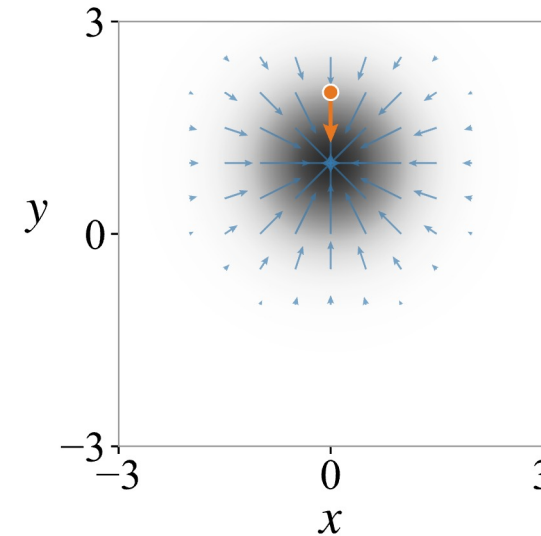
Recall Part 3: a gradient stores two coupled feature components per pixel.

Rotating the image moves the pixels and rotates their gradient vectors.

Original image and gradient



90° counterclockwise rotation



$$\nabla[L_{\beta}f](u) = R_{\beta}[\nabla f](R_{-\beta}u)$$

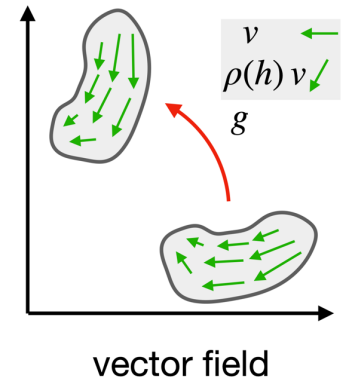
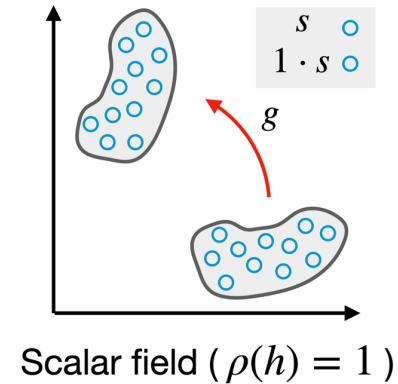
Part 4 chooses channel rules, then builds layers that respect them.

Feature values have geometric types

Three scalar channels and one 3D vector have the same array size.

Scalars retain their values at the moved location.

A vector also rotates its components together.



Spatial Matrices and Channel Representations

The spatial matrix acts differently on scalar, vector, and tensor values.

$$h \in H \leq O(d), \quad h \in \mathbb{R}^{d \times d}, \quad h^{-1} = h^\top$$

$$\rho : H \rightarrow \text{GL}(V), \quad \rho(h_1 h_2) = \rho(h_1) \rho(h_2)$$

Scalar

$$s \in \mathbb{R}, \quad \rho(h)s = s$$

Vector

$$v \in \mathbb{R}^d, \quad \rho(h)v = hv$$

Matrix tensor

$$A \in \mathbb{R}^{d \times d}, \quad \rho_{\text{tens}}(h)[A] = hAh^\top$$

Each tensor stores a matrix at one location, e.g. local covariance.

The transformed tensor keeps the same shape.

Tensor Values and Flattened Coordinates

Stack the tensor's columns to store its entries as a channel vector.

$$\text{vec}(hAh^\top) = \underbrace{(h \otimes h)}_{\rho_{\text{vec}}(h)} \text{vec}(A)$$

$$\text{vec}(A) \in \mathbb{R}^{d^2}, \quad \rho_{\text{vec}}(h) \in \mathbb{R}^{d^2 \times d^2}$$

Example: a 2D quarter-turn exchanges the tensor's principal axes.

$$h = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad hAh^\top = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

$$(h \otimes h)(2, 0, 0, 1)^\top = (1, 0, 0, 2)^\top$$

The larger matrix transforms the entries. Reshaping returns a tensor.

The action on a typed field

Use translations and a closed subgroup of orthogonal transformations.

$$H \leq O(d), \quad E = \mathbb{R}^d \rtimes H, \quad f : \mathbb{R}^d \rightarrow V$$

$$[\pi_\rho(t, h)f](x) = \rho(h)f(h^{-1}(x - t))$$

Inverse lookup retrieves the old location.

