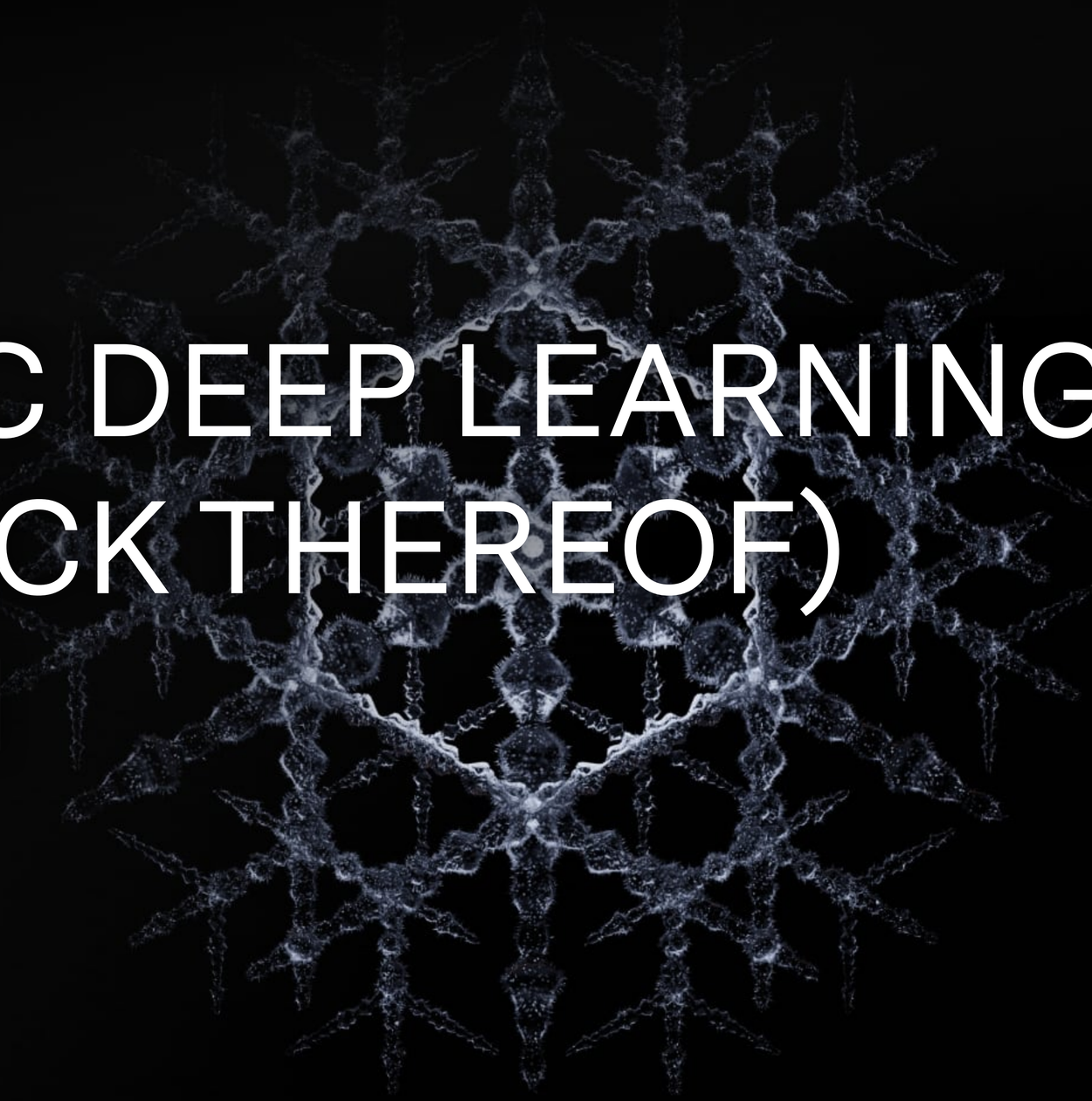
ADVANCES IN COMPUTER VISION

MODULE 1: **Geometry**

GEOMETRIC DEEP LEARNING (OR THE LACK THEREOF) FOR VISION



Prof. Vincent Sitzmann

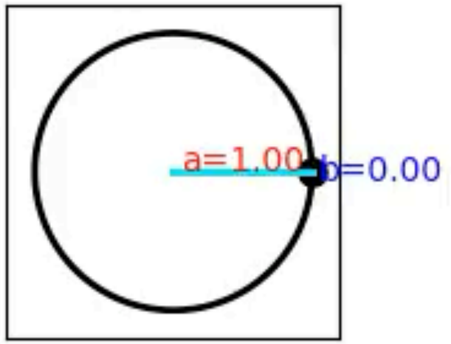
Last time on 6.8300

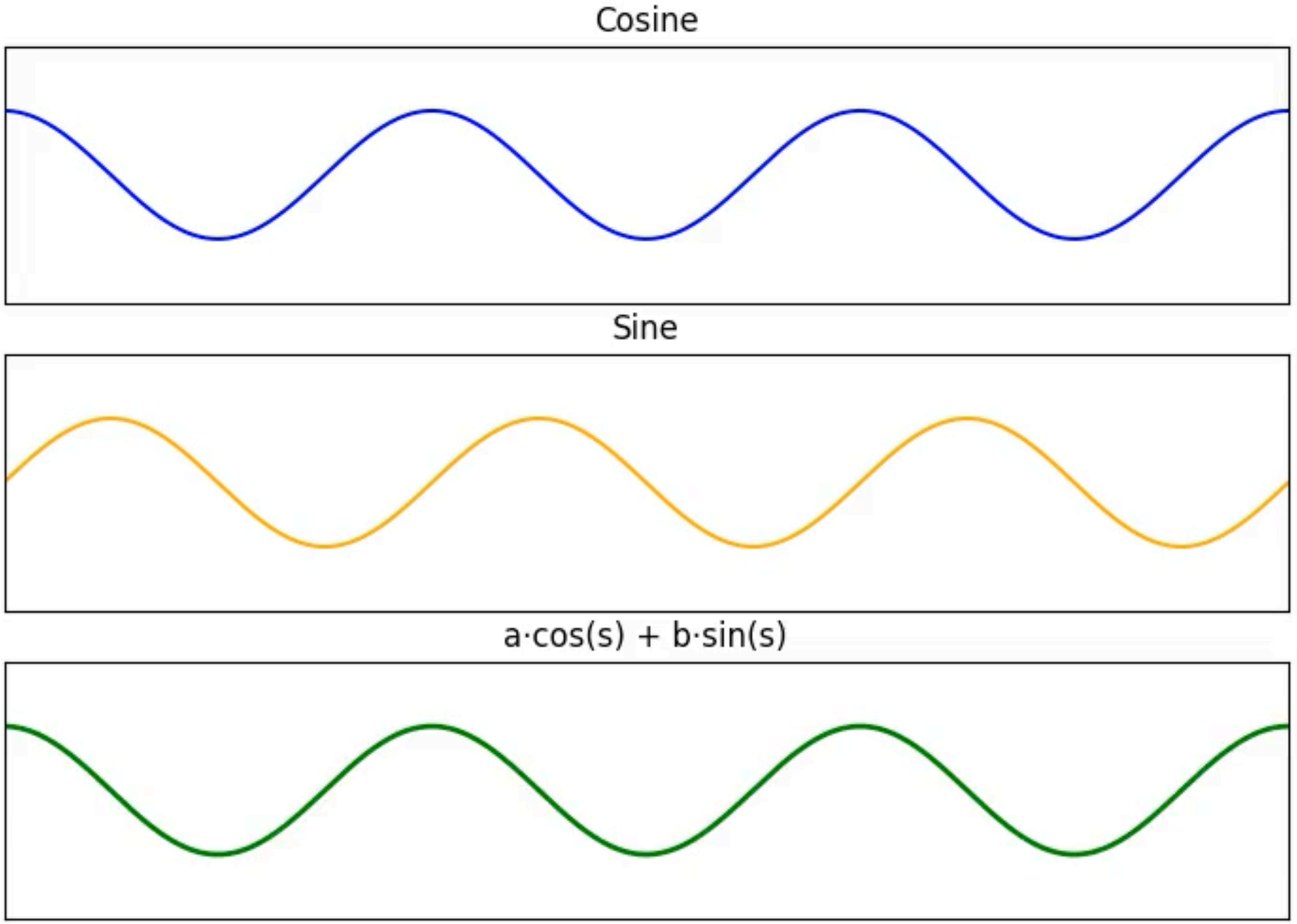
Steerable Bases

Express in
(cos, sin) basis

Translation group acts
via *rotation* of
coefficients!

$$< \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \begin{bmatrix} a \\ b \end{bmatrix} >$$



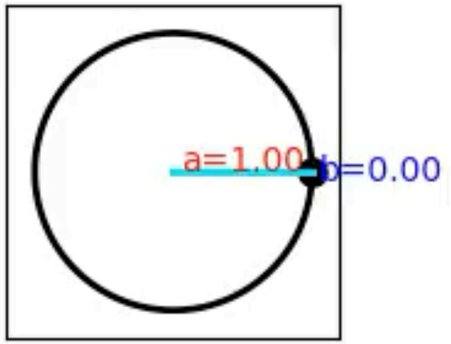


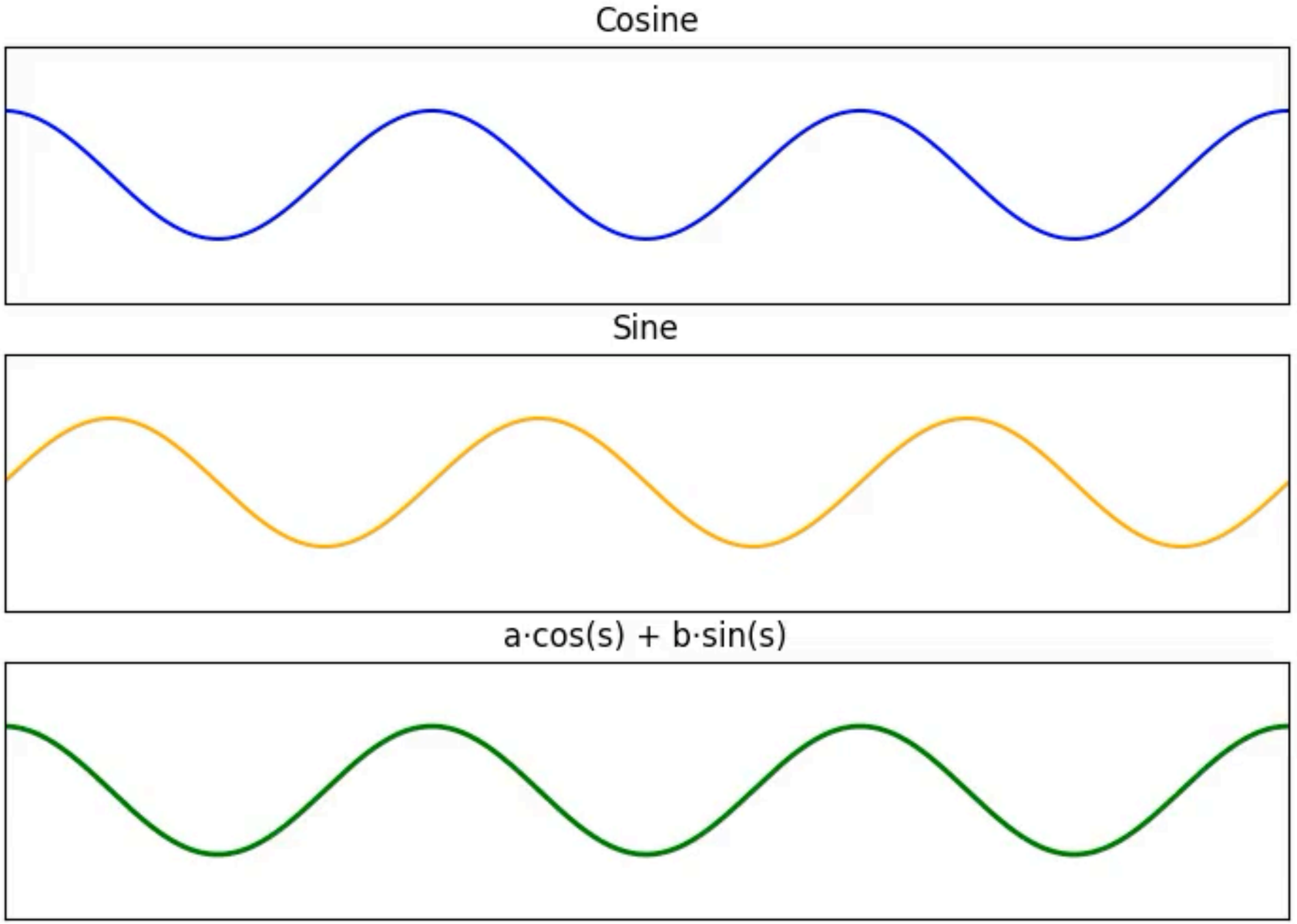
Steerable Bases

Express in
(cos, sin) basis

Translation group acts
via *rotation* of
coefficients!

$$< \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \begin{bmatrix} a \\ b \end{bmatrix} >$$





The Design and Use of Steerable Filters

William T. Freeman and Edward H. Adelson

I. Introduction

ORIENTED filters are used in many vision and image processing tasks, such as texture analysis, edge detection, image data compression, motion analysis, and image enhancement. In mar

With the correct filter set and the correct interpolation rule, it is possible to determine the response of arbitrary orie

a filter of arbitrary orientation without explicitly applying that filter.

the filter outp

The Design and Use of Steerable Filters

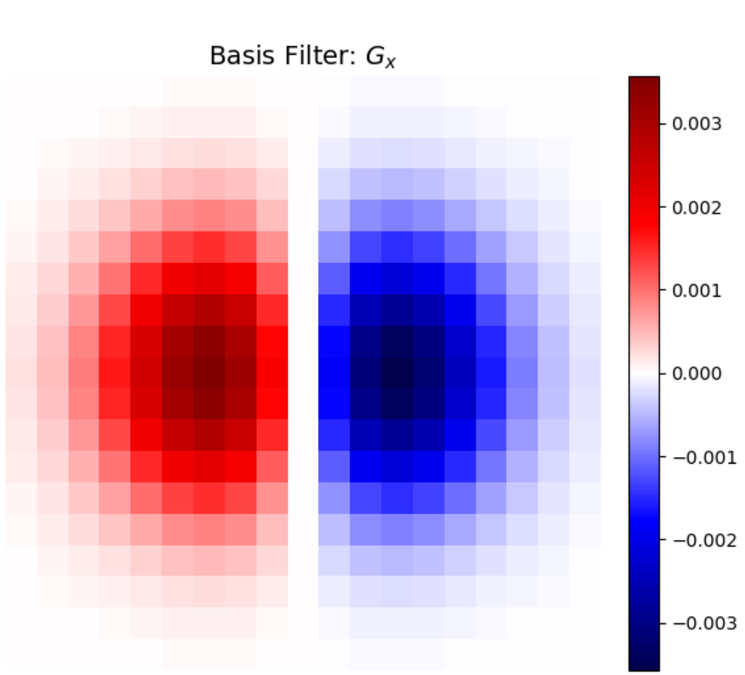
William T. Freeman and Edward H. Adelson

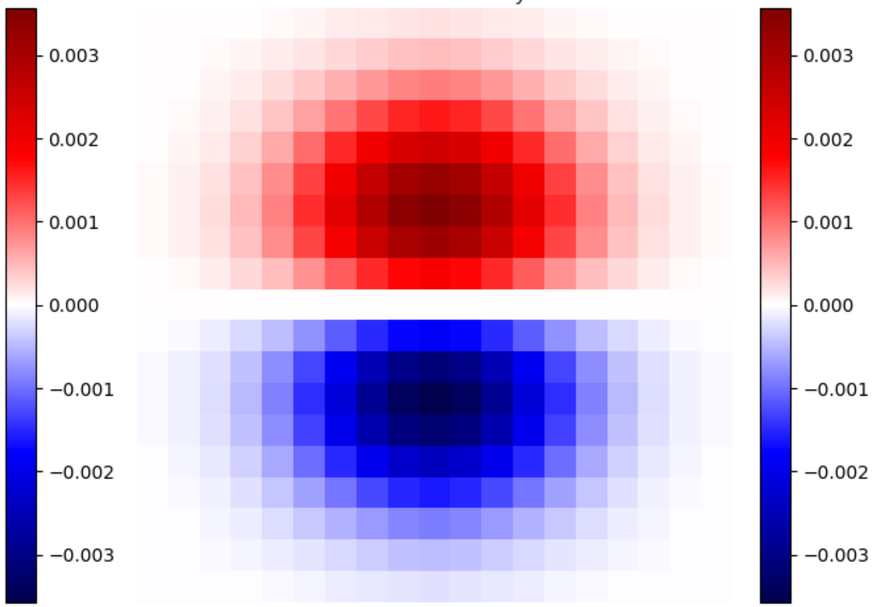
I. Introduction

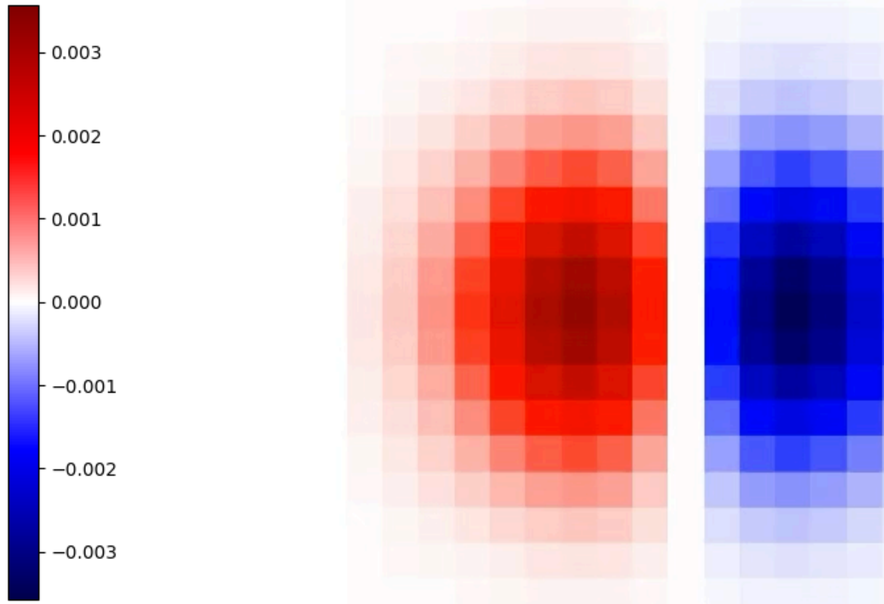
Derives a basis for *rotation-steerable* 2D image filters!

With the correct filter set and the correct interpolation rule, it is possible to determine the response of a filter of arbitrary orientation without explicitly applying that filter.

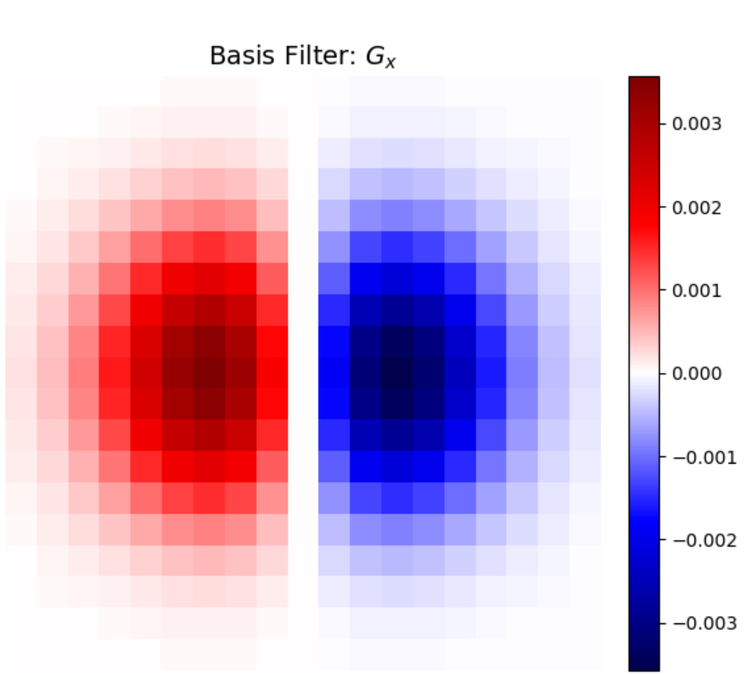
Steerable Filters

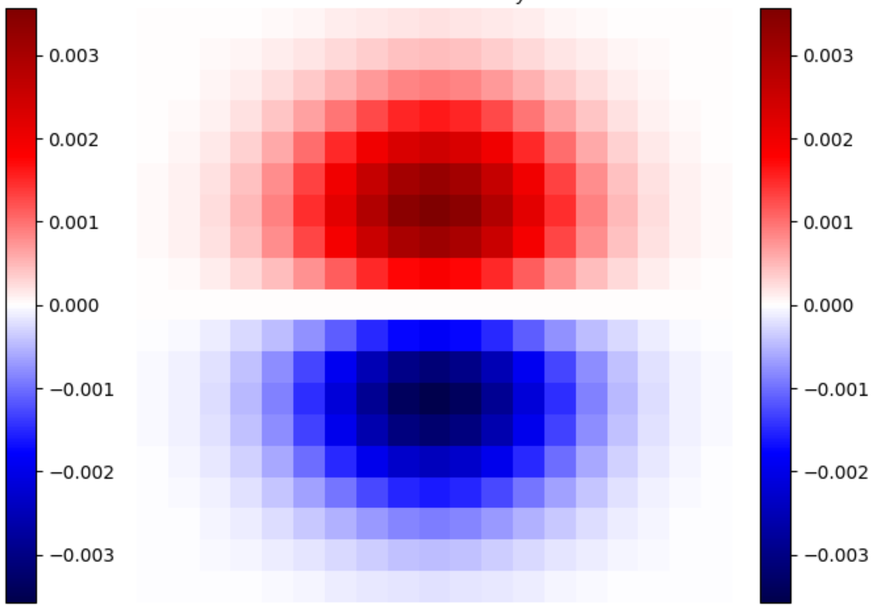


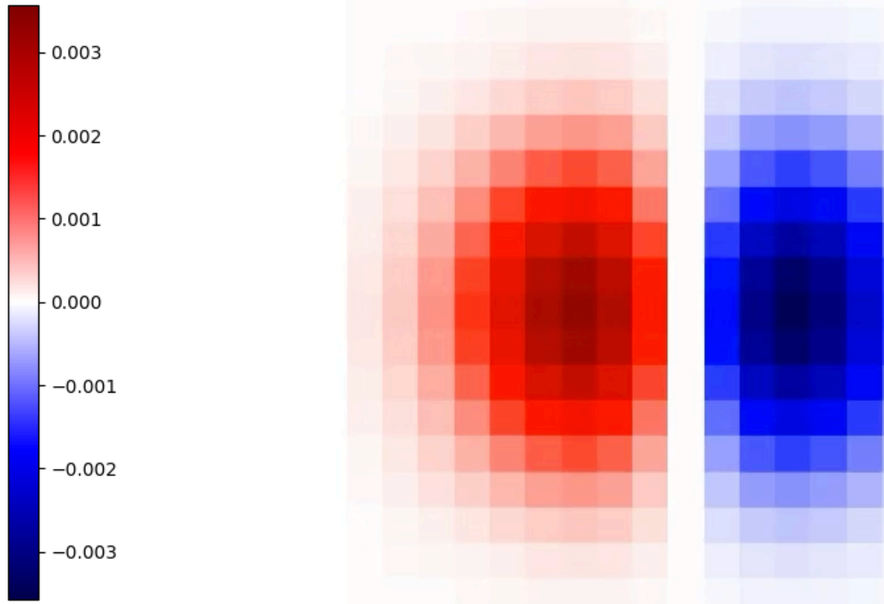




Steerable Filters

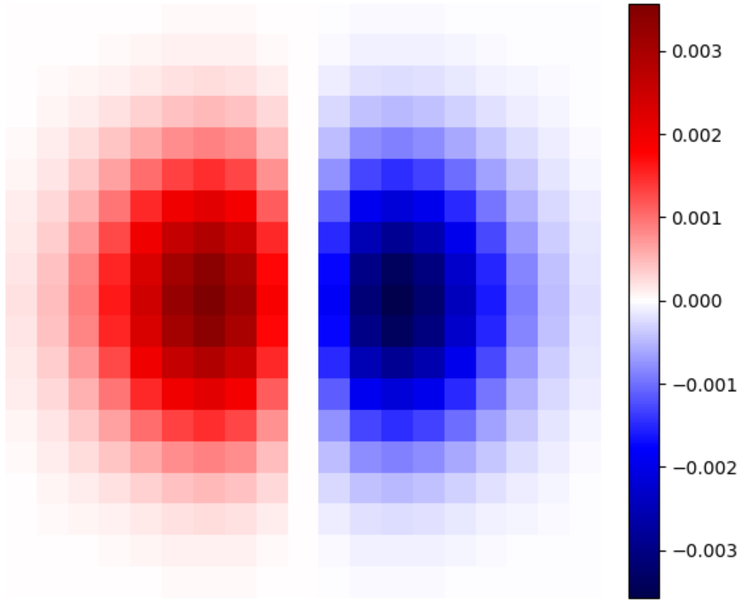




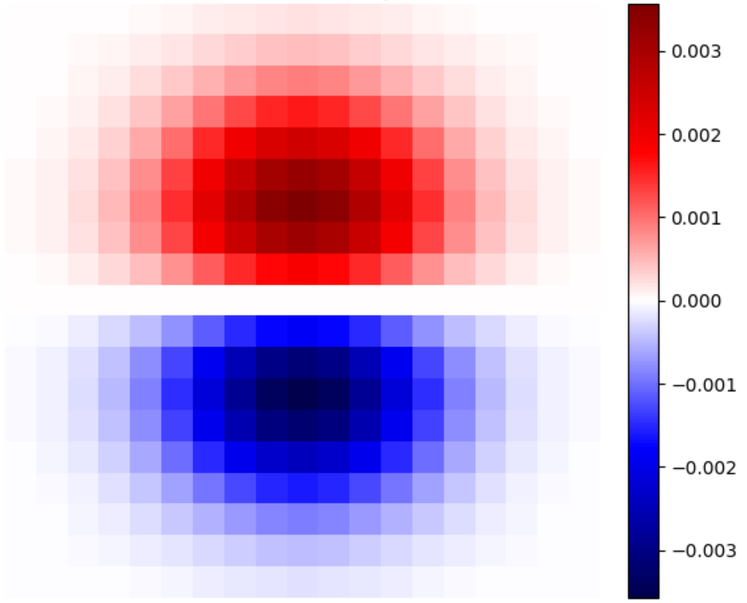


Steerable Filters

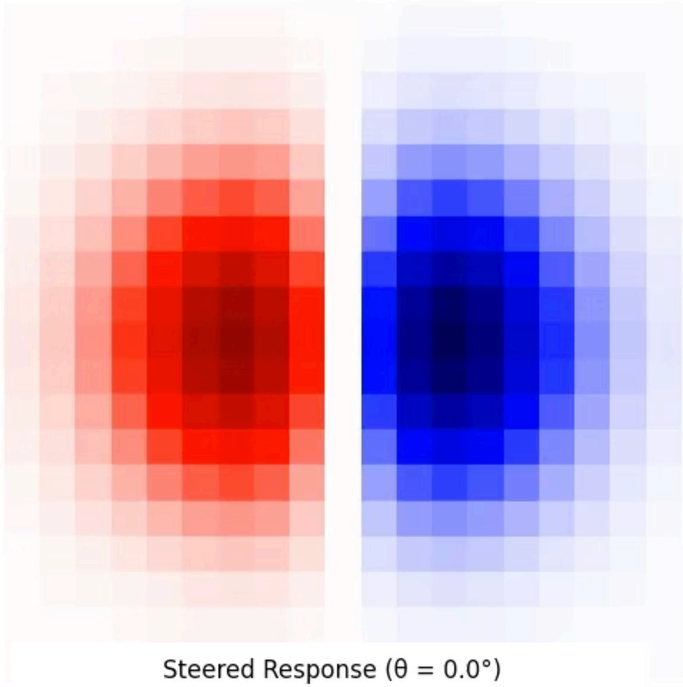
Basis Filter: G_x



Basis Filter: G_y

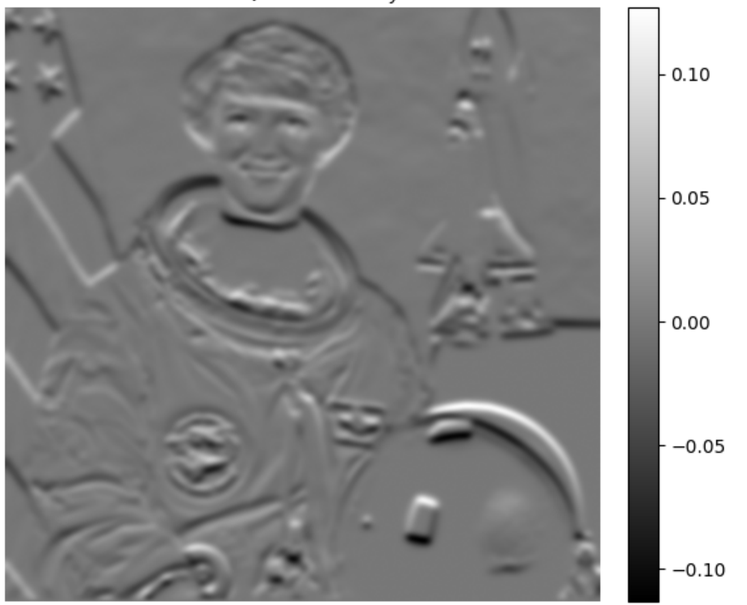


Steered Filter (θ = 0.0°)



Response to G_x

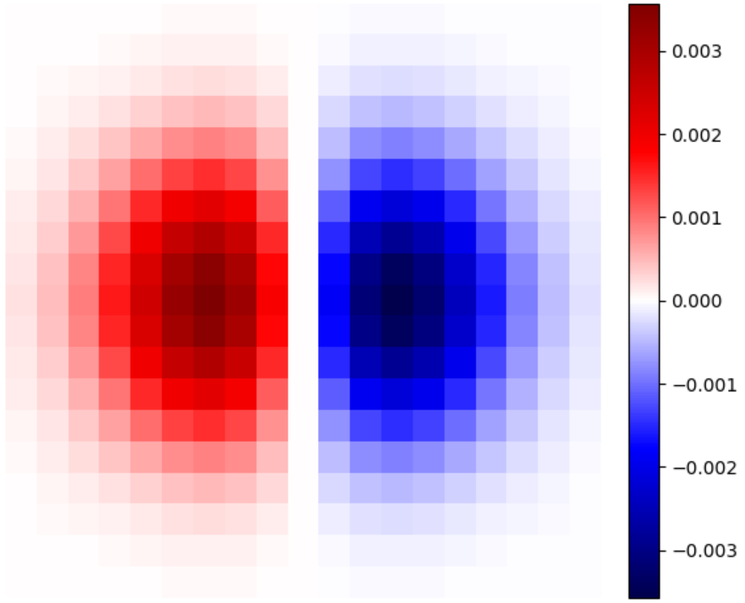
Response to G_y



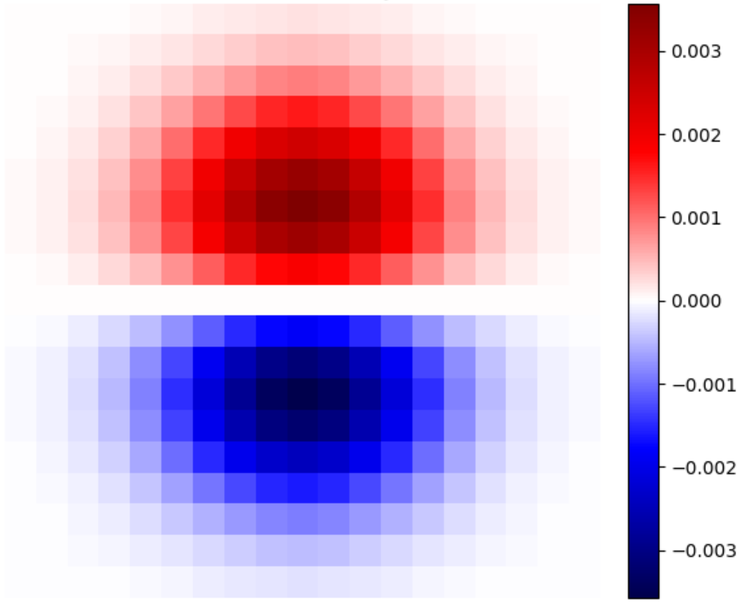
Steered Response (θ = 0.0°)

Steerable Filters

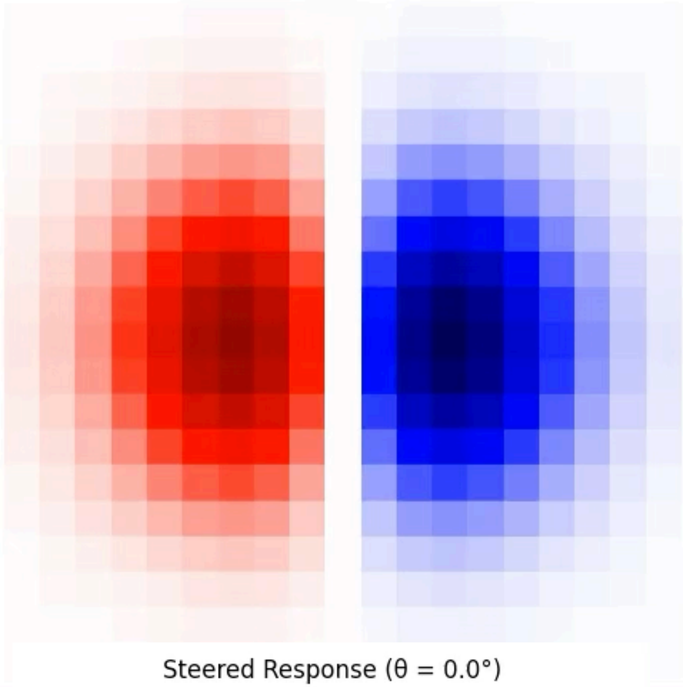
Basis Filter: G_x



Basis Filter: G_y

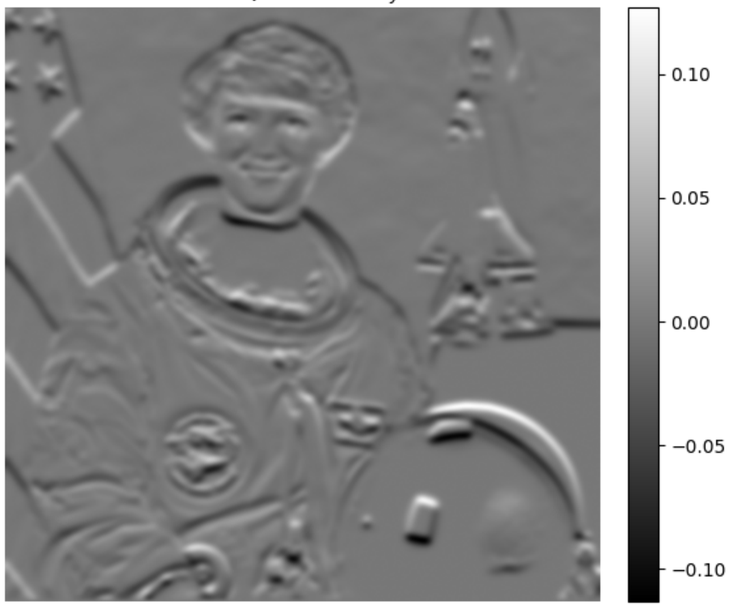


Steered Filter (θ = 0.0°)



Response to G_x

Response to G_y



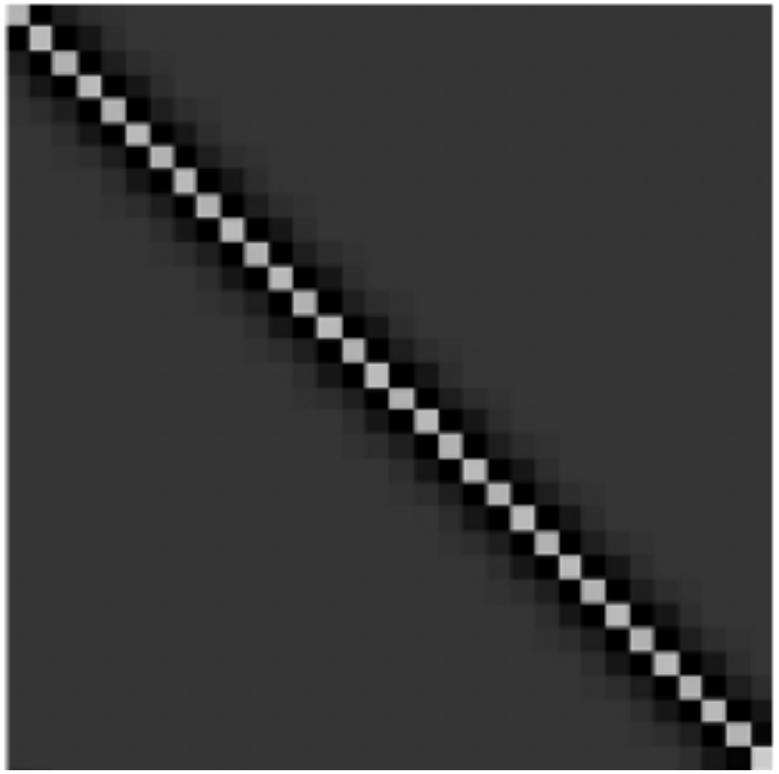
Steered Response (θ = 0.0°)

Recipe for finding a steerable basis in which group action is “simple”

1. Find a *linear operator* such that…
    1. …it *commutes* with the group action. This means that its eigenspace is unchanged under the group action.
    2. …its eigenvectors are guaranteed to be a complete basis for the function space (it is self-adjoint).
2. Compute its Eigendecomposition.
3. The eigenfunctions then **must form a steerable basis** for the group transformation!

Step 1: Find a self-adjoint operator that commutes with translations.

Laplace operator



Gaussian Blur

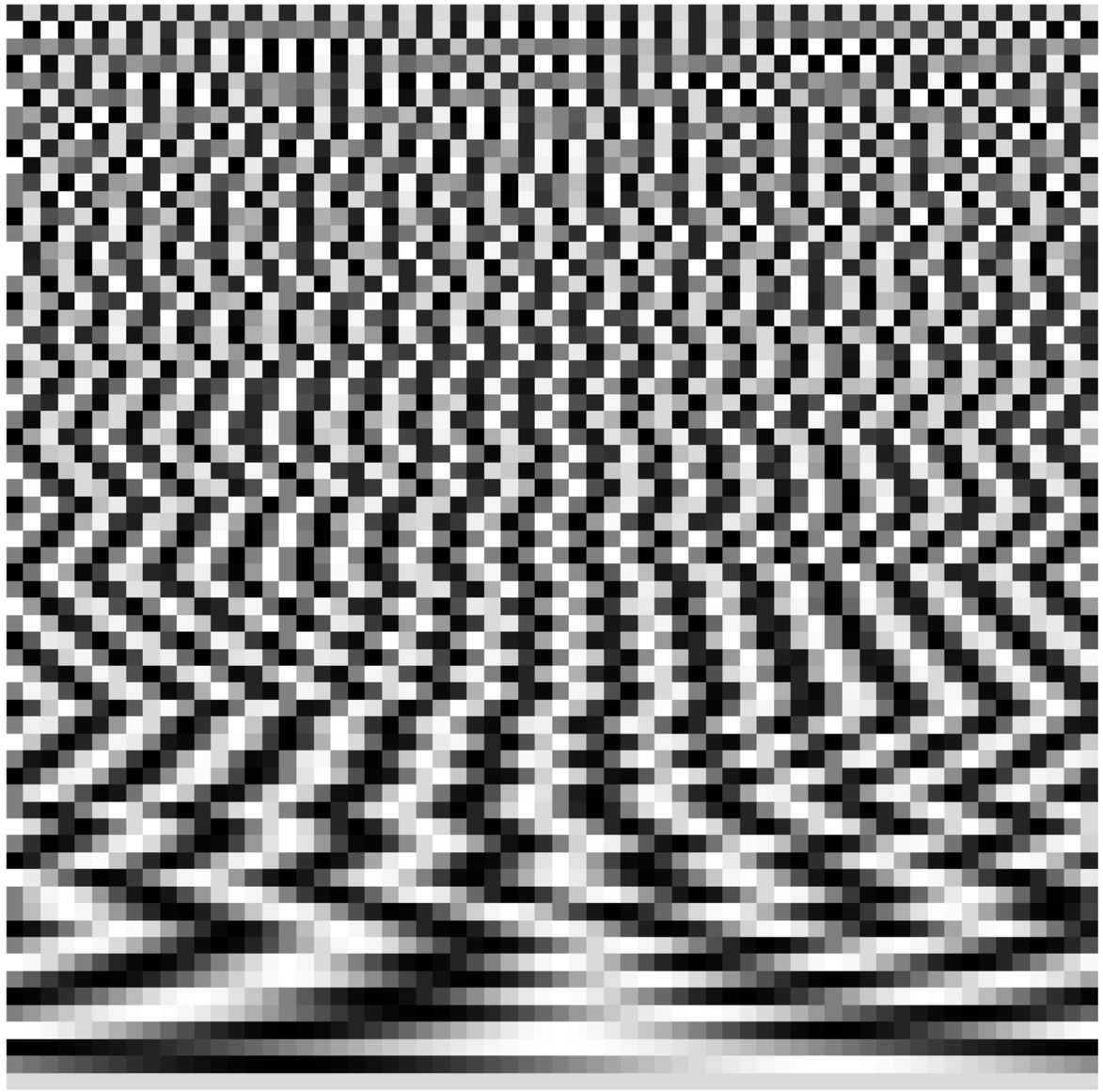
• • •

Any convolution with a centered, symmetric kernel will do!

