

Part 4: Steerable Group Convolutions

Lecturer: Renjie Liao

Revised: Sep. 29, 2026

Note: *LaTeX template courtesy of UC Berkeley EECS dept.***Disclaimer:** *These notes were revised with assistance from GPT-6 Astra.*

The previous note constructed filters whose rotations can be expressed through a small basis. We now use this idea to build layers whose *feature channels themselves* have a prescribed geometric meaning. A temperature is a scalar; a velocity is a vector; a local orientation pattern may require a higher-order tensor. Treating all of these as unrelated scalar channels discards useful structure.

The practical goal is to choose feature types, construct compatible kernels, and implement the resulting layer with familiar tensor operations. The main theoretical result is a necessary and sufficient condition on the kernel. No background in abstract representation theory is assumed beyond matrices and change of basis.

4.1 Two things change when we rotate a feature map

Suppose a point at $x \in \mathbb{R}^d$ moves to $x' = hx + t$, where h is an orthogonal matrix and t is a translation. At the new point, a scalar retains its value, but a vector also rotates:

$$s'(hx + t) = s(x), \quad v'(hx + t) = hv(x). \quad (4.1)$$

For example, rotating a velocity field moves each arrow's base and rotates the arrow. Rotating only the grid locations gives the wrong transformed field. Three ordinary scalar channels and one three-dimensional vector therefore have the same array size but different transformation rules.

A feature type is a channel transformation rule. A *representation* ρ assigns an invertible matrix $\rho(h)$ to each transformation h , with

$$\rho(h_1 h_2) = \rho(h_1) \rho(h_2), \quad \rho(e) = I. \quad (4.2)$$

This ensures that transforming twice agrees with applying the composed transformation. A field $f : \mathbb{R}^d \rightarrow V$ stores a vector of channels in V at every location. Its type is the representation acting on those channels.

Feature	Value at one location	Channel transformation
Scalar	$s \in \mathbb{R}$	$s \mapsto s$
d -dimensional vector	$v \in \mathbb{R}^d$	$v \mapsto hv$
Matrix-valued tensor	$A \in \mathbb{R}^{d \times d}$	$A \mapsto hAh^\top$
m scalar channels	$s \in \mathbb{R}^m$	$s \mapsto s$
m vector channels	$(v_1, \dots, v_m)$	$(hv_1, \dots, hv_m)$

The word *tensor* here describes a transformation law, not merely a multidimensional array in a deep learning framework.

The field action. Let H be a closed subgroup of $O(d)$ and let $E = \mathbb{R}^d \rtimes H$ be the group of translations and these orthogonal transformations. Its multiplication is $(t_1, h_1)(t_2, h_2) = (t_1 + h_1 t_2, h_1 h_2)$. Rewriting (4.1) at a fixed output coordinate gives

$$[\pi_\rho(t, h)f](x) = \rho(h) f(h^{-1}(x - t)). \quad (4.3)$$

The inverse in the argument retrieves the old location; the matrix in front transforms its feature value. We use an *active* convention: the object moves while the coordinate axes remain fixed. A passive change of coordinates uses inverse transformations instead.

4.1.1 Irreducible types and multiplicities

A representation is *reducible* if a nonzero proper subspace is preserved by every $\rho(h)$; otherwise it is *irreducible*. For example, a block containing one scalar and one vector is reducible: rotating the vector never changes the scalar. The standard three-dimensional vector representation of $SO(3)$ is irreducible: no fixed line or plane is preserved by *all* three-dimensional rotations.

Finite-dimensional continuous representations of compact groups can be decomposed into irreducible blocks after a change of basis. Here H is compact because it is a closed subgroup of $O(d)$. We can write

$$V = \bigoplus_{\alpha} (\mathbb{R}^{m_{\alpha}} \otimes V_{\alpha}), \quad \rho(h) = \bigoplus_{\alpha} (I_{m_{\alpha}} \otimes \rho_{\alpha}(h)). \quad (4.4)$$

V_{α} is one irreducible type; m_{α} is its *multiplicity*, meaning the number of independently learned copies. The total channel count is $\sum_{\alpha} m_{\alpha} \dim V_{\alpha}$. Rotations mix components within each copy, whereas learned layers can mix compatible copies.