The leading matrix transforms the channel value.

Here the subscripted field operator differs from Part 1's orientation projection.

Outline

1. Groups and actions
2. Group convolutions
3. Steerable kernels
4. Typed features and $SO(3)$ kernels
 - Typed feature fields
 - **Kernel constraint and concrete kernels**
 - Nonlinearities and trainable layers
 - $SO(3)$ feature types and angular kernels
5. Implementation and verification (optional reading)

Spatial Convolution between Feature Types

Each neighbour contributes to the output through a shared matrix kernel.

$$[C_K f](x) = \int_{\mathbb{R}^d} K(x - y) f(y) dy$$

$$K(z) \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}, \quad z = x - y$$

Rows count output components. Columns count input components.

Scalar to scalar

$$K(z) \in \mathbb{R}^{1 \times 1}$$

Scalar to 3D vector

$$K(z) \in \mathbb{R}^{3 \times 1}$$

Sum the mapped neighbour contributions over space.

Using the displacement alone already gives translation equivariance.

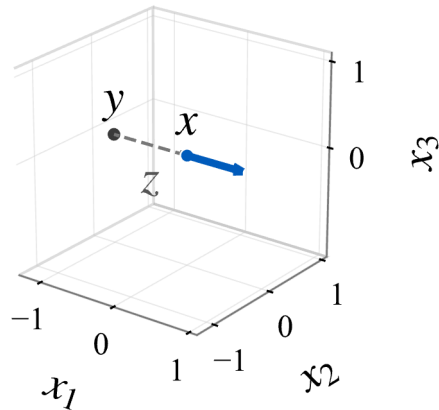
Scalar-to-Vector Convolution in 3D

The neighbour has a 3D position and a scalar input value.

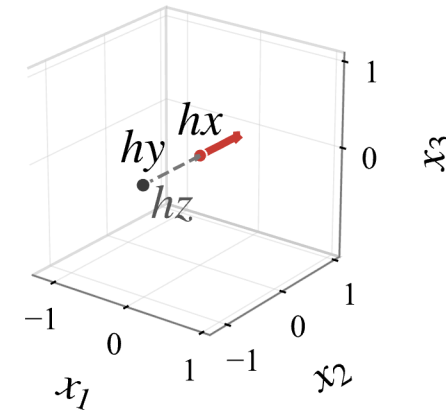
The kernel maps that scalar to a 3D output contribution.

$$s(y) = 2, \quad z = x - y, \quad K(z) = e^{-\|z\|^2} z \in \mathbb{R}^{3 \times 1}$$

Original contribution



90° counterclockwise rotation



$$\rho_{\text{in}}(h) = 1, \quad \rho_{\text{out}}(h) = h$$

$$K(hz) \cdot 2 = hK(z) \cdot 2 = (0, 2e^{-1}, 0)^\top$$

The Necessary and Sufficient Kernel Rule

Assume continuous integrable kernels and continuous channel representations.
Require equivariance on all continuous compactly supported input fields.

$$C_K \pi_{\text{in}}(t, h) = \pi_{\text{out}}(t, h) C_K$$

The kernel rule matches each rotated neighbour contribution:

$$K(hz) \rho_{\text{in}}(h) v = \rho_{\text{out}}(h) K(z) v$$

For every input value, this is equivalent to:

$$\boxed{K(hz) = \rho_{\text{out}}(h) K(z) \rho_{\text{in}}(h)^{-1}}$$

$$\forall h \in H, \quad \forall z \in \mathbb{R}^d$$

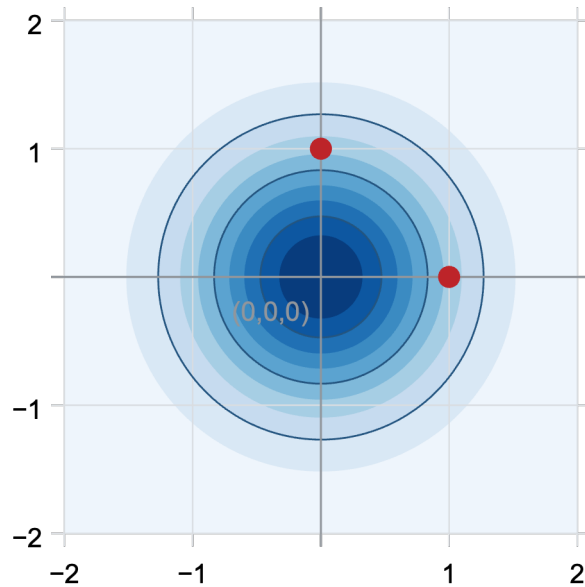
If each neighbour contribution transforms correctly, the integral does too.

