

EECE 576L

Advanced Topics in Deep Learning

Lecture 4: Group Equivariant Deep Learning

Part 2: Group convolutions

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 3D networks
5. Implementation and verification (optional reading)

Outline

1. Groups and actions

2. Group convolutions

- **Lifting convolution: image to poses**
- Group convolution: poses to poses
- Equivariance
- Sampling and network design

3. Steerable kernels

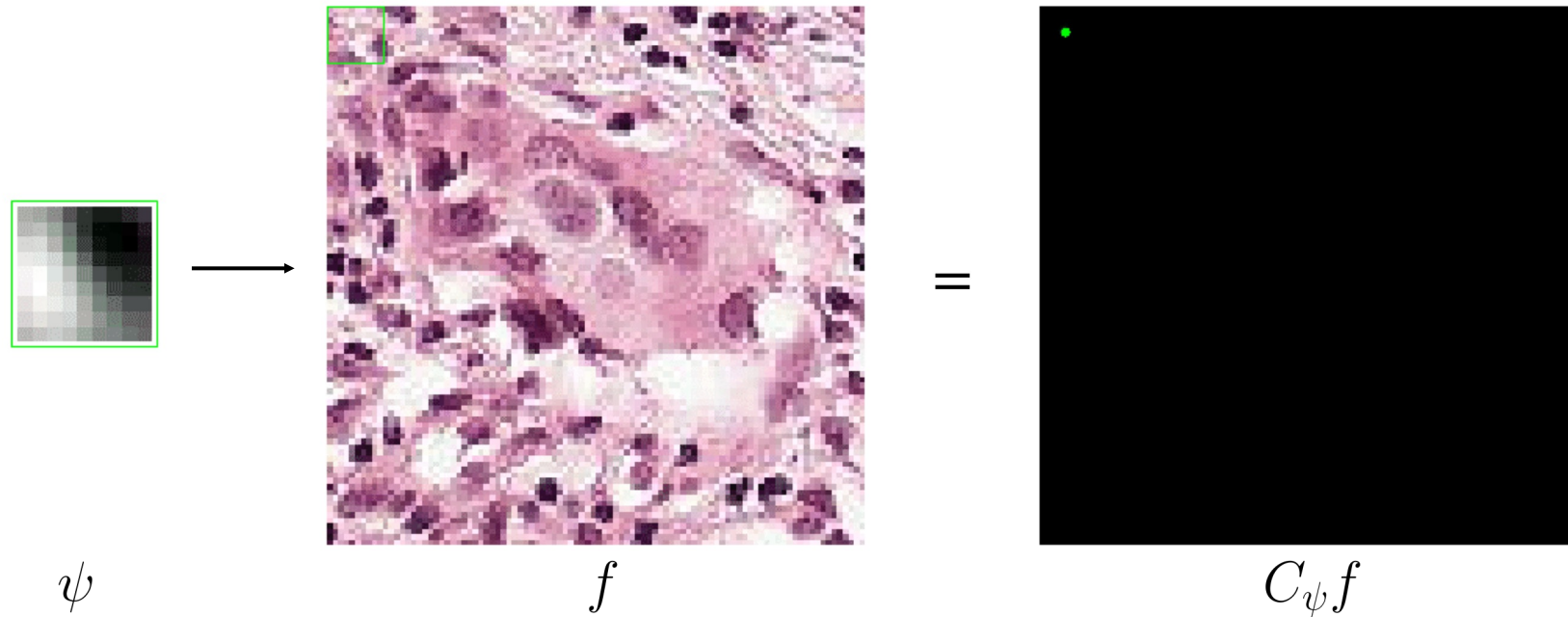
4. Typed features and 3D networks

5. Implementation and verification (optional reading)

Start from familiar convolution

CNNs call cross-correlation “convolution”: the kernel is not flipped.

$$(C_\psi f)(t) = \int_{\mathbb{R}^2} \psi(x - t) f(x) d^2x$$



A fixed filter does not share across rotations

Generic filters need not commute with rotations.

