

Part 3: Steerable Kernels

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.*

3.1 The implementation problem: one filter, many orientations

Suppose an edge detector should recognize the same edge at every orientation. Learning a separate filter for every angle wastes parameters; rotating a small pixel array by interpolation introduces errors. A **steerable filter** offers another route: represent the filter with a few fixed basis functions, then rotate its coefficients analytically. This is the classical steering idea [1]; learning the coefficients makes it useful in CNNs [2].

This note develops that construction for planar rotations. You need vectors, matrices, convolution, and the identities for sine and cosine. The main distinction to keep in mind is:

- **Steering a kernel** constructs its rotated copies.
- **An equivariant layer** additionally specifies how its input and output channels transform. A steerable kernel is a building block, not a guarantee for an otherwise arbitrary network.

Fix the rotation convention first. The group $SO(2)$ is the collection of planar rotations: angles add modulo 2π . Write R_α for a counterclockwise rotation by α . For a scalar image or kernel, define

$$[L_\alpha k](x) = k(R_{-\alpha}x), \quad [L_\alpha k](r, \theta) = k(r, \theta - \alpha). \quad (3.1)$$

The inverse is essential: to find the value at an output position, look up where it came from. In polar coordinates a rotation changes the angle, while leaving the radius unchanged. All signs below follow this convention.

3.2 A two-filter example before the general construction

Choose a radial envelope $q(r)$, and define two real filters

$$\psi_c(r, \theta) = q(r) \cos \theta, \quad \psi_s(r, \theta) = q(r) \sin \theta. \quad (3.2)$$

For example, $q(r) = r \exp[-r^2/(2\sigma^2)]$ gives filters proportional to $x \exp[-(x^2 + y^2)/(2\sigma^2)]$ and $y \exp[-(x^2 + y^2)/(2\sigma^2)]$: two perpendicular edge detectors. The factor r makes their values at the origin well defined.

The angle-subtraction identities give

$$\begin{bmatrix} L_\alpha \psi_c \\ L_\alpha \psi_s \end{bmatrix} = \underbrace{\begin{bmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{bmatrix}}_{D(\alpha)} \begin{bmatrix} \psi_c \\ \psi_s \end{bmatrix}. \quad (3.3)$$

In particular, $L_{\pi/2} \psi_c = \psi_s$. This is a useful sign check in code. If the learned filter is $k = a\psi_c + b\psi_s$, its rotated coefficients are

$$\begin{bmatrix} a_\alpha \\ b_\alpha \end{bmatrix} = D(\alpha)^\top \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}. \quad (3.4)$$

The transpose is not a second convention: basis functions and their coefficients occupy opposite sides of the dot product.

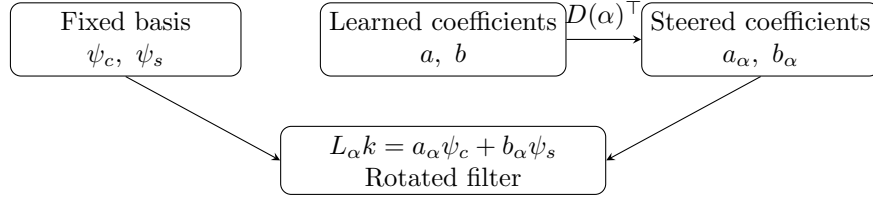


Figure 3.1: Rotation changes two coefficients; the spatial basis is reused. An arbitrary angle does not require resampling an already sampled filter.

Steer responses instead of filters. For the real-valued CNN convolution (with Part 2's convention) $[f \star k](x) = \int_{\mathbb{R}^2} f(x+u)k(u) du$, linearity gives

$$f \star (L_\alpha k) = a_\alpha (f \star \psi_c) + b_\alpha (f \star \psi_s). \quad (3.5)$$

Thus two basis responses suffice to recover every orientation of this particular filter family, before a nonlinearity. Moving a nonlinear activation inside this linear combination would generally change the result.