Scalar-to-Scalar Radial Kernels

Input and output store one scalar value at each location.

$$z \in \mathbb{R}^3, \quad h \in \text{SO}(3) \subset \mathbb{R}^{3 \times 3}, \quad K(z) \in \mathbb{R}^{1 \times 1}$$

$$K(hz) = K(z), \quad K(z) = a(r), \quad r = \|z\|$$



Planar slice.

Rotations connect every direction at one radius.

Equal-length displacements receive equal scalar weights.

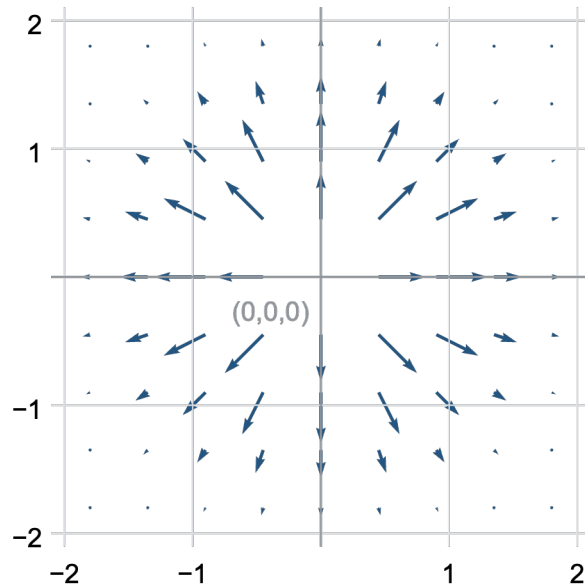
$$a(r) = e^{-r^2}$$

Scalar-to-Vector Radial Kernels

A scalar input contributes to a three-component vector output.

$$z \in \mathbb{R}^3, \quad h \in \text{SO}(3) \subset \mathbb{R}^{3 \times 3}, \quad K(z) \in \mathbb{R}^{3 \times 1}$$

$$K(hz) = hK(z), \quad K(z) = a(r)u, \quad r = \|z\| > 0, \quad u = z/r$$



Planar slice.

Rotations about the displacement leave it fixed.
The kernel vector must lie along this direction.

$$K(z) = e^{-\|z\|^2} z$$

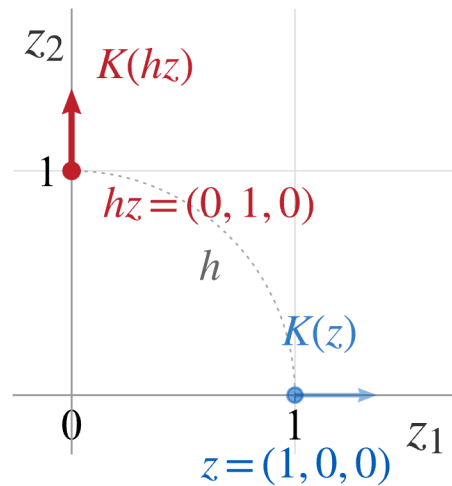
A continuous radial-arrow kernel vanishes at the origin.

The Kernel Rule at Two Evaluation Points

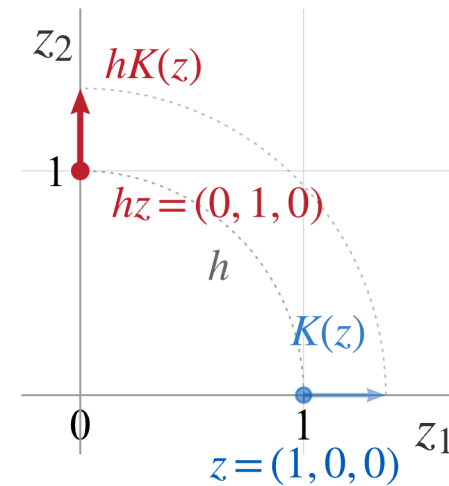
Let the spatial matrix be a quarter-turn about the third coordinate axis.