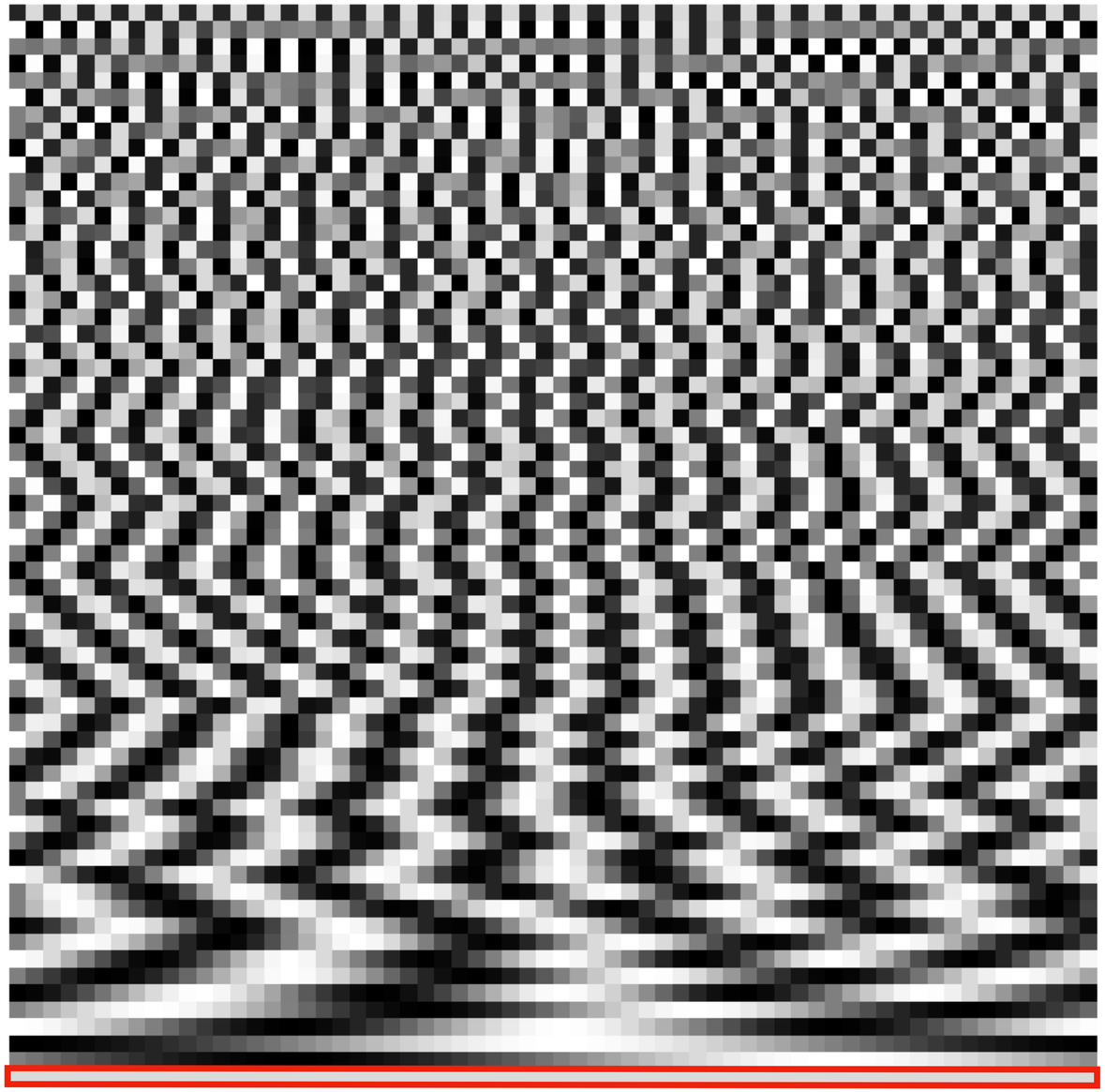
Step 2: Compute its Eigenspectrum

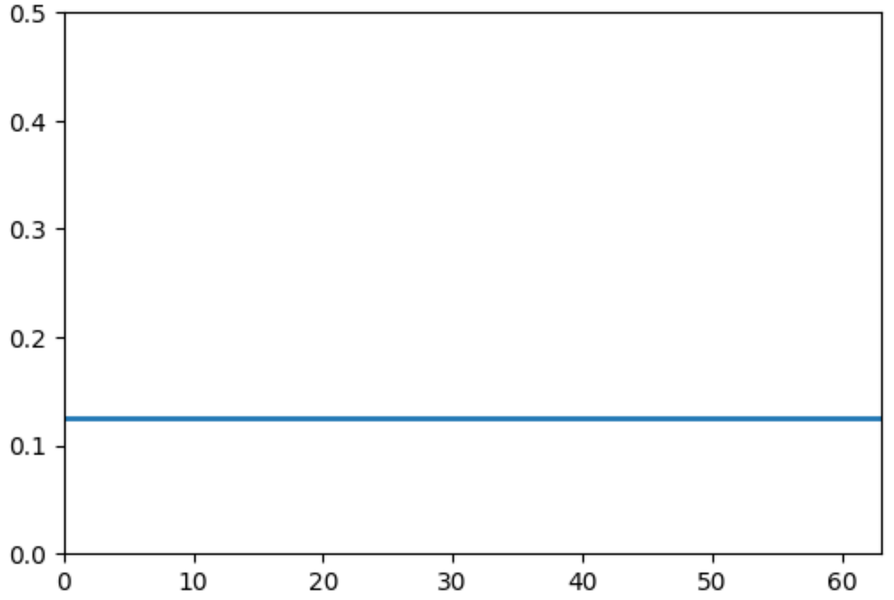
```python
import numpy as np

n = 128

# Create Toeplitz matrix with periodic BCs
main_diag = -2 * np.ones(n)
off_diag = np.ones(n-1)

L = np.diag(main_diag) + np.diag(off_diag, 1) + np.diag(off_diag, -1)
L[0, -1] = 1
L[-1, 0] = 1

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eigh(L)
```

Eigenfunctions

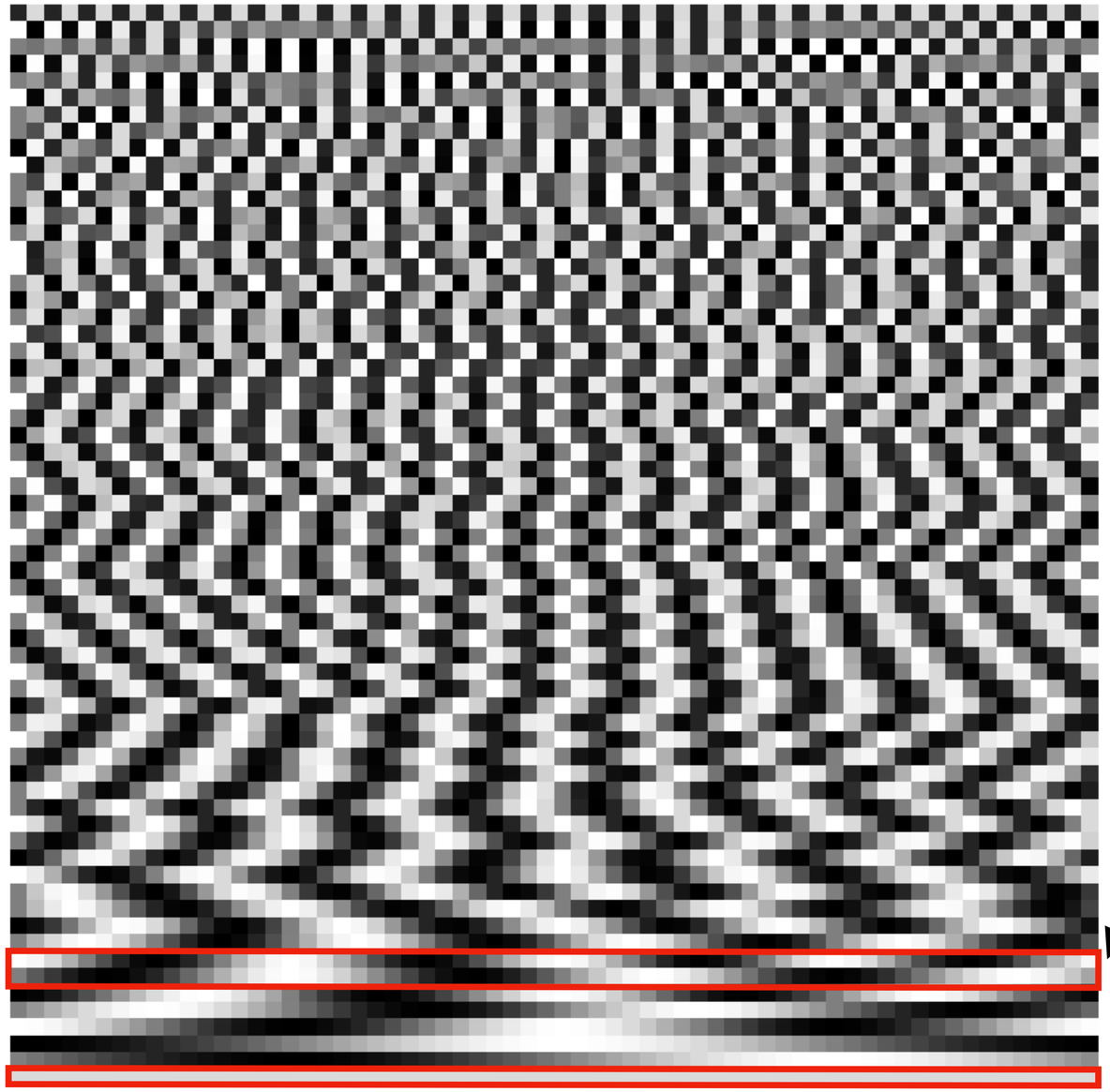


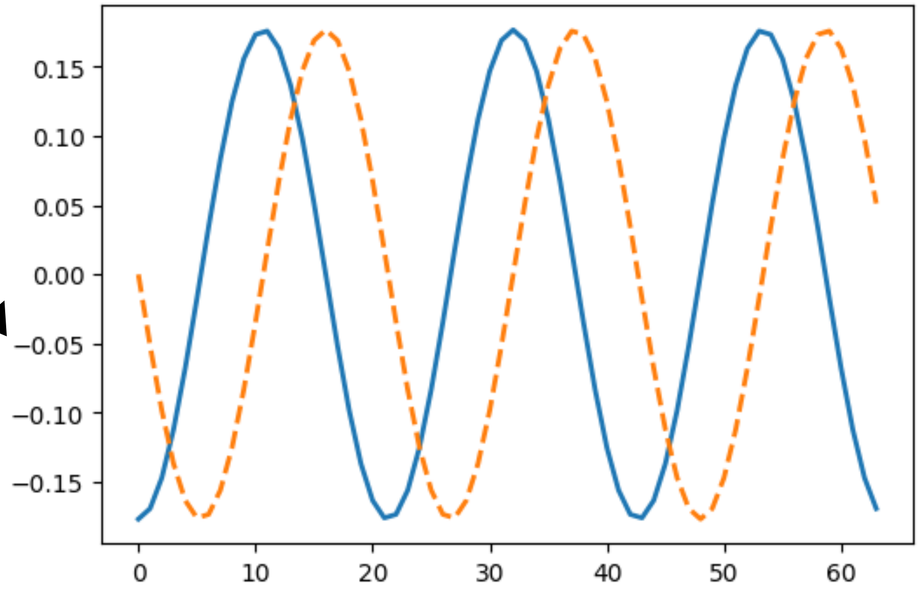
Eigenfunctions

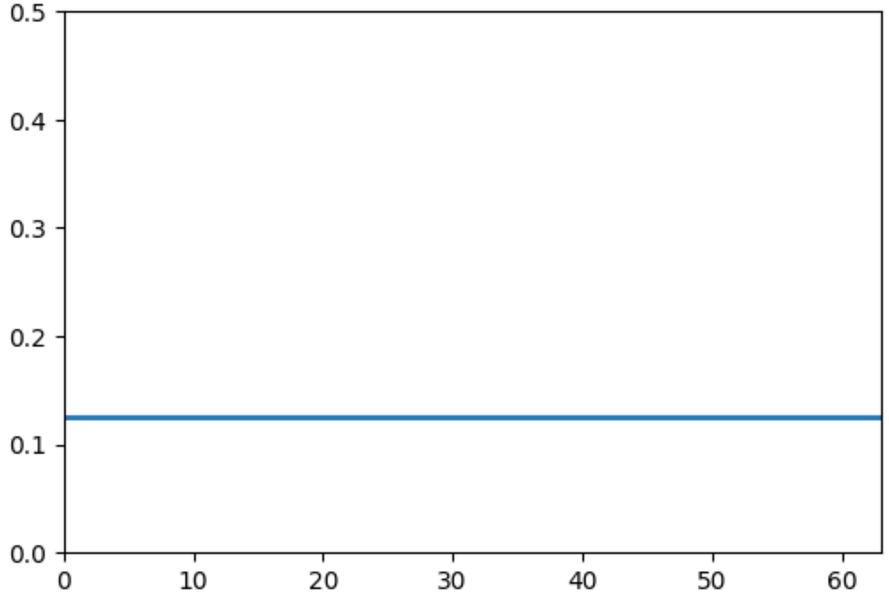




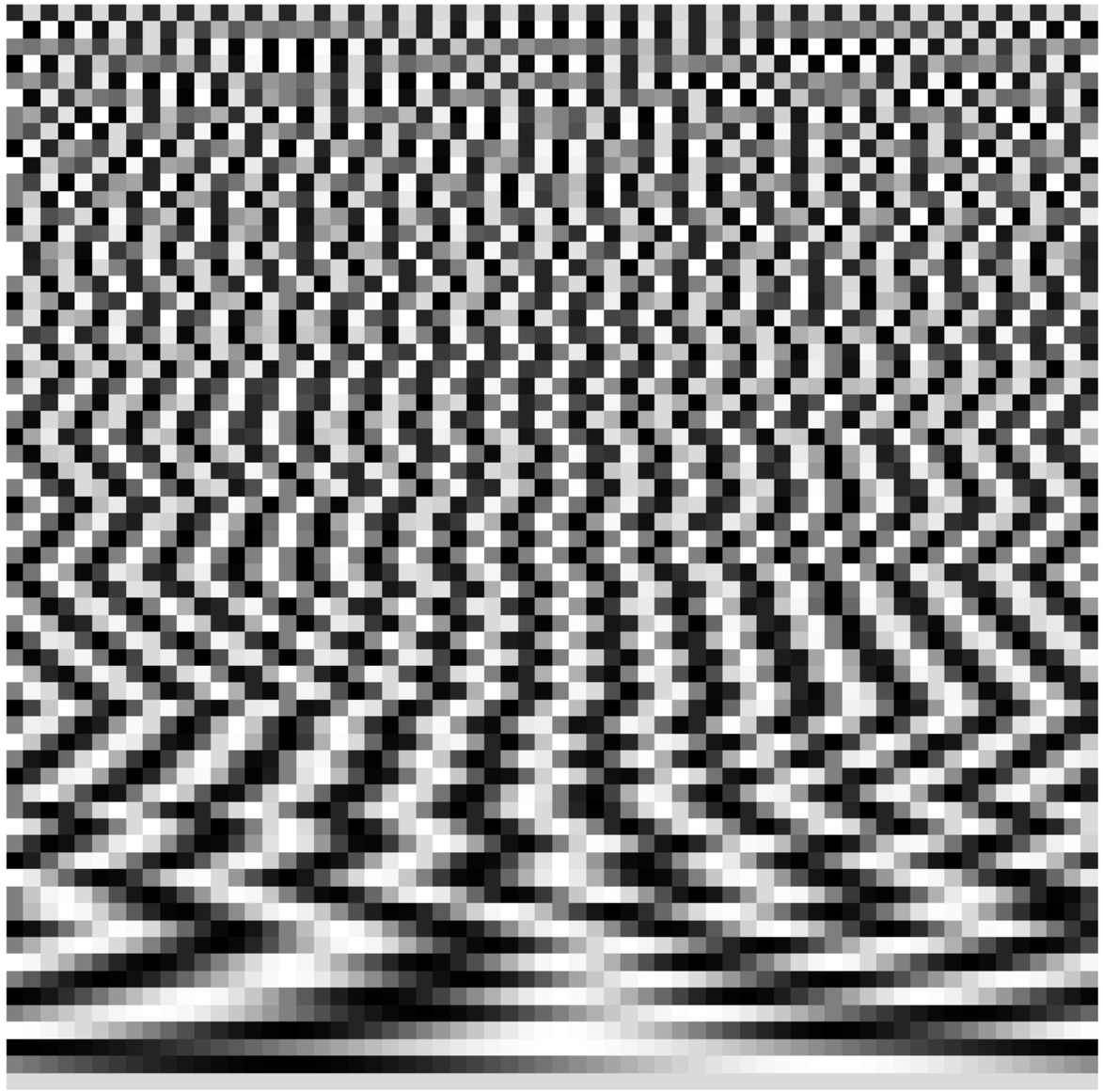
Eigenfunctions





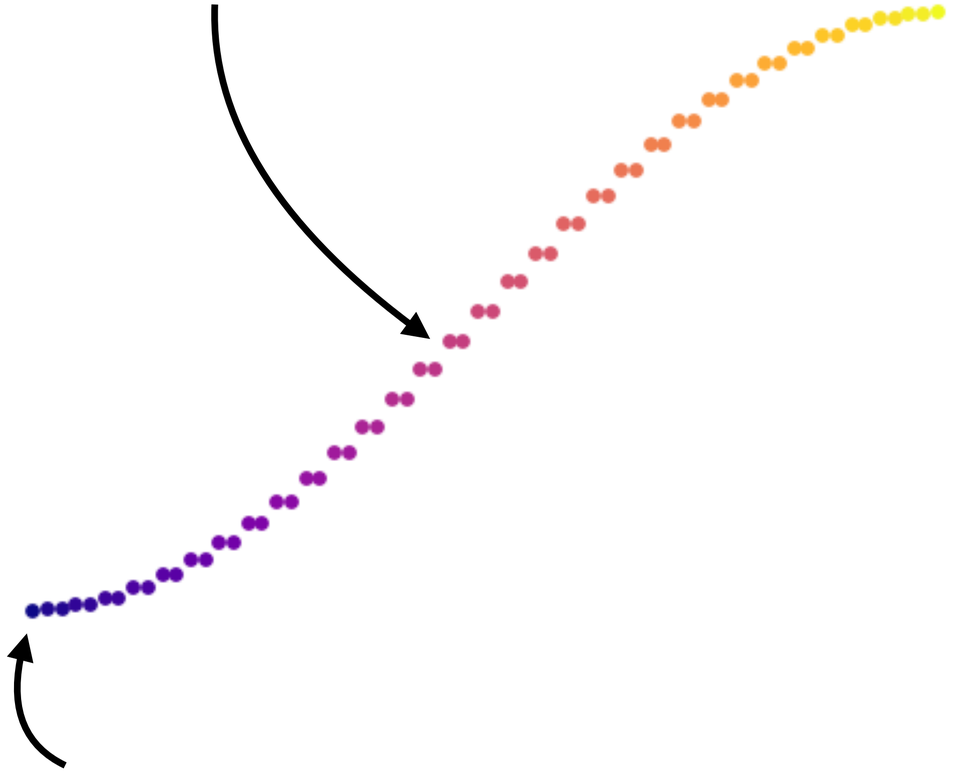


Eigenfunctions



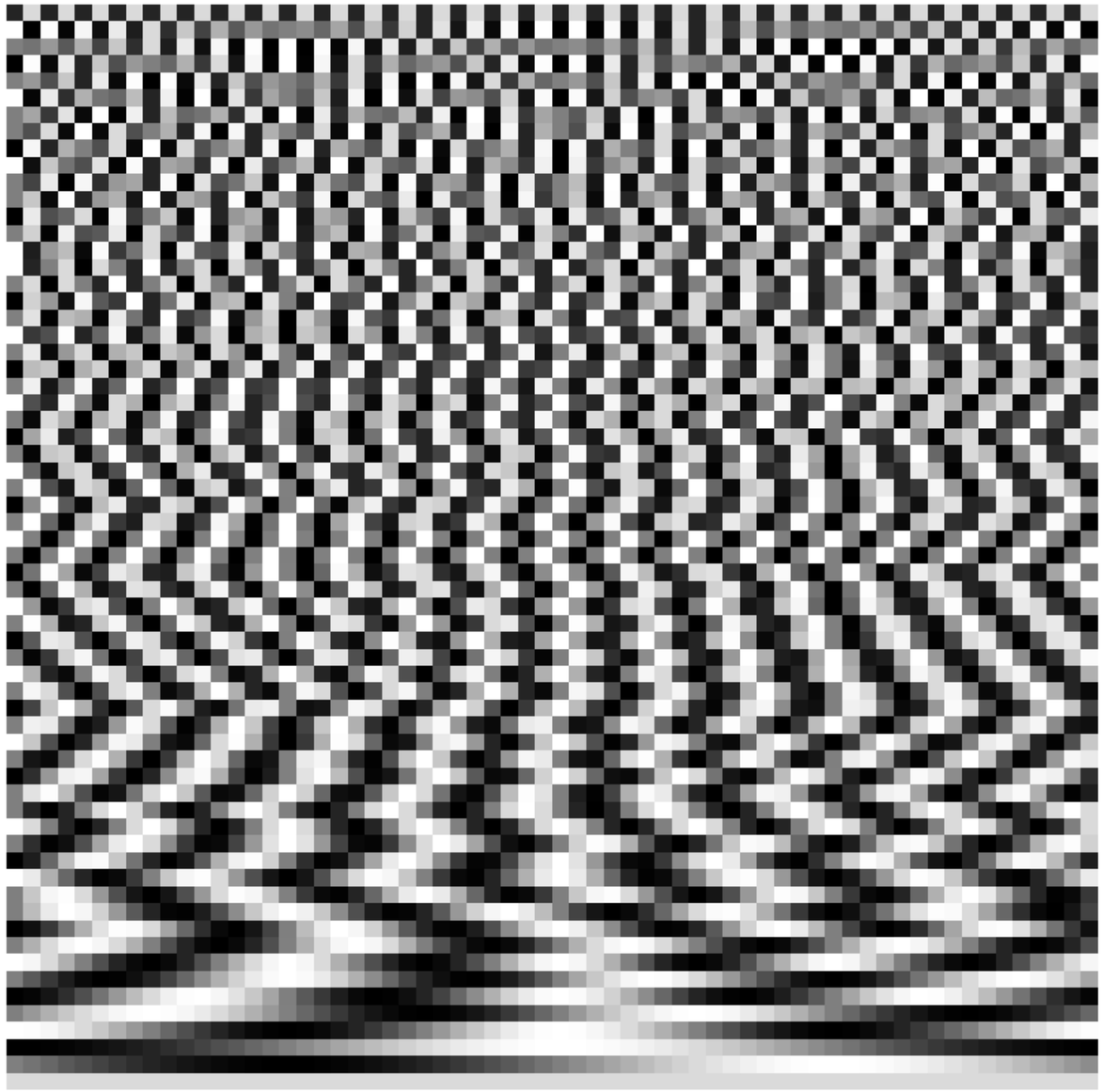
Eigenvalues

Then, pairs of equal eigenvalues



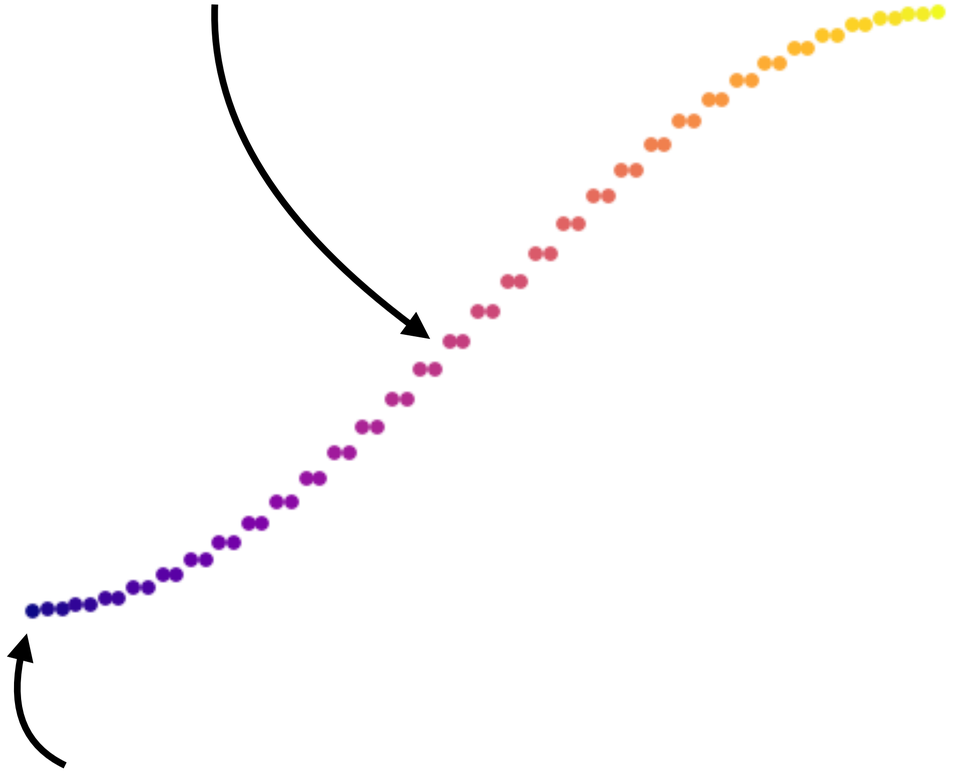
One “zero” eigenvalue

Eigenfunctions



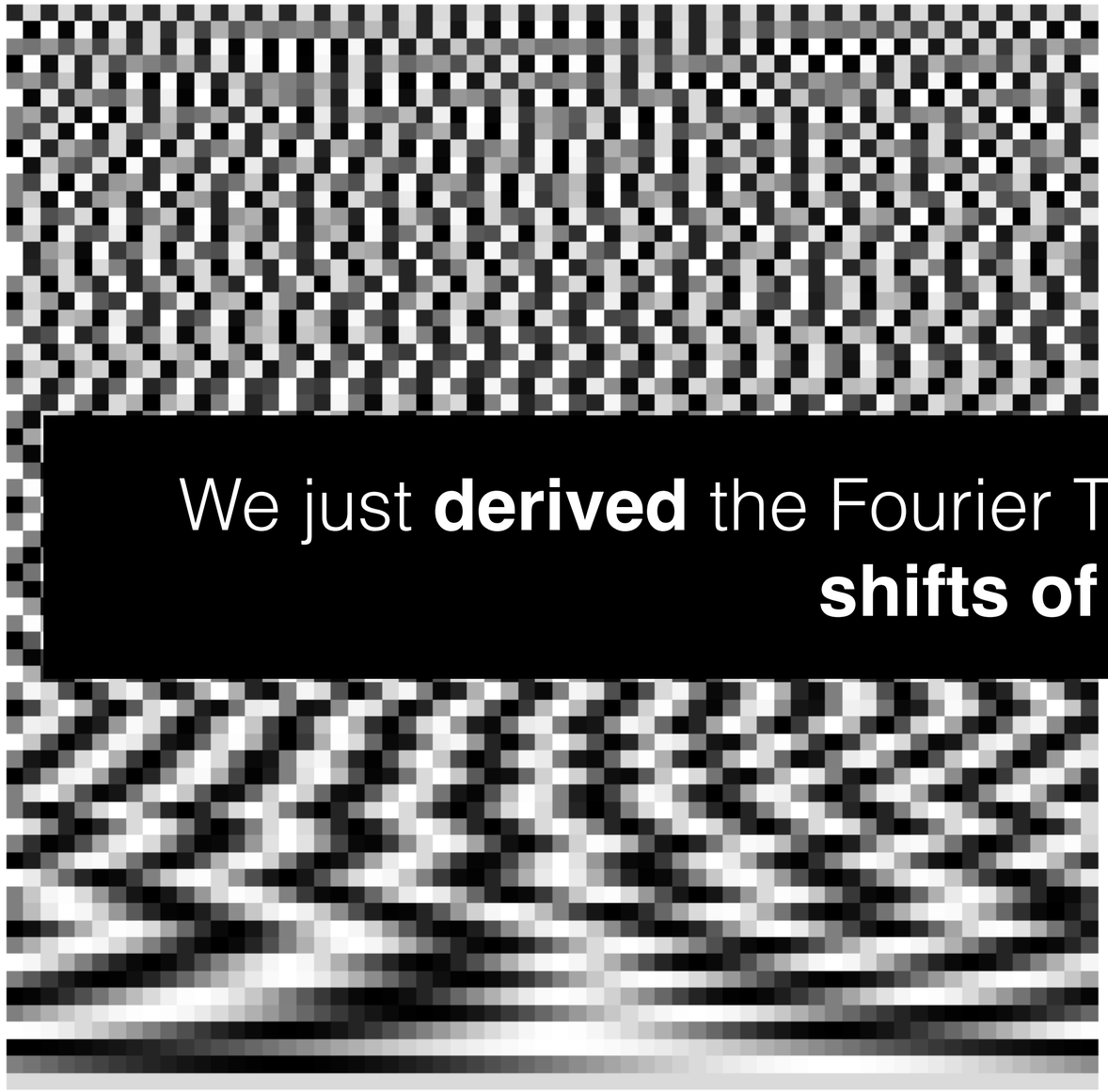
Eigenvalues

Then, pairs of equal eigenvalues



One “zero” eigenvalue

Eigenfunctions



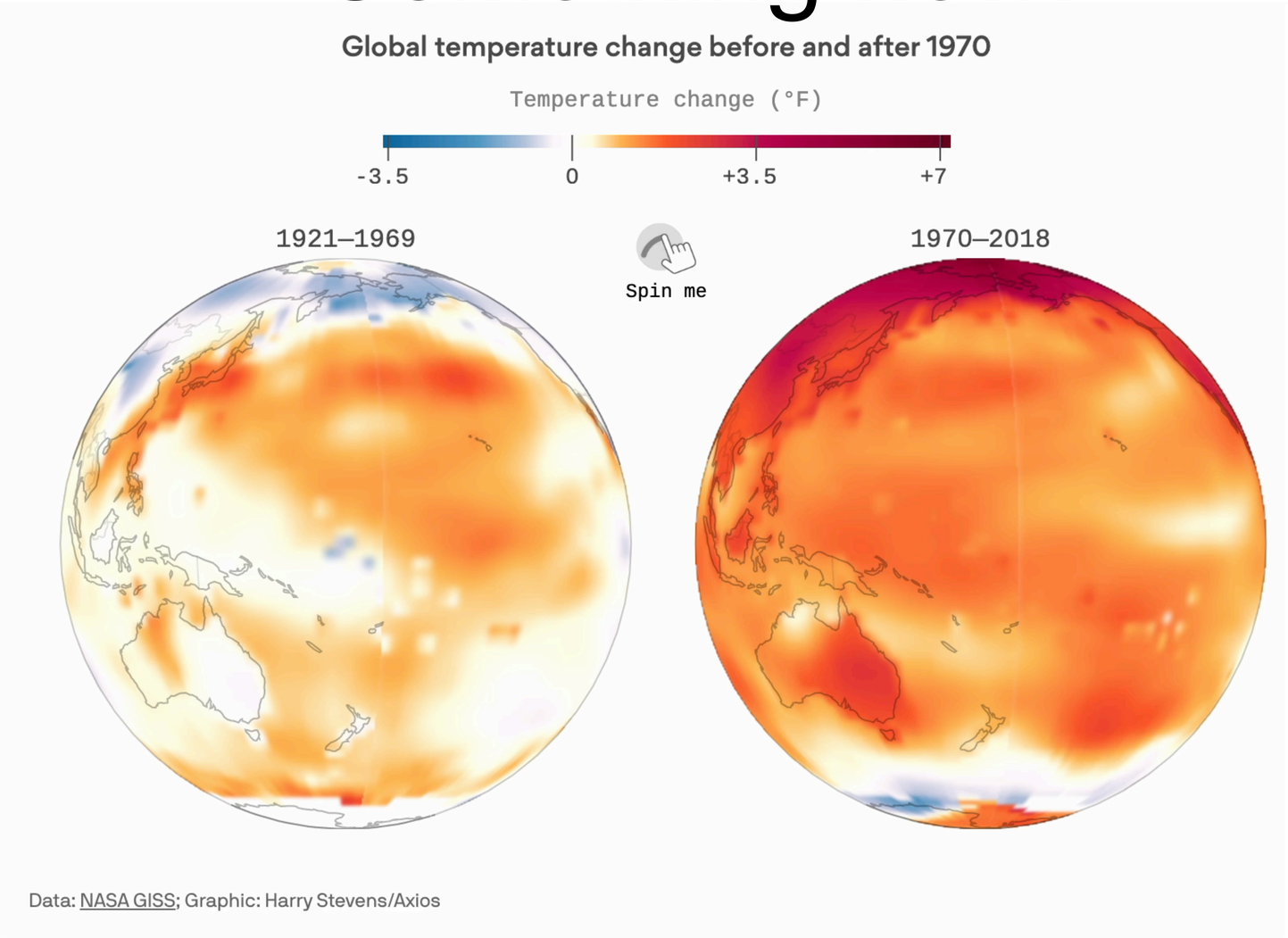
Eigenvalues

Then, pairs of equal eigenvalues

One “zero” eigenvalue

We just **derived** the Fourier Transform only from considering **shifts of functions**!

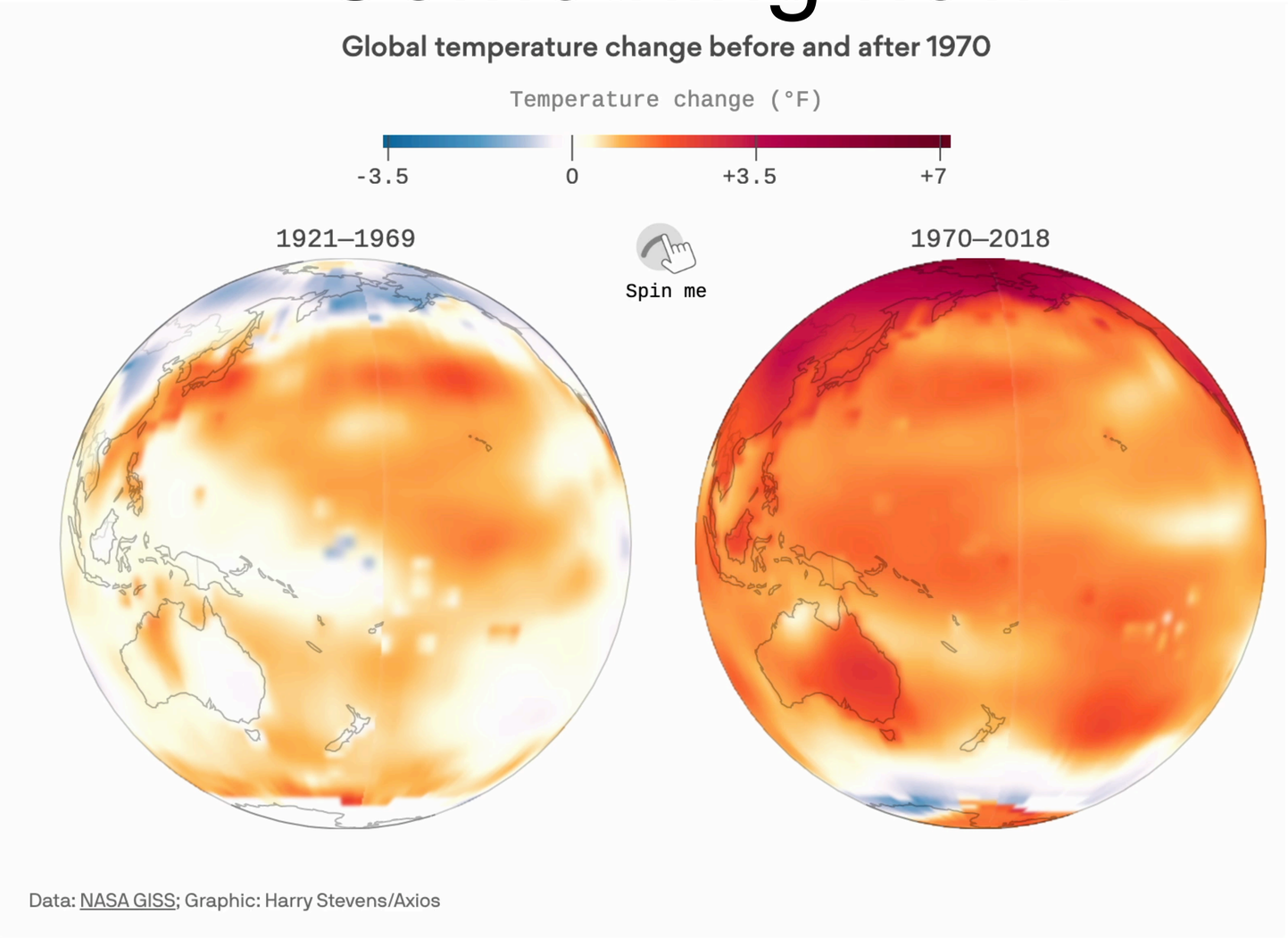
Something new!



Source: https://www.axios.com/2019/05/25/climate-change-earth-temperature-change

Is there a *rotation-steerable basis* for these functions?

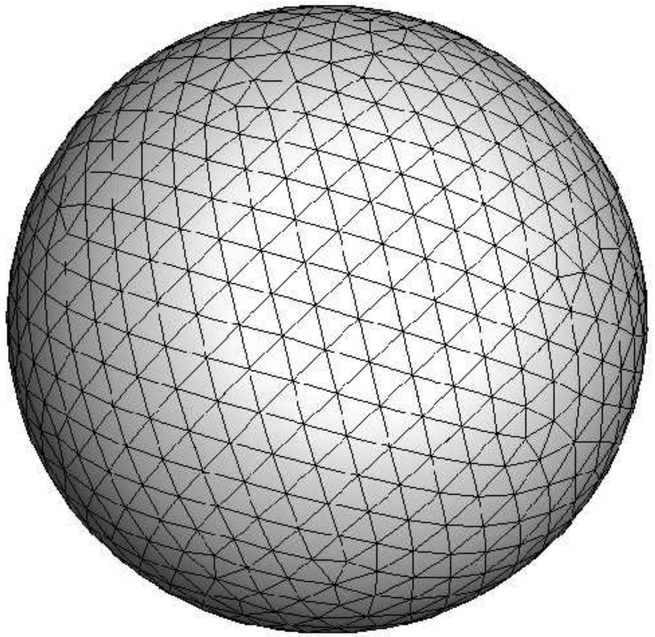
Something new!

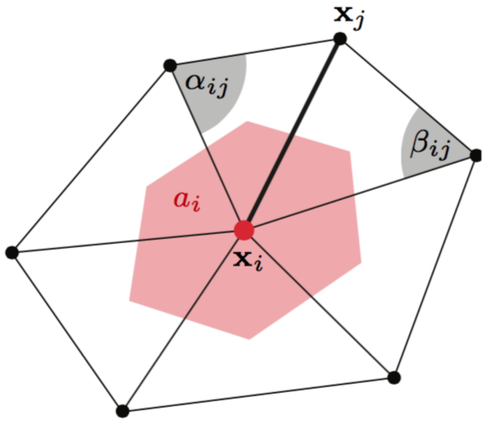


Source: https://www.axios.com/2019/05/25/climate-change-earth-temperature-change

Is there a *rotation-steerable basis* for these functions?