3.3 Circular harmonics and the finite basis we actually learn

The two-filter example extends by replacing θ with $m\theta$. For an integer angular frequency $m \geq 1$ and a radial index j , use

$$\psi_{jm}^c(r, \theta) = q_{jm}(r) \cos(m\theta), \quad \psi_{jm}^s(r, \theta) = q_{jm}(r) \sin(m\theta). \quad (3.6)$$

Keep both members of a pair with the *same* radial envelope. They mix by $D(m\alpha)$ under rotation. Frequency $m = 0$ contributes only $\psi_{j0}(r, \theta) = q_{j0}(r)$, which is unchanged by rotation. Frequency 1 captures a directional pattern; larger m permits more angular oscillations.

For a shorter derivation, combine cosine and sine into the complex **circular harmonic** $e^{im\theta}$:

$$L_\alpha(q(r)e^{im\theta}) = e^{-im\alpha} q(r)e^{im\theta}. \quad (3.7)$$

A spatial rotation becomes multiplication by a phase. Complex numbers are a convenient notation, not an implementation requirement; the real pairs above give the same construction using ordinary real tensors.

Theorem 3.1 (Exact steering in a finite invariant basis) *Let $\Psi(x) = (\psi_1(x), \dots, \psi_{n_b}(x))^\top$ be a finite collection of functions satisfying $\Psi(R_{-\alpha}x) = D(\alpha)\Psi(x)$ for every x and α . Then every filter $k_w(x) = w^\top \Psi(x)$ satisfies*

$$L_\alpha k_w(x) = (D(\alpha)^\top w)^\top \Psi(x). \quad (3.8)$$

Consequently, every learned choice of w can be steered exactly in the continuous domain without leaving this function space.

Proof: Substitute the assumed basis transformation: $k_w(R_{-\alpha}x) = w^\top D(\alpha)\Psi(x) = (D(\alpha)^\top w)^\top \Psi(x)$. ■

Learning radial profiles. With J radial functions for each $m = 0, \dots, M$, a practical real filter is

$$k(r, \theta) = \sum_{j=1}^J a_{j0} q_{j0}(r) + \sum_{m=1}^M \sum_{j=1}^J q_{jm}(r) [a_{jm} \cos(m\theta) + b_{jm} \sin(m\theta)]. \quad (3.9)$$

The envelopes are fixed; the coefficients a, b are trainable. Thus the learned radial profiles are linear combinations of fixed radial functions. There are $n_b = J(1 + 2M)$ real coefficients per input-channel/base-filter pair. Choosing radial functions is a modeling decision: compact radial bumps, Gaussian rings, or other smooth localized functions are possible.

One concrete radial family. For fixed widths $\sigma_j > 0$, one possible choice is

$$q_{jm}(r) = \left(\frac{r}{\sigma_j}\right)^m \exp\left[-\frac{r^2}{2\sigma_j^2}\right], \quad j = 1, \dots, J, \quad m = 0, \dots, M.$$

For example, choose $J = 3$ and $(\sigma_1, \sigma_2, \sigma_3) = (0.5, 1, 1.5)$ in the chosen spatial units. The index j selects a radial width; m selects an angular frequency. For $m = 0$, this is a radial Gaussian with $q_{j0}(0) = 1$; omit the identically zero sine term. For $m > 0$, define both harmonics to be zero at the origin. The factor r^m makes these Gaussian-weighted Cartesian harmonics smooth there. The widths and profiles remain fixed; learn only the expansion coefficients a, b .

What completeness means. For any $k \in L^2(\mathbb{R}^2)$, its angular Fourier expansion is

$$k(r, \theta) = \sum_{m \in \mathbb{Z}} c_m(r) e^{im\theta}, \quad c_m(r) = \frac{1}{2\pi} \int_0^{2\pi} k(r, \theta) e^{-im\theta} d\theta. \quad (3.10)$$

The equality is in L^2 , with planar measure $r dr d\theta$; slices are defined for almost every r . It need not be a pointwise identity for every pixel of an arbitrary square-integrable function. Real kernels satisfy $c_{-m}(r) = \overline{c_m(r)}$.

The *infinite* angular family is complete. A finite bandlimit $|m| \leq M$ omits higher angular frequencies, and a finite radial family adds a separate approximation. Neither is a complete basis for all image kernels. Nevertheless, the finite family in (3.9) is *closed under every rotation*: truncation limits expressiveness but does not itself spoil the exact continuous steering identity.