$$K(z) = e^{-\|z\|^2} z, \quad z = (1, 0, 0)^\top, \quad hz = (0, 1, 0)^\top$$

Rotate the location,
then evaluate the kernel



Rotate the original
location and kernel vector



$$K(hz) = hK(z) = (0, e^{-1}, 0)^\top$$

Outline

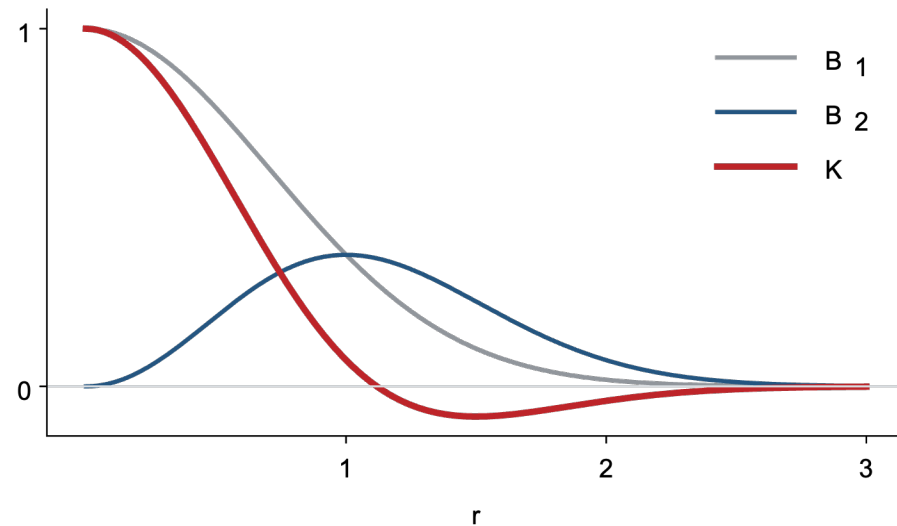
1. Groups and actions
2. Group convolutions
3. Steerable kernels
4. Typed features and $SO(3)$ kernels
 - Typed feature fields
 - Kernel constraint and concrete kernels
 - **Nonlinearities and trainable layers**
 - $SO(3)$ feature types and angular kernels
5. Implementation and verification (optional reading)

Trainable Kernels from a Fixed Basis

Scalar-to-scalar example: Gaussian-weighted radial monomials.

$$B_q(z) = r^{2(q-1)}e^{-r^2}, \quad q = 1, \dots, Q, \quad r = \|z\|$$

$$K(z) = \sum_{q=1}^Q w_q B_q(z)$$



Chosen fixed functions

$$B_1(z) = e^{-r^2}$$

$$B_2(z) = r^2 e^{-r^2}$$

Learned mixture

$$K = B_1 - 0.8B_2$$

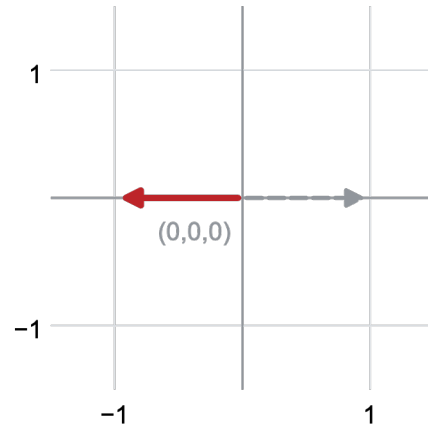
Every mixture remains radial and rotation equivariant.

Componentwise ReLU can break a vector type

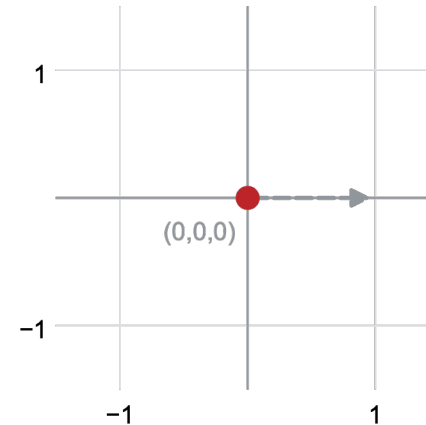
Activations must respect the channel transformation.

$$v = (1, 0, 0)^\top, \quad h = \text{diag}(-1, -1, 1), \quad hv = (-1, 0, 0)^\top$$

ReLU, then rotate



Rotate, then ReLU



$$h \text{ReLU}(v) = (-1, 0, 0)^\top \neq (0, 0, 0)^\top = \text{ReLU}(hv)$$

Use ReLU on invariant scalar channels and permutation channels.
Avoid componentwise ReLU on vector features.