Cutting to the Chase: The Laplace-Beltrami Operator





$$(\Delta u)_i \approx \underbrace{\frac{1}{2\alpha_i}}_{M} \underbrace{\sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{x}_i - \mathbf{x}_j)}_{C}$$

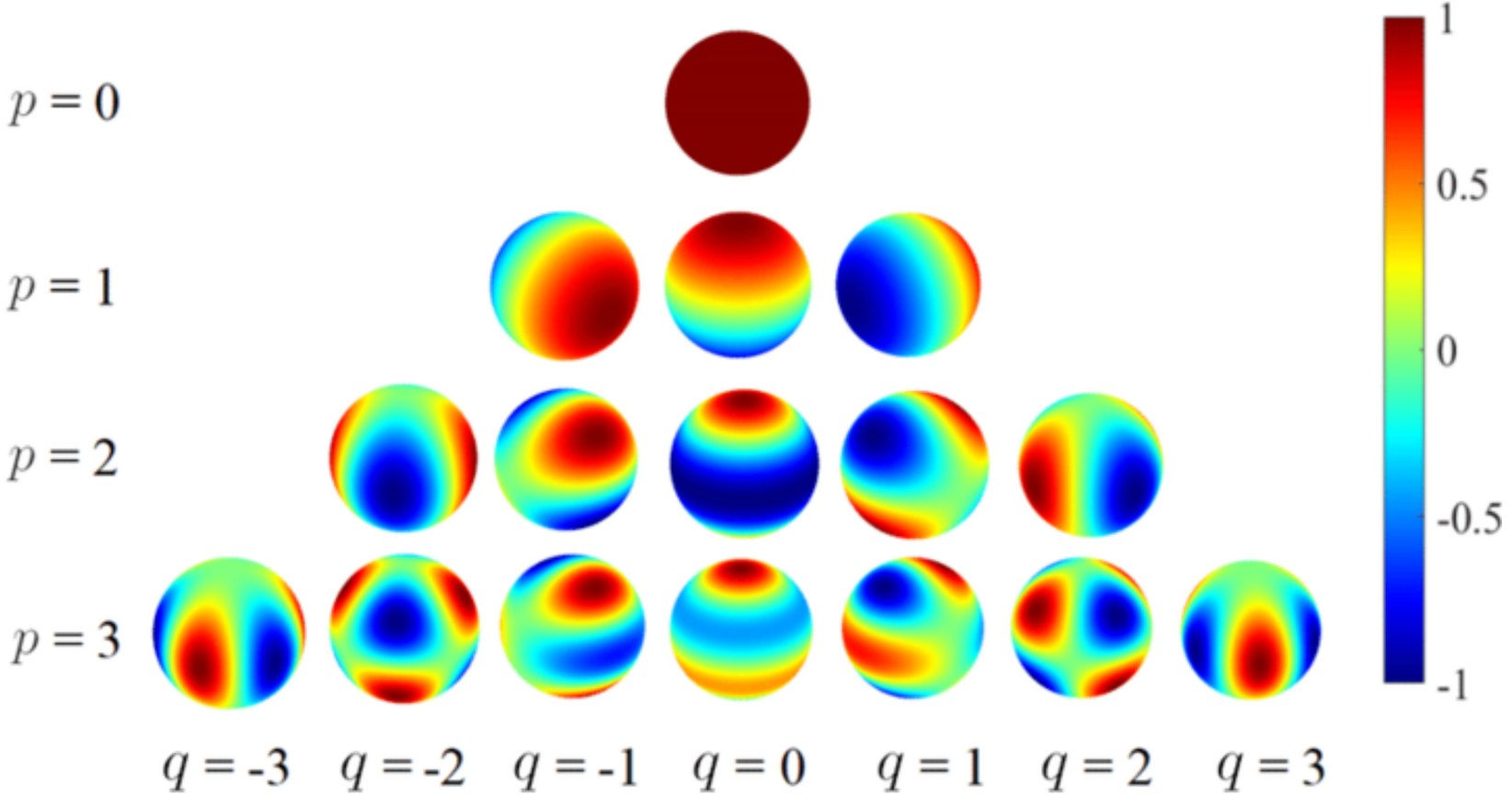
$$L = MC$$

Swiss Army Laplacian, Solomon, Crane and Vouga, https://www.cs.cmu.edu/~kmcrane/Projects/Other/SwissArmyLaplacian.pdf

user10024395 (https://math.stackexchange.com/users/139772/user10024395), How to intepret cotangent laplacian?, URL (version: 2017-11-09): https://math.stackexchange.com/q/2512014

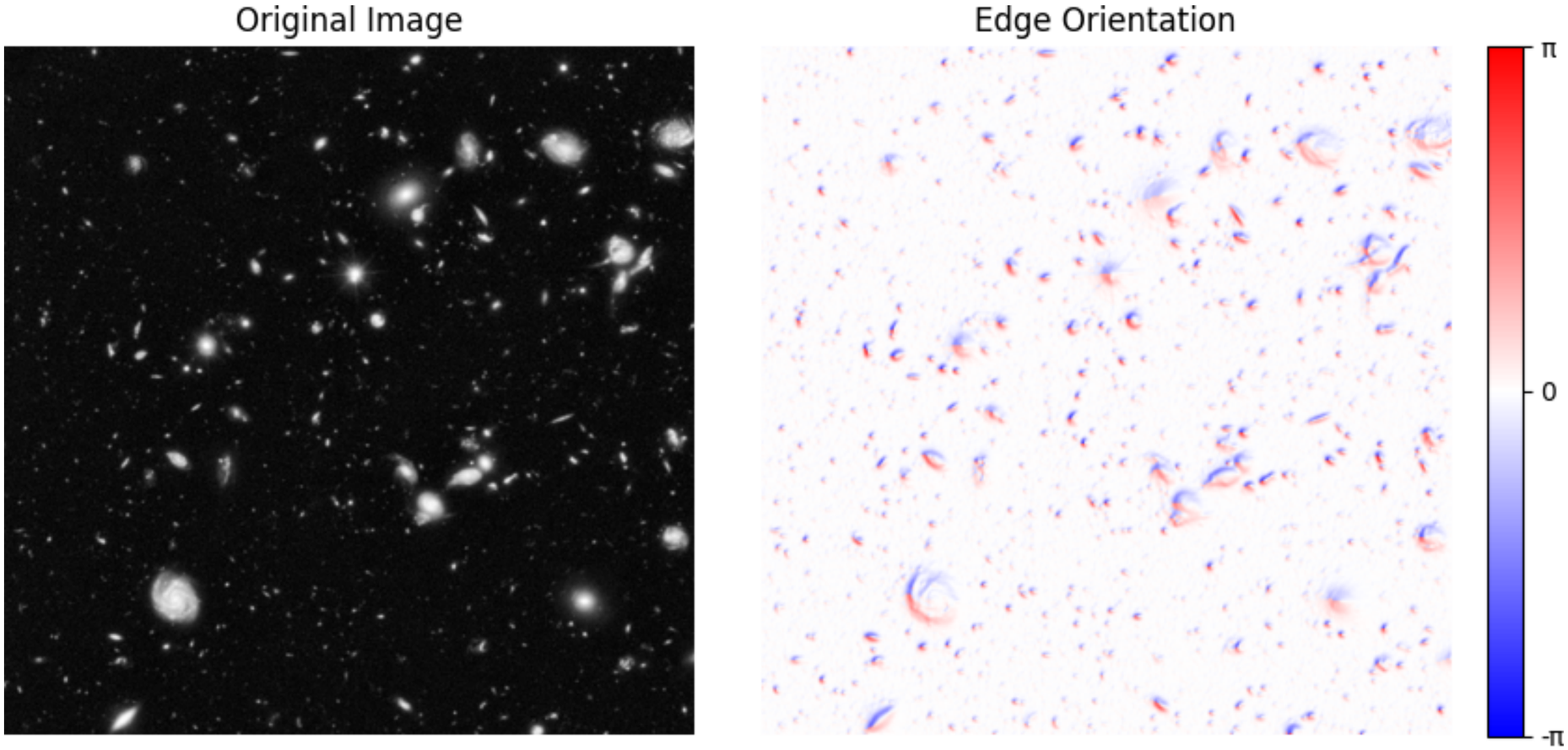
The Spherical Harmonics

1. A Fourier Basis for functions on the sphere!
2. Are rotation-steerable



PSet 2 is out!

https://github.com/6-8300/pset-2/



Fourier Transform, steerable bases, steerable filters!

Most Slides Adapted From:

UNIVERSITY OF AMSTERDAM

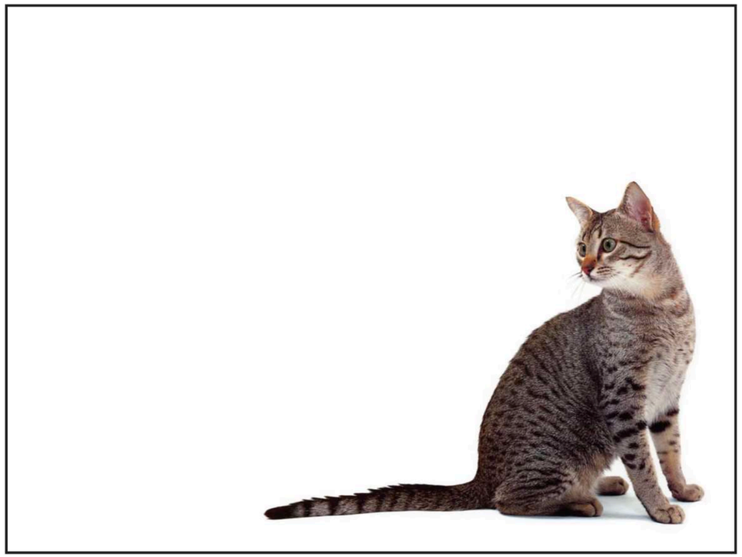
AMLAB Amsterdam Machine Learning Lab

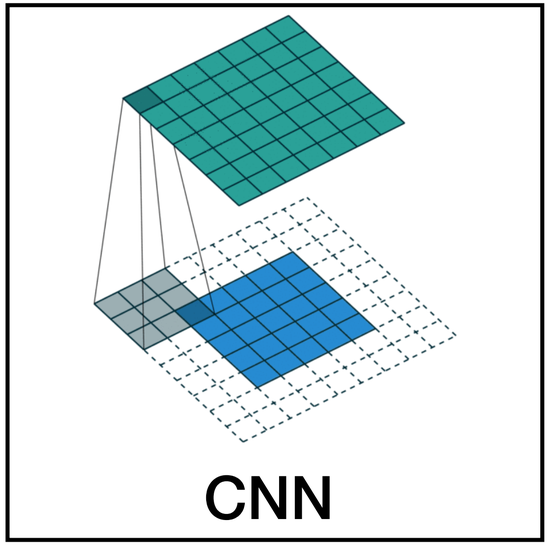
Group Equivariant Deep Learning

Erik Bekkers, Amsterdam Machine Learning Lab, University of Amsterdam
This mini-course serves as a module with the UvA Master AI course Deep Learning 2 https://uvadl2c.github.io/

Has some of the most amazing material on group theory for a computer science audience!

$\mathbb{R}^{H \times W \times 3}$





$\mathbb{R}^{H \times W \times N}$

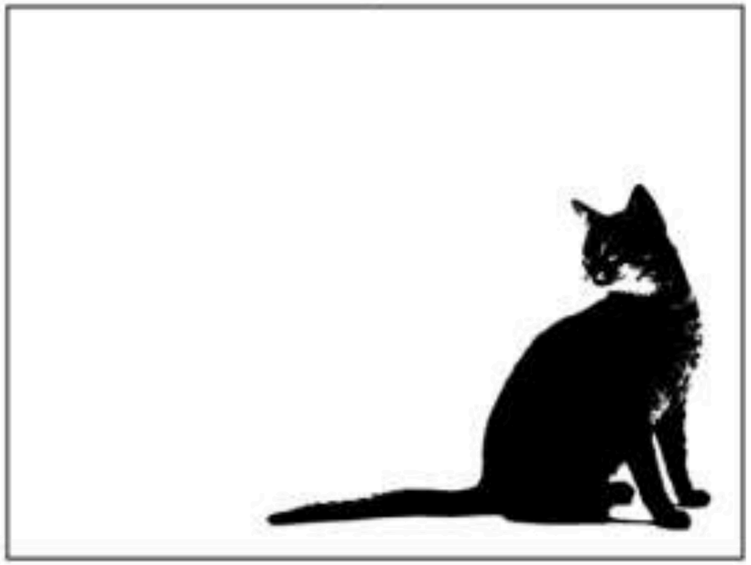
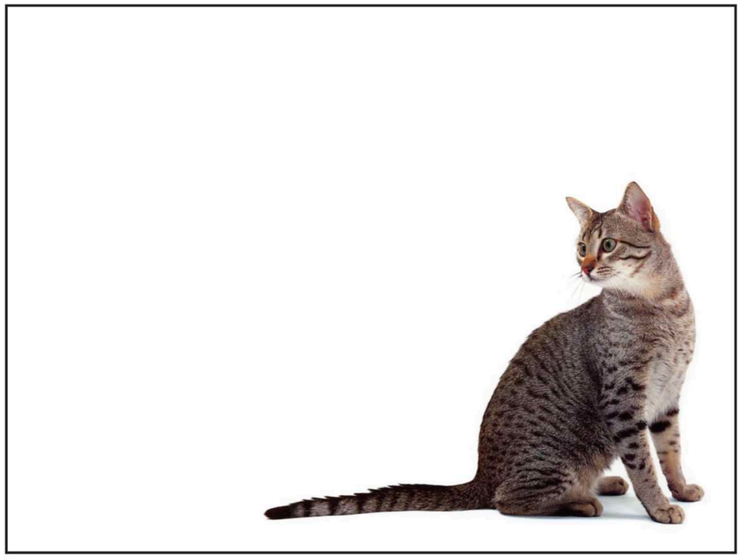
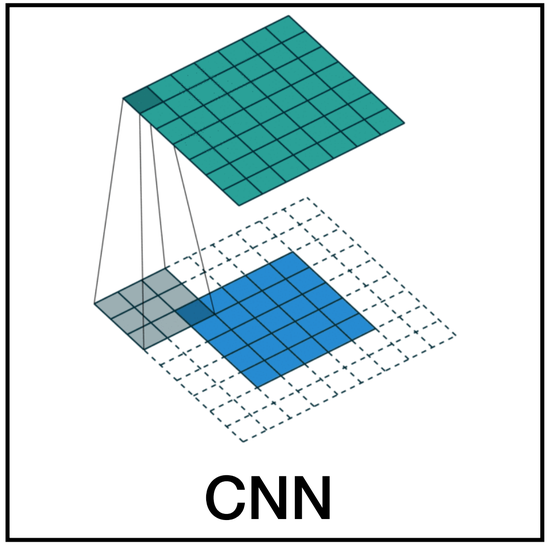


Image: Bernhard Kainz

$\mathbb{R}^{H \times W \times 3}$





$\mathbb{R}^{H \times W \times N}$

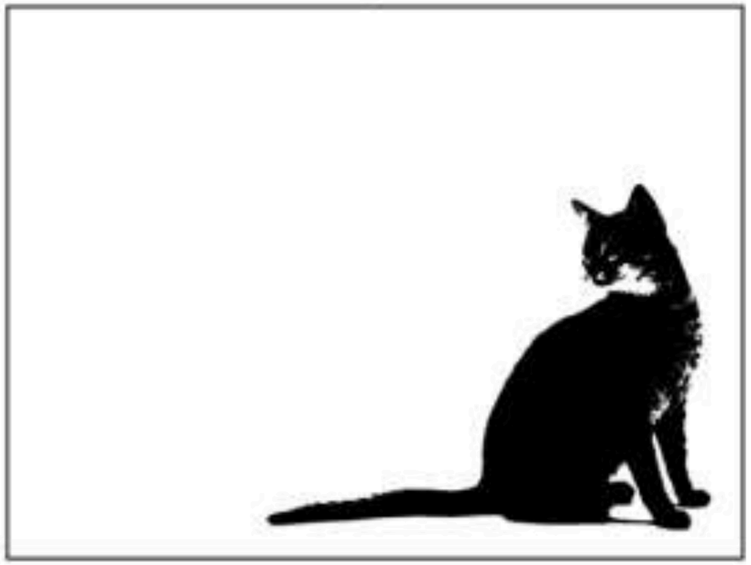


Image: Bernhard Kainz

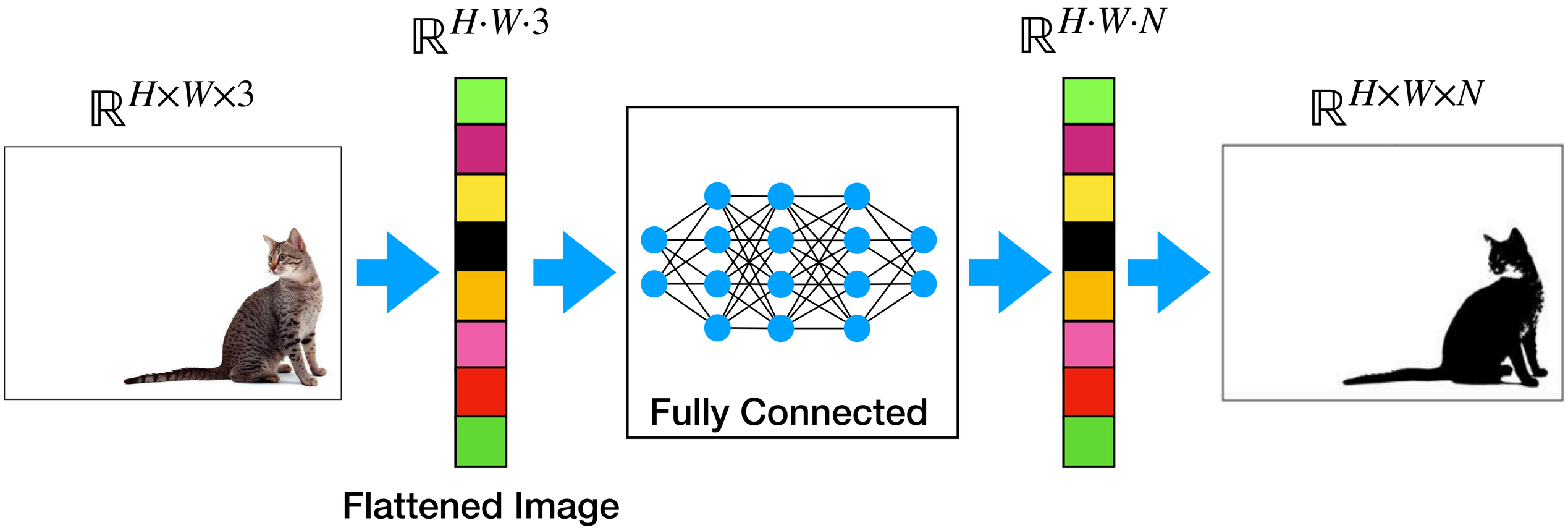


Image: Bernhard Kainz

Why is this a bad idea?

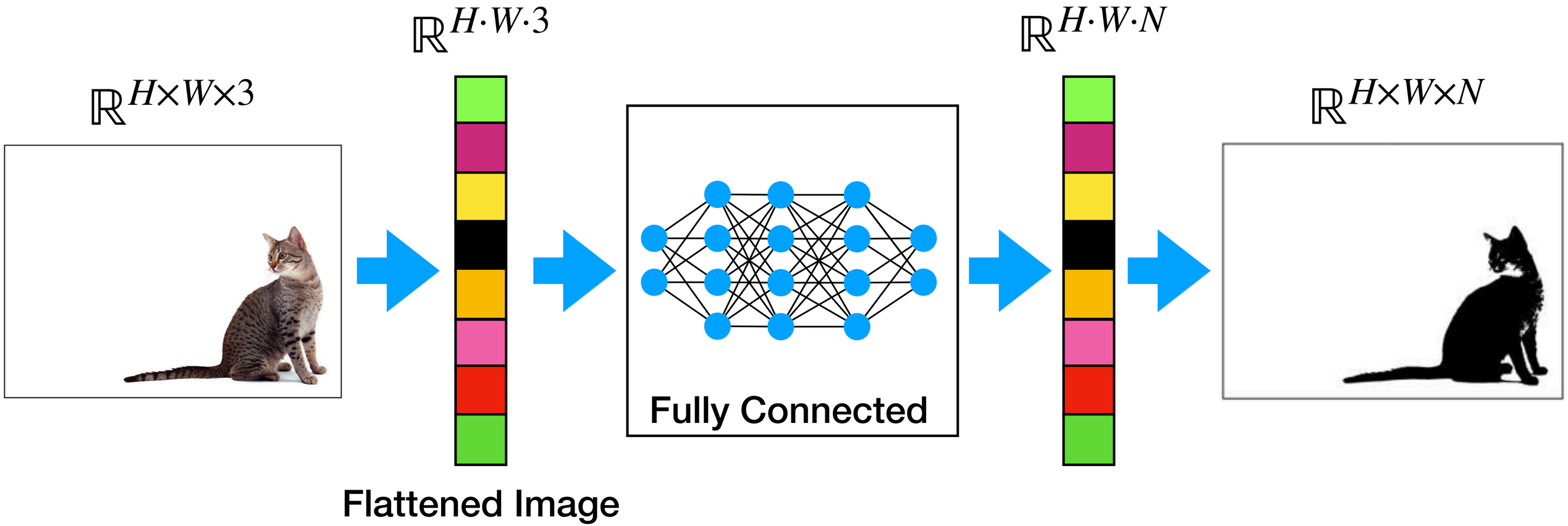


Image: Bernhard Kainz

Why is this a bad idea?

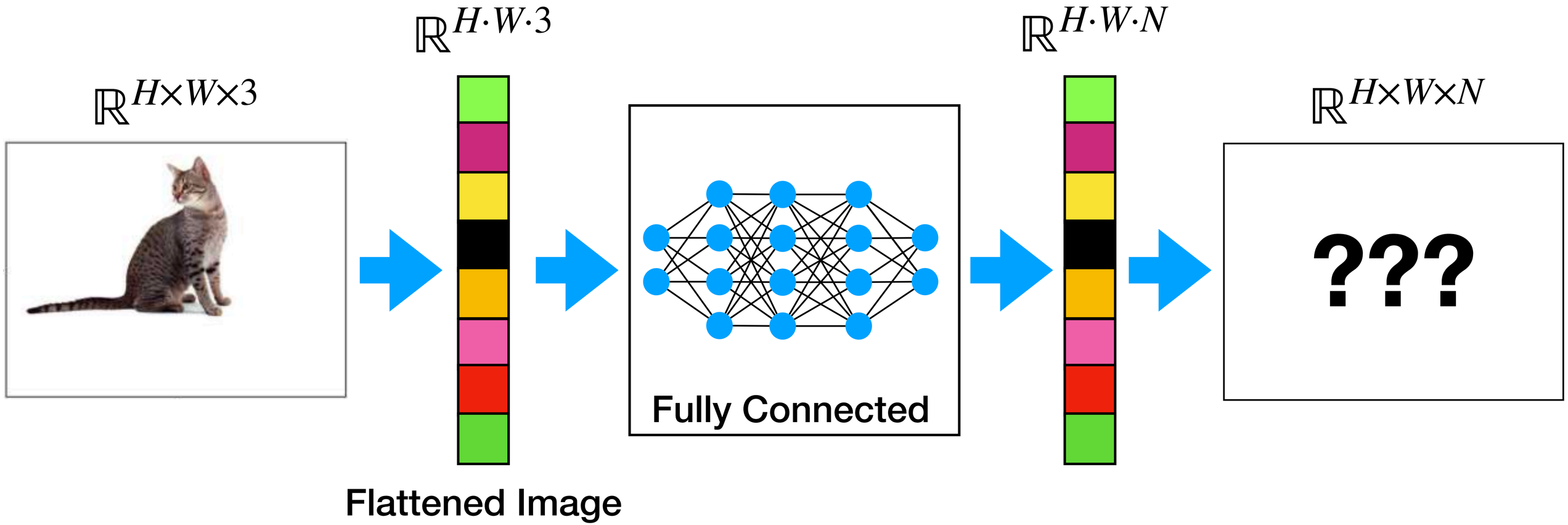
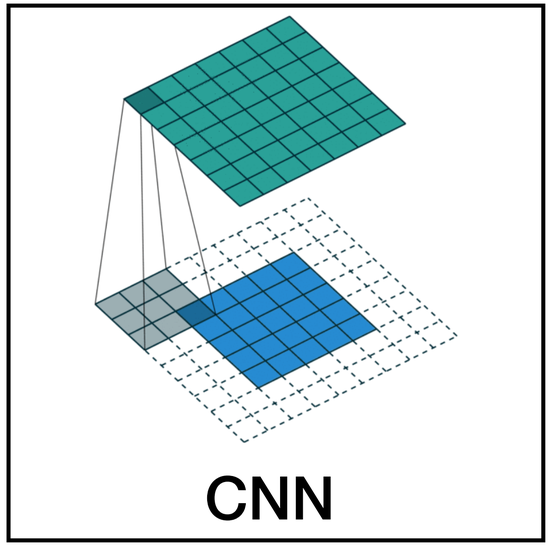


Image: Bernhard Kainz

$\mathbb{R}^{H \times W \times 3}$



$\mathbb{R}^{H \times W \times N}$

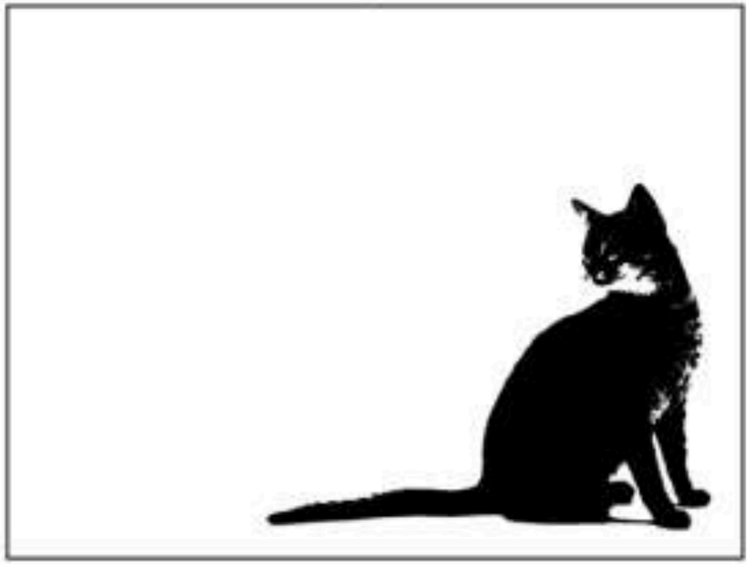
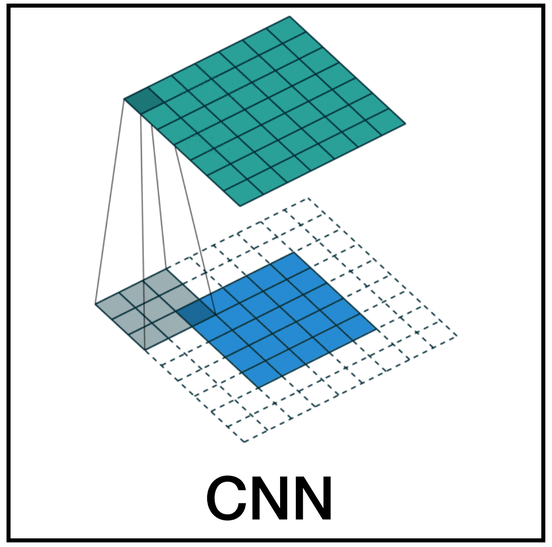


Image: Bernhard Kainz

$\mathbb{R}^{H \times W \times 3}$



$\mathbb{R}^{H \times W \times N}$

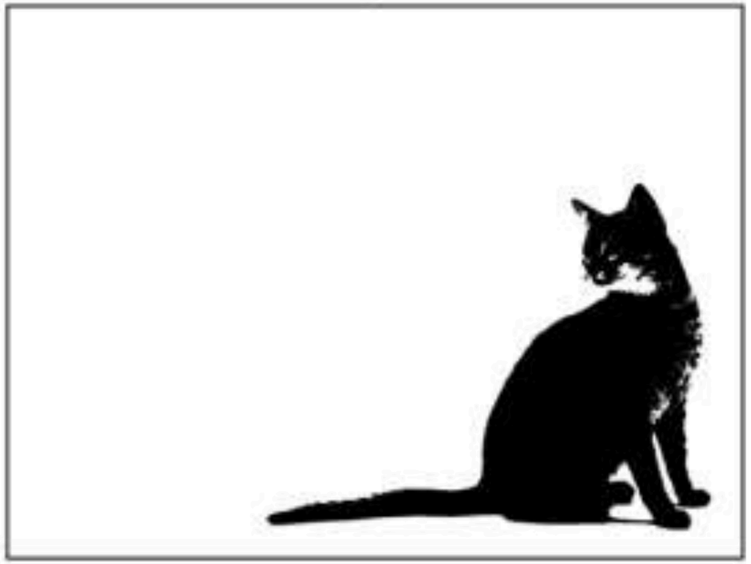
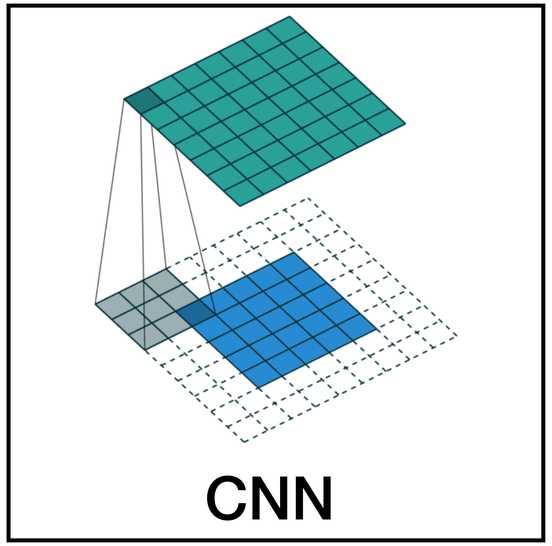
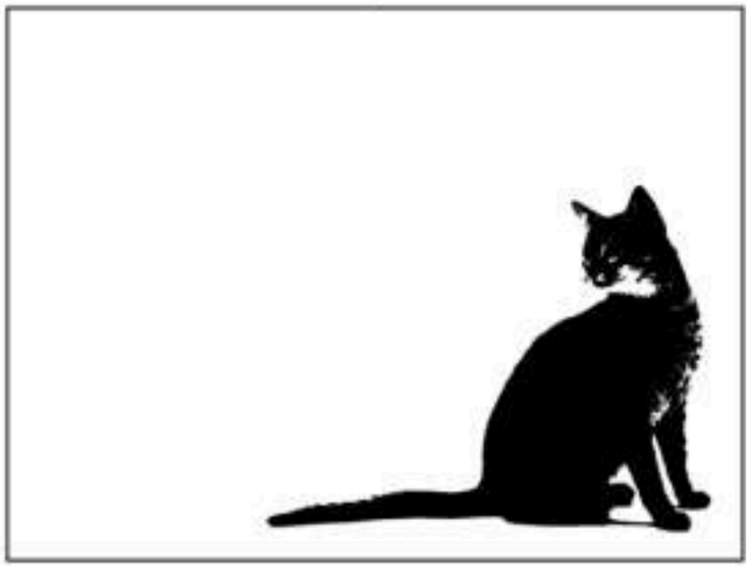


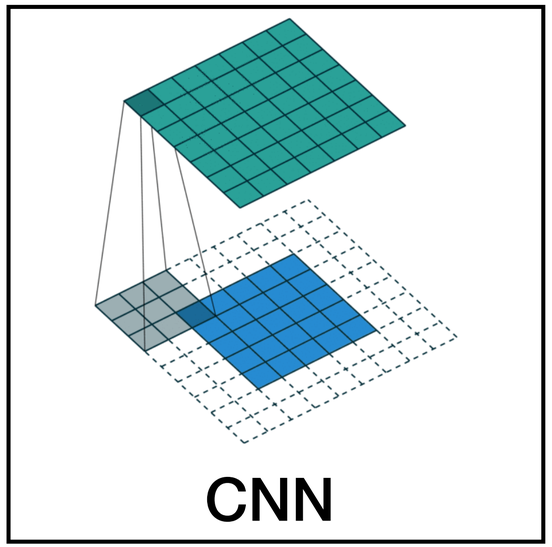
Image: Bernhard Kainz

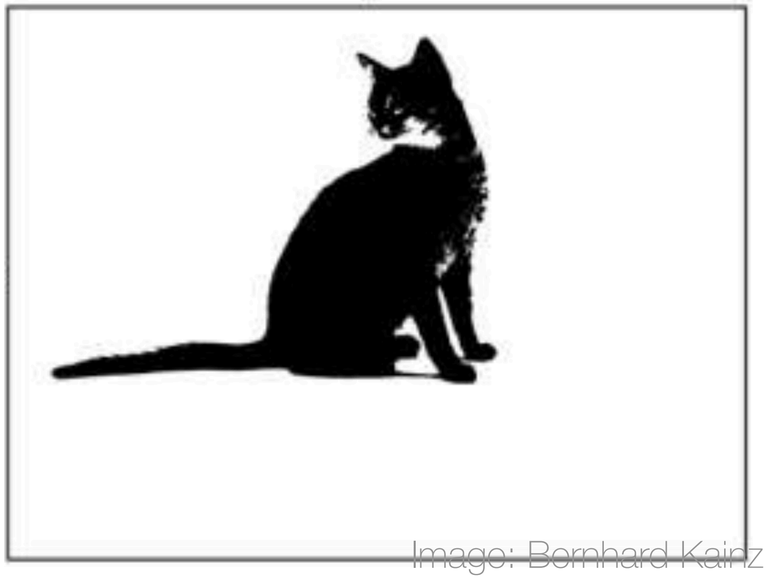
$\mathbb{R}^{H \times W \times 3}$



$\mathbb{R}^{H \times W \times N}$

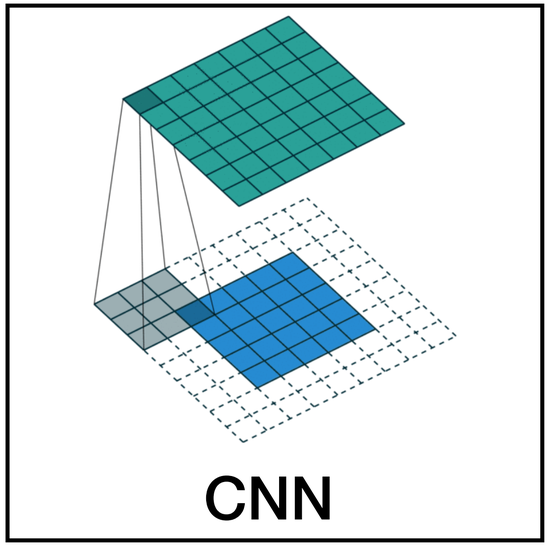




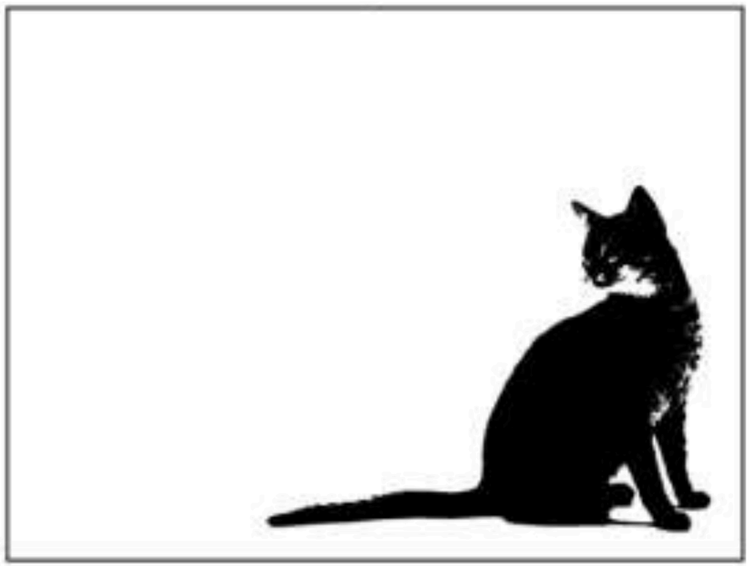


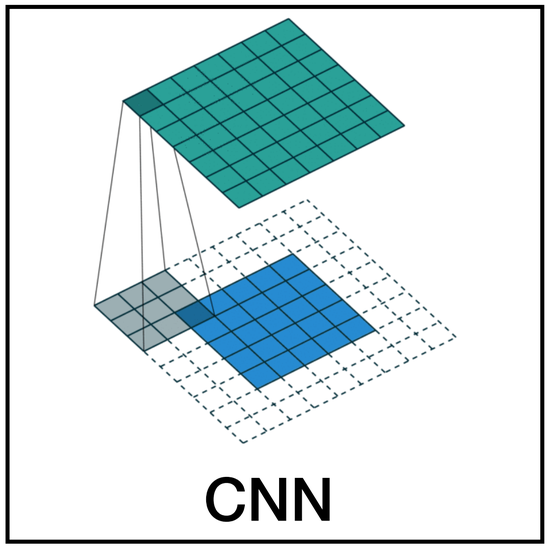
“Shift Equivariance”

$\mathbb{R}^{H \times W \times 3}$



$\mathbb{R}^{H \times W \times N}$





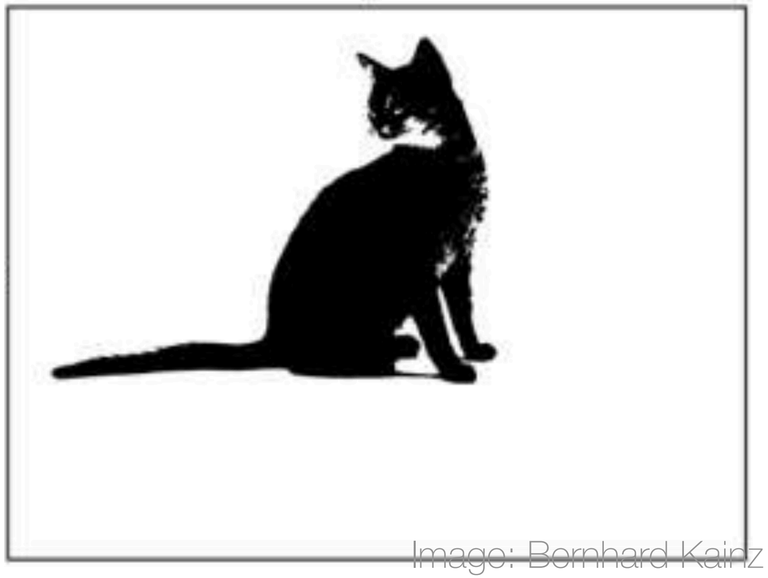


Image: Bernhard Kainz

Recall: Group Representations

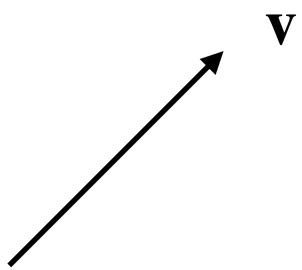
A representation $\rho : G \rightarrow GL(V)$ is a function from G to the general linear group $GL(V)$.

That is $\rho(g)$ is a linear transformation that is parameterized by group elements $g \in G$ that transforms some vector $\mathbf{v} \in V$ (e.g. an image) such that

$$\rho(g') \circ \rho(g)[\mathbf{v}] = \rho(g' \cdot g)\,[\mathbf{v}]$$

Slide credit: Erik Bekkers, University of Amsterdam

Recall: Group Representations



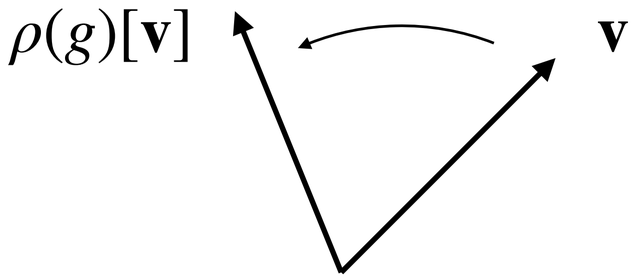
A representation $\rho : G \rightarrow GL(V)$ is a function from G to the general linear group $GL(V)$.