$$C_\psi T_R \neq T_R C_\psi$$

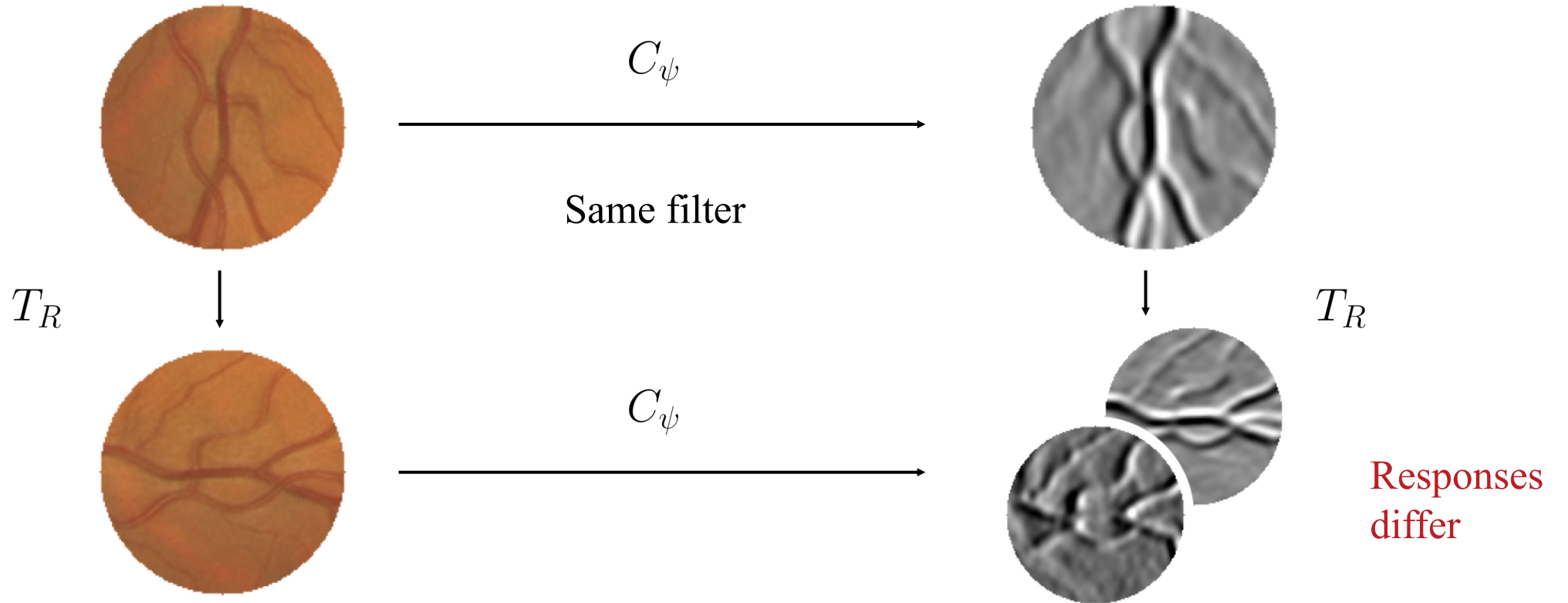
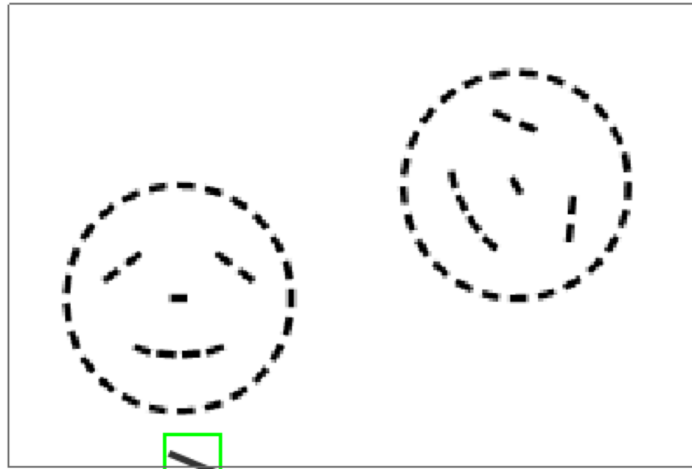


Image input and lifted output

Lifting stores a response for each detector position and orientation.