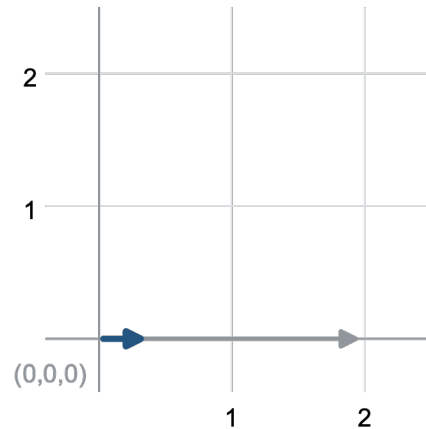
Norm Gating for Geometric Features

An alternative to componentwise ReLU is one scalar gate for the whole vector.

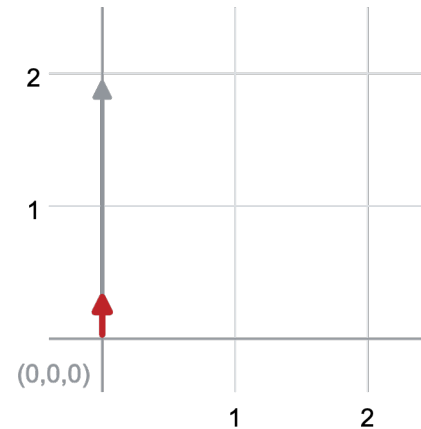
$$\sigma(v) = \gamma(\|v\|^2)v, \quad \|\rho(h)v\|^2 = \|v\|^2$$

Assume orthogonal channel actions. Both routes use the same gate.

Original vector



Rotated vector



$$\gamma(q) = \frac{1}{1+q}, \quad (2, 0, 0)^\top \mapsto (0.4, 0, 0)^\top$$

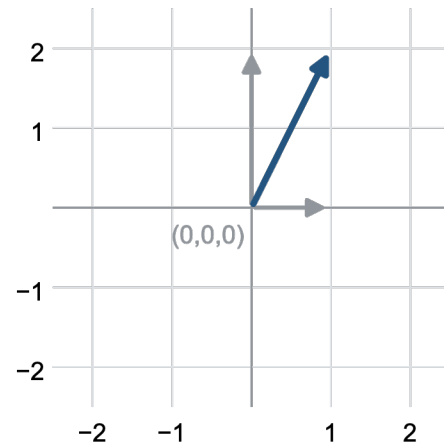
Pooling vectors keeps a vector output

Pool across a region that moves with the input.

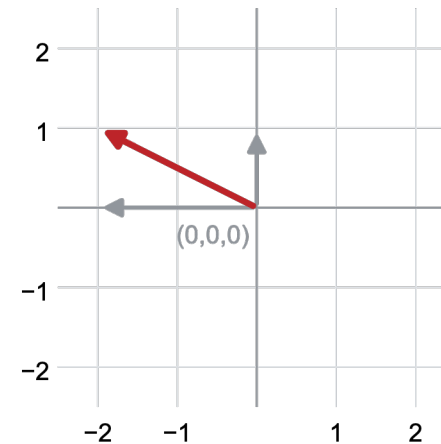
Each index labels one sample vector, not a vector component.

$$v_i \in \mathbb{R}^d, \quad i = 1, \dots, N, \quad \sum_{i=1}^N h v_i = h \sum_{i=1}^N v_i$$

Vector sum



Rotated vector sum



Sum or mean preserves the vector type.

Componentwise max generally breaks rotation equivariance.