That is $\rho(g)$ is a linear transformation that is parameterized by group elements $g \in G$ that transforms some vector $\mathbf{v} \in V$ (e.g. an image) such that

$$\rho(g') \circ \rho(g)[\mathbf{v}] = \rho(g' \cdot g)\,[\mathbf{v}]$$

Slide credit: Erik Bekkers, University of Amsterdam

Recall: Group Representations



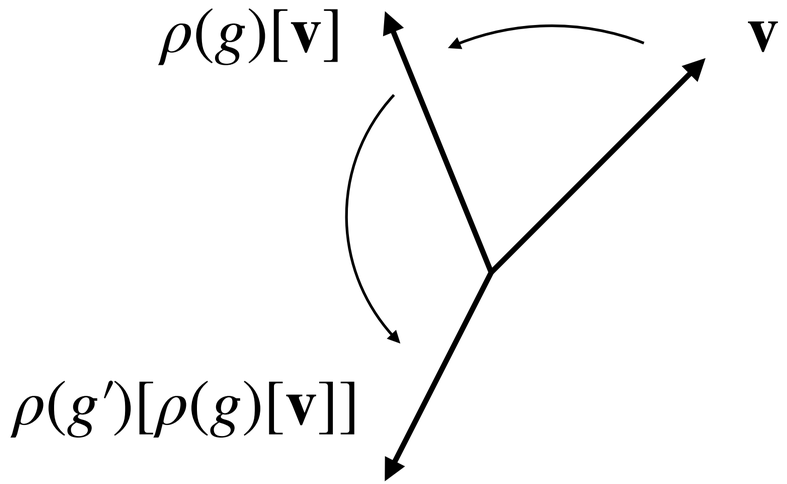
A representation $\rho : G \rightarrow GL(V)$ is a function from G to the general linear group $GL(V)$.

That is $\rho(g)$ is a linear transformation that is parameterized by group elements $g \in G$ that transforms some vector $\mathbf{v} \in V$ (e.g. an image) such that

$$\rho(g') \circ \rho(g)[\mathbf{v}] = \rho(g' \cdot g)\,[\mathbf{v}]$$

Slide credit: Erik Bekkers, University of Amsterdam

Recall: Group Representations



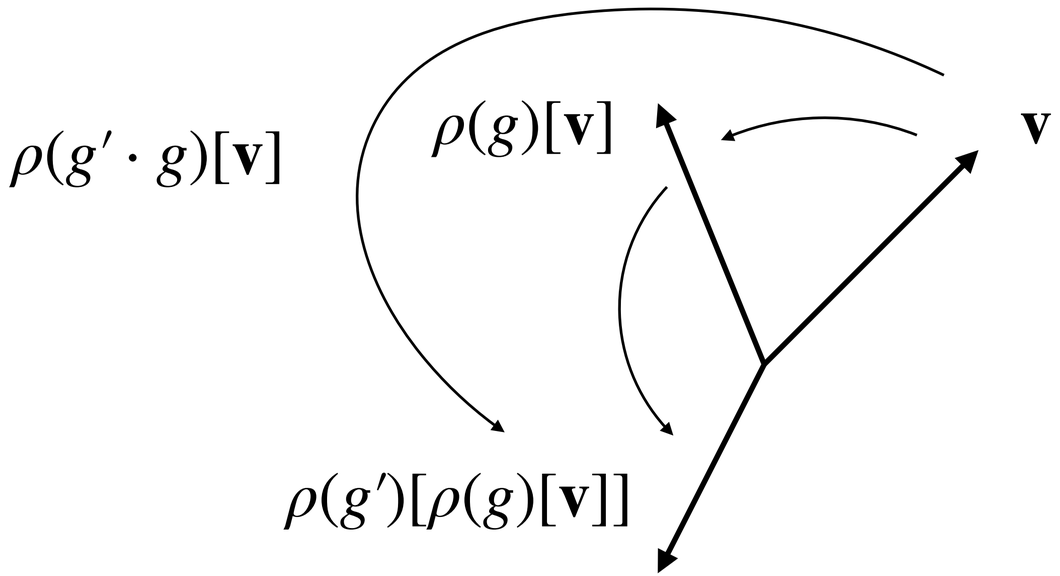
A representation $\rho : G \rightarrow GL(V)$ is a function from G to the general linear group $GL(V)$.

That is $\rho(g)$ is a linear transformation that is parameterized by group elements $g \in G$ that transforms some vector $\mathbf{v} \in V$ (e.g. an image) such that

$$\rho(g') \circ \rho(g)[\mathbf{v}] = \rho(g' \cdot g)\,[\mathbf{v}]$$

Slide credit: Erik Bekkers, University of Amsterdam

Recall: Group Representations



A representation $\rho : G \rightarrow GL(V)$ is a function from G to the general linear group $GL(V)$.

That is $\rho(g)$ is a linear transformation that is parameterized by group elements $g \in G$ that transforms some vector $\mathbf{v} \in V$ (e.g. an image) such that

$$\rho(g') \circ \rho(g)[\mathbf{v}] = \rho(g' \cdot g)\,[\mathbf{v}]$$

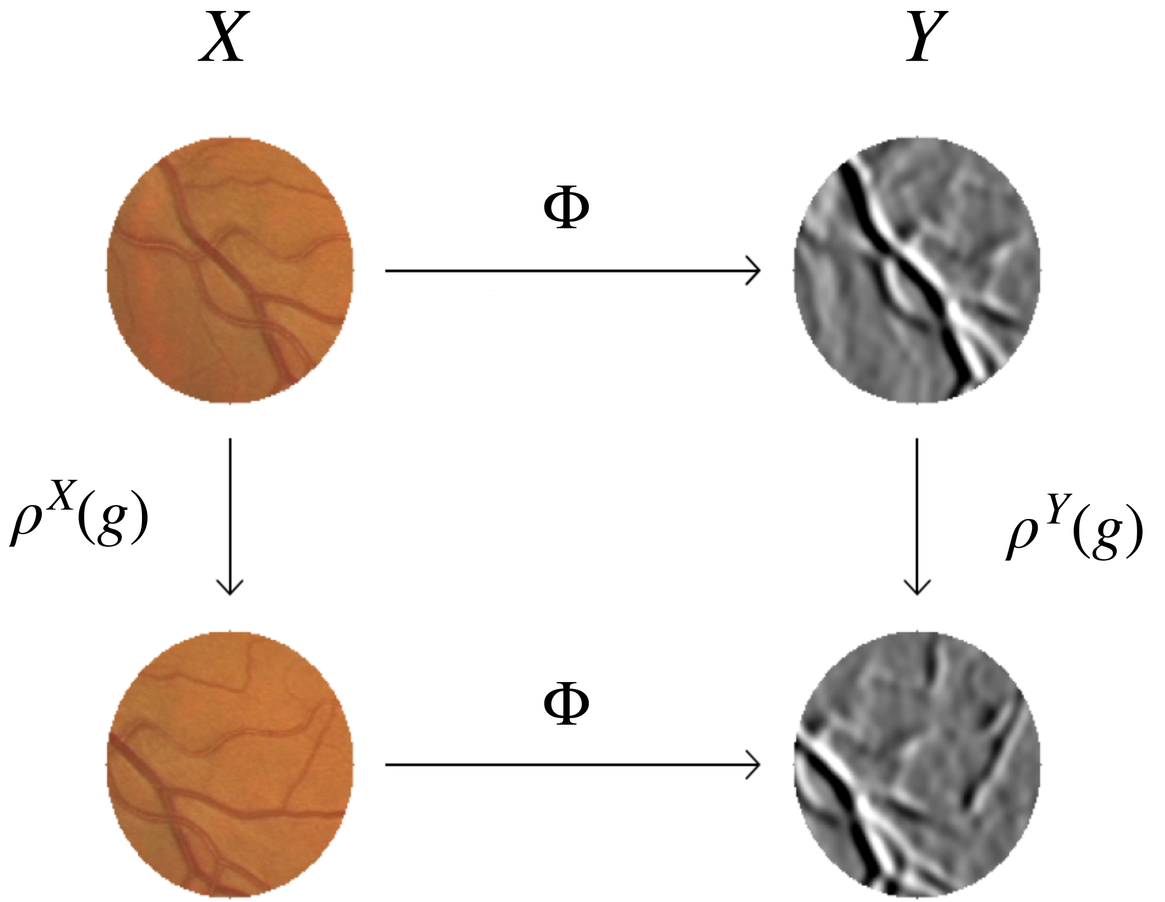
Slide credit: Erik Bekkers, University of Amsterdam

Equivariance

Equivariance is a property of a function $\Phi : X \to Y$ (such as a neural network layer) by which it commutes with the group action:

$$\rho^Y(g) \circ \Phi = \Phi \circ \rho^X(g)$$

group representation action on X

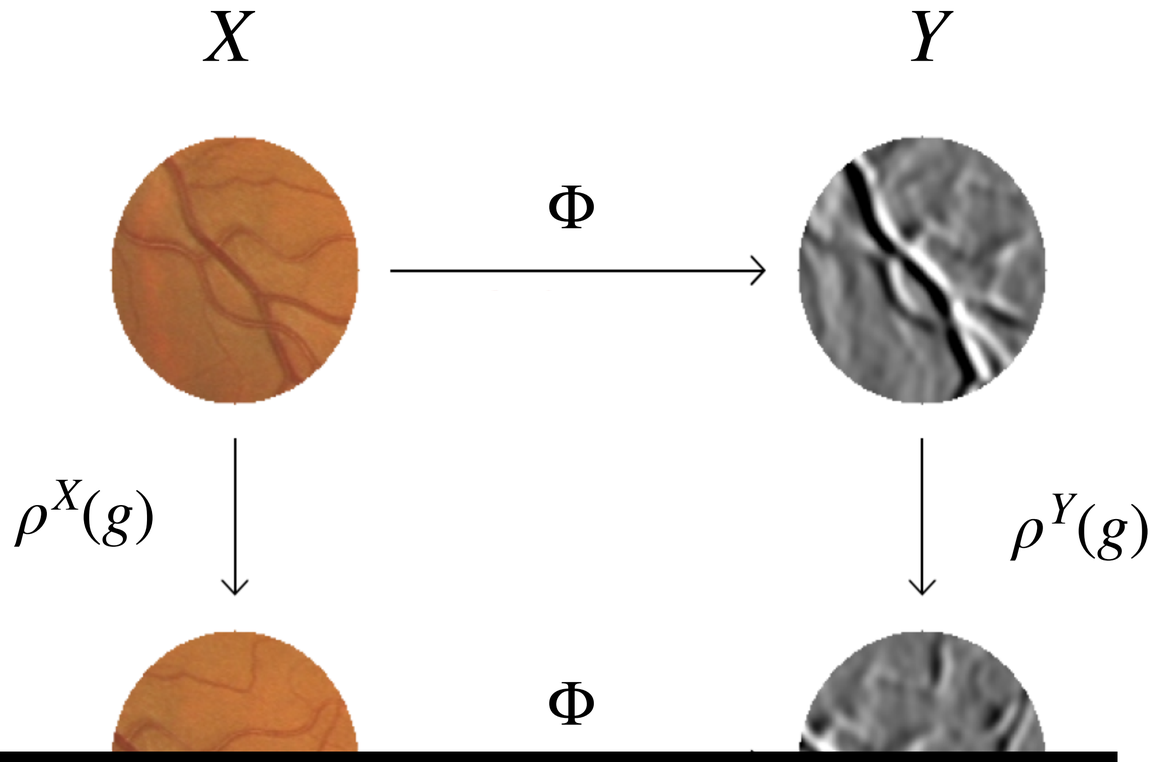


Slide credit: Erik Bekkers, University of Amsterdam

Equivariance

Equivariance is a property of a function $\Phi : X \to Y$ (such as a neural network layer) by which it commutes with the group action:

$$\rho^Y(g) \circ \Phi = \Phi \circ \rho^X(g)$$



gro

A neural network layer is equivariant wrt to a group G if it commutes with the corresponding transformation

Slide credit: Erik Bekkers, University of Amsterdam

Invariance

Invariance is the special case where the action acting on output is *identity:*

$$\Phi = \Phi \circ \rho^{X}(g)$$

Example: Convolution with rotation-symmetric kernel is SO(2) invariant

Slide credit: Erik Bekkers, University of Amsterdam

Invariance

Invariance is the special case where the action acting on output is *identity:*

$$\Phi = \Phi \circ \rho^X(g)$$

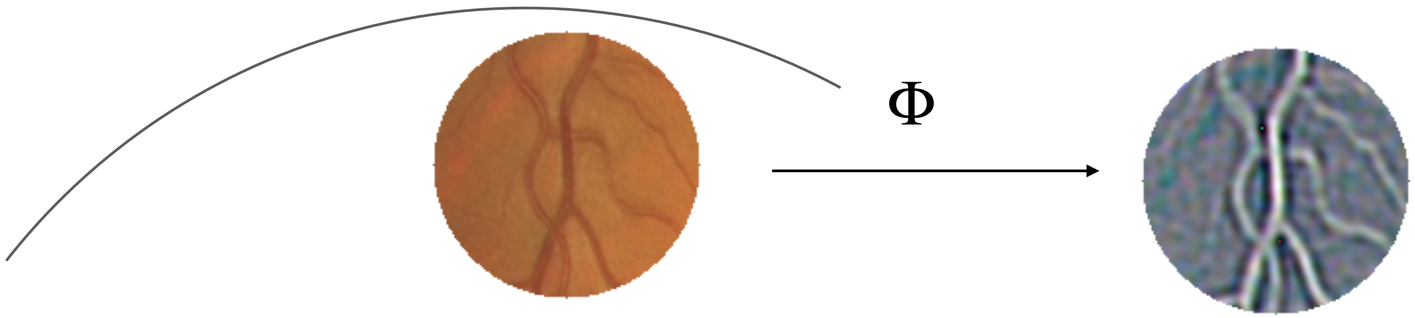
Example: Convolution with rotation-symmetric kernel is SO(2) invariant

Slide credit: Erik Bekkers, University of Amsterdam

Invariance

Invariance is the special case where the action acting on output is *identity:*

$$\Phi = \Phi \circ \rho^{X}(g)$$



Conv2D(input ,)

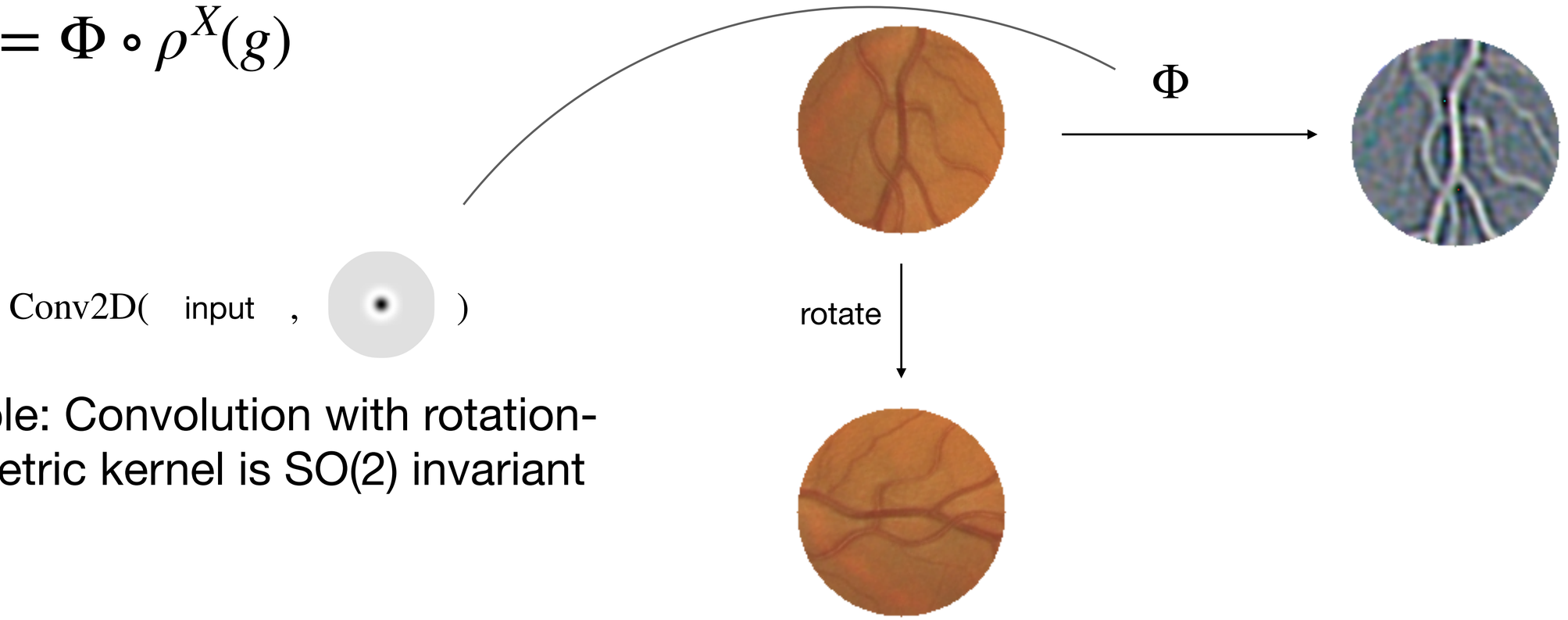
Example: Convolution with rotation-symmetric kernel is SO(2) invariant

Slide credit: Erik Bekkers, University of Amsterdam

Invariance

Invariance is the special case where the action acting on output is *identity:*

$$\Phi = \Phi \circ \rho^X(g)$$



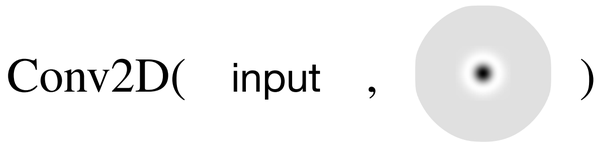
Example: Convolution with rotation-symmetric kernel is SO(2) invariant

Slide credit: Erik Bekkers, University of Amsterdam

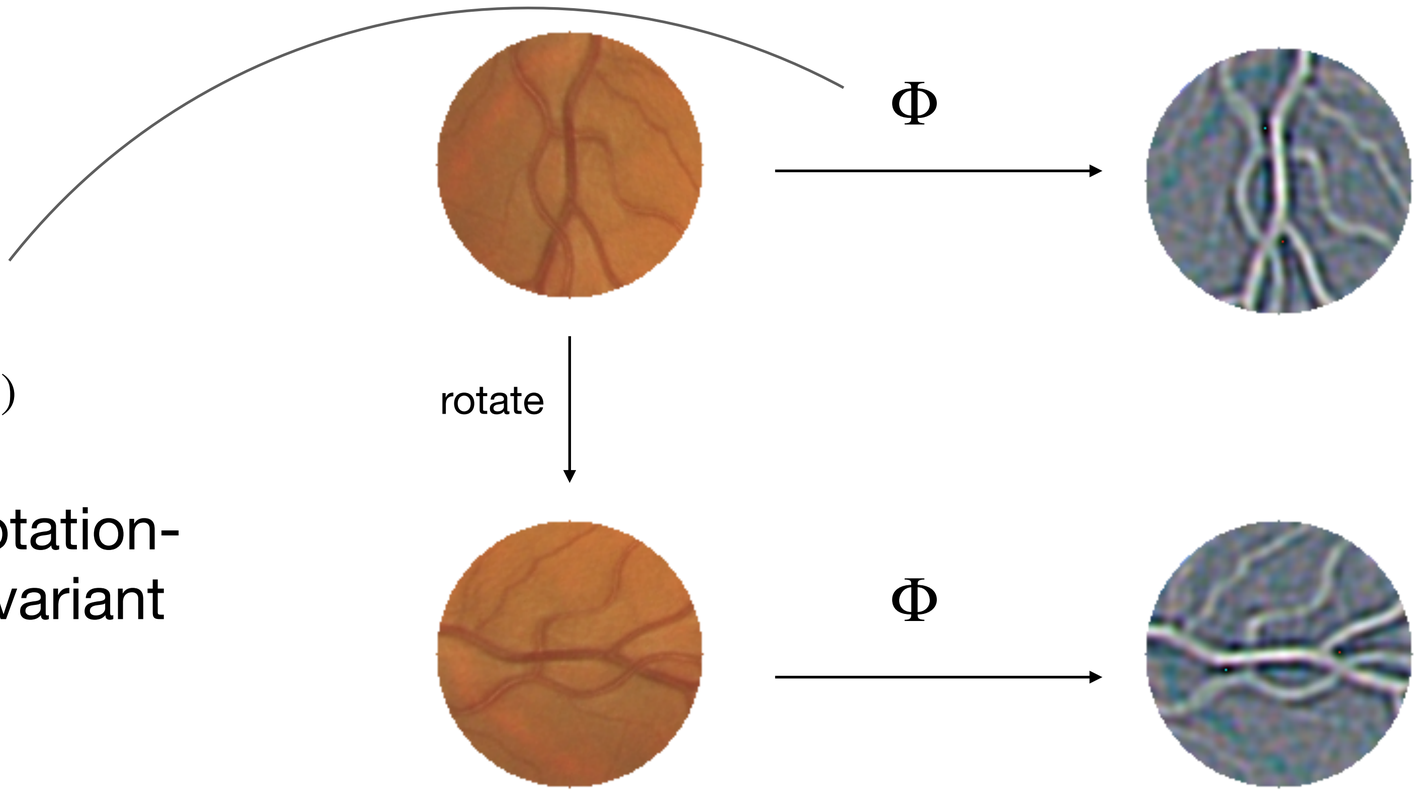
Invariance

Invariance is the special case where the action acting on output is *identity:*

$$\Phi = \Phi \circ \rho^X(g)$$



Example: Convolution with rotation-symmetric kernel is SO(2) invariant



Slide credit: Erik Bekkers, University of Amsterdam

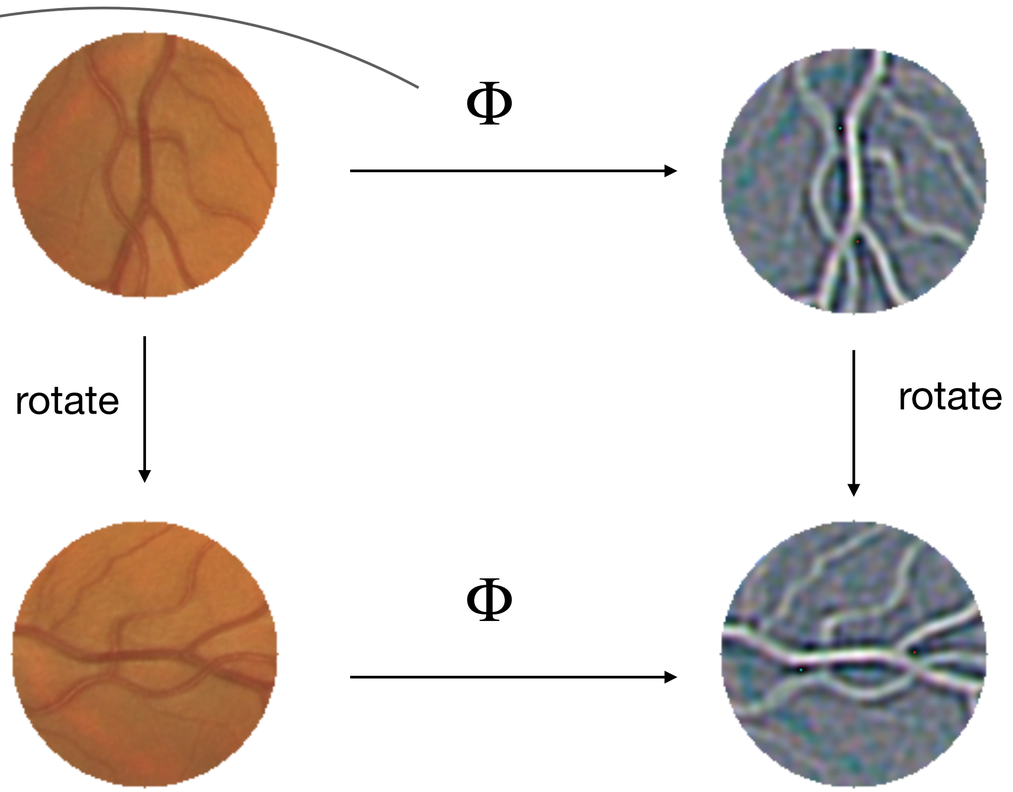
Invariance

Invariance is the special case where the action acting on output is *identity:*

$$\Phi = \Phi \circ \rho^X(g)$$

Conv2D(input ,)

Example: Convolution with rotation-symmetric kernel is SO(2) invariant



Slide credit: Erik Bekkers, University of Amsterdam

Geometric guarantees (equivariance)

CNNs are translation equivariant

Via convolutions

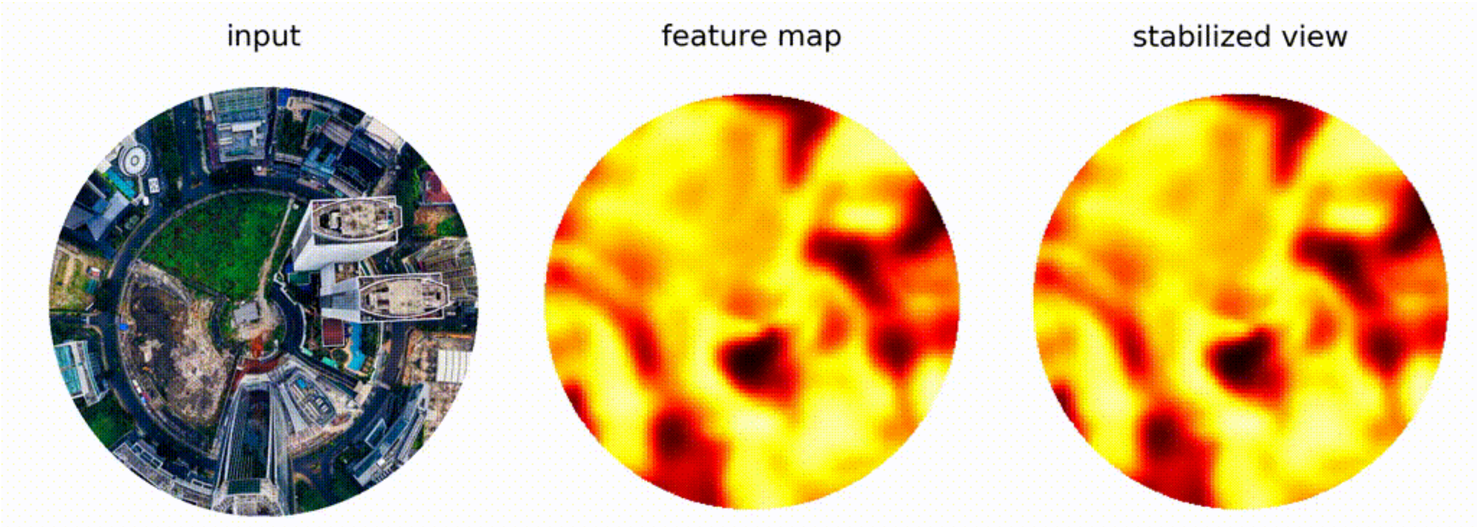
Geometric guarantees (equivariance)

CNNs are translation equivariant

Via convolutions

Geometric guarantees (equivariance)

CNN

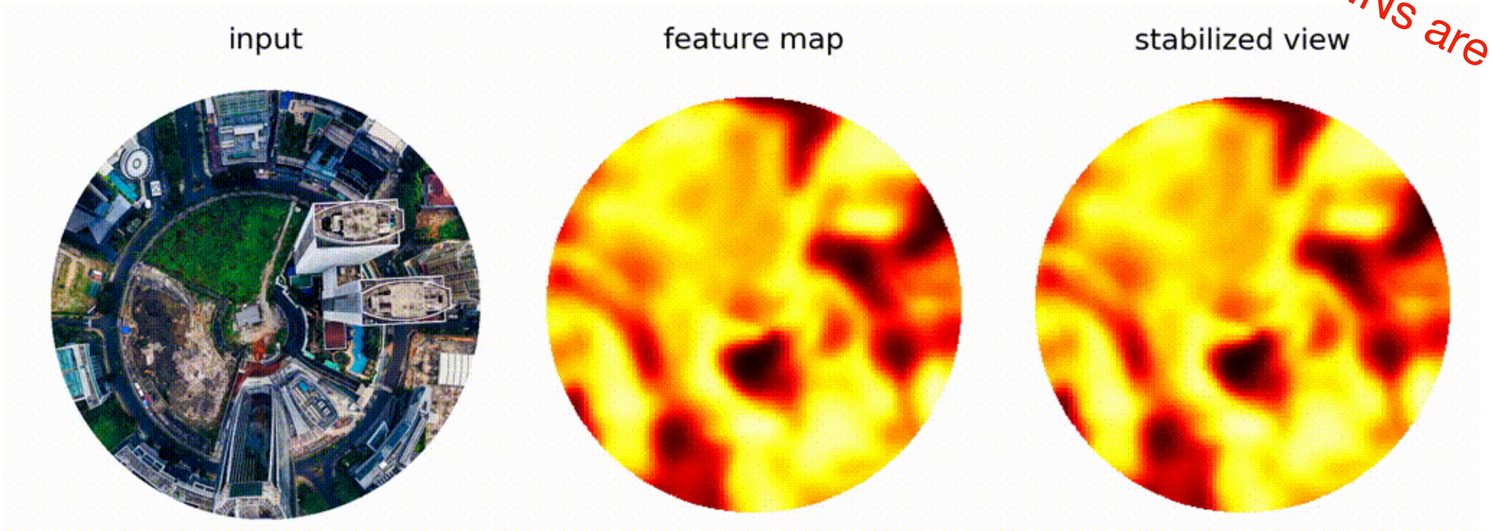


Figures source:
https://github.com/QUVA-Lab/e2cnn

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Geometric guarantees (equivariance)

CNN



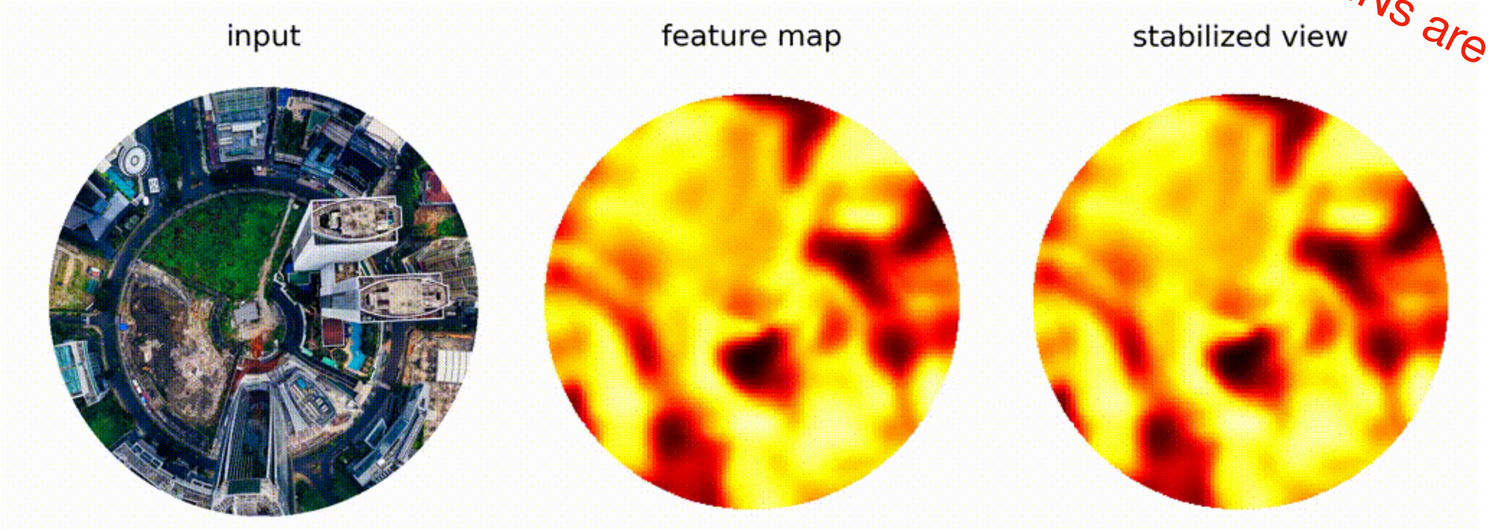
CNNs are not rotation equivariant

Figures source:
https://github.com/QUVA-Lab/e2cnn

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

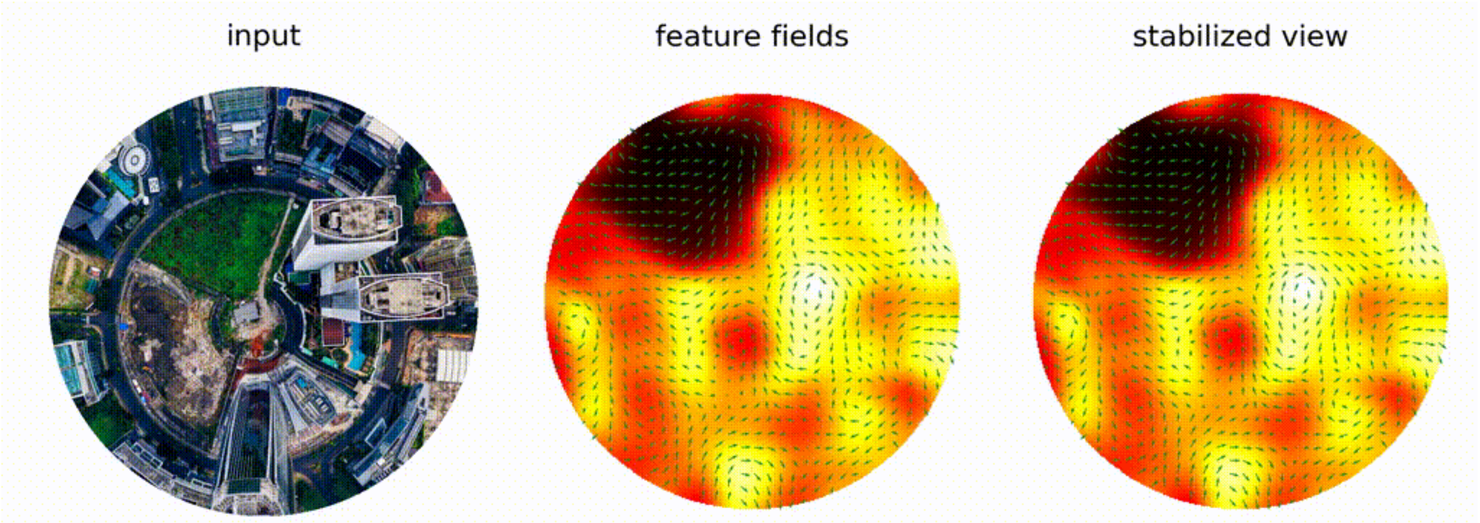
Geometric guarantees (equivariance)

CNN



CNNs are not rotation equivariant

Equivariant NN



Figures source:
https://github.com/QUVA-Lab/e2cnn

Cross-correlations

$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = \int_{\mathbb{R}^2} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

Slide credit: Erik Bekkers, University of Amsterdam

Cross-correlations

Representation of the translation group!

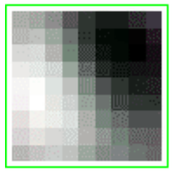
$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = \int_{\mathbb{R}^2} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}' = (\mathscr{L}_g \, k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

Slide credit: Erik Bekkers, University of Amsterdam

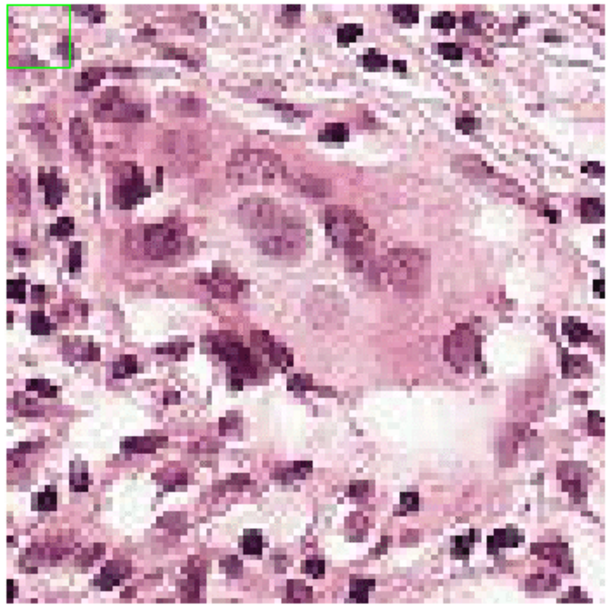
Cross-correlations

Representation of the translation group!