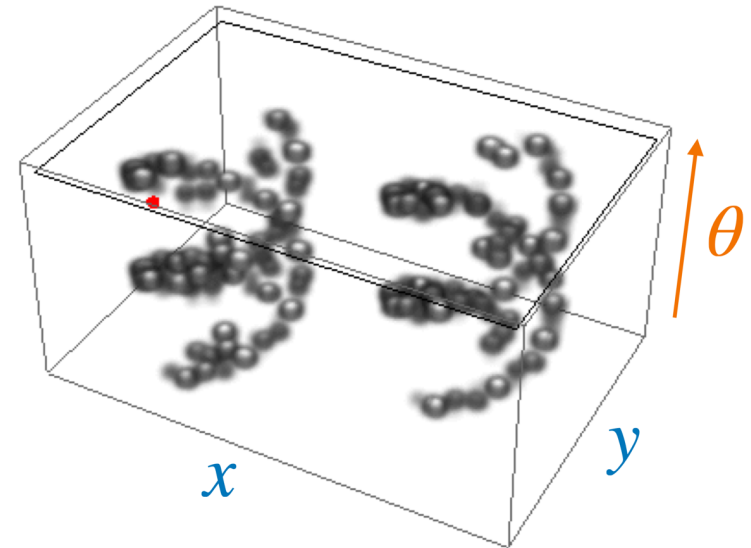
$$f : \mathbb{R}^2 \rightarrow \mathbb{R}^{C_{\text{in}}}, \quad F = \mathcal{L}_\psi f : G \rightarrow \mathbb{R}^{C_{\text{out}}}, \quad G = \text{SE}(2)$$

Image input



$\mathcal{L}_\psi$
Lifting operator

Lifted output field



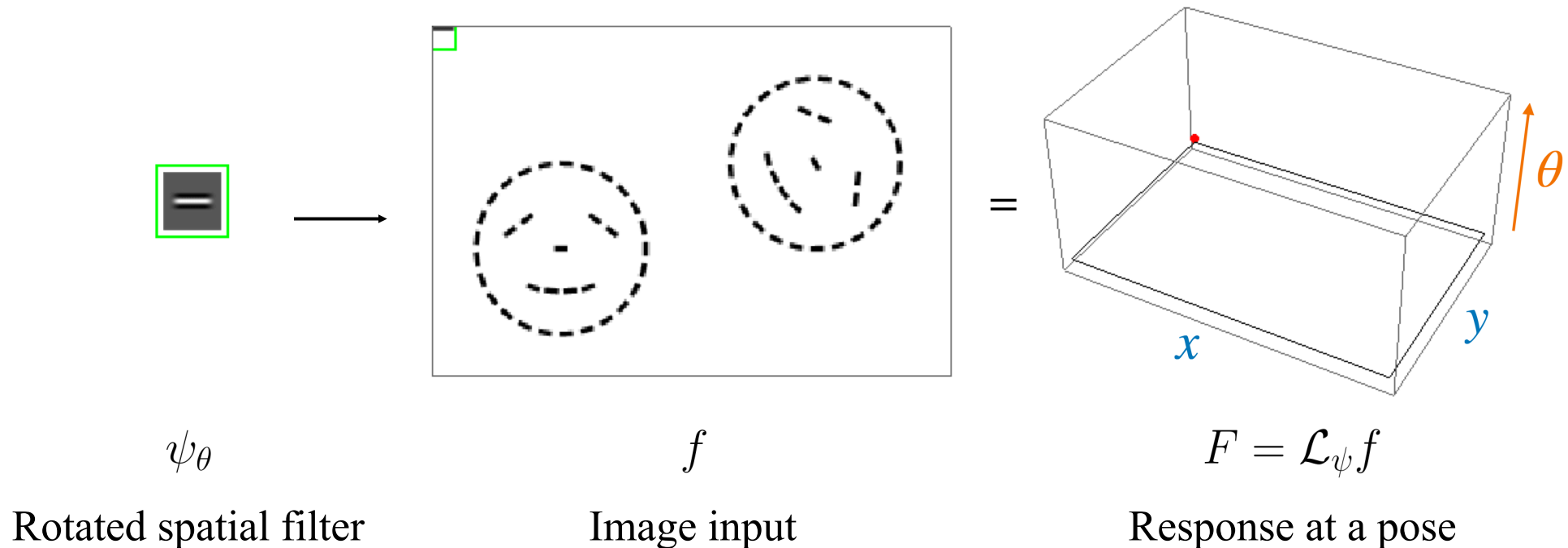
The third coordinate is detector angle, rather than physical depth.