Real versus complex coordinates. Irreducibility depends on the scalar field. For example, the usual two-dimensional rotation type is irreducible over $\mathbb{R}$, but splits into two one-dimensional complex types. In this note the network stores real features. The $SO(3)$ types of degree $\ell = 0, 1, 2, \dots$ have real realizations of dimension $2\ell + 1$. Complex spherical harmonics are an alternative coordinate system, not a requirement to train a complex-valued network. Representation matrices, harmonic functions, and coupling coefficients must use a consistent basis.

4.2 The kernel constraint: the key result

Consider the matrix-valued spatial convolution

$$[C_K f](x) = \int_{\mathbb{R}^d} K(x - y) f(y) dy, \quad K(z) \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}. \quad (4.5)$$

The input and output types may differ. For example, a layer can map a scalar density to a vector field. Sharing a kernel over displacement $x - y$ already gives translation equivariance. Rotation equivariance imposes an additional constraint.

Theorem 4.1 (Equivariance of a spatial convolution) *Let K be a continuous, integrable matrix-valued kernel, and let the input and output carry continuous representations ρ_{in} and ρ_{out} of $H \subseteq O(d)$. Then, on all continuous compactly supported input fields,*

$$C_K \pi_{\text{in}}(t, h) = \pi_{\text{out}}(t, h) C_K \quad \text{for every } (t, h) \in E \quad (4.6)$$

if and only if

$$\boxed{K(hz) = \rho_{\text{out}}(h) K(z) \rho_{\text{in}}(h)^{-1} \quad \text{for every } h \in H, z \in \mathbb{R}^d.} \quad (4.7)$$

Proof: Evaluate both possible orders of operations at $x = hq + t$. In the first, substitute $y = hu + t$; its Jacobian has absolute determinant one:

$$[C_K \pi_{\text{in}}(t, h) f](hq + t) = \int K(h(q - u)) \rho_{\text{in}}(h) f(u) du, \quad (4.8)$$

$$[\pi_{\text{out}}(t, h) C_K f](hq + t) = \int \rho_{\text{out}}(h) K(q - u) f(u) du. \quad (4.9)$$

The kernel constraint makes these integrands equal, proving sufficiency. Conversely, equality of the integrals for every compactly supported test field implies equality of their continuous matrix-valued coefficients. Setting $z = q - u$ gives $K(hz) \rho_{\text{in}}(h) = \rho_{\text{out}}(h) K(z)$, which is exactly (4.7). ■

What the theorem buys us. We can enforce equivariance by constructing the kernel once, rather than checking a network on every possible transformed input. This is the basis of steerable CNNs [1, 2]. The theorem characterizes the convolution operator in (4.5); it does not claim that every abstract linear operator has a continuous integrable kernel. Distributional kernels, including the identity operator, require a broader functional-analytic formulation.

Kernel displacement in a framework call. Many libraries compute $\sum_z A(z)f(x+z)$ rather than $\sum_z K(z)f(x-z)$. They agree when $A(z) = K(-z)$. Both conventions obey the same steerability constraint, but the sign matters for directional kernels such as scalar-to-vector maps. Choose one displacement convention and use it consistently in kernel sampling, message passing, and tests.

4.3 Concrete scalar and vector kernels

Take $H = SO(3)$, $z \neq 0$, $r = \|z\|$, and $u = z/r$. The following examples follow directly from (4.7).

Scalar to scalar: a radial kernel. Both representations equal 1, so $K(hz) = K(z)$. Rotations can map any direction to any other at a fixed radius. Therefore

$$K(z) = a(r). \quad (4.10)$$

This layer can learn which distances matter, but its scalar kernel cannot favor an absolute direction.