$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = \int_{\mathbb{R}^2} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}' = (\mathscr{L}_g \, k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$



$\star_{\mathbb{R}^2}$



$=$

k
2D convolution kernel

f^{in}
2D feature map

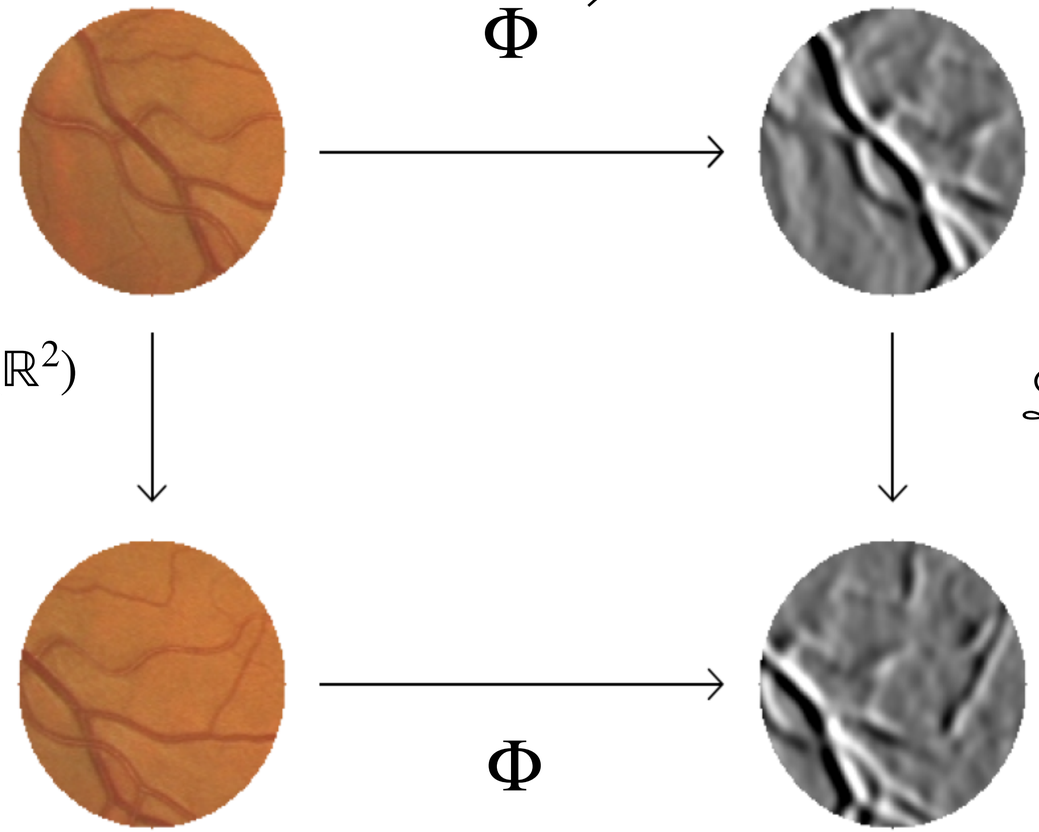
f^{out}
2D feature map (after ReLU)

Slide credit: Erik Bekkers, University of Amsterdam

Equivariance

Convolutions/cross-correlations are translation equivariant

$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = (\mathscr{L}^{\mathbb{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)}_{(\mathbf{x})} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$



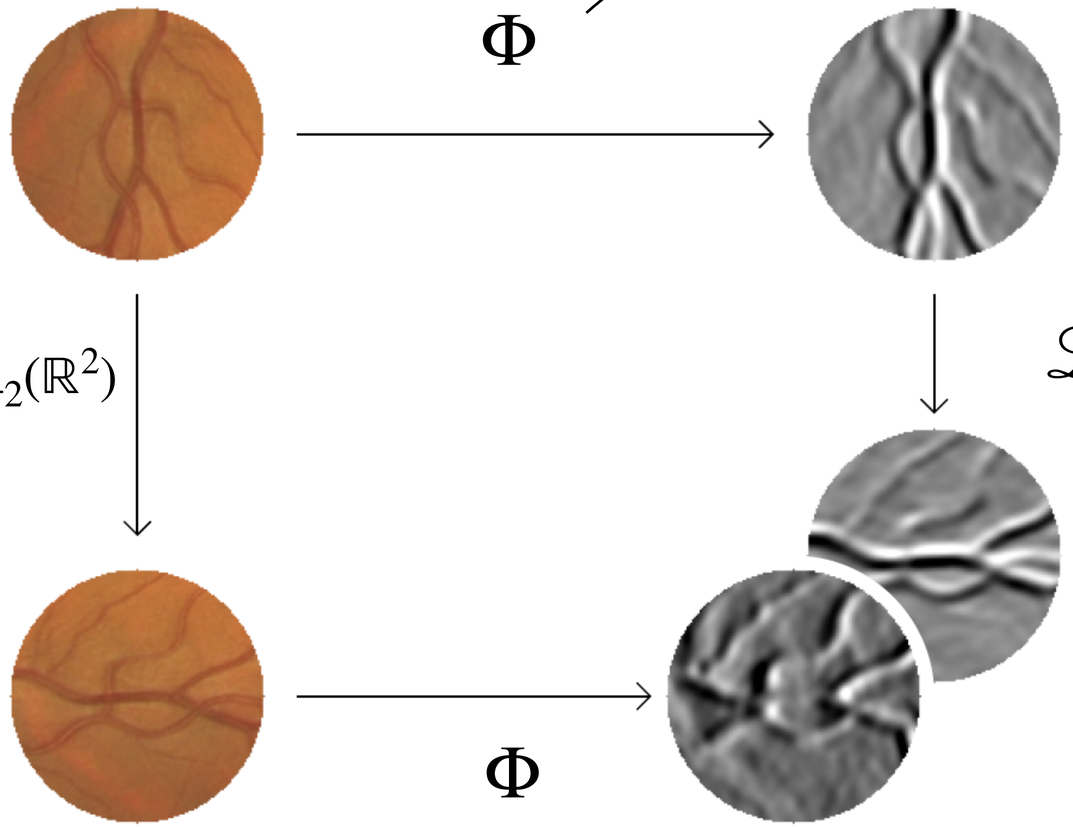
Representation of the translation group

Slide credit: Erik Bekkers, University of Amsterdam

Conventional convolutions do not commute with Rotations

$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = (\mathscr{L}_{(\mathbf{x})}^{\mathbb{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

Representation of the translation group



$\mathscr{L}_{\theta}^{SO(2) \to \mathbb{L}_2(\mathbb{R}^2)}$

$\mathscr{L}_{g}^{SO(2) \to \mathbb{L}_2(\mathbb{R}^2)}$

Representation of the rotation group

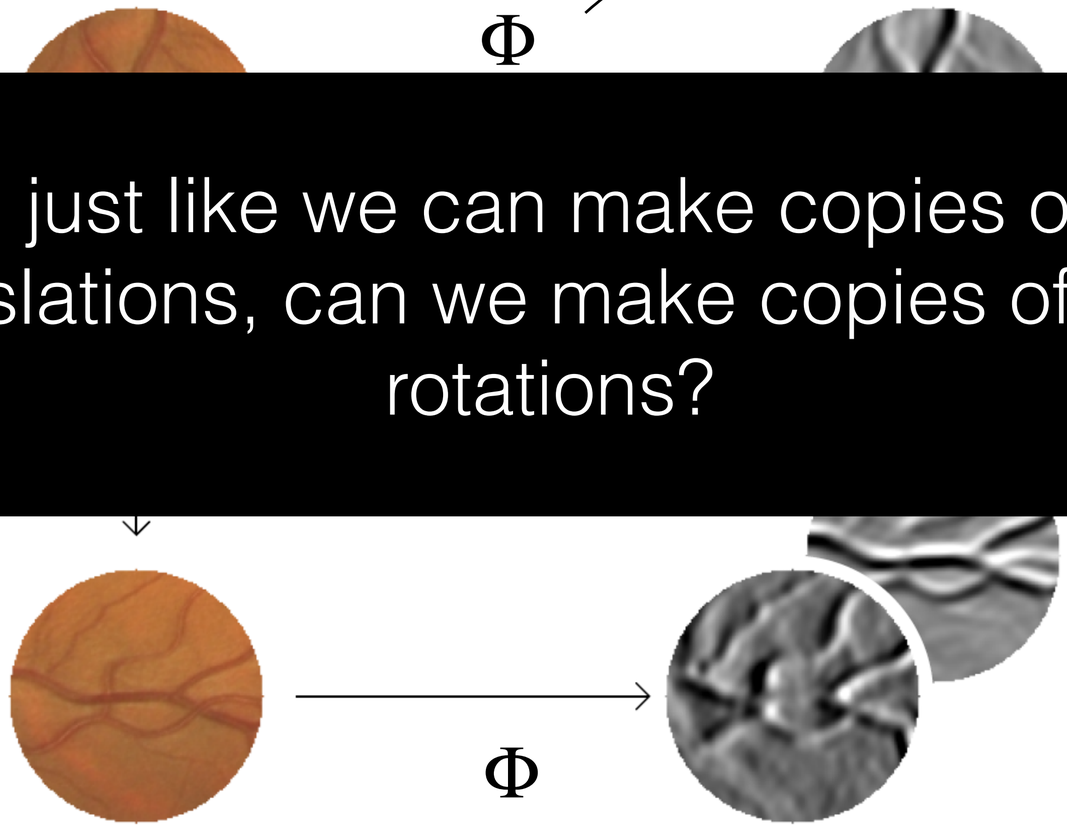
Slide credit: Erik Bekkers, University of Amsterdam

Conventional convolutions do not commute with Rotations

$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = (\mathscr{L}_{(\mathbf{x})}^{\mathbb{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

Representation of the

Simple solution: just like we can make copies of the filter across 2D spatial translations, can we make copies of the filter across rotations?



Representation of the rotation group

Slide credit: Erik Bekkers, University of Amsterdam

SE(2) equivariant cross-correlations

Representation of the roto-translation group!

Lifting correlations: $(k \tilde{\star} f)(\mathbf{x}, \theta) = (\mathscr{L}_g^{SE(2) \to \mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$

Slide credit: Erik Bekkers, University of Amsterdam

SE(2) equivariant cross-correlations

Representation of the roto-translation group!

Lifting correlations: $(k \tilde{\star} f)(\mathbf{x}, \theta) = (\mathscr{L}_{g}^{SE(2) \to \mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)} = (\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)} \mathscr{L}_{\theta}^{SO(2) \to \mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$

translation *rotation*

Slide credit: Erik Bekkers, University of Amsterdam

SE(2) equivariant cross-correlations

Representation of the roto-translation group!

$k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}))$

Lifting correlations: $(k \tilde{\star} f)(\mathbf{x}, \theta) = (\mathscr{L}_g^{SE(2)\to\mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)} = (\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(\mathbb{R}^2)} \mathscr{L}_\theta^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$

translation *rotation*

SE(2) equivariant cross-correlations

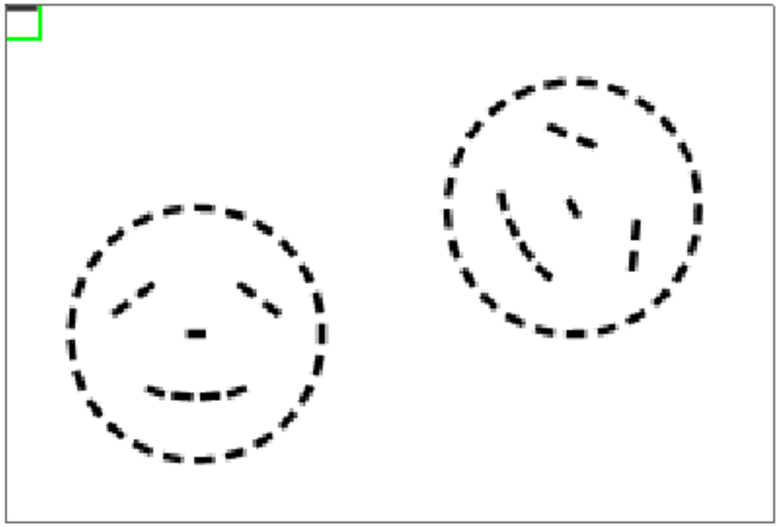
Representation of the roto-translation group!

$k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}))$

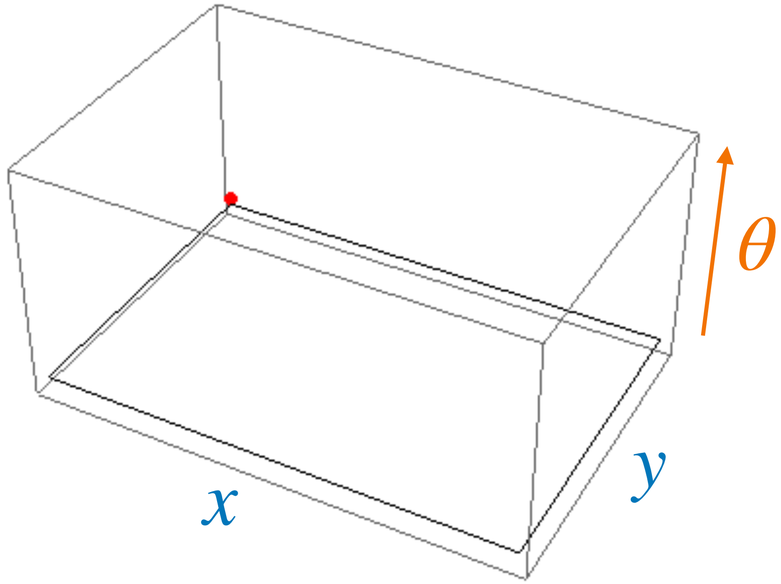
Lifting correlations: $(k \tilde{\star} f)(\mathbf{x},\theta) = (\mathscr{L}_g^{SE(2)\to\mathbb{L}_2(\mathbb{R}^2)} k\,, f)_{\mathbb{L}_2(\mathbb{R}^2)} = (\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(\mathbb{R}^2)} \mathscr{L}_\theta^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)} k\,, f)_{\mathbb{L}_2(\mathbb{R}^2)}$

translation *rotation*

$\star_{\mathbb{R}^2}$



$=$



$\mathscr{L}_\theta^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)} k$
Rotated 2D convolution kernel

f^{in}
2D feature map

f^{out}
3D (*SE(2)*) feature map (after ReLU)

Slide credit: Erik Bekkers, University of Amsterdam

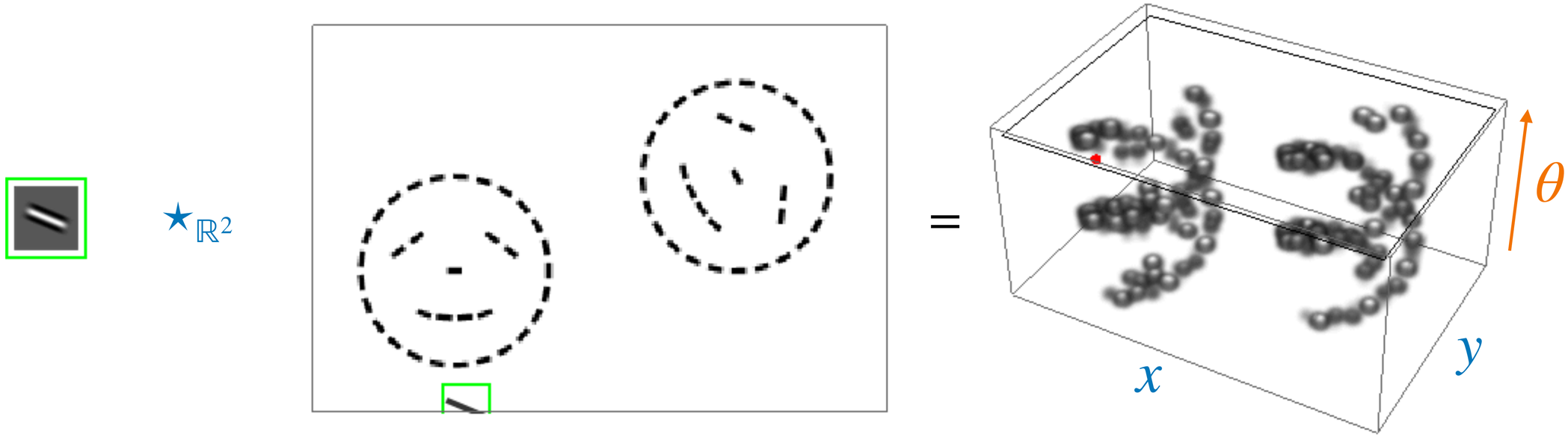
SE(2) equivariant cross-correlations

Representation of the roto-translation group!

$k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}))$

Lifting correlations: $(k \tilde{\star} f)(\mathbf{x},\theta) = (\mathscr{L}_g^{SE(2)\to\mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)} = (\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(\mathbb{R}^2)} \mathscr{L}_\theta^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$

translation *rotation*



$\mathscr{L}_\theta^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)} k$
Rotated 2D convolution kernel

f^{in}
2D feature map

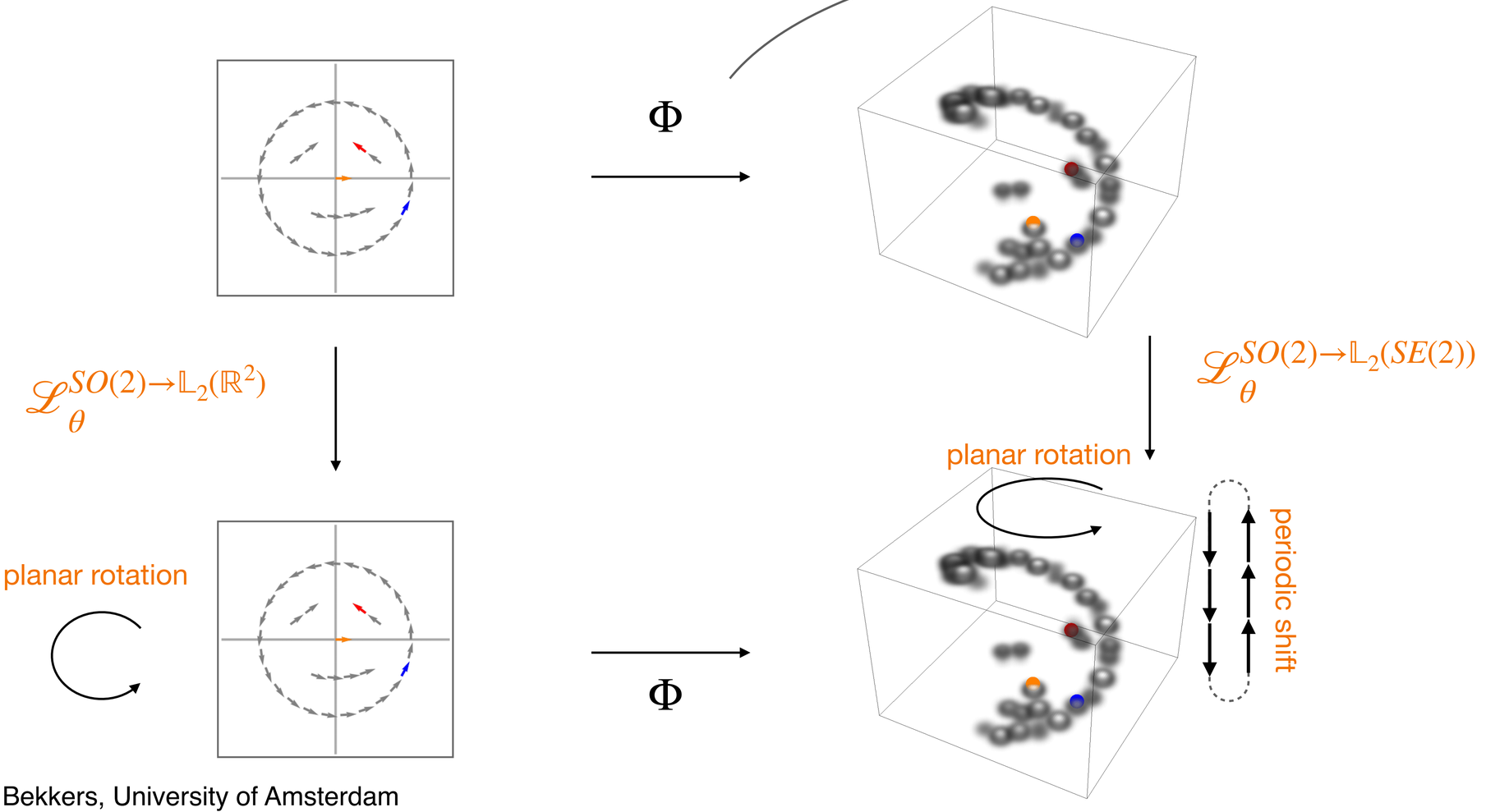
f^{out}
3D (*SE(2)*) feature map (after ReLU)

Slide credit: Erik Bekkers, University of Amsterdam

Equivariance

SE(2) group **lifting convolutions** are roto-translation equivariant

$$(k \tilde{\star} f)(\mathbf{x}, \theta) = (\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)} \mathscr{L}_{\theta}^{SO(2) \to \mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

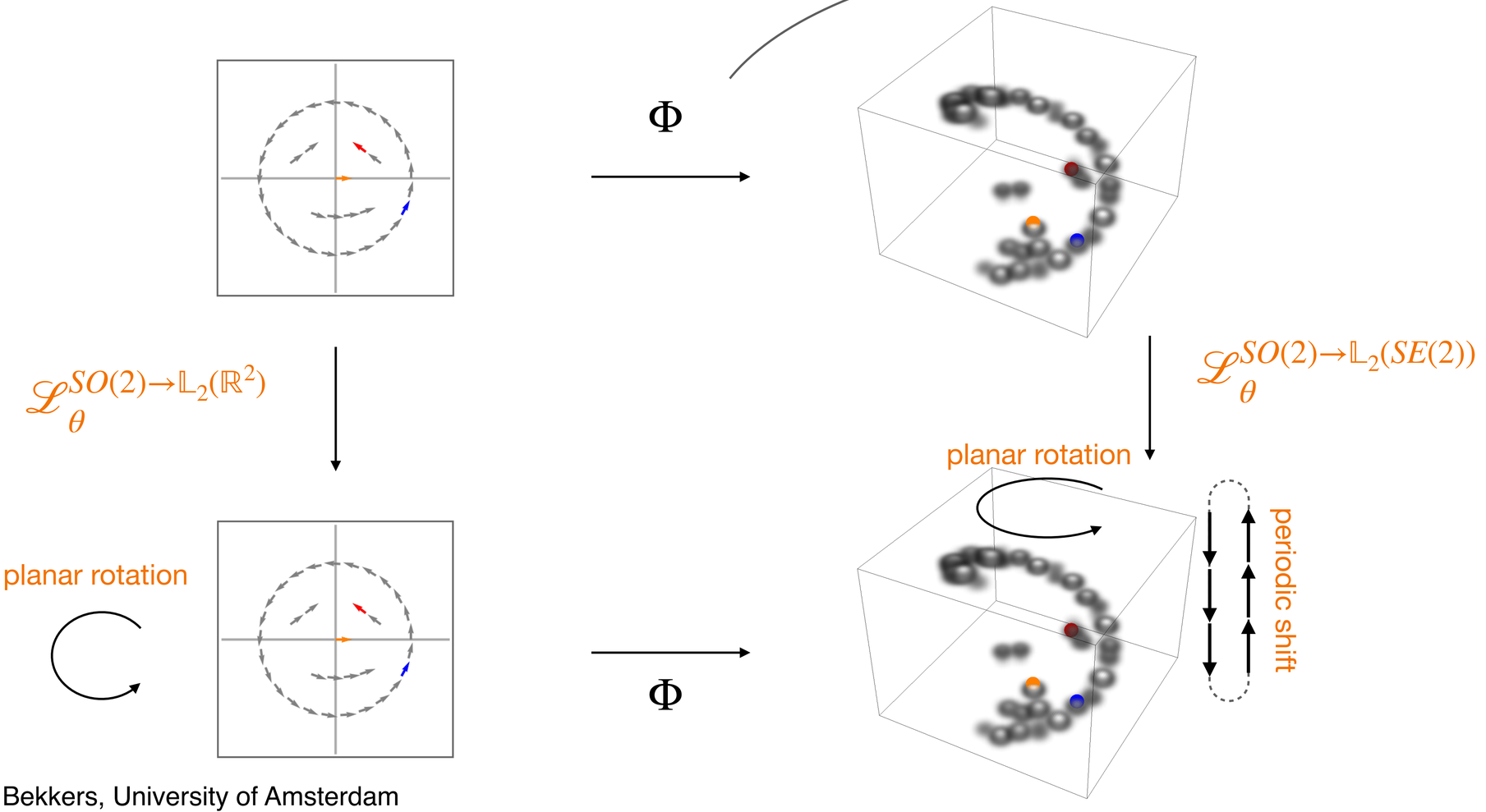


Slide credit: Erik Bekkers, University of Amsterdam

Equivariance

SE(2) group **lifting convolutions** are roto-translation equivariant

$$(k \tilde{\star} f)(\mathbf{x}, \theta) = (\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)} \mathscr{L}_{\theta}^{SO(2) \to \mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

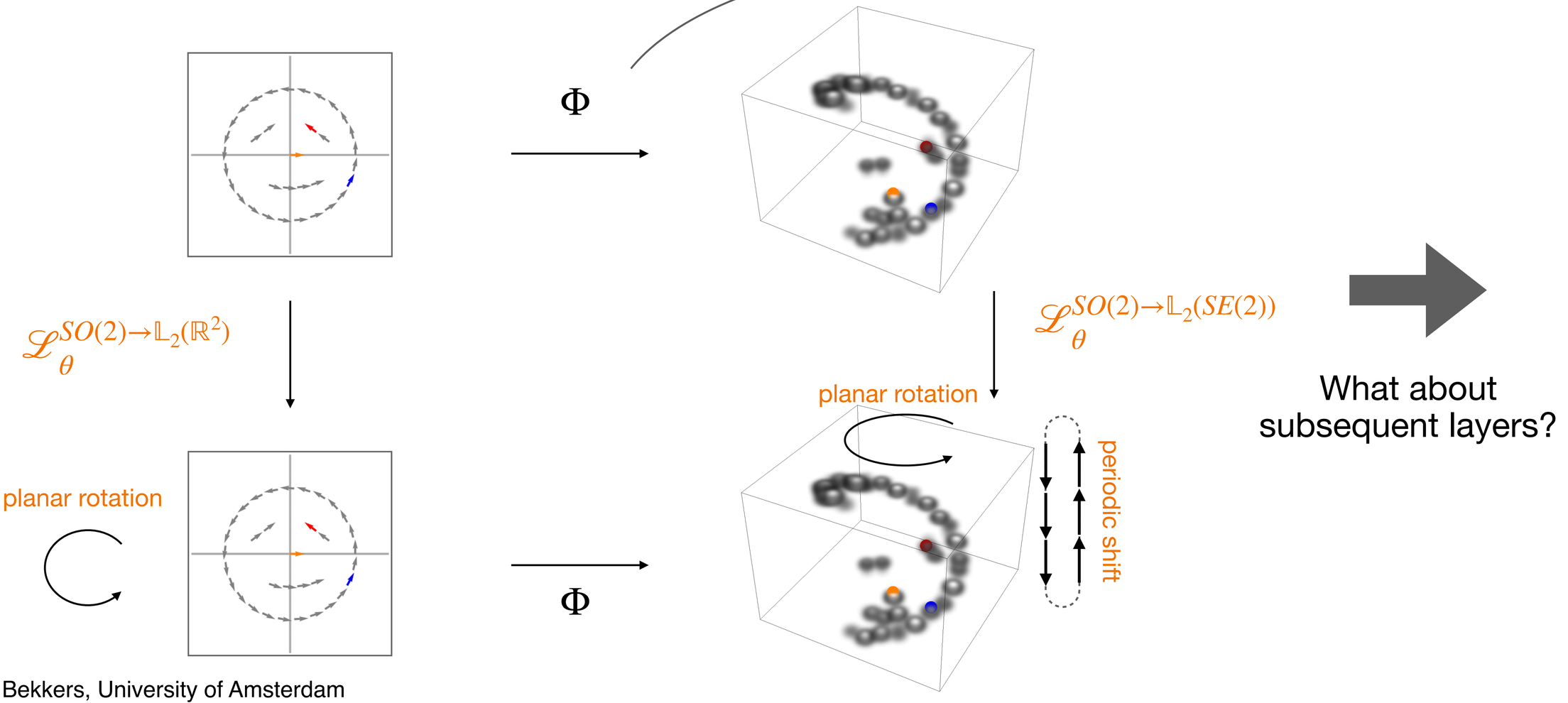


Slide credit: Erik Bekkers, University of Amsterdam

Equivariance

SE(2) group **lifting convolutions** are roto-translation equivariant

$$(k \tilde{\star} f)(\mathbf{x}, \theta) = (\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)} \mathscr{L}_{\theta}^{SO(2) \to \mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$



Slide credit: Erik Bekkers, University of Amsterdam

SE(2) equivariant cross-correlations

Group correlations:

$$(k \star f)(\mathbf{x}, \theta) = (\mathscr{L}_{g}^{SE(2)\to\mathbb{L}_2(SE(2))}\, k\,, f)_{\mathbb{L}_2(SE(2))} = (\overbrace{\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(SE(2))}\, \mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2))}\, k}^{k(\mathbf{R}_{\theta}^{-1}(\mathbf{x}'-\mathbf{x}),\mathbf{R}_{\theta'-\theta})}\,, f)_{\mathbb{L}_2(SE(2)}$$

translation *rotation*

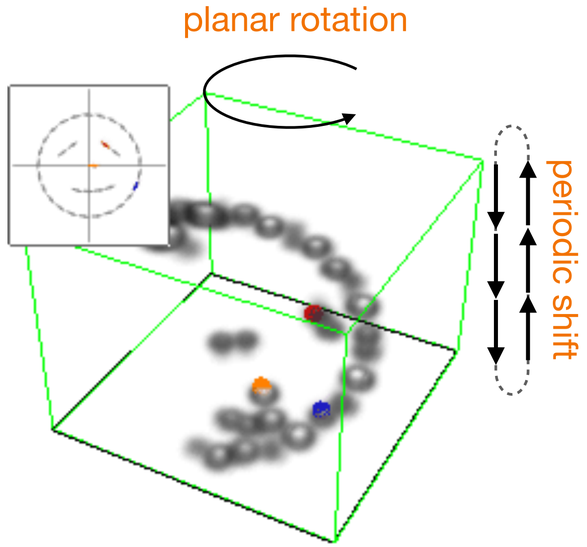
Slide credit: Erik Bekkers, University of Amsterdam

SE(2) equivariant cross-correlations

Group correlations:

$$(k \star f)(\mathbf{x}, \theta) = (\mathscr{L}_{g}^{SE(2)\to\mathbb{L}_2(SE(2))}\, k\,, f)_{\mathbb{L}_2(SE(2))} = (\overbrace{\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(SE(2))}\, \mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2))}\, k}^{k(\mathbf{R}_{\theta}^{-1}(\mathbf{x}'-\mathbf{x}),\mathbf{R}_{\theta'-\theta})}\,, f)_{\mathbb{L}_2(SE(2)}$$

translation *rotation*



$\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2)} k$

Rotated SE(2) convolution kernel

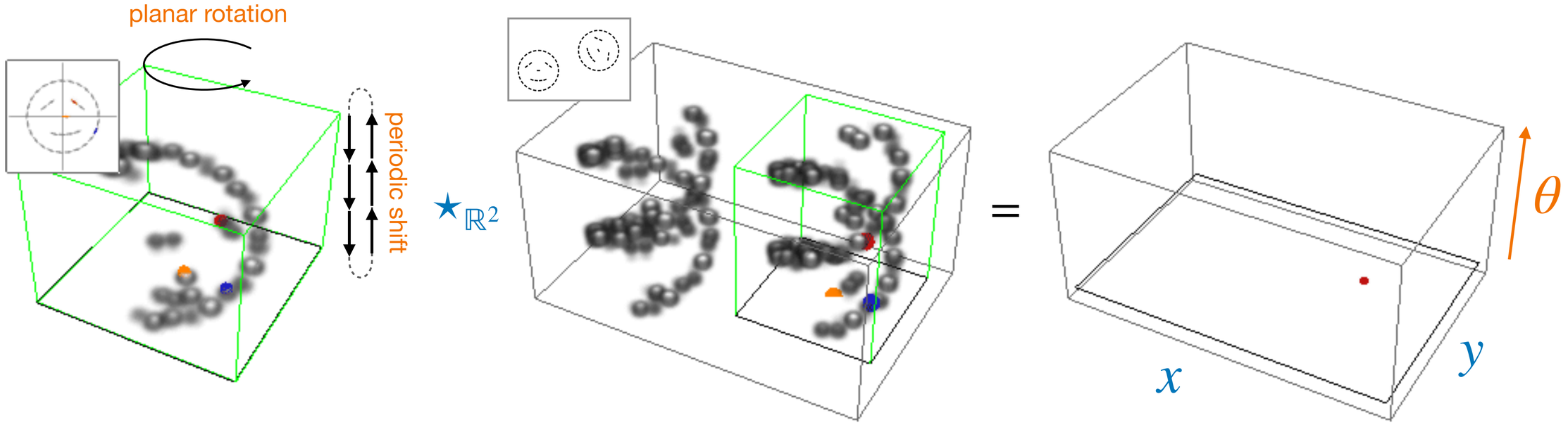
Slide credit: Erik Bekkers, University of Amsterdam

SE(2) equivariant cross-correlations

Group correlations:

$$(k \star f)(\mathbf{x}, \theta) = (\mathscr{L}_{g}^{SE(2)\to\mathbb{L}_2(SE(2))}\, k\,, f)_{\mathbb{L}_2(SE(2))} = (\overbrace{\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(SE(2))}\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2))}\, k}^{k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}),\mathbf{R}_{\theta'-\theta})}\,, f)_{\mathbb{L}_2(SE(2)}$$

translation *rotation*



$\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2)}k$
Rotated SE(2) convolution kernel

f^{in}
SE(2) feature map

f^{out}
SE(2) feature map (after ReLU)

Slide credit: Erik Bekkers, University of Amsterdam

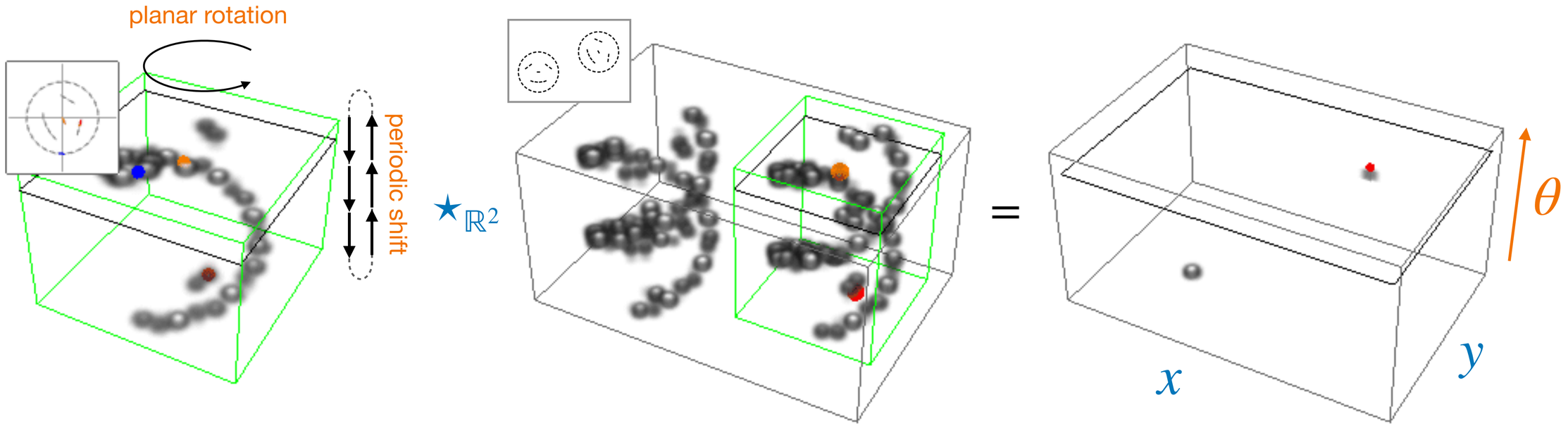
SE(2) equivariant cross-correlations

Group correlations:

$$(k \star f)(\mathbf{x}, \theta) = (\mathscr{L}_{g}^{SE(2)\to\mathbb{L}_2(SE(2))}\, k\,, f)_{\mathbb{L}_2(SE(2))} = (\overbrace{\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(SE(2))}\, \mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2))}\, k}\,, f)_{\mathbb{L}_2(SE(2)}$$

$k(\mathbf{R}_{\theta}^{-1}(\mathbf{x}'-\mathbf{x}), \mathbf{R}_{\theta'-\theta})$

translation *rotation*



$\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2)}k$
Rotated SE(2) convolution kernel

f^{in}
SE(2) feature map

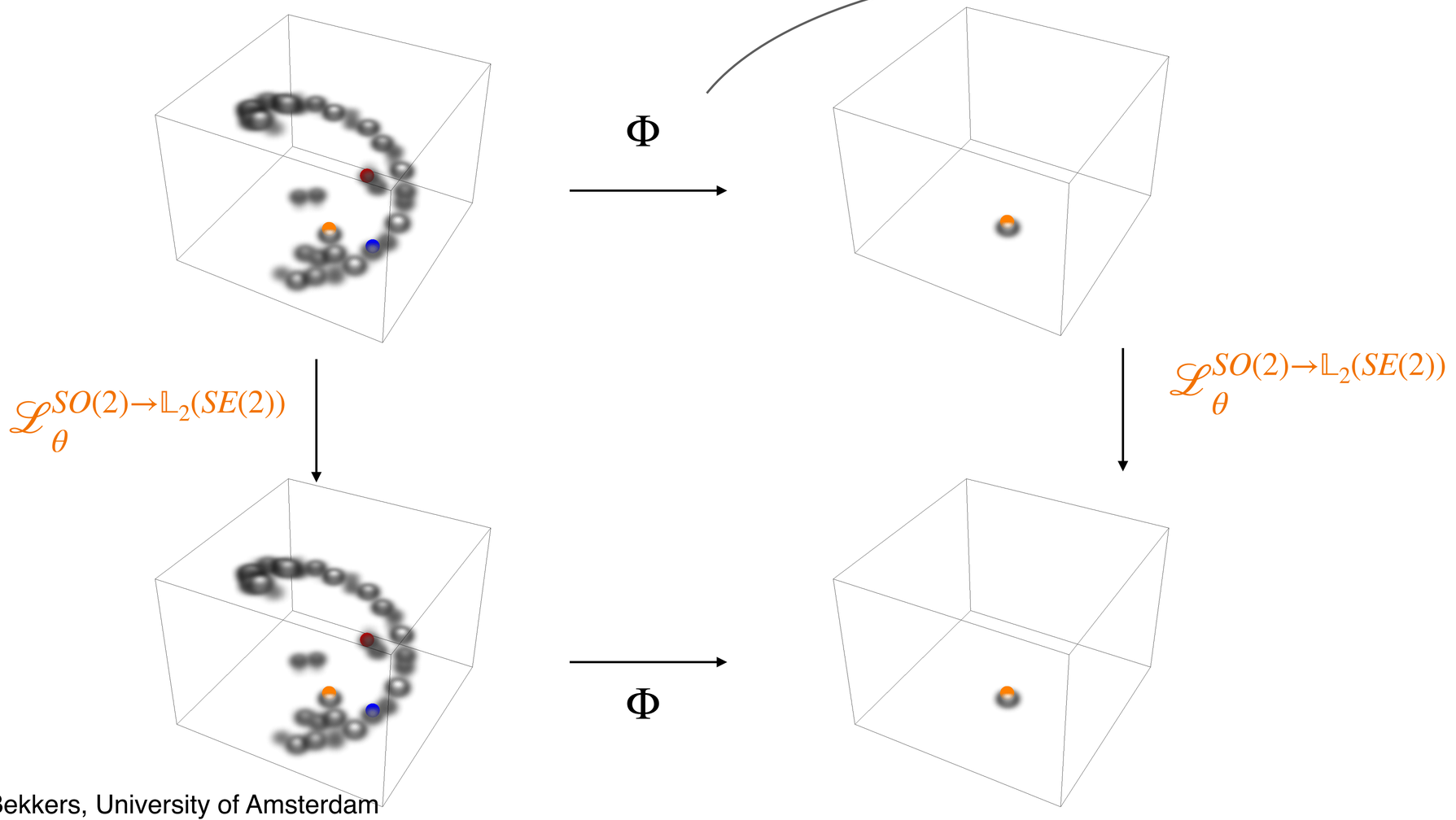
f^{out}
SE(2) feature map (after ReLU)

Slide credit: Erik Bekkers, University of Amsterdam

Equivariance

SE(2) group convolutions are roto-translation equivariant

$$(k \star f)(\mathbf{x}, \theta) = (\mathscr{L}_{\mathbf{x}}^{\mathbb{R}^2 \to \mathbb{L}_2(SE(2))} \mathscr{L}_{\theta}^{SO(2) \to \mathbb{L}_2(SE(2))} k \, , f)_{\mathbb{L}_2(SE(2))}$$

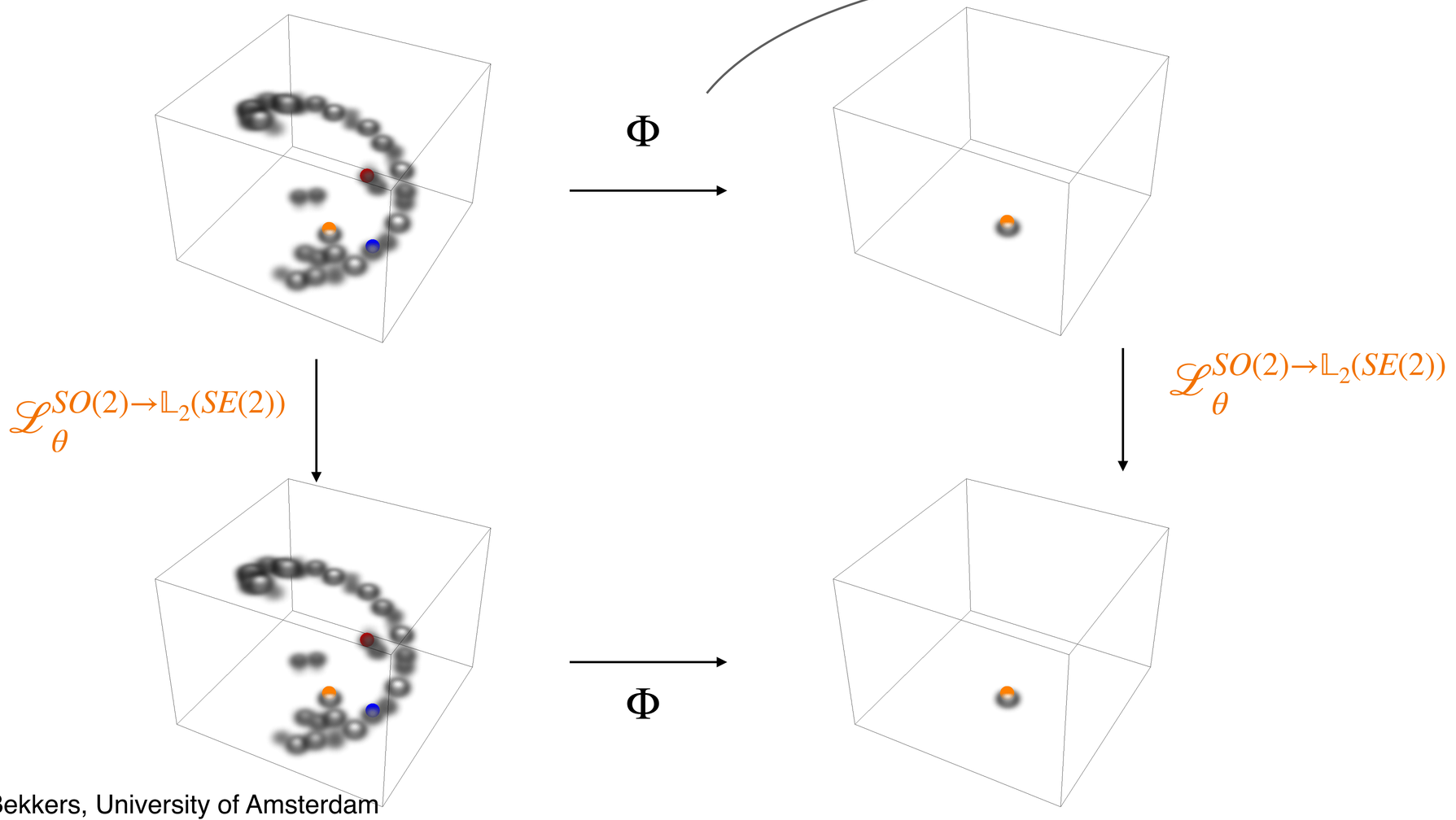


Slide credit: Erik Bekkers, University of Amsterdam

Equivariance

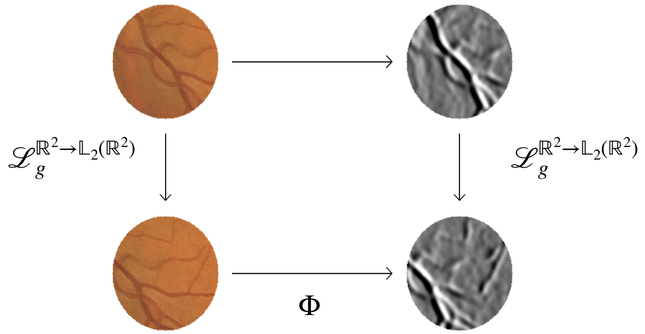
SE(2) group convolutions are roto-translation equivariant

$$(k \star f)(\mathbf{x}, \theta) = (\mathscr{L}_{\mathbf{x}}^{\mathbb{R}^2 \to \mathbb{L}_2(SE(2))} \mathscr{L}_{\theta}^{SO(2) \to \mathbb{L}_2(SE(2))} k \, , f)_{\mathbb{L}_2(SE(2))}$$