Lifting convolution: one detector pose

The first group-convolution layer maps an image to a pose field.

[Play animation at 6:48](#) 

$$F(t, \theta) = (\mathcal{L}_\psi f)(t, \theta) = \int_{\mathbb{R}^2} \psi(R_{-\theta}(x - t)) f(x) d^2x$$

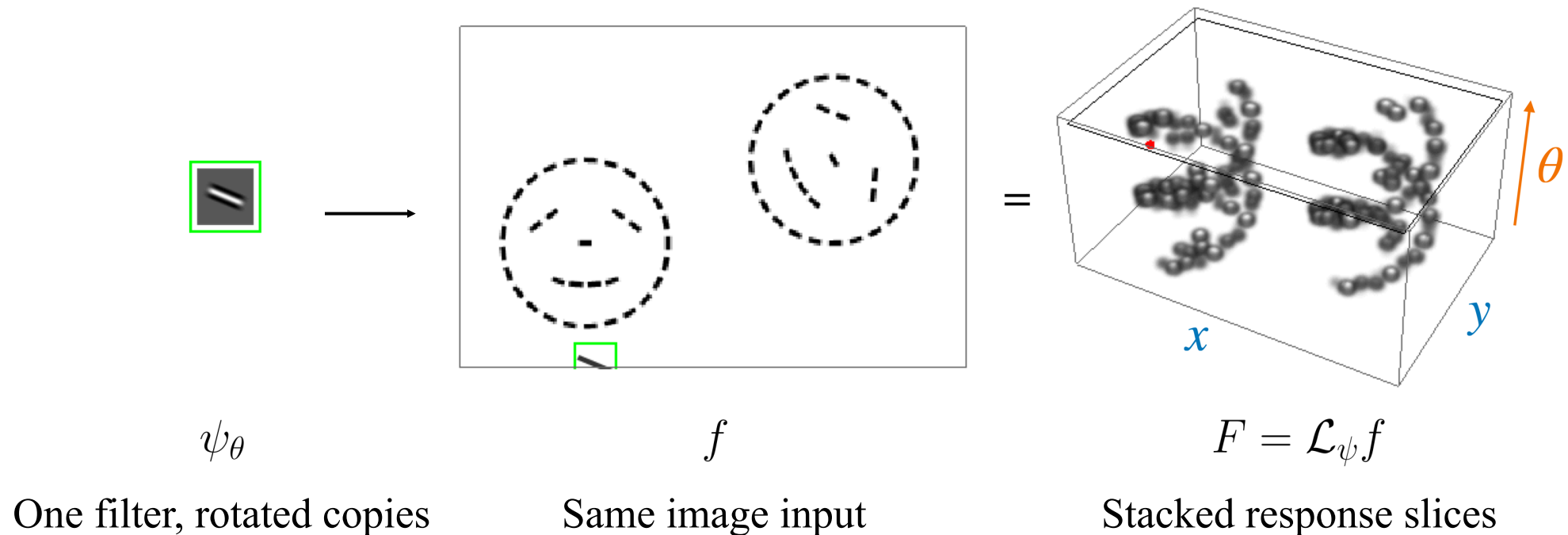


Lifting convolution: all detector orientations

At each angle, an ordinary spatial convolution creates one slice.

[Play animation at 7:08](#) 

$$\psi_{\theta}(\delta) = \psi(R_{-\theta}\delta), \quad F(t, \theta) = (C_{\psi_{\theta}}f)(t)$$