Outline

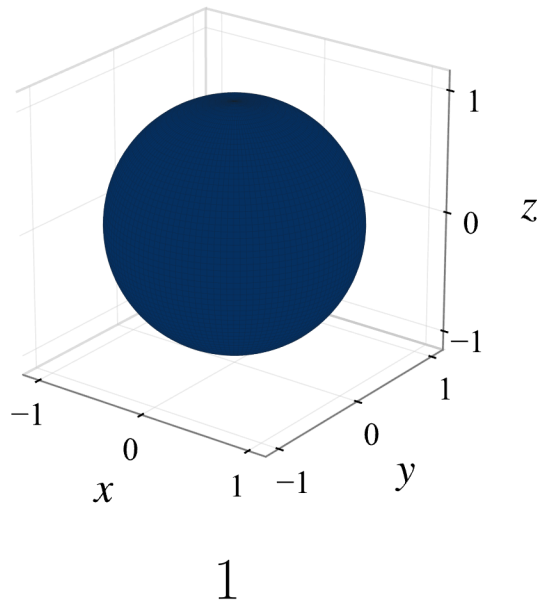
1. Groups and actions
2. Group convolutions
3. Steerable kernels
4. Typed features and $\text{SO}(3)$ kernels
 - Typed feature fields
 - Kernel constraint and concrete kernels
 - Nonlinearities and trainable layers
 - **$\text{SO}(3)$ feature types and angular kernels**
5. Implementation and verification (optional reading)

Scalar, Vector, and Quadrupole Degrees

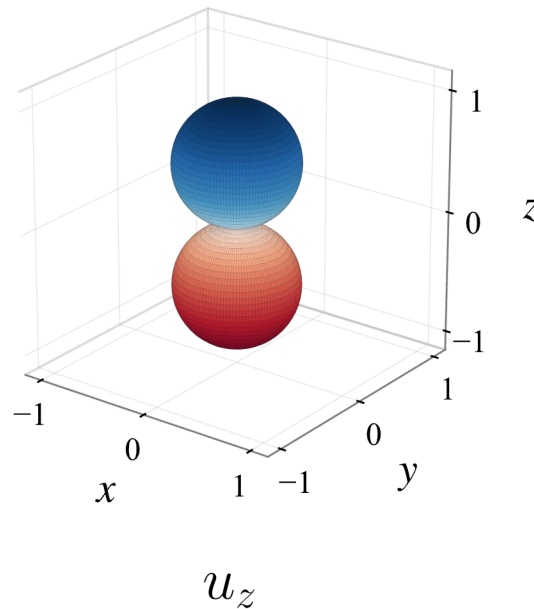
Each degree has a channel block. The plots show one harmonic per degree.

$$\dim V_\ell = 2\ell + 1$$

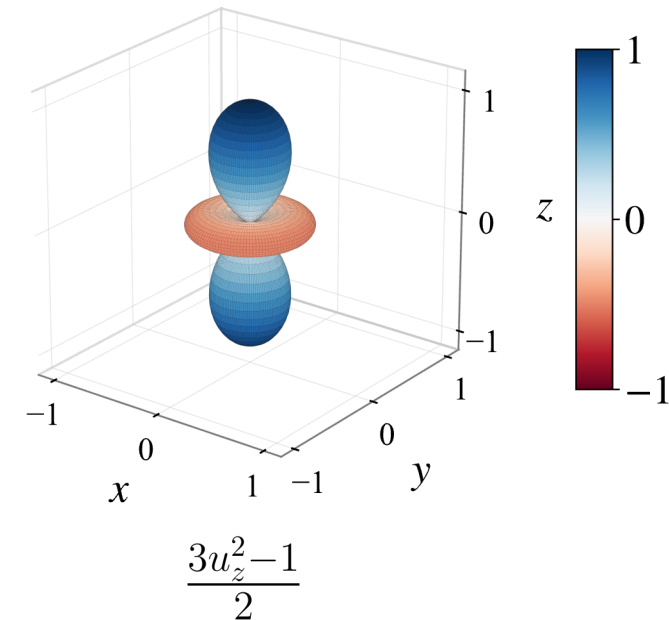
Degree 0: scalar



Degree 1: vector



Degree 2: tensor



3D polar plots: radius shows magnitude and color shows signed value.

Spherical Harmonics as Direction Features

A unit direction is mapped to a fixed vector of angular functions.

$$u \in S^2, \quad \ell = 0, 1, 2, \dots, \quad Y^\ell(u) \in \mathbb{R}^{2\ell+1}$$

Spatial rotations mix these functions by a channel matrix.

$$Y^\ell(hu) = D^\ell(h)Y^\ell(u)$$

$$h \in \text{SO}(3) \subset \mathbb{R}^{3 \times 3}, \quad D^\ell(h) \in \mathbb{R}^{(2\ell+1) \times (2\ell+1)}$$

Examples in convenient fixed normalizations:

Degree 0: constant

$$Y^0(u) = 1, \quad D^0(h) = 1$$

Degree 1: direction vector

$$Y^1(u) = u, \quad D^1(h) = h$$

The next slide makes the five degree-two components explicit.