Scalar to vector: a radial arrow. Now $K(z)$ is a three-dimensional column and must satisfy $K(hz) = hK(z)$. Rotations around the axis u leave z fixed, so they must also leave $K(z)$ fixed. The only such vectors are parallel to u :

$$K(z) = a(r)u, \quad [C_K s](x) = \int a(\|x-y\|) \frac{x-y}{\|x-y\|} s(y) dy. \quad (4.11)$$

Each neighbor contributes an arrow along its displacement; the output sums these arrows. Equivariance forces $K(0) = 0$. For a continuous kernel, require $a(r) \rightarrow 0$ as $r \rightarrow 0$, or parameterize $K(z) = b(r)z$ with a bounded continuous b near zero.

Vector to vector: preserve, project, or turn. The general $SO(3)$ -equivariant kernel at nonzero displacement has the form

$$K(z) = a(r)I + b(r)uu^\top + c(r)[u]_\times, \quad [u]_\times v = u \times v. \quad (4.12)$$

Applied to an input vector, the three terms scale it, extract its component along the displacement, and turn its transverse component by a cross product. Arbitrary radial functions specify how these effects depend on distance. At the origin the kernel must be a multiple of I ; one sufficient continuity condition is that $a(r)$ has a limit and $b(r), c(r) \rightarrow 0$.

Why only these terms? Fix u . The kernel must commute with every rotation around u . On the transverse plane, a commuting map is a combination of identity and a 90° rotation; along u , it is a scalar. These are precisely the three operations in (4.12).

Reflections change the answer. For $O(3)$, distinguish a polar vector, which transforms as $v \mapsto hv$, from an axial vector, which transforms as $v \mapsto \det(h)hv$. The cross product of two polar vectors is axial. Thus the $c(r)[u]_\times$ term is forbidden for an $O(3)$ -equivariant *polar-to-polar* map; the first two terms remain valid. A rotation-only model cannot distinguish these parity types, so the intended symmetry group must be chosen before constructing a kernel.

4.4 Compatible nonlinearities, biases, and readouts

A valid linear layer is only one part of a network. A pointwise nonlinearity must satisfy

$$\sigma(\rho(h)v) = \rho(h)\sigma(v). \quad (4.13)$$

For invariant scalar channels, an ordinary ReLU is valid. For vector components, it generally is not. Let $v = (1, 0, 0)^\top$ and let h rotate by 180° around the third axis. Then $\text{ReLU}(hv) = 0$ but $h \text{ReLU}(v) = (-1, 0, 0)^\top$.

An implementation-friendly replacement. Choose orthogonal representation matrices, which is possible for the compact groups considered here. Then $\|\rho(h)v\|^2 = \|v\|^2$, so

$$\sigma(v) = \gamma(\|v\|^2) v \quad (4.14)$$

is equivariant for any scalar function γ . It changes the magnitude using invariant information while retaining the prescribed direction transformation. More generally, an invariant scalar gate g computed by the network can multiply a typed feature: $\sigma(v, g) = \gamma(g)v$. These formulas are well-defined at $v = 0$ and need no division by its norm. Tensor-product nonlinearities offer another option, introduced below.

Biases and feature mixing. A spatially constant additive bias must satisfy $\rho_{\text{out}}(h)b = b$ for every h . Arbitrary biases are allowed in trivial scalar blocks; a nontrivial irreducible vector block has only the zero invariant bias. A channel mixing matrix W is valid only if $W\rho_{\text{in}}(h) = \rho_{\text{out}}(h)W$. In a fixed real $SO(3)$ irrep basis, arbitrary learned mixing between copies of the same degree has form $A \otimes I_{2\ell+1}$.

Pooling does not automatically make vectors invariant. If $v_i \mapsto hv_i$, then $\sum_i v_i \mapsto h \sum_i v_i$: sum and mean pooling produce an equivariant *vector*, not an invariant prediction. An invariant graph-level scalar can instead pool scalar channels or invariants such as $\|v_i\|^2$. A vector-valued task, such as force prediction, should retain the vector output. Pooling spatial fields also requires transforming the domain or using an invariant domain.