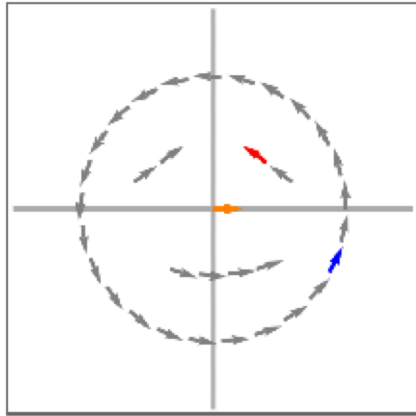


Motion of a lifted feature field

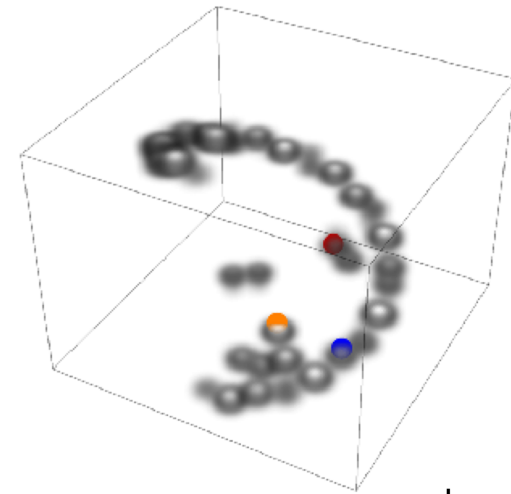
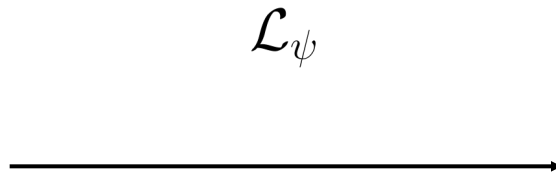
Image motion moves both positions and orientation indices.

[Play animation at 8:23](#) 

$$a = (u, R_\alpha), \quad (t, \theta) \mapsto (u + R_\alpha t, \theta + \alpha)$$



Transformed image



Spatial warp + angle shift

$$(L_a F)(t, \theta) = F(R_{-\alpha}(t - u), \theta - \alpha)$$

Outline

1. Groups and actions

2. Group convolutions

- Lifting convolution: image to poses
- **Group convolution: poses to poses**
- Equivariance
- Sampling and network design

3. Steerable kernels

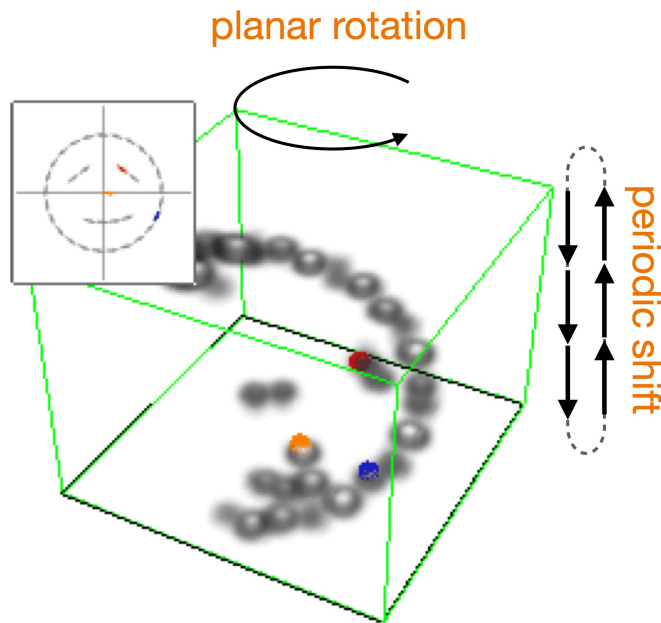
4. Typed features and 3D networks

5. Implementation and verification (optional reading)

Hidden filters detect arrangements of oriented parts

Combine oriented edges into corners and other patterns.

[Play animation at 12:02](#) 



A filter now spans:

- Relative spatial displacement
- Relative orientation
- Input and output feature channels

$$\psi : \mathbb{R}^2 \times S^1 \rightarrow \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}$$

A 3D pose domain: two spatial coordinates and one periodic angle.

Group convolution on pose fields

The lifted field is the first hidden layer's input. Its output is another pose field.

$$F : G \rightarrow \mathbb{R}^{C_{\text{in}}}, \quad \psi : G \rightarrow \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}, \quad Y : G \rightarrow \mathbb{R}^{C_{\text{out}}}$$

Generic group convolution

$$Y(g) = (C_\psi F)(g) = \int_G \psi(g^{-1}h) F(h) d\mu(h)$$

Kernel weights depend on the **relative position and orientation**.