3.4 From the equations to a convolution layer

We first implement a **lifting layer**: scalar image channels in, orientation-indexed response channels out. Let C_{in} be the number of scalar input channels, C the number of learned base filters, A the number of sampled orientations, and s an odd kernel size.

3.4.1 Build the basis once

1. Choose M , radial envelopes, and centered Cartesian coordinates (u, v) on the $s \times s$ kernel grid. Use mathematical coordinates (u right, v up); array row indices normally increase downward. Compute $r = \sqrt{u^2 + v^2}$ and $\theta = \text{atan2}(v, u)$.
2. Evaluate (3.6) and the $m = 0$ terms. A continuous nonzero-frequency basis function must vanish at the origin; choose $q_{jm}(r) \rightarrow 0$ as $r \rightarrow 0$ for $m > 0$ and explicitly set those center entries to zero. Using $q_{jm}(r) = r^m h_j(r^2)$ with smooth h_j is one smooth choice. An arbitrary value returned by $\text{atan2}(0, 0)$ does not resolve this issue.
3. Prefer radial support or a radial taper that fits inside the grid. A square cutoff is not preserved by arbitrary rotations. Check sampled norms and rank: overly high frequencies or redundant radial functions can produce nearly zero or linearly dependent sampled filters.
4. Store the sampled basis $B \in \mathbb{R}^{n_b \times s \times s}$ as a buffer. Cache its analytically steered versions $B_{\ell d_{uv}} = \psi_d(R_{-\alpha_\ell}(u, v))$ for $\alpha_\ell = 2\pi\ell/A$.

If a basis pair is rescaled, use a common scale for its sine and cosine members to retain the simple rotation matrix. More general basis changes are allowed, but the steering matrices must be changed accordingly.

3.4.2 Learn coefficients, synthesize weights, call a standard kernel

The trainable tensor is $w \in \mathbb{R}^{C \times C_{\text{in}} \times n_b}$. The oriented convolution weights are

$$W_{cliuv} = \sum_{d=1}^{n_b} w_{cid} B_{lduv}, \quad W \in \mathbb{R}^{C \times A \times C_{\text{in}} \times s \times s}. \quad (3.11)$$

The following PyTorch-style forward pass performs this contraction and one ordinary `conv2d` call. Assume that `B` is a fixed buffer, `w` is a parameter, and `x` has shape `[N, Cin, H, W]`:

```
# B: [A, n_b, s, s]; w: [C, Cin, n_b]
filters = torch.einsum("cid,aduv->caiuv", w, B)
filters = filters.reshape(C * A, Cin, s, s)
y = torch.nn.functional.conv2d(x, filters, padding=s // 2)
y = y.reshape(x.shape[0], C, A, y.shape[-2], y.shape[-1])
```

Gradients flow through the linear contraction into `w`. Basis expansion followed by ordinary convolution is also the implementation pattern used by steerable CNN libraries [4]. Cache expanded filters for inference only while parameters are unchanged.

Parameter and compute accounting. Ignoring bias, this layer learns $CC_{\text{in}}n_b$ parameters, independent of A . Learning every orientation independently would use $CAC_{\text{in}}s^2$ parameters. For example, $C = 16$, $C_{\text{in}} = 3$, $J = 3$, $M = 2$, and $A = 8$ gives $n_b = 15$: 720 learned coefficients versus 18,816 weights in eight independent 7×7 filters per base channel. The output still stores CA channels; weight sharing does not remove orientation-dependent compute or memory. For small n_b and many orientations, computing basis responses once and mixing them, as in (3.5), is an alternative to benchmark.

3.5 What equivariance this construction provides

An orientation index is part of the feature. For a continuous image f , define

$$F_f(x, \alpha) = [f \star L_\alpha k](x). \quad (3.12)$$

Rotating the input by β gives

$$F_{L_\beta f}(x, \alpha) = F_f(R_{-\beta}x, \alpha - \beta). \quad (3.13)$$

To verify this, substitute $u = R_\beta v$ in the convolution integral: the input becomes $f(R_{-\beta}x + v)$ and the filter becomes $k(R_{-(\alpha-\beta)}v)$. Thus a rotation moves spatial locations *and* shifts orientation channels. This is precisely why retaining the orientation axis matters.