Slide credit: Erik Bekkers, University of Amsterdam

2D cross-correlation (translation equivariant)

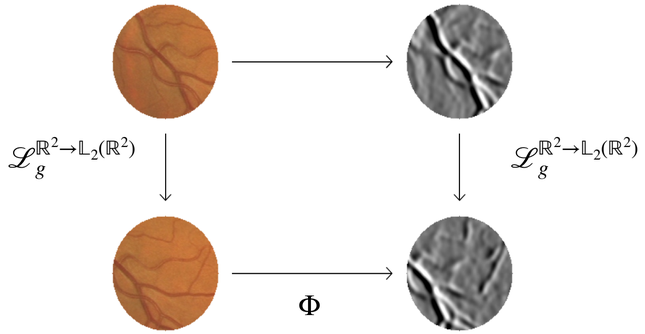


$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = (\mathscr{L}_{\mathbf{x}}^{\mathbb{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)} k, f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

$$= \int_{\mathbb{R}^2} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

Slide credit: Erik Bekkers, University of Amsterdam

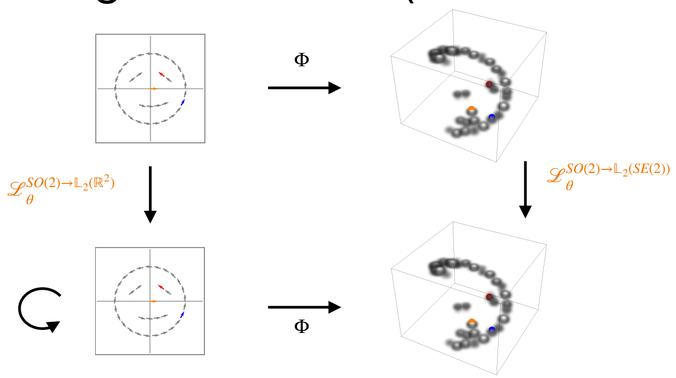
2D cross-correlation (translation equivariant)



$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = (\mathscr{L}_{\mathbf{x}}^{\mathbb{R}^2 \to \mathbb{L}_2(\mathbb{R}^2))} k, f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

$$= \int_{\mathbb{R}^2} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

SE(2) lifting correlations (roto-translation equivariant)

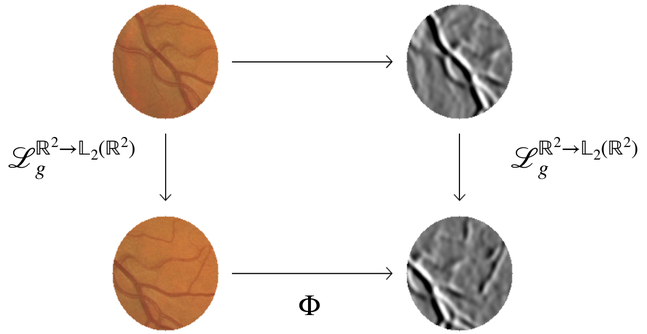


$$(k \tilde{\star} f)(\mathbf{x}, \theta) = (\mathscr{L}_g^{SE(2) \to \mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

$$= \int_{\mathbb{R}^2} k(\mathbf{R}_\theta^{-1}(\mathbf{x}' - \mathbf{x})) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

Slide credit: Erik Bekkers, University of Amsterdam

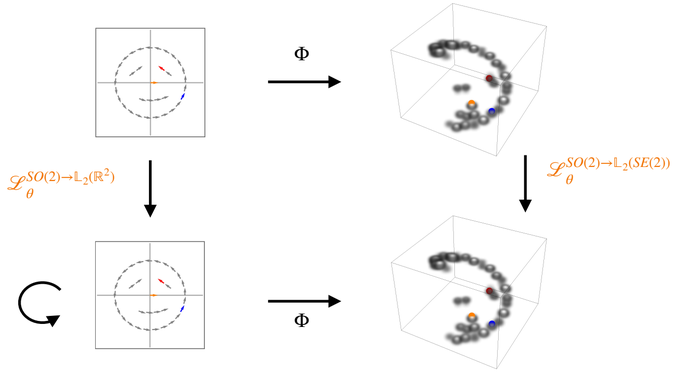
2D cross-correlation (translation equivariant)



$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = (\mathscr{L}_{\mathbf{x}}^{\mathbb{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)} k, f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

$$= \int_{\mathbb{R}^2} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

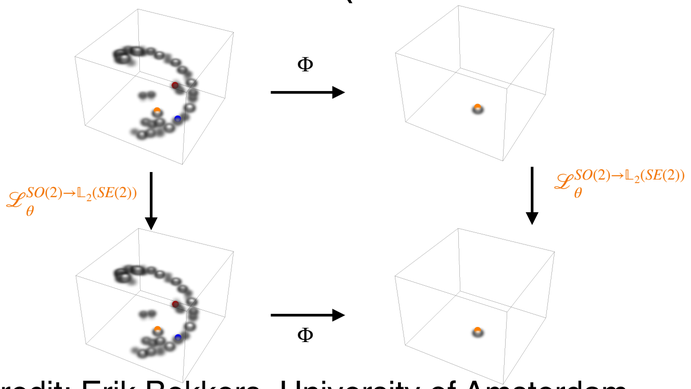
SE(2) lifting correlations (roto-translation equivariant)



$$(k \tilde{\star} f)(\mathbf{x}, \theta) = (\mathscr{L}_g^{SE(2) \to \mathbb{L}_2(\mathbb{R}^2)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

$$= \int_{\mathbb{R}^2} k(\mathbf{R}_\theta^{-1}(\mathbf{x}' - \mathbf{x})) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

SE(2) G-correlations (roto-translation equivariant)



$$(k \tilde{\star} f)(\mathbf{x}, \theta) = (\mathscr{L}_g^{SE(2) \to \mathbb{L}_2(SE(2))} k \, , f)_{\mathbb{L}_2(SE(2)}$$

$$= \int_{\mathbb{R}^2} \int_{S^1} k(\mathbf{R}_\theta^{-1}(\mathbf{x}' - \mathbf{x}), \theta' - \theta \mod 2\pi) f(\mathbf{x}', \theta') \mathrm{d}\mathbf{x}' \mathrm{d}\theta'$$

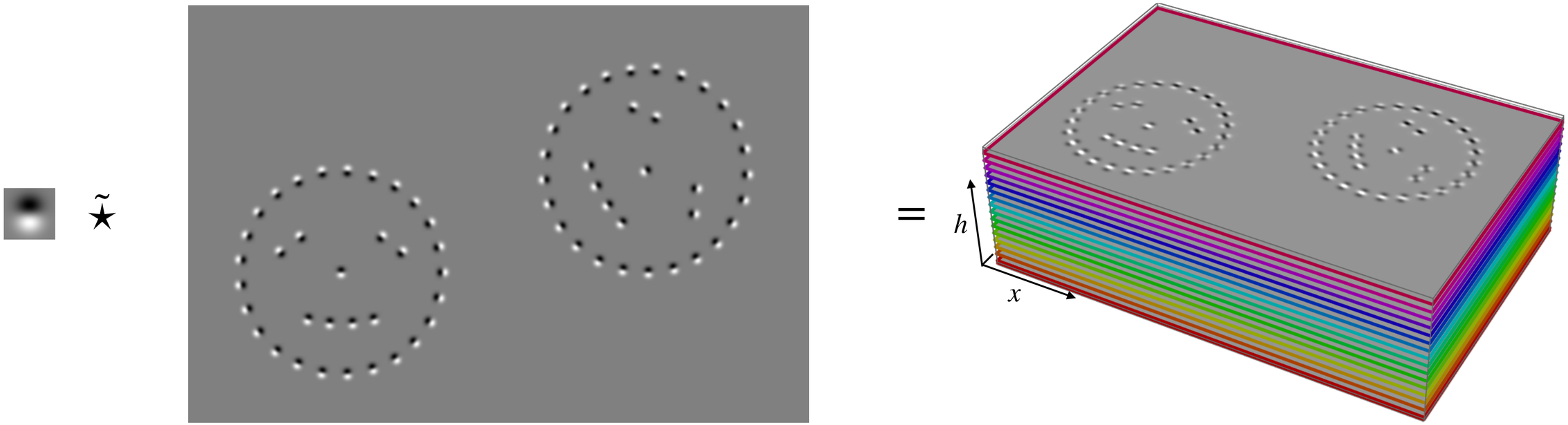
Slide credit: Erik Bekkers, University of Amsterdam

Recap: Regular Group Convolutions

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)}$$



2D convolution kernel 2D input feature map *SE(2)* output feature map

Slide credit: Erik Bekkers, University of Amsterdam

Recap: Regular Group Convolutions

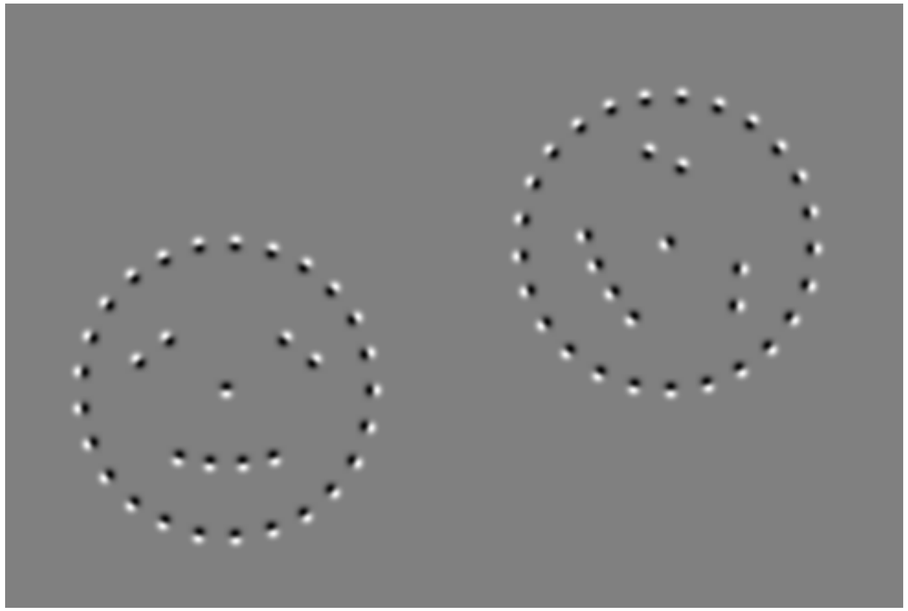
Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

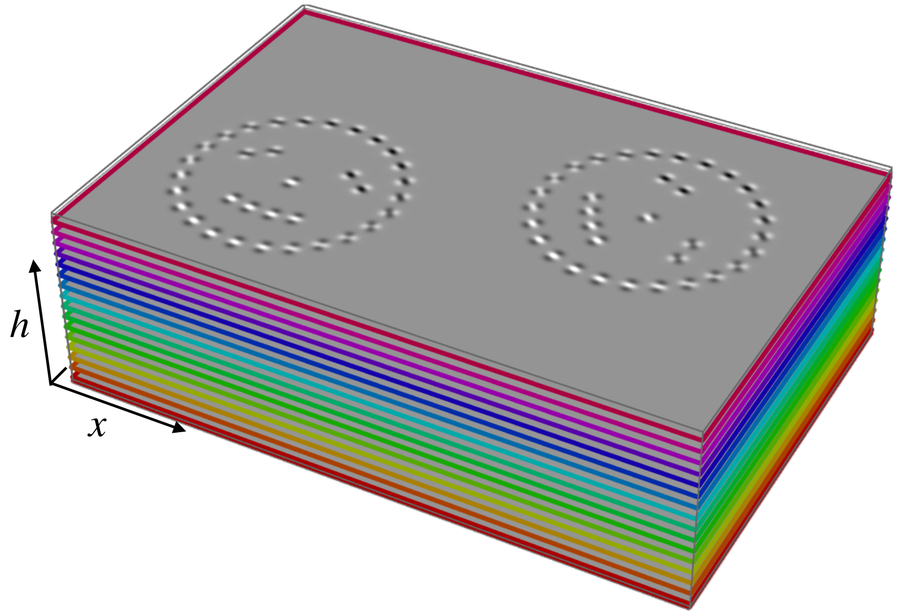
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$\tilde{\star}$



$=$



2D convolution kernel

2D input feature map

SE(2) output feature map

Slide credit: Erik Bekkers, University of Amsterdam

Recap: Regular Group Convolutions

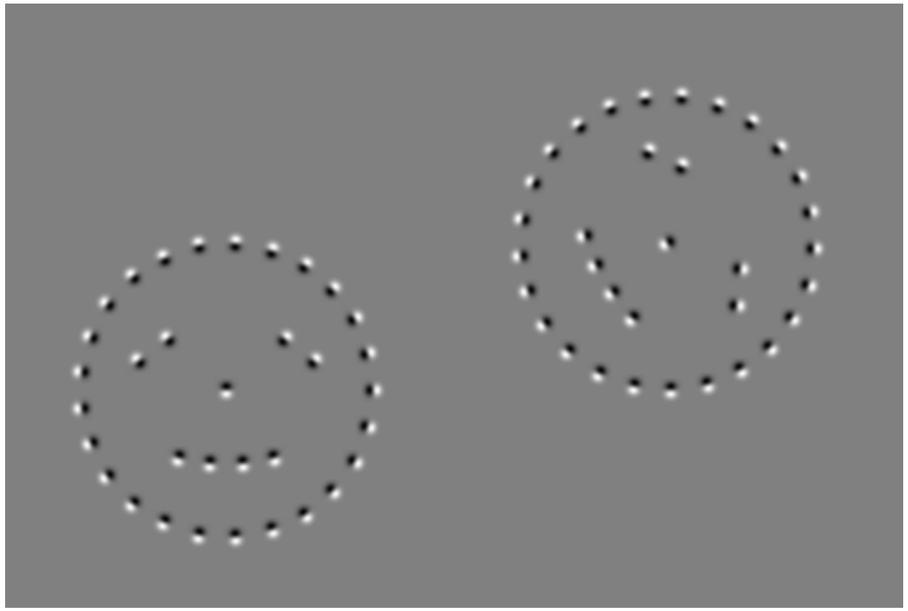
Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

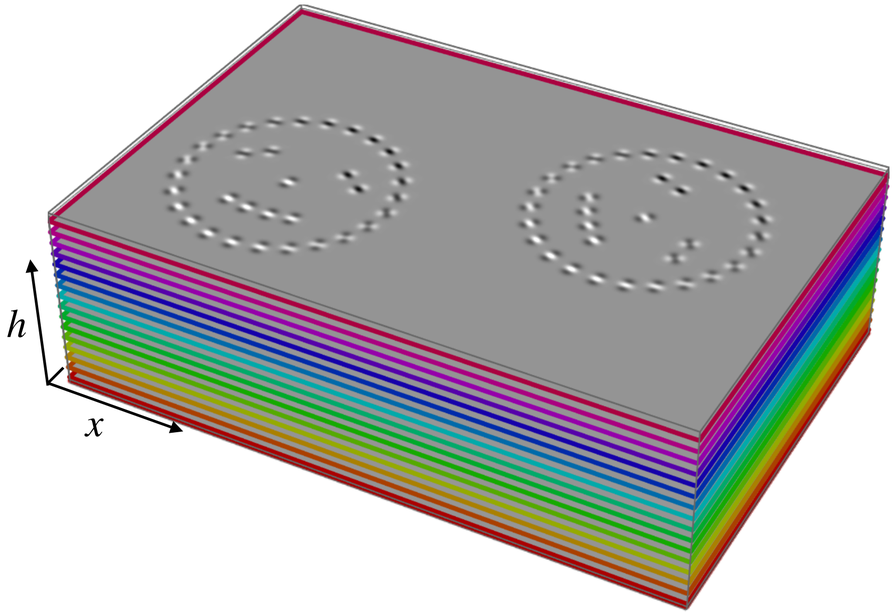
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$\tilde{\star}$



$=$



2D convolution kernel

2D input feature map

SE(2) output feature map

Slide credit: Erik Bekkers, University of Amsterdam

Recap: Regular Group Convolutions

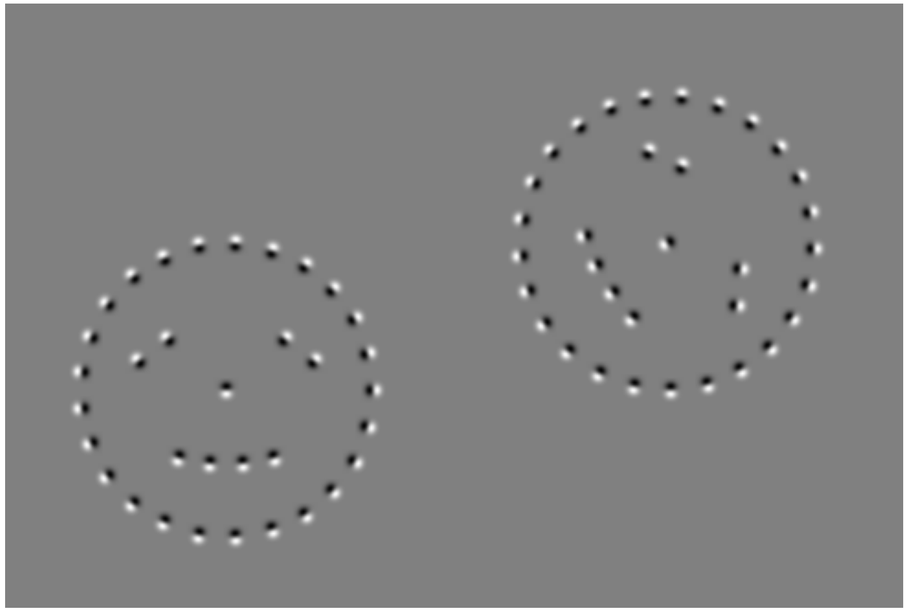
Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

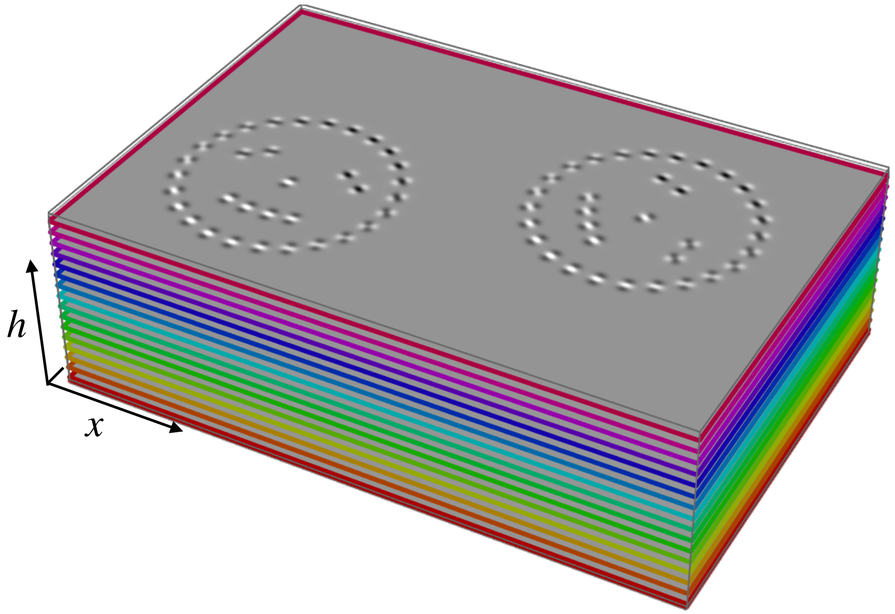
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}')f(\mathbf{x}')\mathrm{d}\mathbf{x}' = \int_{\mathbb{R}^2} k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}))f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

$\tilde{\star}$



=



2D convolution kernel

2D input feature map

SE(2) output feature map

Slide credit: Erik Bekkers, University of Amsterdam

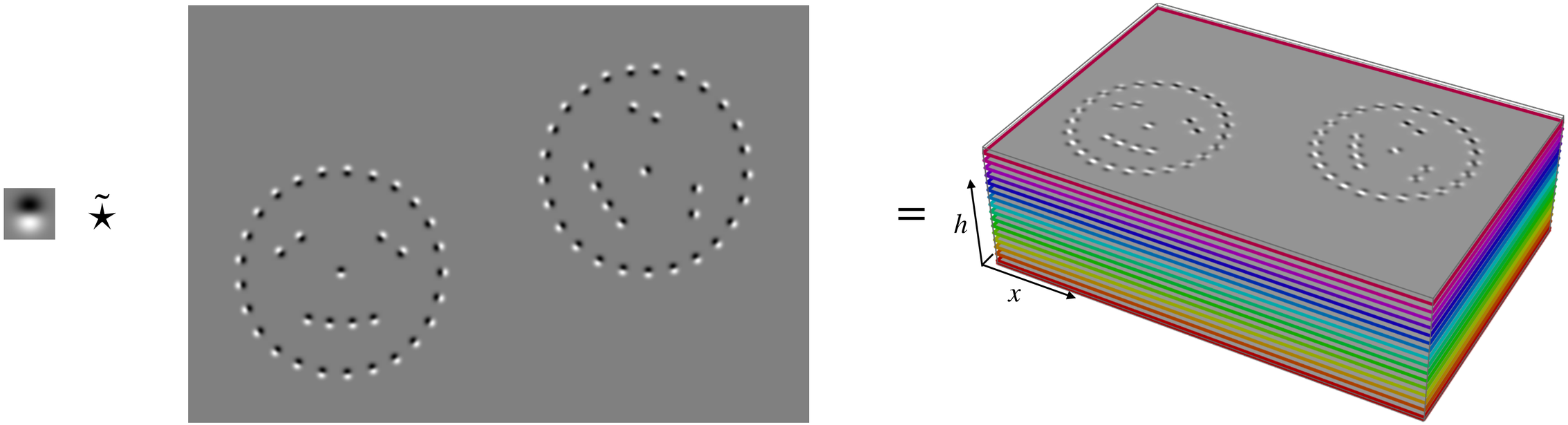
Recap: Regular Group Convolutions

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}')f(\mathbf{x}')\mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$



2D convolution kernel — 2D input feature map — *SE(2)* output feature map

Slide credit: Erik Bekkers, University of Amsterdam

Recap: Regular Group Convolutions

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

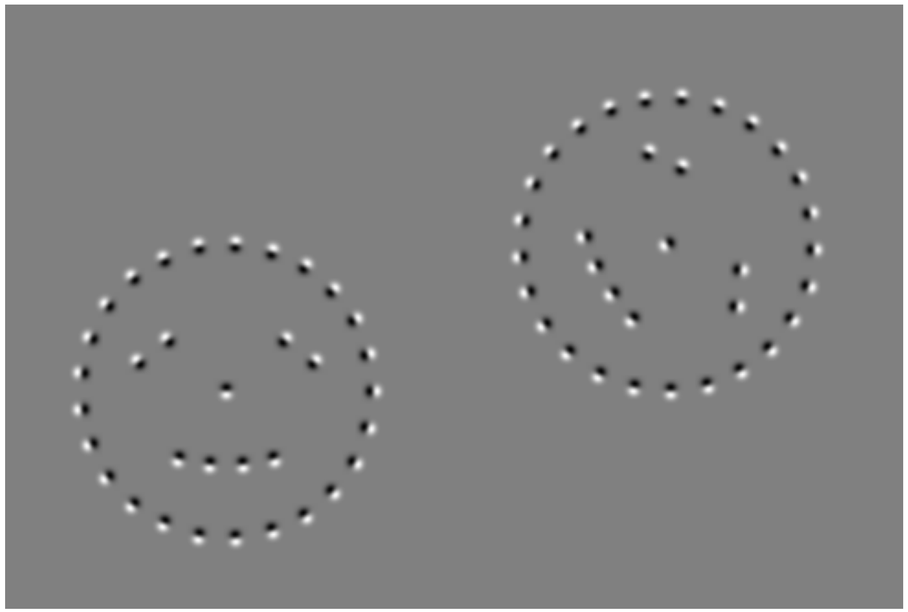
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

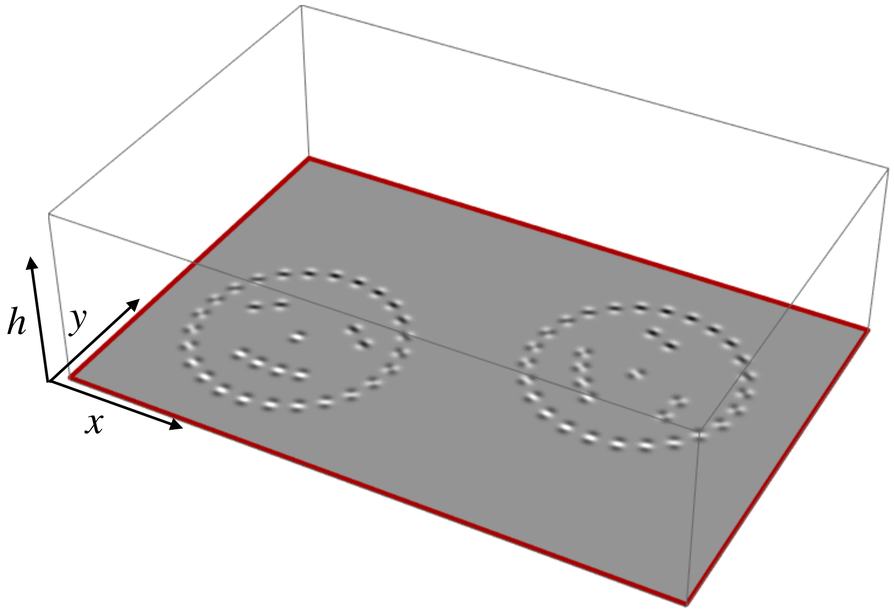
---

translation
"template matching"

$\star_{\mathbb{R}^2}$



$=$



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map

Slide credit: Erik Bekkers, University of Amsterdam

Recap: Regular Group Convolutions

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

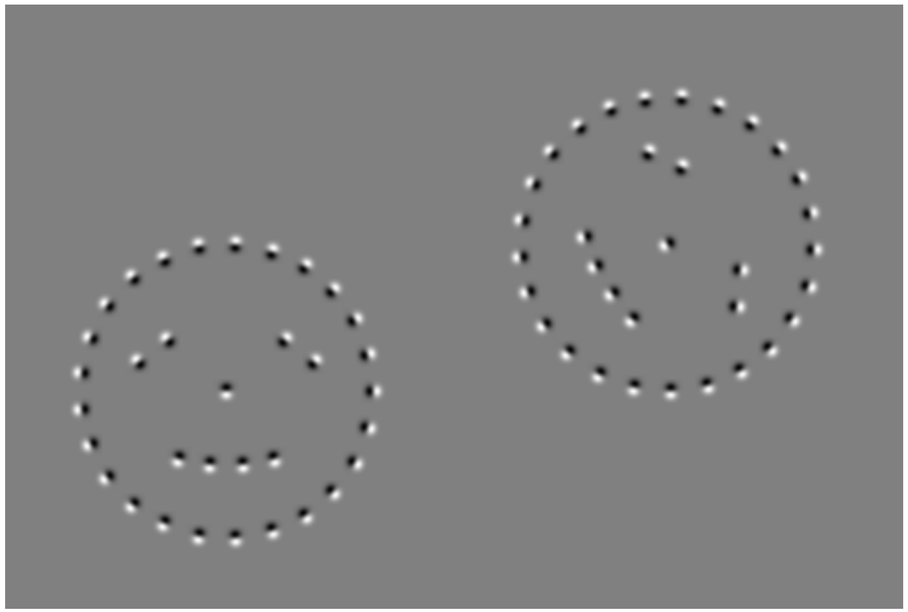
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}')f(\mathbf{x}')\mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

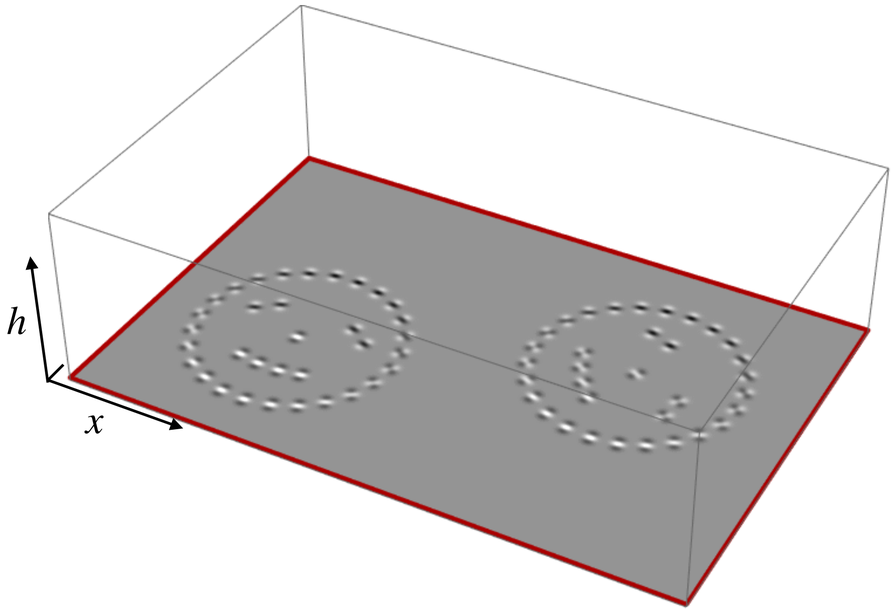
---

translation
"template matching"

$\star_{\mathbb{R}^2}$



$=$



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map

Slide credit: Erik Bekkers, University of Amsterdam

Recap: Regular Group Convolutions

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

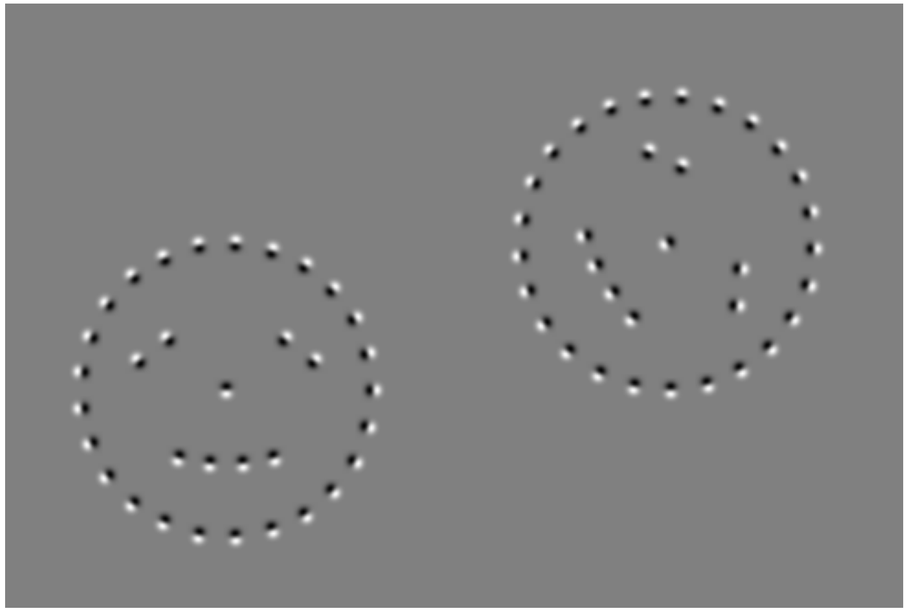
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

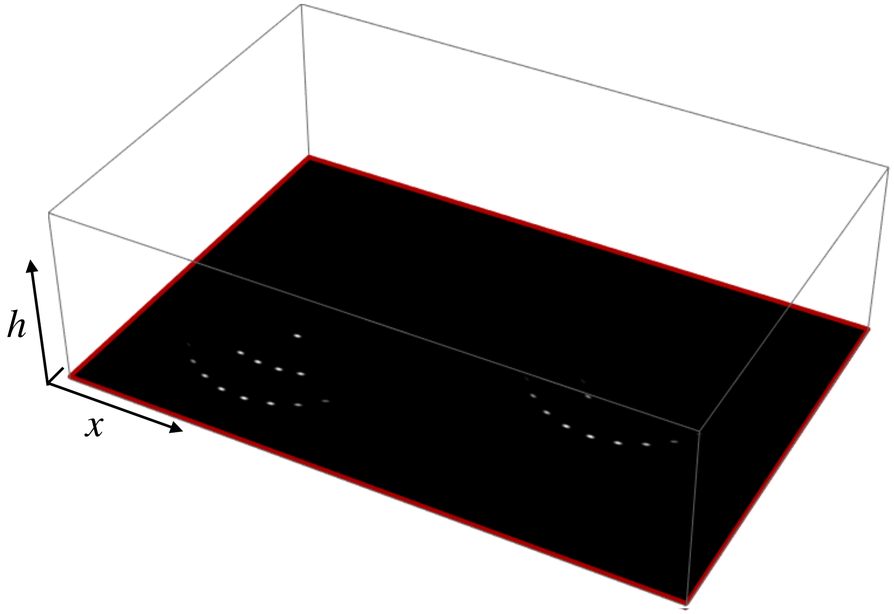
---

translation
"template matching"

$\star_{\mathbb{R}^2}$



$=$



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map (after ReLU)

Slide credit: Erik Bekkers, University of Amsterdam

Recap: Regular Group Convolutions

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

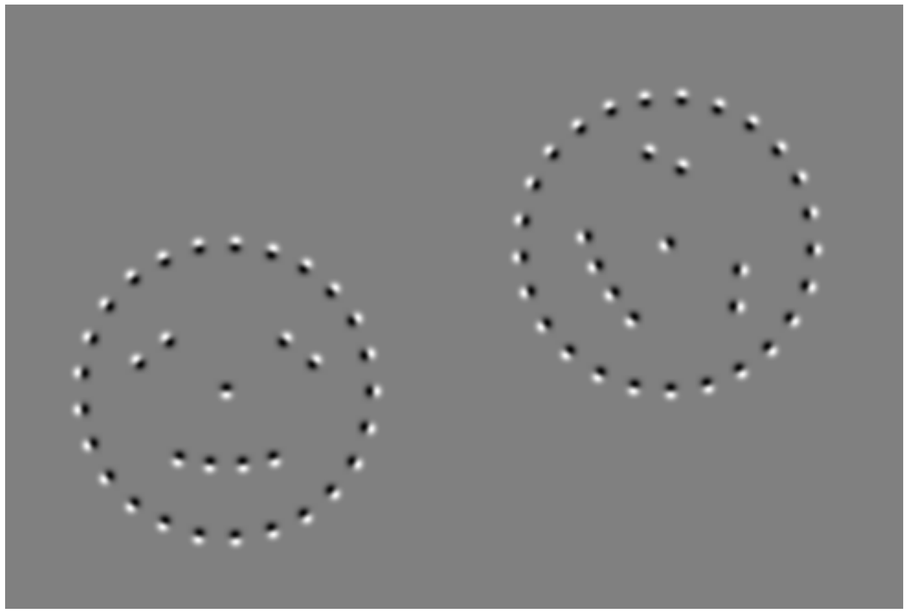
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

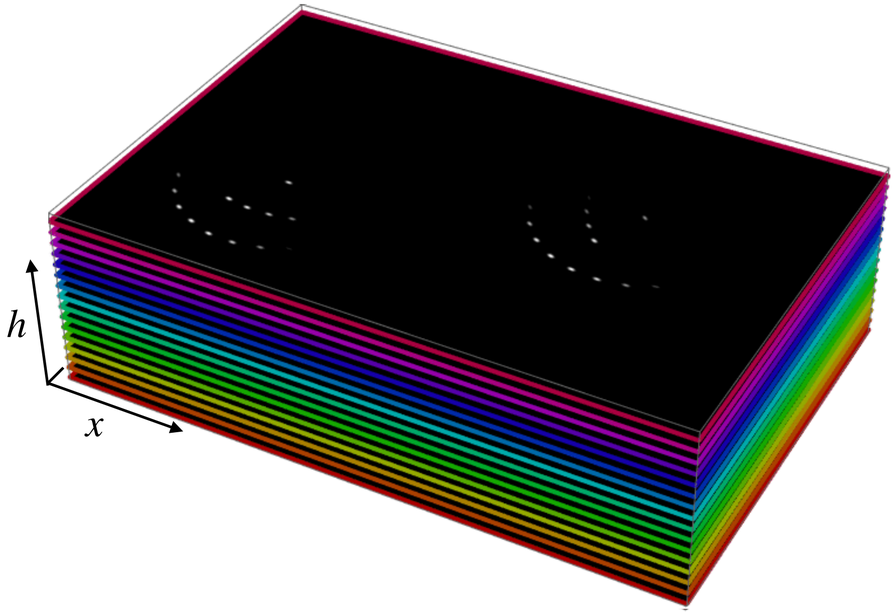
---

translation
"template matching"

$\star_{\mathbb{R}^2}$



$=$



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map (after ReLU)

Slide credit: Erik Bekkers, University of Amsterdam

Recap: Regular Group Convolutions

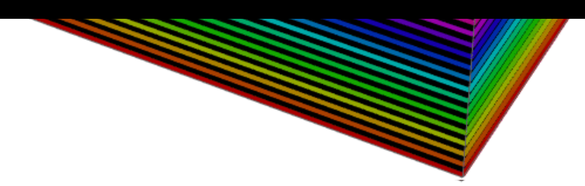
Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

$$(k \tilde{\star} f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \, , f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

Problem: Cost grows linearly with the discretization of the group and combinatorially with the *number of groups* - if you want to be scale- *and* rotation-equivariant, you need to re-scale every rotation of every kernel etc…



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map (after ReLU)

Slide credit: Erik Bekkers, University of Amsterdam

This happens naturally in neural networks to some degree!

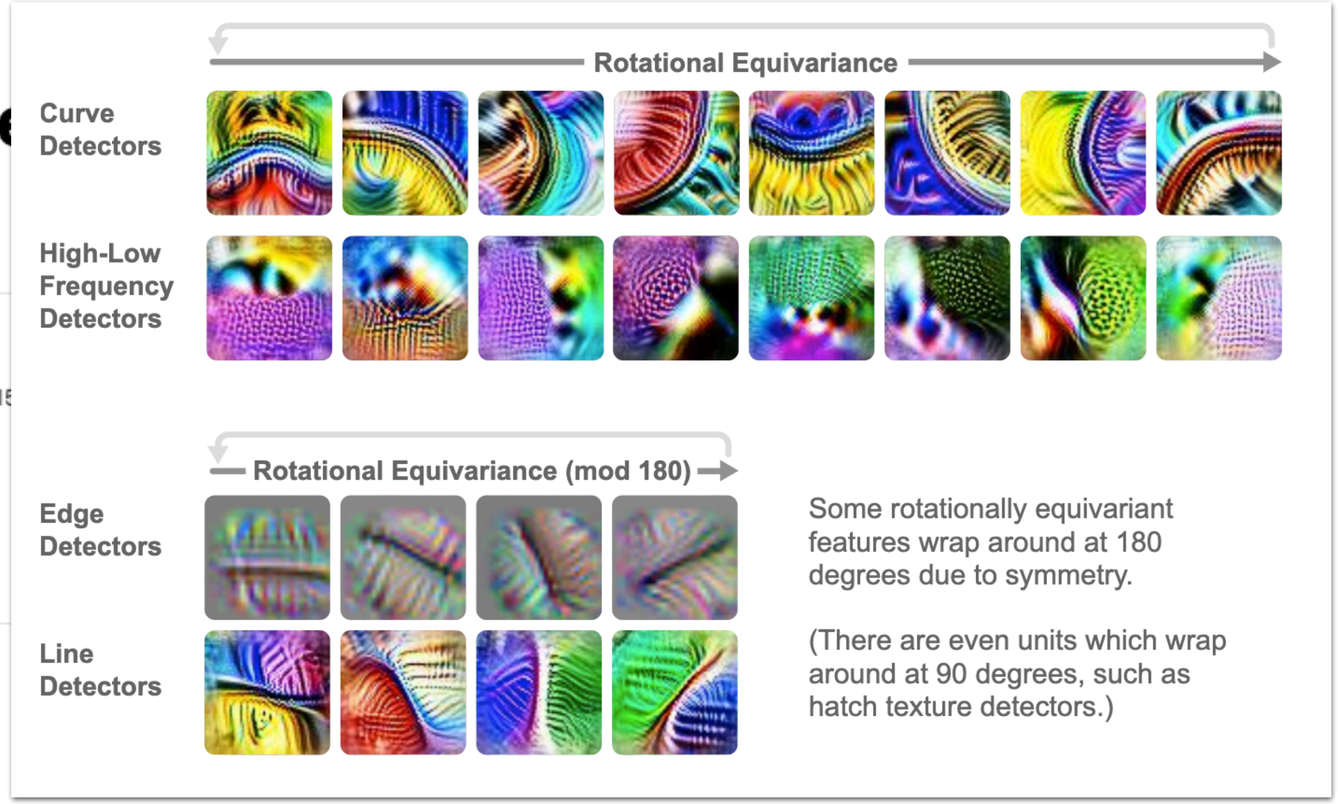
Naturally Occurring Equivariance in Neural Networks

AUTHORS	AFFILIATIONS	PUBLISHED	DOI
Chris Olah	OpenAI	Dec. 8, 2020	10.23915/distill.00024.004
Nick Cammarata	OpenAI		
Chelsea Voss	OpenAI		
Ludwig Schubert			
Gabriel Goh	OpenAI		

This happens naturally in neural networks to some degree!