SE(2): integrate over space and average over angles

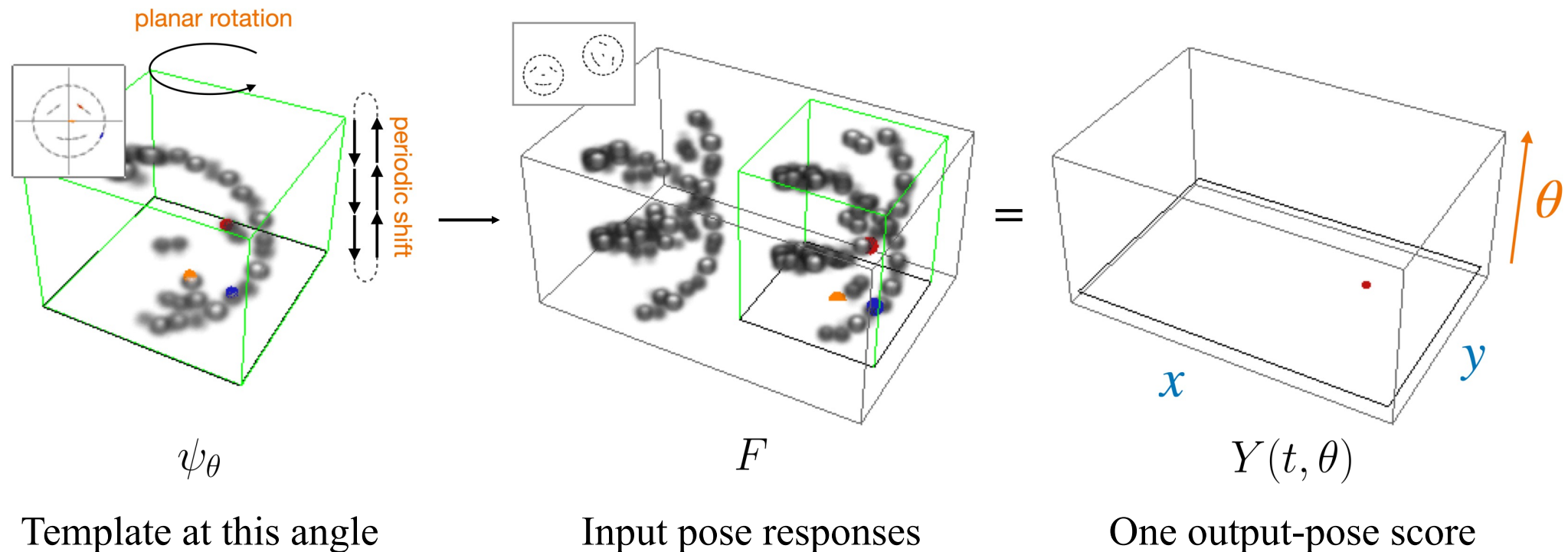
$$Y(t, \theta) = \int_{\mathbb{R}^2} \int_0^{2\pi} \psi(R_{-\theta}(x - t), \varphi - \theta) F(x, \varphi) \frac{d\varphi}{2\pi} d^2x$$

One template, one output pose

Left: align the template. Middle: input features. Right: one matching score.

[Play animation at 12:27](#) 