4.5 From a constrained kernel to a trainable layer

The constraint (4.7) is linear in K . If $B_1, \dots, B_Q$ each satisfy it, so does

$$K(z) = \sum_{q=1}^Q w_q B_q(z). \quad (4.15)$$

The basis fixes the angular transformation law; gradient descent learns its coefficients. A finite radial basis or a finite angular cutoff selects a model subspace. Equivariance is retained, but an arbitrary continuous kernel need not lie in that finite subspace.

A concrete tensor layout. For one input type of dimension d_{in} with m_{in} copies and one output type of dimension d_{out} with m_{out} copies, sample the basis at P spatial offsets:

$$B \in \mathbb{R}^{P \times Q \times d_{\text{out}} \times d_{\text{in}}}, \quad (4.16)$$

$$W \in \mathbb{R}^{m_{\text{out}} \times m_{\text{in}} \times Q}, \quad (4.17)$$

$$K_{p,a,o,b,i} = \sum_q W_{a,b,q} B_{p,q,o,i}. \quad (4.18)$$

The materialized kernel has shape $[P, m_{\text{out}} d_{\text{out}}, m_{\text{in}} d_{\text{in}}]$. An explicit tensor contraction is

```
# B: [offset, basis, out_component, in_component]
# W: [out_copy, in_copy, basis]
K = einsum("pqoi,abq->paobi", B, W)
K = K.reshape(P, m_out * d_out, m_in * d_in)
```

Concatenate such blocks for the chosen input/output types. On a fixed grid, basis samples can be cached; during training only W changes. A standard convolution routine then consumes the assembled kernel after reordering its axes. Avoid detaching K from the differentiation graph during training.

Sampling is a separate issue. The continuous kernel can be exactly steerable while its sampled grid implementation is only approximately rotation equivariant. Arbitrary rotations do not preserve a cubic lattice; interpolation, truncation, stride, and boundary padding can introduce additional errors. Lattice-preserving transformations allow exact index permutations, subject to compatible boundary handling. Increasing sampling resolution is not a substitute for checking the actual implemented layer.

4.6 What the $SO(3)$ building blocks mean

For real $SO(3)$ irreducible types, the degree ℓ determines the number of components:

Degree	Dimension	Geometric interpretation
0	1	Scalar: unchanged by rotation.
1	3	Vector: components mix as a three-dimensional rotation.
2	5	Symmetric traceless matrix Q : $Q \mapsto hQh^T$; six symmetric entries minus one trace constraint.
ℓ	$2\ell + 1$	Higher-order angular information with a prescribed rotation law.

An $\ell = 2$ feature is not five unrelated scalar channels. Its five entries jointly describe a directional pattern, just as three vector components jointly describe an arrow.

4.6.1 Spherical harmonics encode the direction of an edge

Real spherical harmonics collect functions of a unit direction $u \in S^2$ into a vector $Y^\ell(u) \in \mathbb{R}^{2\ell+1}$. We choose conventions so that

$$Y^\ell(hu) = D^\ell(h)Y^\ell(u), \quad D^\ell(h)^T D^\ell(h) = I. \quad (4.19)$$

The degree-zero harmonic is constant. Degree one is proportional, up to basis ordering, to (u_x, u_y, u_z) . Degree two can be represented by the coordinates, in an orthonormal basis of symmetric traceless matrices, of $uu^T - I/3$. Higher degrees resolve increasingly detailed angular variation.

This agrees with Part 3's inverse lookup convention: $[L_h Y^\ell](u) = Y^\ell(h^{-1}u) = D^\ell(h)^{-1}Y^\ell(u)$. Consequently the coefficients of $c^T Y^\ell(u)$ transform as $c \mapsto D^\ell(h)c$, since D^ℓ is orthogonal. The five raw entries of a traceless matrix are not orthonormal coordinates; use an orthonormal basis when applying an ordinary Euclidean norm gate.

The same degree ℓ labels two closely related objects: a *learned feature vector* transforms by $D^\ell(h)$, whereas the *fixed function* Y^ℓ converts a direction into a vector transforming by that matrix. In a kernel, spherical harmonics supply geometry; learned radial coefficients decide how strongly to use it. This construction is used in tensor field networks and steerable CNNs [3, 1].