Degree Two as Five Tensor Coordinates

A symmetric traceless 3D tensor has five independent components.

$$Q(u) = uu^\top - \frac{1}{3}I_3, \quad Q^\top = Q, \quad \text{tr } Q = 0$$

Use orthonormal coordinates for two diagonal and three off-diagonal terms.

$$Y^2(u) = \begin{bmatrix} (u_x^2 - u_y^2)/\sqrt{2} \\ (2u_z^2 - u_x^2 - u_y^2)/\sqrt{6} \\ \sqrt{2}u_xu_y \\ \sqrt{2}u_xu_z \\ \sqrt{2}u_yu_z \end{bmatrix}$$

$$Q(hu) = hQ(u)h^\top$$

$$Y^2(hu) = D^2(h)Y^2(u)$$

The full tensor has nine entries.
Symmetry and zero trace leave five.

$$Y^2(u) \in \mathbb{R}^5$$

$$D^2(h) \in \mathbb{R}^{5 \times 5}$$

Opposite directions give the same tensor. This map describes an unsigned axis.

Harmonic Degrees Define Feature Types

A 90° rotation about the third axis acts on vector and tensor features.

$$v_\ell \in \mathbb{R}^{2\ell+1}, \quad v_\ell \mapsto D^\ell(h)v_\ell, \quad \rho(h) = D^\ell(h)$$

Three vector components

$$D^1(h) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Five tensor coordinates

$$D^2(h) = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Higher degrees: seven functions at degree 3, nine at degree 4.

General basis derivation: Note 4, spherical-harmonics appendix.

Part 3 rotates the function basis using inverse lookup:

$$[L_h Y^\ell](u) = Y^\ell(h^{-1}u) = D^\ell(h)^{-1} Y^\ell(u)$$

Spherical Harmonics in a Steerable Kernel

For a scalar input and a degree-specific output, combine radius and direction.

$$\rho_{\text{in}}(h) = 1, \quad \rho_{\text{out}}(h) = D^\ell(h), \quad K(z) \in \mathbb{R}^{(2\ell+1) \times 1}$$

$$K(z) = a(r)Y^\ell(u), \quad r = \|z\| > 0, \quad u = z/r$$

Rotations preserve distance and rotate the unit direction.

$$\|hz\| = r, \quad \frac{hz}{\|hz\|} = hu$$

The harmonic rotation law gives the required kernel transformation.

$$K(hz) = a(r)Y^\ell(hu) = D^\ell(h)K(z)$$

Degree zero gives radial weights. Degree one gives the radial arrows.

A Basis for General Typed Kernels

Vector and tensor inputs require matrix-valued basis kernels.

$$B_q(hz) = \rho_{\text{out}}(h)B_q(z)\rho_{\text{in}}(h)^{-1}$$

- Use radial profiles and spherical harmonics for geometry.
- Couple geometry to the input and output feature types.
- Learn coefficients of these fixed compatible kernels.

Typed layers preserve geometric meaning as the 3D input rotates.

Harmonic properties and coupling: Note 4, Section 4.6.
Point-cloud algorithms remain in Note 4, Section 4.7.

Questions?

Typed features and $SO(3)$ steerable kernels

Implementation and verification: optional Note 5.

References

- [1] Bekkers. Group Equivariant Deep Learning, Lectures 2.4, 2.5, 3.3. University of Amsterdam.
- [2] Weiler & Cesa (2019). General $E(2)$ -Equivariant Steerable CNNs. NeurIPS.
- [3] Weiler et al. (2018). 3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data. NeurIPS.
- [4] Cohen, Geiger & Weiler (2019). A General Theory of Equivariant CNNs on Homogeneous Spaces. NeurIPS.

References (continued)

- [5] Thomas et al. (2018). Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds.
- [6] e3nn documentation. Spherical Harmonics.
- [7] e3nn documentation. Tensor Product.
- [8] e3nn documentation. Norm-Based Activation.