$$\psi_{\theta}(\delta, \varphi) = \psi(R_{-\theta}\delta, \varphi - \theta)$$

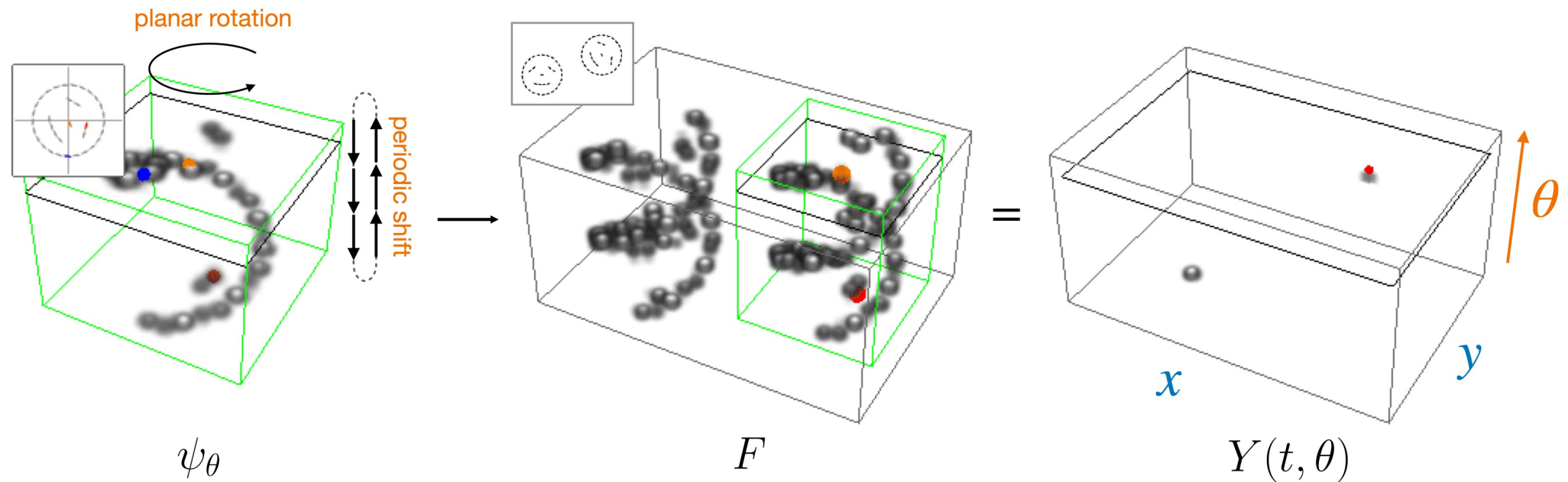


The same template at another angle

Rotate offsets and shift the angle index together. The input field stays fixed.

[Play animation at 12:52](#) 

$$\psi_{\theta}(\delta, \varphi) = \psi(R_{-\theta}\delta, \varphi - \theta)$$



Same template, new angle

Same input responses

Old (gray), new (red) scores

Outline

1. Groups and actions

2. Group convolutions

- Lifting convolution: image to poses
- Group convolution: poses to poses
- **Equivariance**
- Sampling and network design

3. Steerable kernels

4. Typed features and 3D networks

5. Implementation and verification (optional reading)

Group-convolution equivariance

A hidden group-convolution layer commutes with a motion of its input field.

$$(L_a F)(h) = F(a^{-1}h)$$

Input pose field

$$F : G \rightarrow \mathbb{R}^{C_{\text{in}}}$$

Oriented edge or corner responses.

Hidden group-convolution layer

$$Y = C_\psi F$$

A learned template detects their configuration.

Move input, then convolve

$$C_\psi(L_a F)$$

=

Convolve, then move output

$$L_a(C_\psi F)$$

Both routes move a corner response to the same location and orientation.

Lifting and hidden-layer interfaces

The lifting operator connects an image action to a pose-field action.

Image motion

$$(T_a f)(x) = f(a^{-1} \cdot x)$$

Pose-field motion

$$(L_a F)(g) = F(a^{-1}g)$$

Lifting convolution: image to poses

$$F = \mathcal{L}_\psi f$$

$$\mathcal{L}_\psi T_a = L_a \mathcal{L}_\psi$$

Transform pixels, then lift.
The result equals transforming the lifted field.

Hidden convolution: poses to poses

$$Y = C_\psi F$$

$$C_\psi L_a = L_a C_\psi$$

Transform a hidden field, then convolve.
The result equals transforming the output field.

Equivariance proof: pose fields

Rigid motions preserve spatial area and the angular integration weights.

1. Expand convolution with the moved input field.

$$(C_\psi L_a F)(g) = \int_G \psi(g^{-1}h) F(a^{-1}h) d\mu(h)$$

2. Change the input variable and rewrite the relative pose.

$$h = ak, \quad d\mu(h) = d\mu(k), \quad g^{-1}a = (a^{-1}g)^{-1}$$

$$\int_G \psi((a^{-1}g)^{-1}k) F(k) d\mu(k) = (C_\psi F)(a^{-1}g)$$

3. Recognize the left-regular action on the output.

$$(C_\psi L_a F)(g) = (L_a C_\psi F)(g)$$

Equivariance proof: lifting

The input variable is an image point. Rigid motions preserve planar area.