No hidden assumption on the signal. The full collection of spherical harmonics is a complete orthogonal basis for square-integrable scalar functions on S^2 . A finite cutoff $\ell \leq L$ is a band-limited approximation, not an assertion that every angular signal has finitely many modes. Also, the transformation of a harmonic *basis function* and that of an expansion's *coefficients* must not be mixed: inverse rotations and transposes depend on which object is being transformed.

4.6.2 Coupling two types with a tensor product

If a has type ℓ_1 and b has type ℓ_2 , their outer product has $(2\ell_1 + 1)(2\ell_2 + 1)$ entries. A fixed change of basis separates these into types

$$V_{\ell_1} \otimes V_{\ell_2} \cong \bigoplus_{\ell=|\ell_1-\ell_2|}^{\ell_1+\ell_2} V_{\ell}. \quad (4.20)$$

The change-of-basis entries are called Clebsch–Gordan coefficients. For implementation, regard them as fixed tensors that project an outer product into compatible output types.

For two ordinary vectors a, b , the familiar decomposition is

$$\underbrace{a^\top b}_{\ell=0, \text{ 1 component}}, \quad \underbrace{a \times b}_{\ell=1, \text{ 3 components}}, \quad \underbrace{\frac{ab^\top + ba^\top}{2} - \frac{a^\top b}{3} I}_{\ell=2, \text{ 5 components}}. \quad (4.21)$$

Normalization and a fixed basis change turn these expressions into orthonormal coupling coefficients. The dimensions add to $1 + 3 + 5 = 9$, exactly the size of the outer product. This example is the useful geometric meaning of the general decomposition.

The allowed kernel paths. An angular feature $Y^J(u)$ can couple with an input of degree ℓ_{in} to produce degree ℓ_{out} only when

$$|\ell_{\text{out}} - \ell_{\text{in}}| \leq J \leq \ell_{\text{out}} + \ell_{\text{in}}. \quad (4.22)$$

A basis kernel for such a path is

$$[B^J(u)]_{o,i} = \sum_{m=-J}^J C_{\ell_{\text{in}},i;\ell_{\text{out}},o}^{J,m} Y_m^J(u). \quad (4.23)$$

The fixed coupling tensor C ensures that B^J obeys the kernel constraint. Multiplying by radial functions and learning coefficients completes the parameterization. For scalar-to-vector maps, only $J = 1$ is allowed, recovering the radial arrow in (4.11). For vector-to-vector maps, $J = 0, 1, 2$ give three paths, consistent with (4.12).

For $O(3)$, each type also has a parity $p \in \{+1, -1\}$ under inversion. Natural degree- J harmonics have parity $(-1)^J$, so a path additionally requires $p_{\text{out}} = p_{\text{in}}(-1)^J$. Library implementations can construct these compatible paths without deriving special functions by hand [5, 4].

4.7 A point-cloud layer is a sparse steerable convolution

Let positions be $x_i \in \mathbb{R}^3$ and typed features be f_i . Choose neighbors by an invariant rule, such as a distance cutoff. With $z_{ij} = x_i - x_j$, define

$$f'_i = \sum_{j \in \mathcal{N}(i)} K(z_{ij}) f_j. \quad (4.24)$$

This is an equivariant message-passing layer: under a rigid motion, displacements become hz_{ij} , and (4.7) gives $K(hz_{ij})\rho_{\text{in}}(h)f_j = \rho_{\text{out}}(h)K(z_{ij})f_j$. Summing therefore produces the correct output type. Relabeling points also relabels the outputs if the same neighbor rule and weights are used everywhere.

Forward pass.

1. Build directed edges $j \rightarrow i$ and compute displacement, distance, and direction.
2. Evaluate real harmonics of each direction. Evaluate radial basis functions or a radial MLP on the distance.
3. Couple each input type with compatible angular types using fixed Clebsch–Gordan tensors. Learn the radial weights and mixing between copies.
4. Sum edge messages at each destination. Apply a compatible self-interaction and nonlinearity.