Naturally Occurring Equivariance in Neural Networks

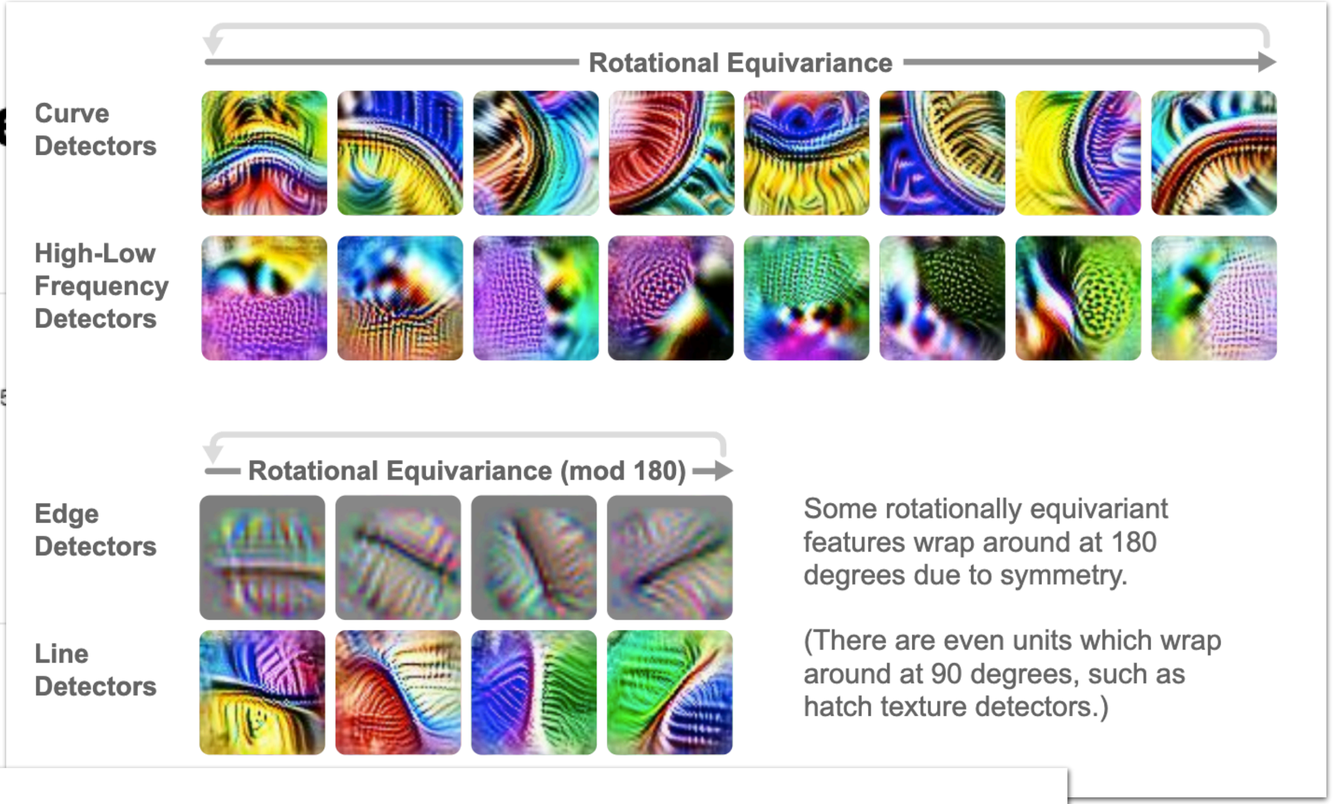
AUTHORS	AFFILIATIONS	PUBLISHED	DOI
Chris Olah	OpenAI	Dec. 8, 2020	10.2391
Nick Cammarata	OpenAI		
Chelsea Voss	OpenAI		
Ludwig Schubert			
Gabriel Goh	OpenAI		



This happens naturally in neural networks to some degree!

Naturally Occurring Equivariance Neural Networks

AUTHORS	AFFILIATIONS	PUBLISHED	DOI
Chris Olah	OpenAI	Dec. 8, 2020	10.23915
Nick Cammarata	OpenAI		
Chelsea Voss	OpenAI		
Ludwig Schubert			
Gabriel Goh	OpenAI		



One can test that these are genuinely rotated versions of the same feature by taking examples that cause one to fire, rotating them, and checking that the others fire as expected.

Recall: Steerable basis

A vector $Y(x) = \begin{pmatrix} \vdots \\ Y_l(x) \\ \vdots \end{pmatrix} \in \mathbb{K}^L$ with (basis) functions $Y_l \in \mathbb{L}_2(X)$ is steerable if

$$\forall_{g \in G}: \quad Y(g\,x) = \rho(g) Y(x),$$

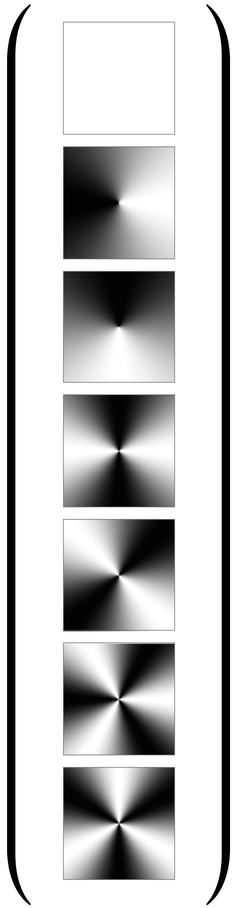
where $g\,x$ denotes the action of G on X and $\rho(g) \in \mathbb{K}^{L \times L}$ is a representation of G.

I.e., we can transform all basis functions simply by taking a linear combination of the original basis functions.

Slide credit: Erik Bekkers, University of Amsterdam

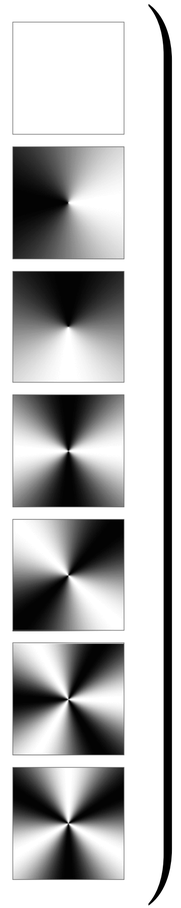
Circular Harmonics: Rotation-Steerable Functions

$$Y(\mathbf{R}_\theta^{-1}\,\mathbf{x}) \quad = \quad \rho(\mathbf{R}_\theta^{-1}) \quad Y(\mathbf{x})$$



$=$

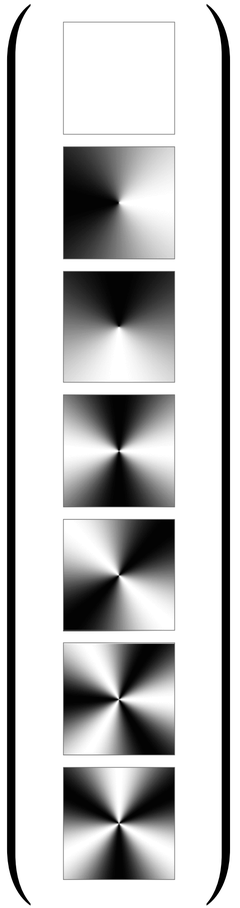
$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \cos\theta & \sin\theta & 0 & 0 & 0 & 0 \\
0 & -\sin\theta & \cos\theta & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \cos 2\theta & \sin 2\theta & 0 & 0 \\
0 & 0 & 0 & -\sin 2\theta & \cos 2\theta & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \cos 3\theta & \sin 3\theta \\
0 & 0 & 0 & 0 & 0 & -\sin 3\theta & \cos 3\theta
\end{pmatrix}$$



Slide credit: Erik Bekkers, University of Amsterdam

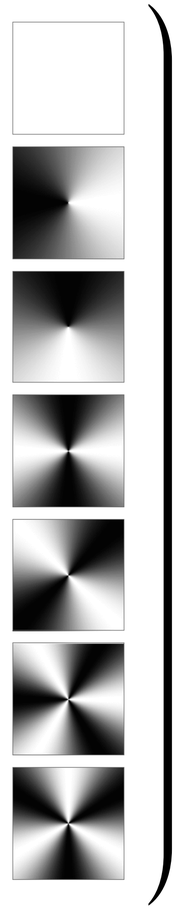
Circular Harmonics: Rotation-Steerable Functions

$$Y(\mathbf{R}_\theta^{-1}\,\mathbf{x}) \quad = \quad \rho(\mathbf{R}_\theta^{-1}) \quad Y(\mathbf{x})$$



$=$

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \cos\theta & \sin\theta & 0 & 0 & 0 & 0 \\
0 & -\sin\theta & \cos\theta & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \cos 2\theta & \sin 2\theta & 0 & 0 \\
0 & 0 & 0 & -\sin 2\theta & \cos 2\theta & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \cos 3\theta & \sin 3\theta \\
0 & 0 & 0 & 0 & 0 & -\sin 3\theta & \cos 3\theta
\end{pmatrix}$$



Slide credit: Erik Bekkers, University of Amsterdam

Two dimensional rotation-steerable functions

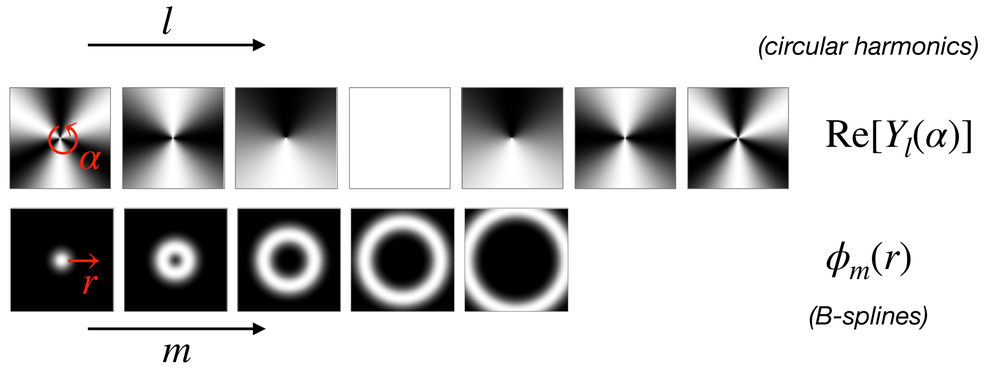
- Consider polar-separable convolution kernel:

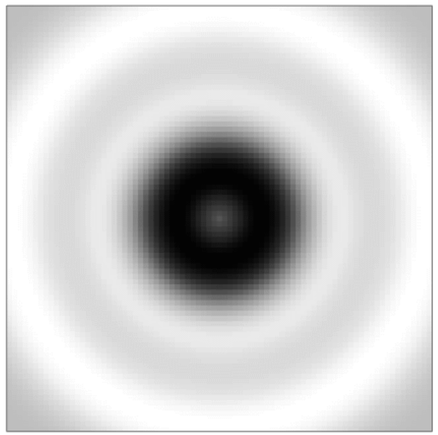
$$k(\mathbf{x}\,|\,\mathbf{w}) = k^{\rightarrow}(r\,|\,\mathbf{w})\,k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}),$$

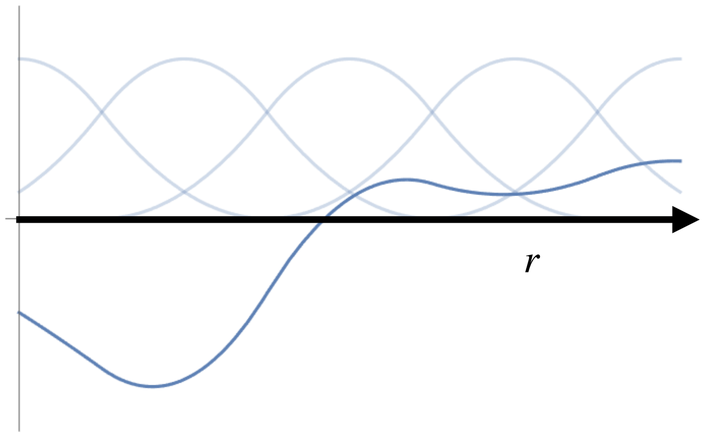
- with $k^{\circlearrowleft}$ in an $SO(2)$ steerable basis, and $k^{\rightarrow}$ in some radial basis:

$$k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}) = \sum_l \overline{w}_l Y_l(\alpha), \qquad \text{e.g., with} \qquad Y_l(\alpha) = e^{i\,l\,\alpha},$$

$$k^{\rightarrow}(r\,|\,\mathbf{w}) = \sum_m w_m \phi_m(r)$$







Slide credit: Erik Bekkers, University of Amsterdam

Two dimensional rotation-steerable functions

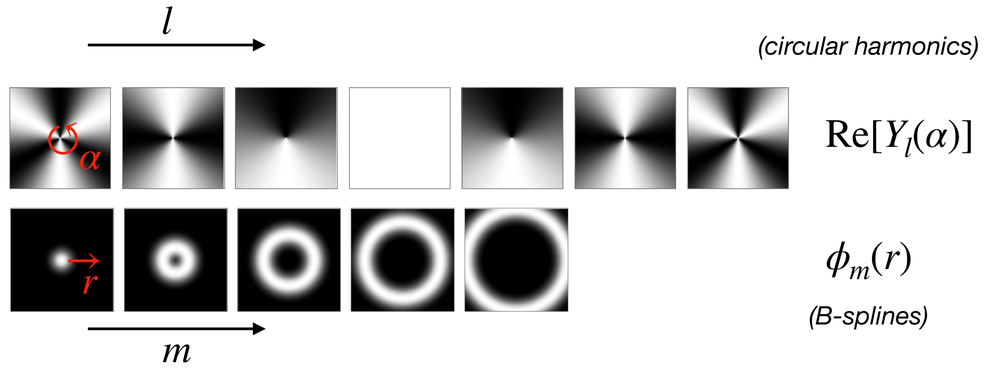
- Consider polar-separable convolution kernel:

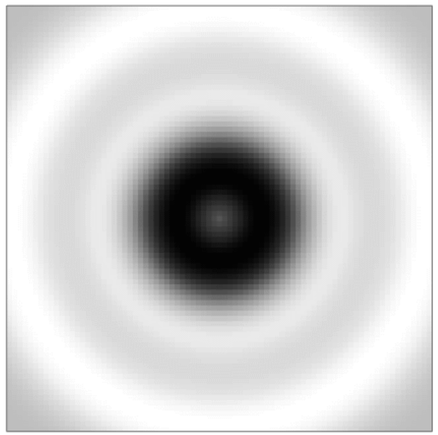
$$k(\mathbf{x}\,|\,\mathbf{w}) = k^{\rightarrow}(r\,|\,\mathbf{w})\,k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}),$$

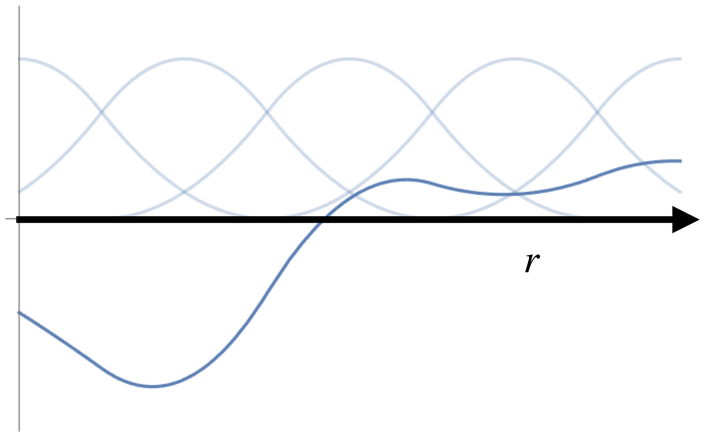
- with $k^{\circlearrowleft}$ in an $SO(2)$ steerable basis, and $k^{\rightarrow}$ in some radial basis:

$$k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}) = \sum_l \overline{w}_l Y_l(\alpha), \qquad \text{e.g., with} \qquad Y_l(\alpha) = e^{i\,l\,\alpha},$$

$$k^{\rightarrow}(r\,|\,\mathbf{w}) = \sum_m w_m \phi_m(r)$$







Slide credit: Erik Bekkers, University of Amsterdam

Two dimensional rotation-steerable functions

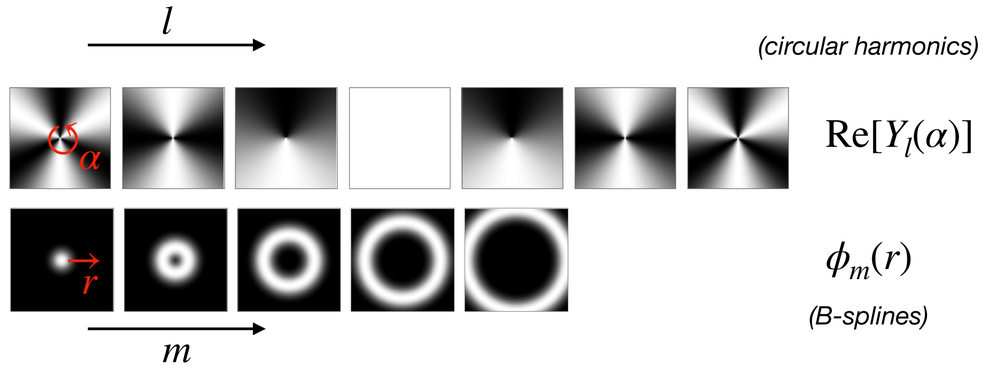
- Consider polar-separable convolution kernel:

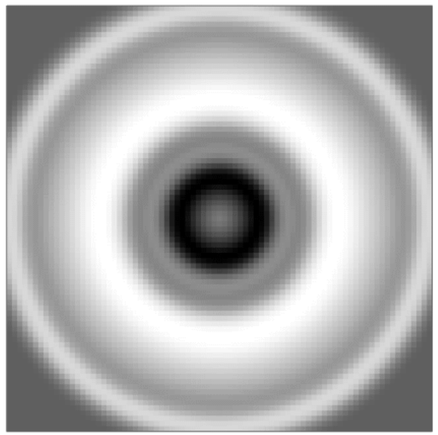
$$k(\mathbf{x}\,|\,\mathbf{w}) = k^{\rightarrow}(r\,|\,\mathbf{w})\,k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}),$$

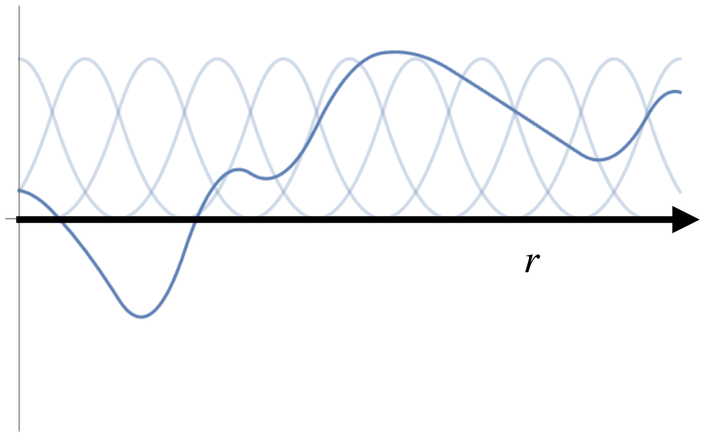
- with $k^{\circlearrowleft}$ in an $SO(2)$ steerable basis, and $k^{\rightarrow}$ in some radial basis:

$$k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}) = \sum_l \overline{w}_l Y_l(\alpha), \qquad \text{e.g., with} \qquad Y_l(\alpha) = e^{i\,l\,\alpha},$$

$$k^{\rightarrow}(r\,|\,\mathbf{w}) = \sum_m w_m \phi_m(r)$$







Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \,\tilde{\star}\, f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

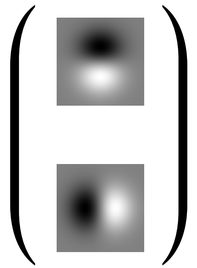
Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \,\tilde{\star}\, f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$k(\mathbf{x} \,|\, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^{\dagger} Y(\mathbf{x})$$



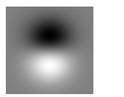
Slide credit: Erik Bekkers, University of Amsterdam

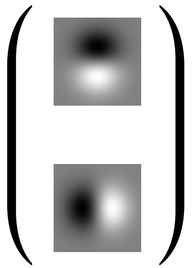
Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \,\tilde{\star}\, f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$k(h^{-1}\,\mathbf{x} \,|\, \hat{\mathbf{w}}) = (\rho(h)\hat{\mathbf{w}})^{\dagger} \quad Y(\mathbf{x})$$





Slide credit: Erik Bekkers, University of Amsterdam

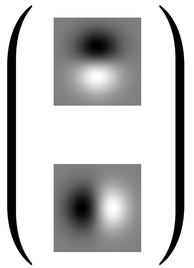
Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \tilde{\star} f)(\mathbf{x}, h)$$

$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$k(h^{-1}\mathbf{x} \,|\, \hat{\mathbf{w}}) = (\rho(h)\hat{\mathbf{w}})^\dagger \quad Y(\mathbf{x})$$

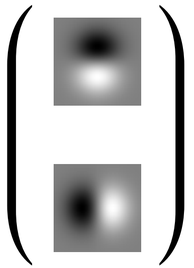


Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$): $(k \tilde{\star} f)(\mathbf{x}, h)$

$$k(h^{-1}\,\mathbf{x} \,|\, \hat{\mathbf{w}}) = (\rho(h)\hat{\mathbf{w}})^{\dagger} \; Y(\mathbf{x})$$



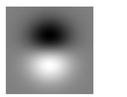
$$= (\rho(h)\,\hat{\mathbf{w}})^{\dagger} \int_{\mathbb{R}^d} Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

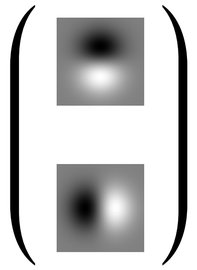
Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$): $(k \tilde{\star} f)(\mathbf{x}, h)$

$$k(h^{-1}\,\mathbf{x} \,|\, \hat{\mathbf{w}}) = (\rho(h)\hat{\mathbf{w}})^{\dagger} \quad Y(\mathbf{x})$$





$$= (\rho(h)\ \hat{\mathbf{w}})^{\dagger}\ \hat{f}^{Y}(\mathbf{x})$$

Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$): $(k \tilde{\star} f)(\mathbf{x}, h)$

$$k(h^{-1}\,\mathbf{x}\,|\,\hat{\mathbf{w}}) = (\rho(h)\hat{\mathbf{w}})^{\dagger}\; Y(\mathbf{x})$$

$$= (\rho(h)\;\hat{\mathbf{w}})^{\dagger}\;\hat{f}^{Y}(\mathbf{x})$$

! Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. IEEE Transactions on Pattern analysis and machine intelligence, 13(9), 891-906.

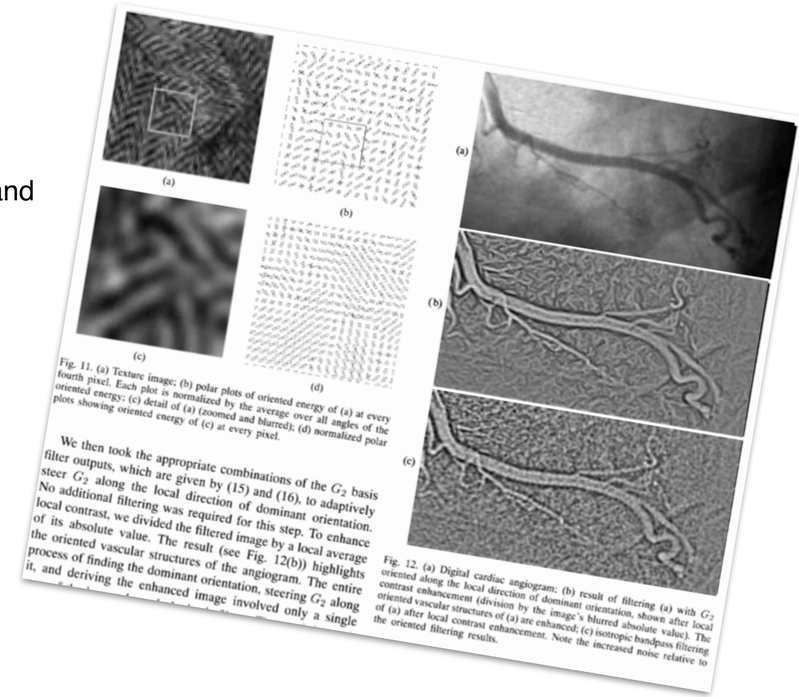


Fig. 11. (a) Texture image; (b) polar plots of oriented energy of (a) at every fourth pixel. Each plot is normalized by the average over all angles of the oriented energy; (c) detail of (a) (zoomed and blurred); (d) normalized polar plots showing oriented energy of (c) at every pixel.

We then took the appropriate combinations of the G_2 basis filter outputs, which are given by (15) and (16). to adaptively steer G_2 along the local direction of dominant orientation. No additional filtering was required for this step. To enhance local contrast, we divided the filtered image by a local average of its absolute value. The result (see Fig. 12(b)) highlights the oriented vascular structures of the angiogram. The entire process of finding the dominant orientation, steering G_2 along it, and deriving the enhanced image involved only a single

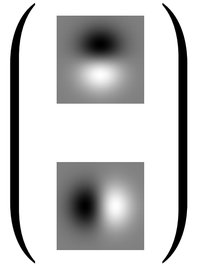
Fig. 12. (a) Digital cardiac angiogram; (b) result of filtering (a) with G_2 oriented along the local direction of dominant orientation, shown after local contrast enhancement (division by the image's blurred absolute value). The oriented vascular structures of (a) are enhanced; (c) isotropic bandpass filtering of (a) after local contrast enhancement. Note the increased noise relative to the oriented filtering results.

Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

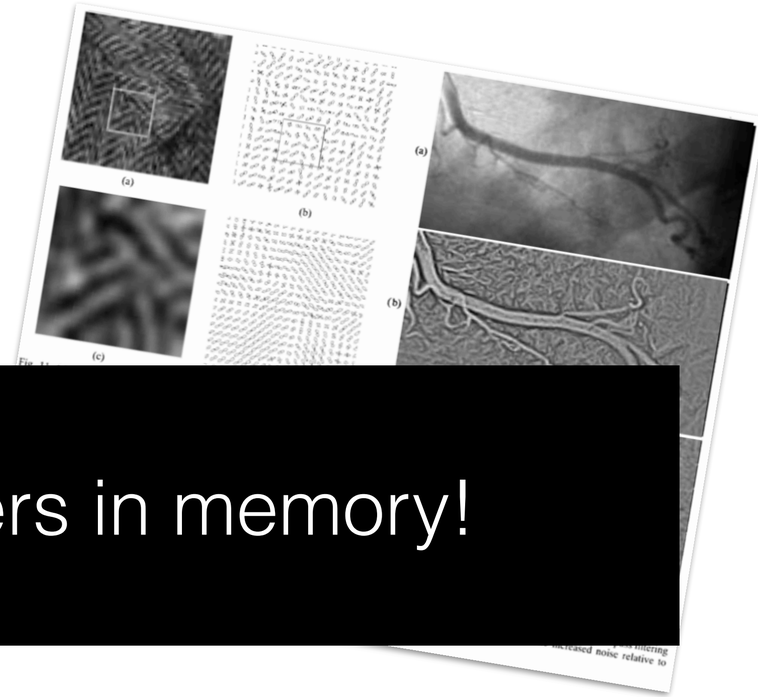
Group convolution ($G = \mathbb{R}^d \rtimes H$): $(k \tilde{\star} f)(\mathbf{x}, h)$

$$k(h^{-1}\,\mathbf{x}\,|\,\hat{\mathbf{w}}) = (\rho(h)\hat{\mathbf{w}})^{\dagger}\; Y(\mathbf{x})$$



$$= (\rho(h)\;\hat{\mathbf{w}})^{\dagger}\;\hat{f}^{Y}(\mathbf{x})$$

! Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. IEEE Transactions on Pattern analysis and machine intelligence, 13(9), 891-906.



Only need to store responses of bases filters in memory!

Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

$$(k \,\tilde{\star}\, f)(\mathbf{x}, \theta) = (\rho(\mathbf{R}_\theta)\hat{\mathbf{w}})^\dagger \, \hat{f}^Y(\mathbf{x})$$

Slightly awkward: Need to "keep track" of $\hat{\mathbf{w}}$ to get response for a particular orientation….

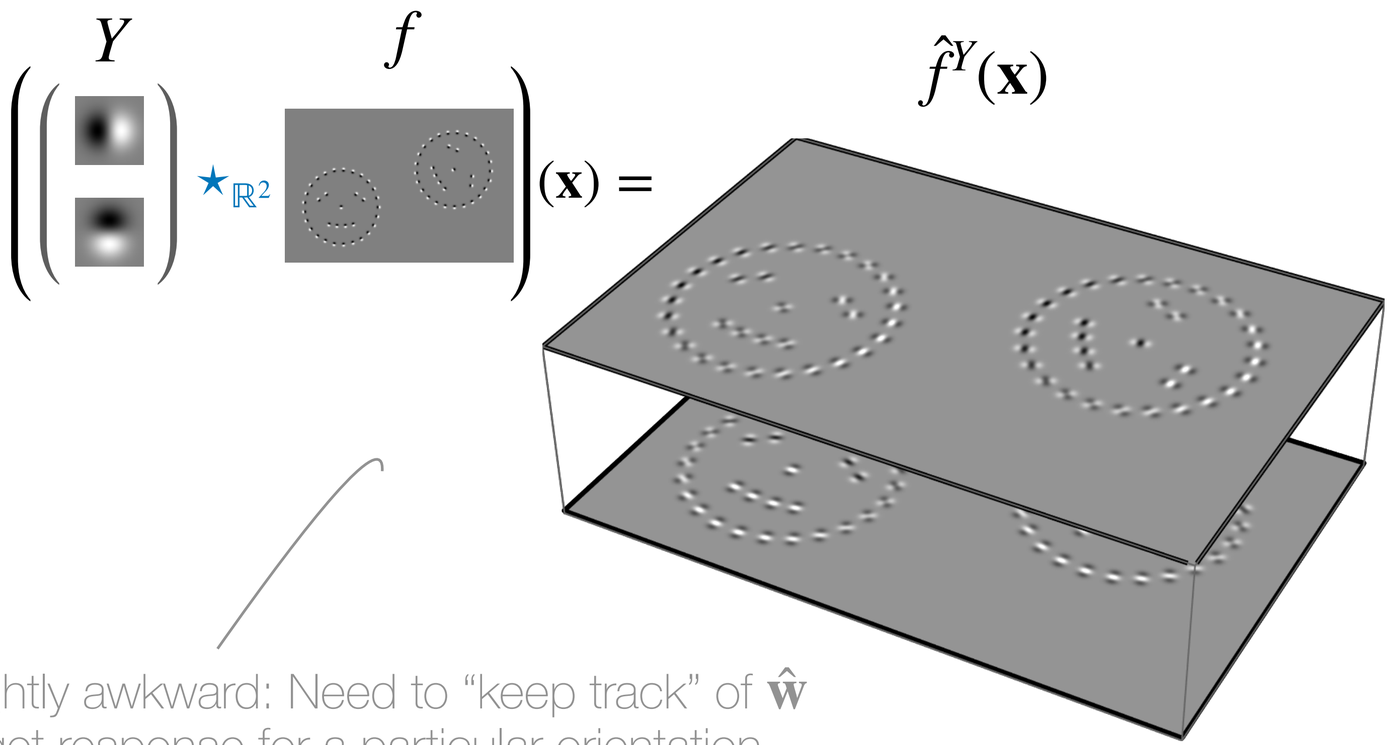
Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

$$(k \,\tilde{\star}\, f)(\mathbf{x}, \theta) = (\rho(\mathbf{R}_\theta)\hat{\mathbf{w}})^\dagger \, \hat{f}^Y(\mathbf{x})$$



Slightly awkward: Need to "keep track" of $\hat{\mathbf{w}}$ to get response for a particular orientation….

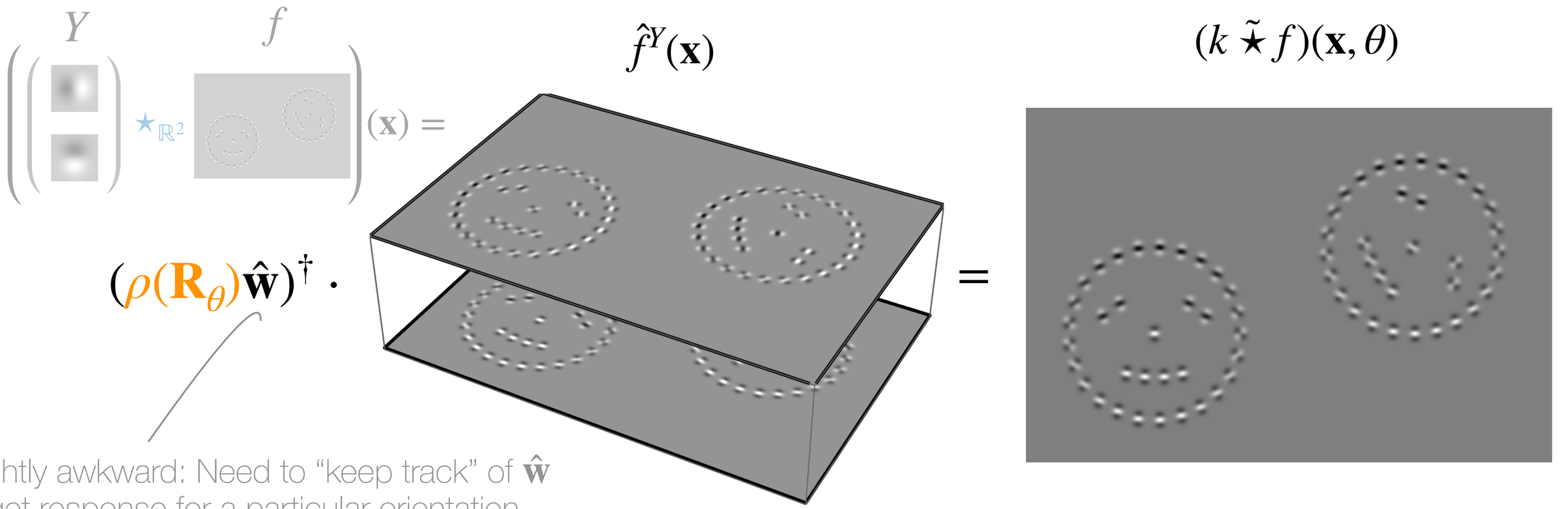
Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

$$(k \,\tilde{\star}\, f)(\mathbf{x}, \theta) = (\rho(\mathbf{R}_\theta)\hat{\mathbf{w}})^\dagger \, \hat{f}^Y(\mathbf{x})$$



Slightly awkward: Need to "keep track" of $\hat{\mathbf{w}}$ to get response for a particular orientation….

Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$): $(k \tilde{\star} f)(\mathbf{x}, h)$

$$k(h^{-1}\,\mathbf{x}\,|\,\hat{\mathbf{w}}) = (\rho(h^{-1})\hat{\mathbf{w}})^T\; Y(\mathbf{x})$$

$$= (\rho(h)\;\hat{\mathbf{w}})^{\dagger}\;\hat{f}^{Y}(\mathbf{x})$$

!

Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$): $(k \tilde{\star} f)(\mathbf{x}, h)$

$$k(h^{-1}\,\mathbf{x}\,|\,\hat{\mathbf{w}}) = (\rho(h^{-1})\hat{\mathbf{w}})^T\; Y(\mathbf{x})$$

$$= (\rho(h)\;\hat{\mathbf{w}})^{\dagger}\;\hat{f}^{Y}(\mathbf{x})$$

!

Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$): $(k \tilde{\star} f)(\mathbf{x}, h)$

$$k(h^{-1}\,\mathbf{x}\,|\,\hat{\mathbf{w}}) = (\rho(h^{-1})\hat{\mathbf{w}})^T\, Y(\mathbf{x})$$

$$= \mathrm{tr}(\, \hat{f}^Y(\mathbf{x})\, \hat{\mathbf{w}}^\dagger\, \rho(h^{-1})\,)$$

!

$\mathbf{a}^T\,\mathbf{b} = \mathrm{tr}(\mathbf{b}\,\mathbf{a}^T)$ and $\rho(h)^\dagger = \rho(h^{-1})$

Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$): $(k \tilde{\star} f)(\mathbf{x}, h)$

$k(h^{-1}\,\mathbf{x}\,|\,\hat{\mathbf{w}}) = (\rho(h^{-1})\hat{\mathbf{w}})^T\; Y(\mathbf{x})$

!

$= \mathrm{tr}(\; \hat{f}(\mathbf{x})\; \rho(h^{-1})\;)$

$\hat{f}(\mathbf{x}) = \hat{f}^Y(\mathbf{x})\; \hat{w}^\dagger$

Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$): $(k \tilde{\star} f)(\mathbf{x}, h)$

$k(h^{-1}\,\mathbf{x}\,|\,\hat{\mathbf{w}}) = (\rho(h^{-1})\hat{\mathbf{w}})^T\ Y(\mathbf{x})$

!

$= \mathcal{F}_H^{-1}[\,\hat{f}(\mathbf{x})\,](h)$

Inverse H-Fourier transform!

Slide credit: Erik Bekkers, University of Amsterdam

Lifting convolution with steerable kernel

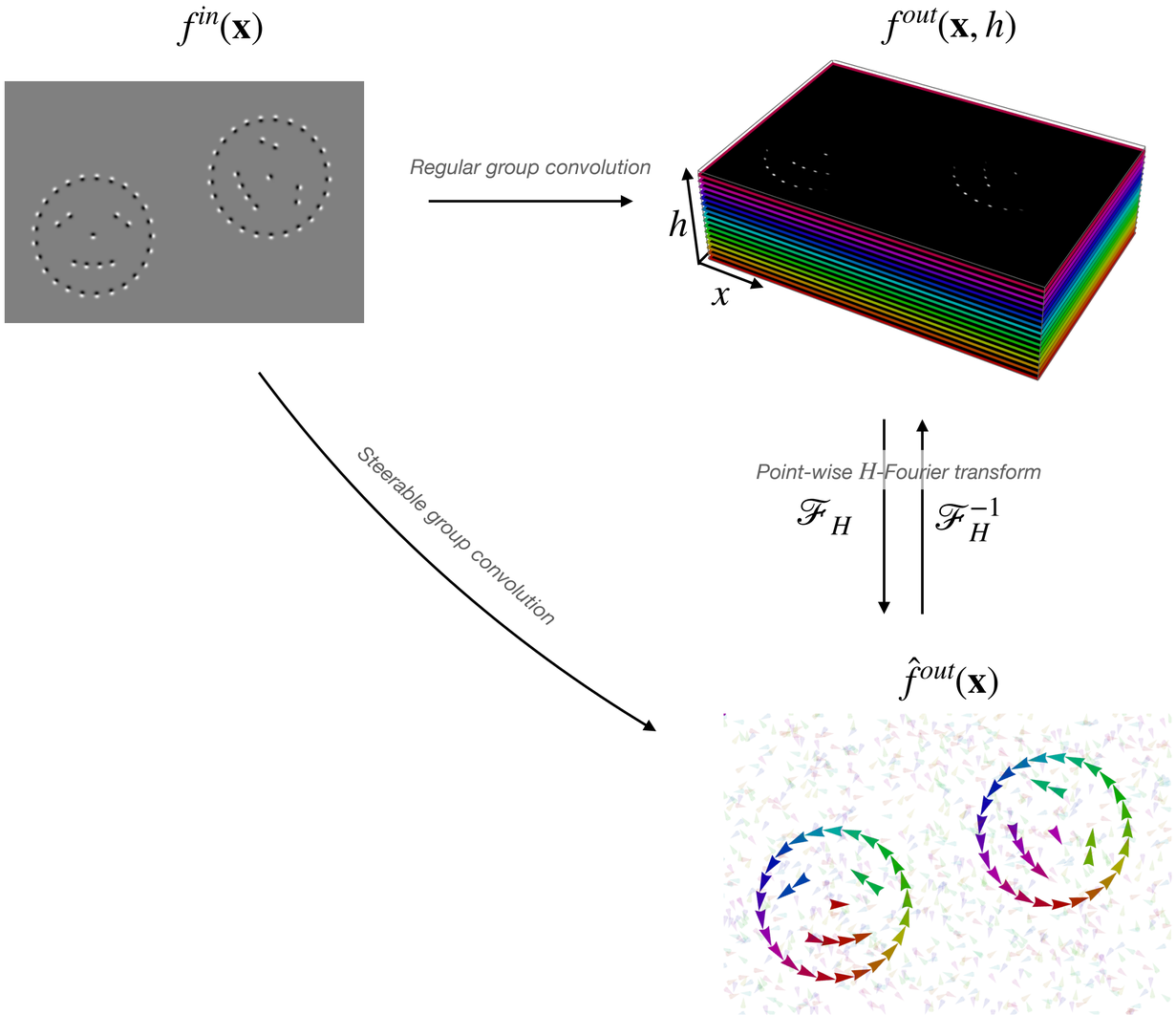
Group convolution ($G = \mathbb{R}^d \rtimes H$): $(k \tilde{\star} f)(\mathbf{x}, h)$

The coefficients $\hat{\mathbf{w}}$ can be absorbed into the features.
The features for a particular pixel location can then be interpreted as a pixel-wise *fourier coefficients with respect to the rotation group*!