When only A angles are stored, rotations by multiples of $2\pi/A$ permute those angles cyclically. Other angles are not simple permutations of this finite tensor. They can be evaluated through the continuous steering formula; reconstructing them from sampled responses requires sufficient angular sampling. For a response bandlimited to M , the usual uniform Fourier reconstruction requires $A \geq 2M + 1$. A pointwise nonlinearity can introduce additional angular frequencies.

Steerable does not mean scalar-to-scalar rotation-equivariant. Suppose the kernel is continuous and integrable, both input and output are ordinary scalar images with no orientation or vector channels, and we require $T_k L_\alpha = L_\alpha T_k$ for all inputs, where $T_k f = f \star k$. The kernel must be rotationally symmetric:

$$k(R_\alpha u) = k(u) \quad \text{for every } \alpha, \quad \text{so } k(r, \theta) = q(r). \quad (3.14)$$

This follows by comparing the two integrals for arbitrary test inputs. A directional edge detector fails this requirement, although its rotated copies can be steered perfectly.

Directional information therefore needs a suitable output transformation law: orientation channels, vector channels, or other feature types. The next note derives the matrix-valued kernel constraints for such types. Applying arbitrary channel mixing or nonlinearities after the lifting layer can destroy its equivariance; the channel transformation law determines which operations are allowed [3].

3.5.1 Separate mathematical guarantees from numerical tests

Analytic steering avoids repeatedly interpolating a sampled filter. It does *not* make an ordinary square pixel grid invariant under every angle. Discretized convolution, image interpolation, boundaries, stride, and angular sampling can all affect a layer's measured equivariance. Use two distinct tests:

1. **Kernel steering test.** At arbitrary continuous coordinates, compare $k_w(R_{-\alpha}x)$ with $k_{D(\alpha)^\top w}(x)$. Include $\alpha = 0, \pi/2$, and a non-grid angle. For the analytic construction, error should be at floating-point scale. This isolates coefficient ordering and sign mistakes.
2. **Layer equivariance test.** Compare the two sides of (3.13) after the actual discrete implementation. Start with a 90° rotation on a square image, A divisible by four, stride one, a centered odd kernel, and a compatible interior crop. Then test arbitrary angles, report relative error, and vary resolution and interpolation. A discrepancy here can be a sampling effect even when the kernel identity is correct.

Also test a single asymmetric edge image to see the direction of the orientation-channel shift. Tying a bias across all orientations of a base filter preserves the permutation rule; separate biases for each orientation generally do not.

Optional derivation: why the angular basis is complete

On the circle use $\langle f, g \rangle = (2\pi)^{-1} \int_0^{2\pi} f(\theta) \overline{g(\theta)} d\theta$. Direct integration gives

$$\langle e^{im\theta}, e^{in\theta} \rangle = \delta_{mn}. \quad (3.15)$$

Orthogonality alone is not completeness. The additional Fourier-series theorem says that trigonometric polynomials are dense in $L^2(S^1)$. One route to this result is that Fejér means approximate continuous periodic functions uniformly, while continuous periodic functions are dense in L^2 . Therefore the orthogonal projections

$$S_M f = \sum_{m=-M}^M \langle f, e^{im\theta} \rangle e^{im\theta} \quad (3.16)$$

converge to f in L^2 . Applying this angular projection to a planar kernel yields (3.10); its squared truncation error is

$$\|k - S_M k\|_{L^2(\mathbb{R}^2)}^2 = 2\pi \int_0^\infty \sum_{|m|>M} |c_m(r)|^2 r dr \rightarrow 0. \quad (3.17)$$

This is a convergence statement about increasingly rich function spaces, not a claim that finitely many harmonics represent every kernel.

References

- [1] W. T. Freeman and E. H. Adelson. The Design and Use of Steerable Filters. *IEEE TPAMI*, 1991.
- [2] M. Weiler, F. A. Hamprecht, and M. Storath. Learning Steerable Filters for Rotation Equivariant CNNs. *CVPR*, 2018.
- [3] M. Weiler and G. Cesa. General $E(2)$ -Equivariant Steerable CNNs. *NeurIPS*, 2019.
- [4] The e2cnn authors. e2cnn neural network modules: basis expansion and R2Conv. Implementation reference.