The radial MLP receives invariant inputs; feeding raw coordinate components to it would generally destroy the guarantee. Invariant scalar edge attributes, such as bond type, can also condition its weights. A normalized sum is valid when its normalization factor is invariant.

The simplest working example. Given scalar features s_j , a scalar-to-vector message can be implemented without any harmonic library:

```
# x: [nodes, 3]; s: [nodes, 1].
# Edges contain distinct points; src[j] sends to dst[j].
z = x[dst] - x[src]           # [edges, 3]
r2 = (z * z).sum(-1, keepdim=True) # [edges, 1]
weight = radial_mlp(r2)       # [edges, 1]
message = weight * z * s[src]  # [edges, 3]
v_out = scatter_sum(message, dst, dim=0, dim_size=num_nodes)
```

This uses $K(z) = b(\|z\|^2)z$, a nonsingular form of (4.11). The kernel is multiplied by the distance cutoff implicit in the edge set. A smooth cutoff can make the layer continuous when neighbors enter or leave the cutoff radius.

Zero displacement has no unit direction. Handle self-edges through a compatible linear map, or use a separately defined kernel value at zero. Radius or nearest-neighbor selection must be consistent under rotations and point permutations; ties and floating-point threshold decisions require care. A point-cloud sum is a learnable discrete operator, not automatically an accurate quadrature of a continuum integral.

A useful layer-level check. Rotate the coordinates by h , rotate every input block by its own $\rho_{\text{in}}(h)$, and run the layer again. Compare with rotating the original output by $\rho_{\text{out}}(h)$. Rotating coordinates alone is insufficient when input features contain vectors or higher types. With the same edges and exact point-coordinate rotations, the algebra above predicts equality up to floating-point error; a rasterized implementation has additional sampling error.

Appendix: where spherical harmonics come from

This derivation is useful for connecting harmonics to Fourier analysis, but is not needed to implement the layer. On the unit sphere, the Laplace operator is

$$\Delta_{S^2} = \frac{1}{\sin \theta} \frac{\partial}{\partial \theta} \left(\sin \theta \frac{\partial}{\partial \theta} \right) + \frac{1}{\sin^2 \theta} \frac{\partial^2}{\partial \phi^2}. \quad (4.25)$$

Its eigenfunctions satisfy $-\Delta_{S^2} Y = \lambda Y$. Write $Y(\theta, \phi) = \Theta(\theta)\Phi(\phi)$. Periodicity in ϕ gives $\Phi(\phi) = e^{im\phi}$ with integer m . Substituting $q = \cos \theta$ gives the associated Legendre equation

$$(1 - q^2)\Theta'' - 2q\Theta' + \left(\lambda - \frac{m^2}{1 - q^2} \right) \Theta = 0. \quad (4.26)$$

Regularity at the poles selects $\lambda = \ell(\ell + 1)$ with $\ell \in \{0, 1, 2, \dots\}$ and $|m| \leq \ell$. The corresponding complex functions have form

$$Y_\ell^m(\theta, \phi) = N_{\ell m} P_\ell^m(\cos \theta) e^{im\phi}, \quad (4.27)$$

where $N_{\ell m}$ normalizes the surface integral and P_ℓ^m is an associated Legendre function. Suitable real and imaginary combinations produce the real basis used above. Degree ℓ has $2\ell + 1$ independent functions because $m = -\ell, \dots, \ell$.

Rotations preserve the spherical Laplacian, so they preserve each degree's eigenspace. The matrix describing a rotation within that eigenspace is $D^\ell(h)$. Thus the special functions and the feature transformation matrices describe two sides of the same construction.

References

- [1] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen. *3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data*. NeurIPS, 2018. arXiv:1807.02547.
- [2] T. S. Cohen, M. Geiger, and M. Weiler. *A General Theory of Equivariant CNNs on Homogeneous Spaces*. NeurIPS, 2019. arXiv:1811.02017.
- [3] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. *Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds*. 2018. arXiv:1802.08219.
- [4] e3nn documentation. Spherical Harmonics. Accessed September 2026.
- [5] e3nn documentation. Tensor Product. Accessed September 2026.