Recall the detector-frame coordinate:

$$g^{-1} \cdot x = R_{-\theta}(x - t)$$

1. Expand lifting with the moved image.

$$(\mathcal{L}_\psi T_a f)(g) = \int_{\mathbb{R}^2} \psi(g^{-1} \cdot x) f(a^{-1} \cdot x) d^2x$$

2. Change coordinates and identify the inverse-moved output pose.

$$x = a \cdot y, \quad d^2x = d^2y, \quad g^{-1} \cdot (a \cdot y) = (a^{-1}g)^{-1} \cdot y$$

$$(\mathcal{L}_\psi T_a f)(g) = (\mathcal{L}_\psi f)(a^{-1}g) = (L_a \mathcal{L}_\psi f)(g)$$

Outline

1. Groups and actions

2. Group convolutions

- Lifting convolution: image to poses
- Group convolution: poses to poses
- Equivariance
- **Sampling and network design**

3. Steerable kernels

4. Typed features and 3D networks

5. Implementation and verification (optional reading)

Two approaches to rotations

Both approaches prescribe how features transform when the image rotates.

1. Store responses at sampled orientations (this part)

$$\theta_j = \frac{2\pi j}{N}, \quad j = 0, \dots, N - 1$$

N counts the sampled orientations.

Rotate one shared filter and shift hidden orientation indices.

2. Encode continuous transformations analytically (Parts 3–4)

A finite steerable basis and typed channels specify the response to any rotation angle without storing an infinite array.

The approaches can be combined: a steerable basis can generate a sampled bank.

Compatible operations in a complete network

After lifting, every hidden feature map carries the same pose transformation rule.

1. Nonlinearity: apply the same scalar function at every pose

$$\sigma(L_a F)(g) = \sigma(F(a^{-1}g)) = (L_a \sigma(F))(g)$$

2. Bias and normalization: share across orientations

Use one bias, scale and offset per learned feature type.

Aggregate normalization statistics over batch, space and orientation.

3. Residual addition: match the feature transformation rules

Both branches must transform in the same way before their features are added.

Readout: retain a map or remove pose

The output transformation should match the prediction task.

Dense prediction: pool orientation and retain position

$$(PF)(t) = \frac{1}{N} \sum_{j=0}^{N-1} F(t, \theta_j)$$

$$P(L_{(u,\alpha)}F)(t) = (PF)(R_{-\alpha}(t - u))$$

A segmentation map still moves with the image. It is spatially equivariant.

Classification: pool spatial positions as well

$$I(F) = \sum_t (PF)(t), \quad I(L_a F) = I(F)$$

Equalities assume the transformation preserves the sampled angles and domain.

Finite grids and continuous rotations

The continuous equivariance formula describes fields. Arrays add sampling choices.

Quarter-turns can permute pixels exactly

Use centered square grids and compatible padding.

Other angles usually interpolate between pixels.

$$R_{\pi/6}(1, 0)^{\top} = (\sqrt{3}/2, 1/2)^{\top} \notin \mathbb{Z}^2$$

Cropping and stride can change the spatial domain

Cropping may lose a feature. Stride may discard its sampling phase.

Steerable bases avoid repeatedly rotating raster kernels (Parts 3–4).

They preserve continuous transformation laws, while spatial sampling still needs care.

An equivariance test for the pose-field network

Use random images and weights in deterministic evaluation mode.

Route 1: transform the image, then apply the network

$$Y_1 = \mathcal{N}(T_a f), \quad (T_a f)(x) = f(a^{-1} \cdot x)$$

Route 2: apply the network, then transform the output poses

$$Y_2 = L_a \mathcal{N}(f), \quad (L_a F)(g) = F(a^{-1} g)$$

$$\varepsilon_{\text{eq}} = \frac{\|Y_1 - Y_2\|_2}{\max(\|Y_1\|_2, \|Y_2\|_2, 10^{-12})}$$

The output action moves positions and shifts orientation indices.

Record angle resolution, interpolation, crop and dtype with the error.

Questions?

Group convolutions

Next: Steerable kernels

References

- [1] Cohen & Welling (2016). Group Equivariant Convolutional Networks. ICML.
- [2] Bekkers. Group Equivariant Deep Learning. University of Amsterdam, Lecture 1.3.
- [3] Bekkers. Regular group convolutional neural networks. Lecture 1.3 video.
- [4] PyTorch documentation. Conv2d (cross-correlation convention).