$= \mathcal{F}_H^{-1}[\, \hat{f}(\mathbf{x}) \,](h)$ Inverse H-Fourier transform!

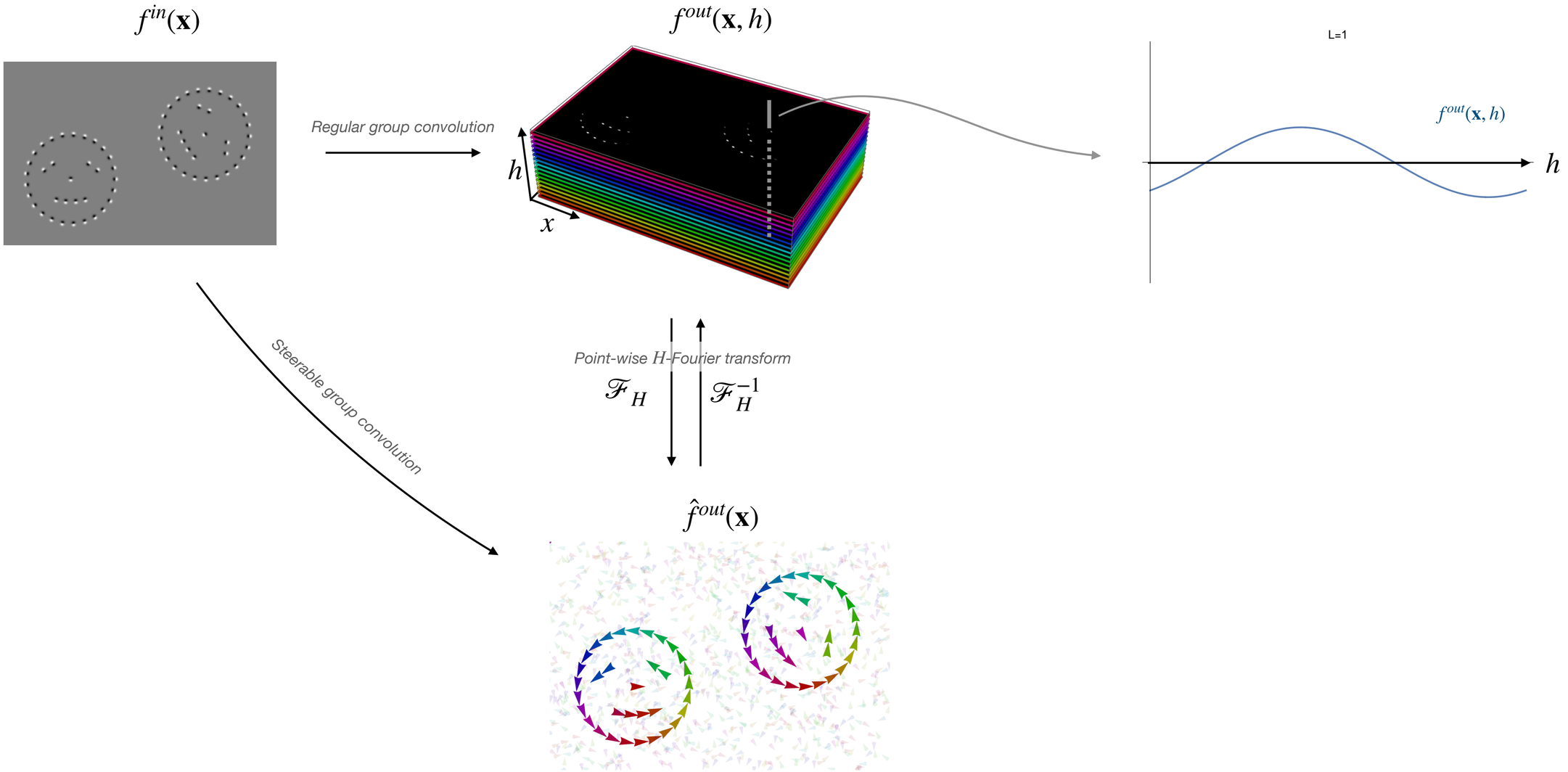
Slide credit: Erik Bekkers, University of Amsterdam

From regular to steerable via a Fourier transform



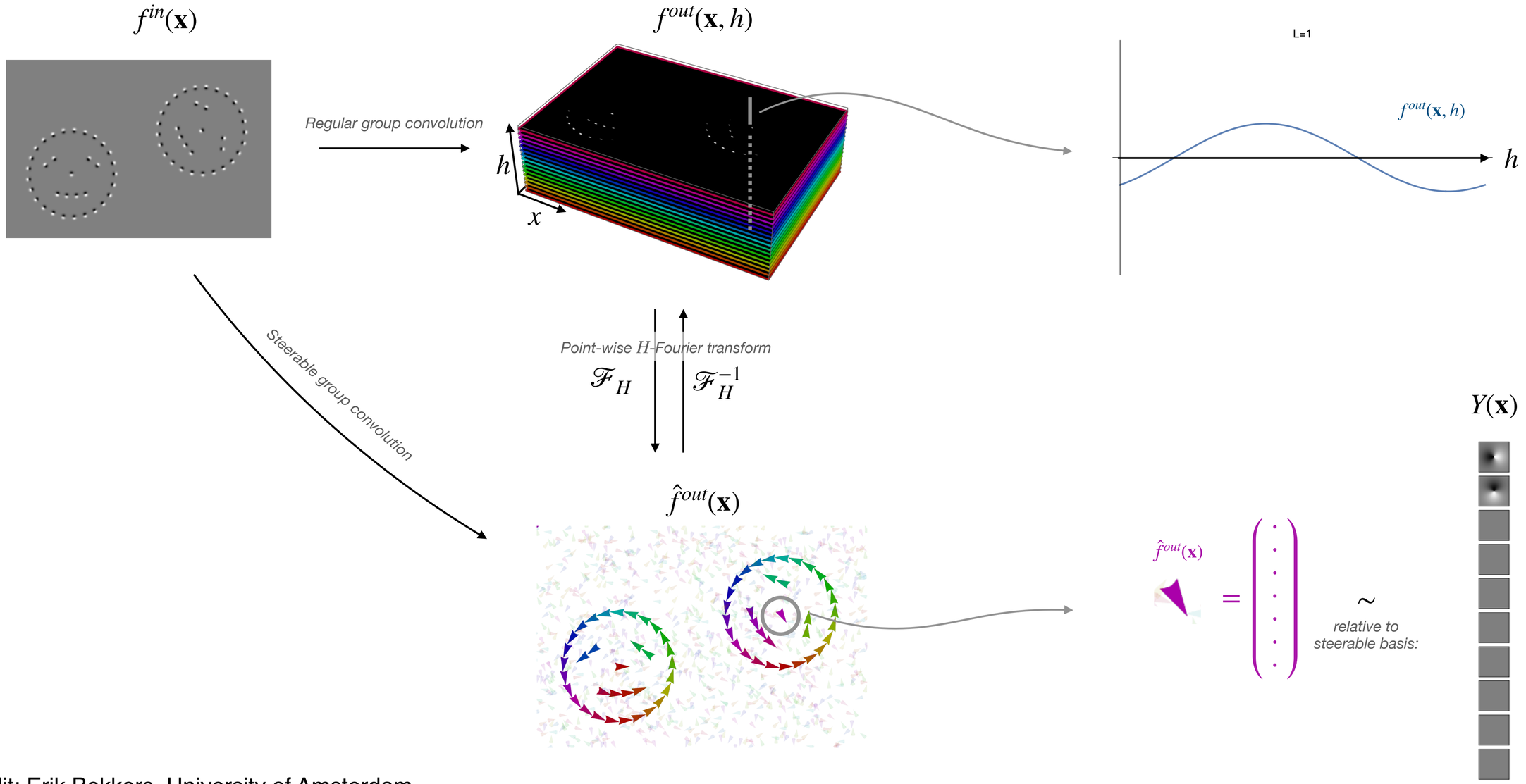
Slide credit: Erik Bekkers, University of Amsterdam

From regular to steerable via a Fourier transform



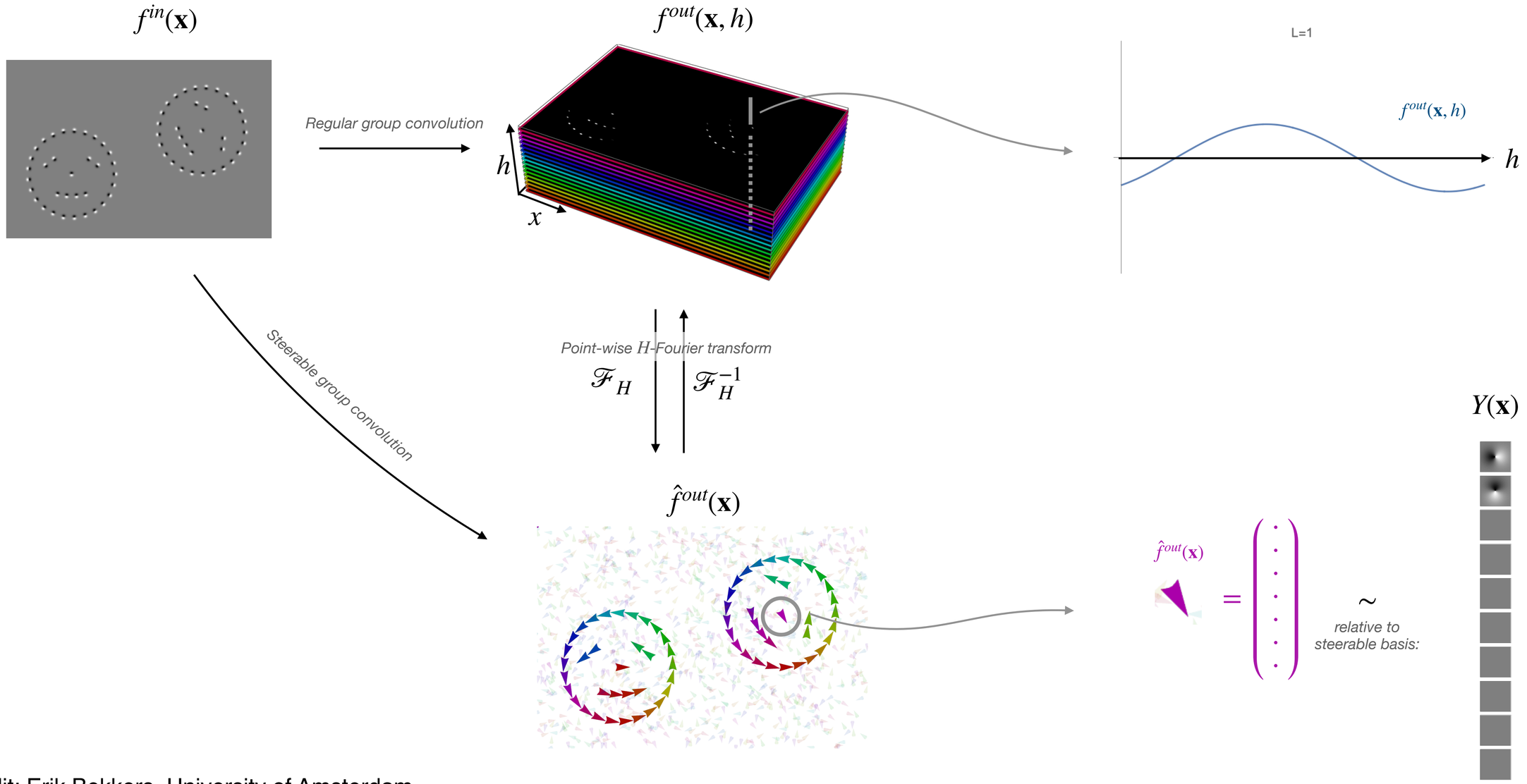
Slide credit: Erik Bekkers, University of Amsterdam

From regular to steerable via a Fourier transform



Slide credit: Erik Bekkers, University of Amsterdam

From regular to steerable via a Fourier transform



Slide credit: Erik Bekkers, University of Amsterdam

From regular to steerable via a Fourier transform

Regular group convolutions:
Domain expanded feature maps

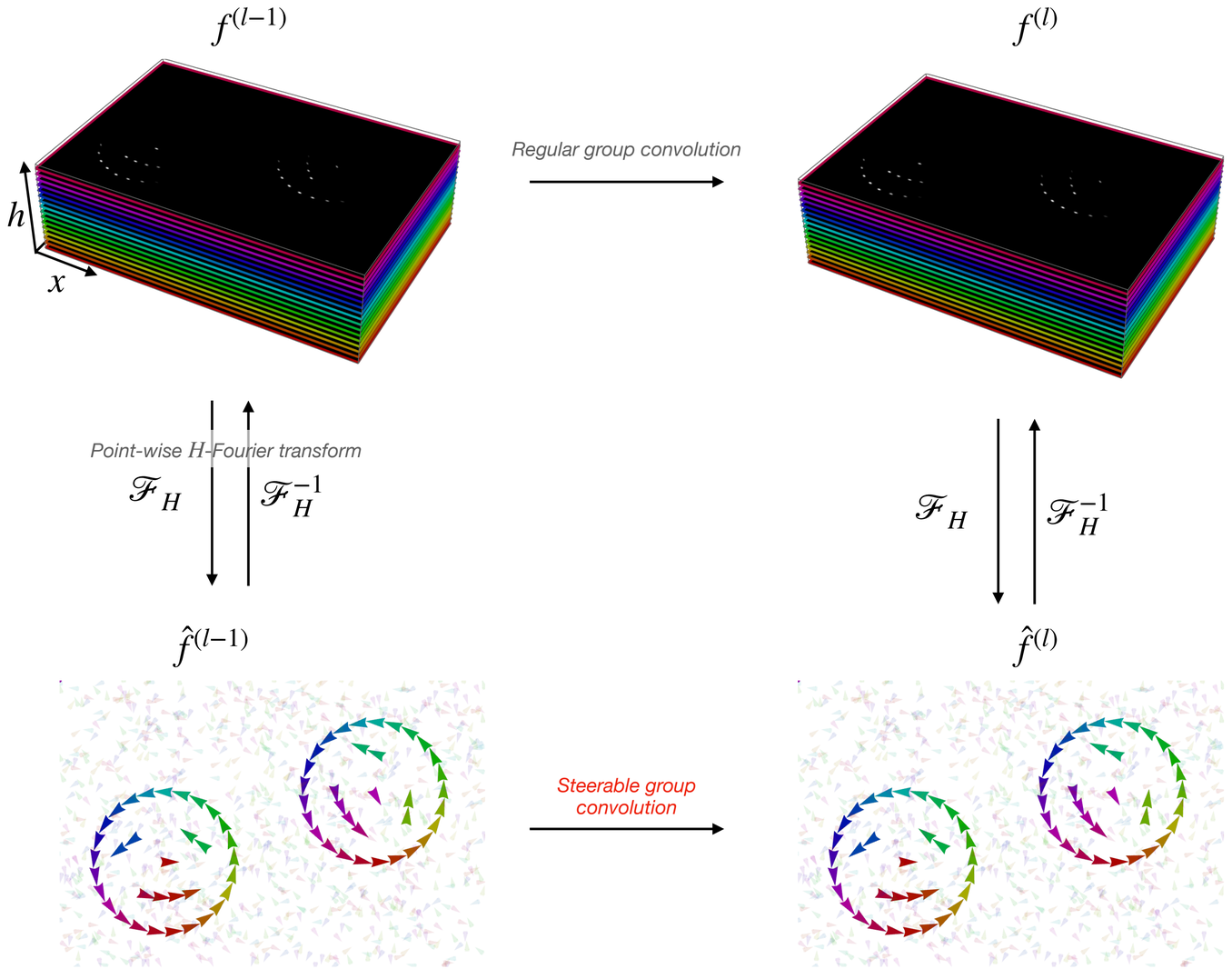
$$f^{(l)} : \mathbb{R}^d \times H \to \mathbb{R}$$

added axis

Steerable group convolutions:
Co-domain expanded feature maps (feature fields)

$$\hat{f}^{(l)} : \mathbb{R}^d \to V_H$$

vector field instead of scalar field
(vectors in V_H transform via group H representations)



Slide credit: Erik Bekkers, University of Amsterdam

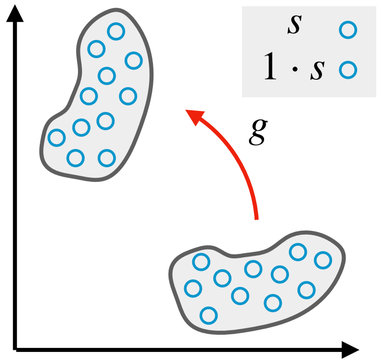
Feature field and induced representation

We call $\hat{f} : \mathbb{R}^d \to \mathbb{R}^{d_\rho}$ a feature vector field, or simply a **feature field**, if its

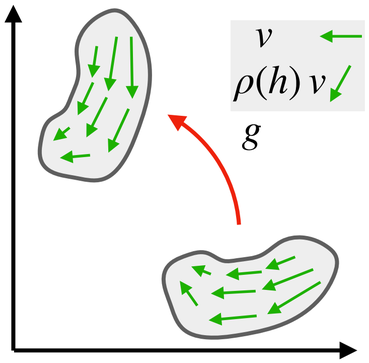
codomain	transforms via a *representation*	$\rho(h)$	of H
domain	transforms via the action	g^{-1}	of $G = (\mathbb{R}^d, +) \rtimes H$

Representation ρ defines the **type** of the field, and together with the group action of $G = (\mathbb{R}^d, +) \rtimes H$ defines the **induced representation**

$$\left(\mathrm{Ind}_H^G[\rho](\mathbf{x}, h)\,\hat{f}\right)(\mathbf{x}') := \rho(h)\,\hat{f}(h^{-1}(\mathbf{x}' - \mathbf{x}))$$



Scalar field ($\rho(h) = 1$)



vector field

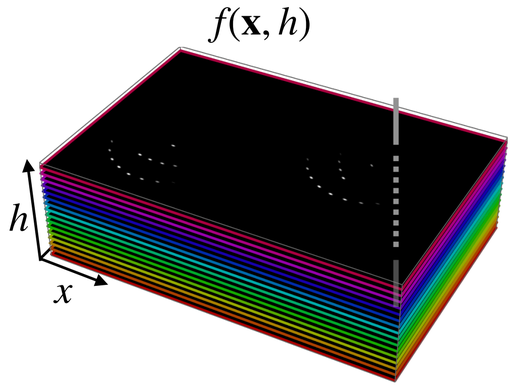
Figure adapted from: Weiler, M., & Cesa, G. (2019). General e (2)-equivariant steerable cnns. NeurIPS
See also https://github.com/QUVA-Lab/e2cnn

Slide credit: Erik Bekkers, University of Amsterdam

Feature field and induced representation

Regular G feature maps: $f(\mathbf{x}, h)$ considered so far can be considered feature fields.

$$(\mathscr{L}_g f)(\mathbf{x}', h') = f(h^{-1}(\mathbf{x}' - \mathbf{x}), h^{-1}h)$$

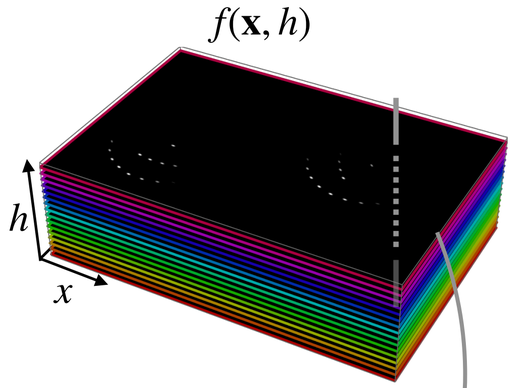


Slide credit: Erik Bekkers, University of Amsterdam

Feature field and induced representation

Regular G feature maps: $f(\mathbf{x}, h)$ considered so far can be considered feature fields.

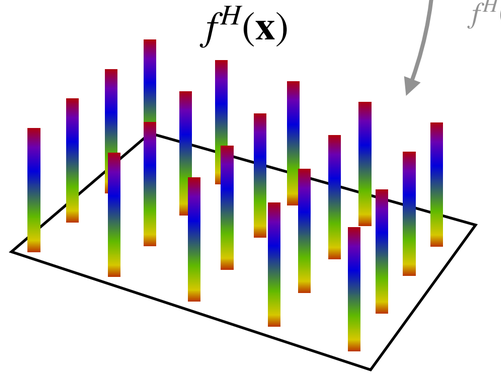
$$(\mathscr{L}_g f)(\mathbf{x}', h') = f(h^{-1}(\mathbf{x}' - \mathbf{x}), h^{-1}h)$$



Regular H feature fields: Let $f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$ be the field of functions $f^H(\mathbf{x}) : H \to \mathbb{R}$ on the subgroup H, then the functions (**fibers**) transform via the regular representation $\mathscr{L}^H_h$

(recall. $\mathscr{L}^H_h f(h') = f(h^{-1}h')$)

$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}^G_H[\ \mathscr{L}^H_h\ ](\mathbf{x}, h) f^H\right)(\mathbf{x}')$$



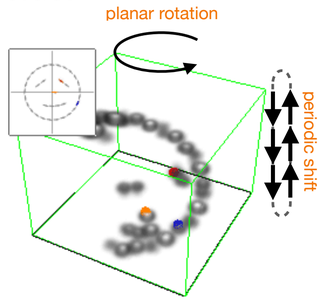
$f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$

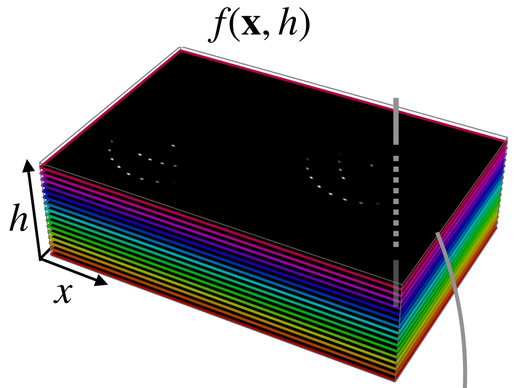
Slide credit: Erik Bekkers, University of Amsterdam

Feature field and induced representation

Regular G feature maps: $f(\mathbf{x}, h)$ considered so far can be considered feature fields.

$$(\mathscr{L}_g f)(\mathbf{x}', h') = f(h^{-1}(\mathbf{x}' - \mathbf{x}), h^{-1}h)$$



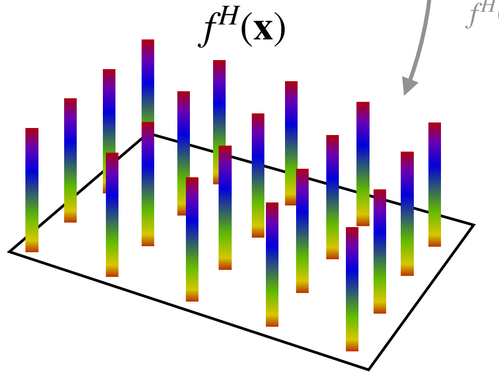


Regular H feature fields: Let $f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$ be the field of functions $f^H(\mathbf{x}) : H \to \mathbb{R}$ on the subgroup H, then the functions (**fibers**) transform via the regular representation $\mathscr{L}_h^H$

(recall. $\mathscr{L}_h^H f(h') = f(h^{-1}h')$)

$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}_H^G[\ \mathscr{L}_h^H\ ](\mathbf{x}, h) f^H\right)(\mathbf{x}')$$

$f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$

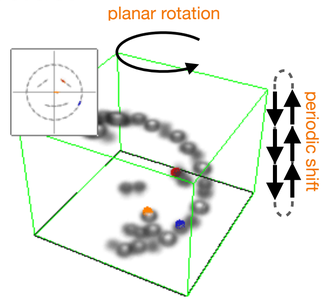


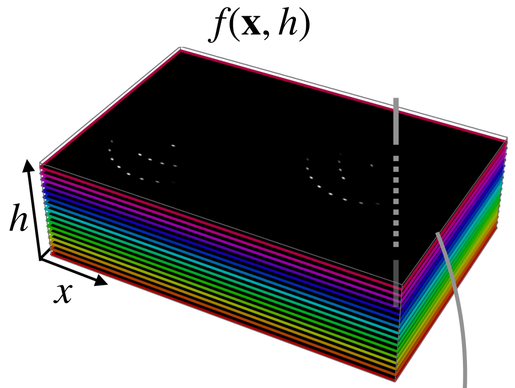
Slide credit: Erik Bekkers, University of Amsterdam

Feature field and induced representation

Regular G feature maps: $f(\mathbf{x}, h)$ considered so far can be considered feature fields.

$$(\mathscr{L}_g f)(\mathbf{x}', h') = f(h^{-1}(\mathbf{x}' - \mathbf{x}), h^{-1}h)$$

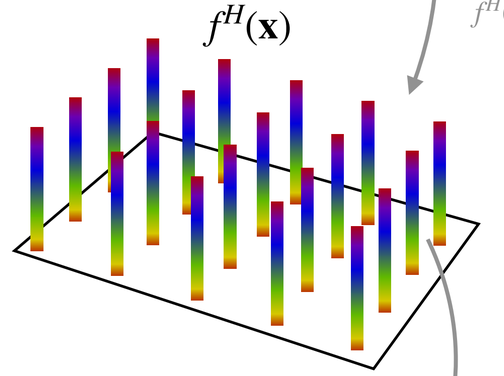




Regular H feature fields: Let $f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$ be the field of functions $f^H(\mathbf{x}) : H \to \mathbb{R}$ on the subgroup H, then the functions (**fibers**) transform via the regular representation $\mathscr{L}^H_h$

(recall. $\mathscr{L}^H_h f(h') = f(h^{-1}h')$)

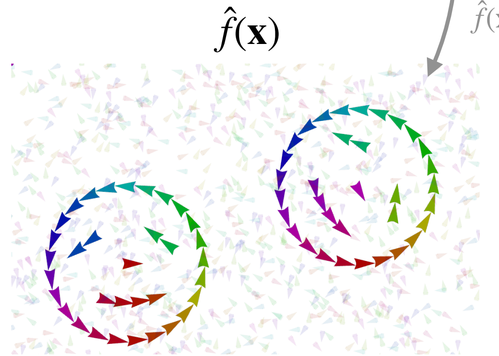
$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}^G_H[\ \mathscr{L}^H_h\ ](\mathbf{x}, h) f^H\right)(\mathbf{x}')$$



$f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$

Steerable H feature fields: Since the fibers $f^H(\mathbf{x})$ are functions on H we can represent them via their Fourier coefficients $\hat{f}(\mathbf{x}) = \mathscr{F}_H[f^H(\mathbf{x})]$. These vectors of coefficients transform via irreps $\rho(h) = \oplus_l \rho_l(h)$

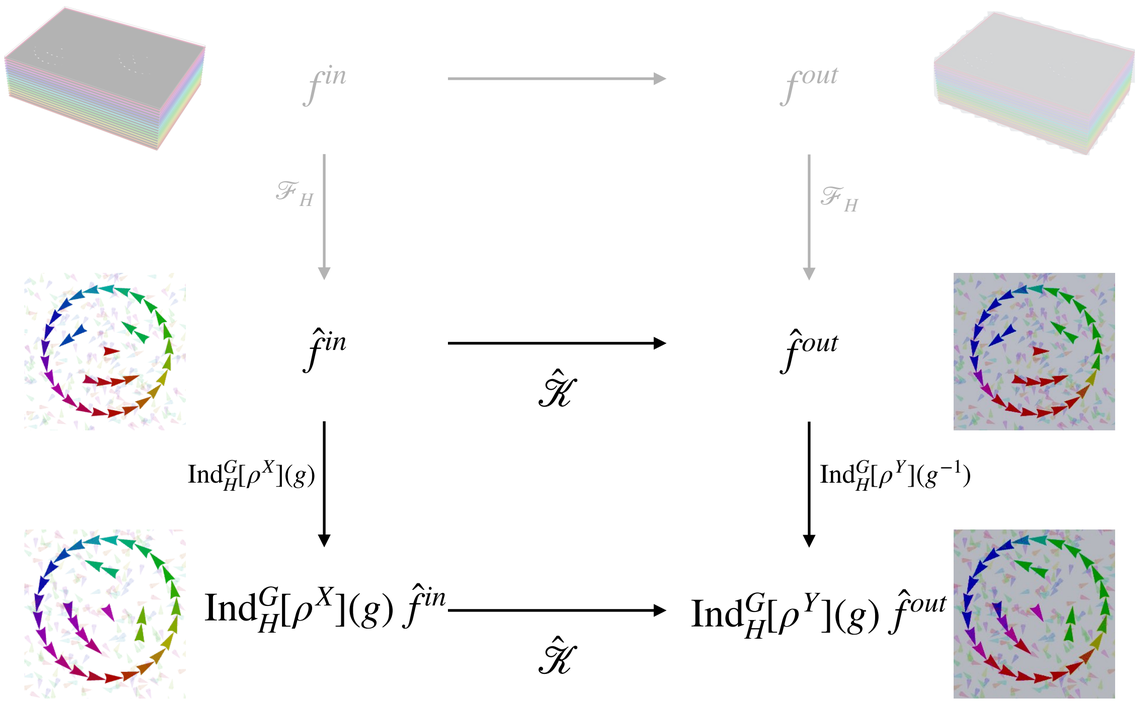
$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}^G_H[\ \mathscr{L}^H_h\ ](\mathbf{x}, h) \hat{f}\right)(\mathbf{x}') \iff \left(\mathrm{Ind}^G_H[\ \rho(h)\ ](\mathbf{x}, h) \hat{f}\right)(\mathbf{x}')$$



$\hat{f}(\mathbf{x}) = \mathscr{F}_H[f^H(\mathbf{x})]$

Slide credit: Erik Bekkers, University of Amsterdam

Steerable group convolutions



If $\mathscr{K}$ is linear

$$\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}, \mathbf{x}')\hat{f}(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

and equivariant

$$\mathscr{K}[\mathrm{Ind}_H^G[\rho^X](g)\ \hat{f}] = \mathrm{Ind}_H^G[\rho^Y](g)\ \mathscr{K}[\hat{f}]$$

Then it is a normal convolution

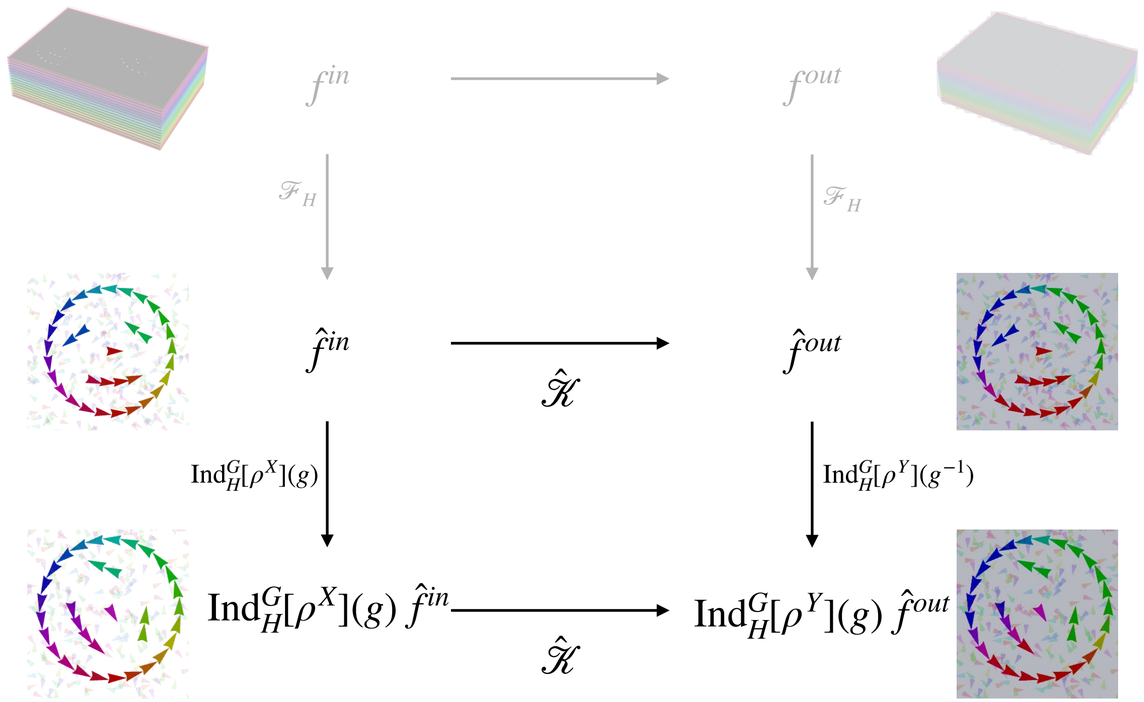
$$\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$$\forall_{h \in H}\ \forall_{\mathbf{x} \in \mathbb{R}^d} : \qquad k(h\,\mathbf{x}) = \rho_Y(h)k(\mathbf{x})\rho_X(h^{-1})$$

Slide credit: Erik Bekkers, University of Amsterdam

Steerable group convolutions



If $\mathscr{K}$ is linear

$$\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}, \mathbf{x}')\hat{f}(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

and equivariant

$$\mathscr{K}[\mathrm{Ind}_H^G[\rho^X](g)\ \hat{f}] = \mathrm{Ind}_H^G[\rho^Y](g)\ \mathscr{K}[\hat{f}]$$

Then it is a normal convolution

$$\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$$\forall_{h \in H}\ \forall_{\mathbf{x} \in \mathbb{R}^d} : \qquad k(h\,\mathbf{x}) = \rho_Y(h)k(\mathbf{x})\rho_X(h^{-1})$$

Problem: The G-steerability constraint! [1,2]

Solution: Expand kernel in steerable basis

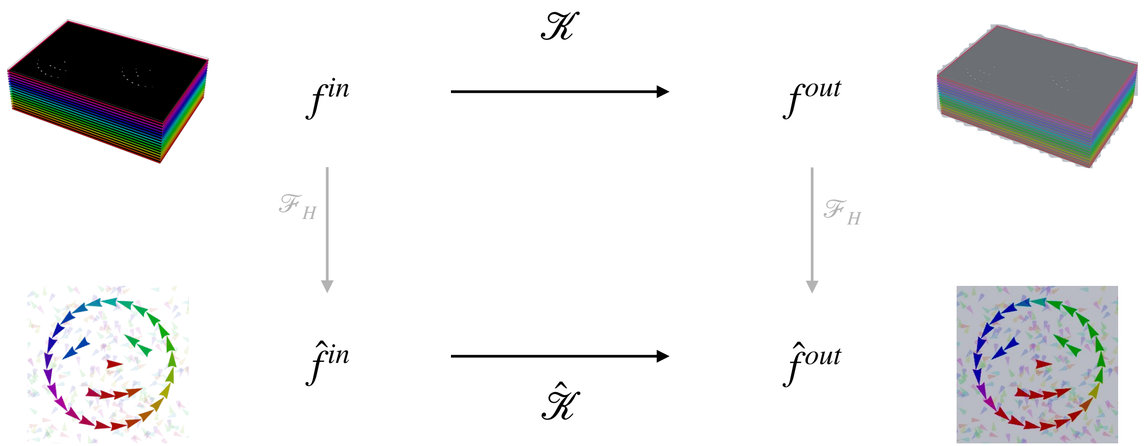
[1] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S. Cohen. 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. In Conference on Neural Information Processing Systems (NeurIPS), 2018a.

[2] Cesa, G., Lang, L., & Weiler, M. (2021, September). A Program to Build E (N)-Equivariant Steerable CNNs. In International Conference on Learning Representations.

Slide credit: Erik Bekkers, University of Amsterdam

Steerable group convolutions

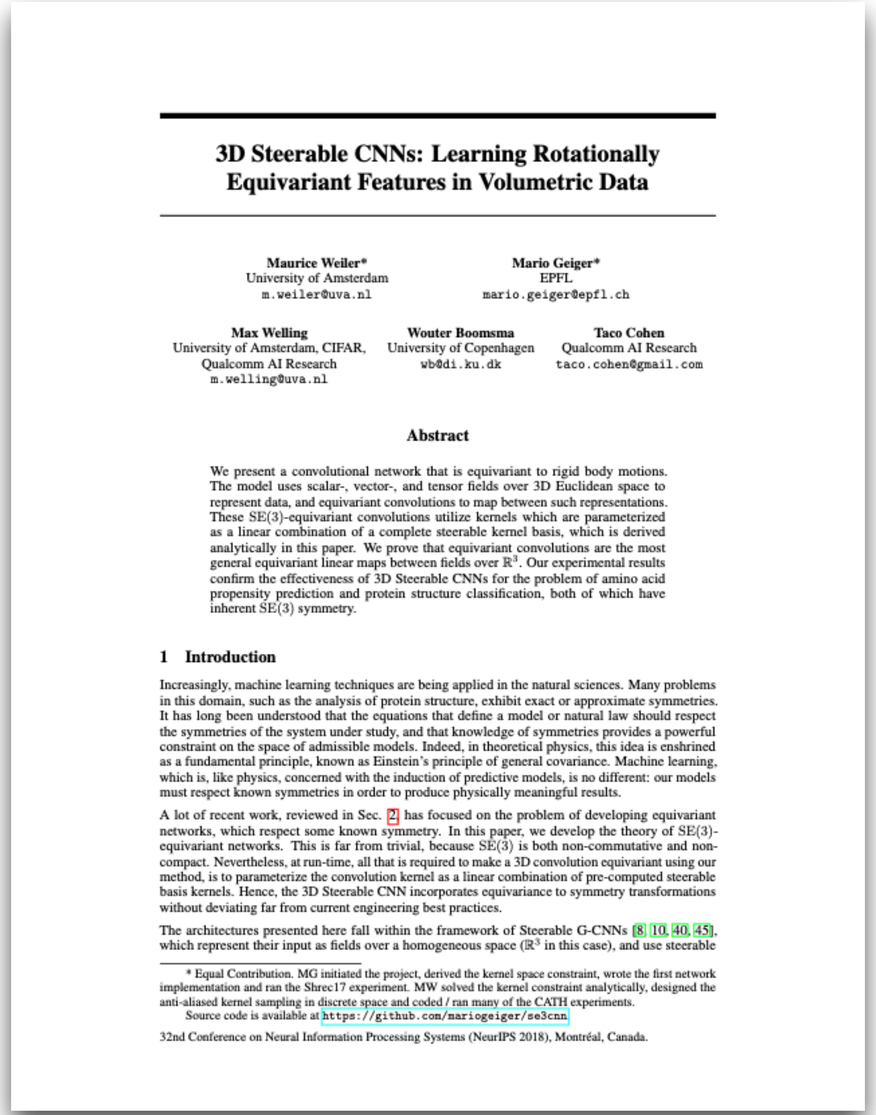
Group convolution $\mathscr{K}[f](g) = \int_G k(g^{-1}g')f(g)\mathrm{d}\mathbf{x}'\mathrm{d}g$



Normal convolution $\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$\forall_{h\in H}\ \forall_{\mathbf{x}\in\mathbb{R}^d}$: $\quad k(g\,\mathbf{x}) = \rho_Y(h)k(\mathbf{x})\rho_X(h^{-1})$



3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data

Maurice Weiler*
University of Amsterdam
m.weiler@uva.nl

Mario Geiger*
EPFL
mario.geiger@epfl.ch

Max Welling
University of Amsterdam, CIFAR,
Qualcomm AI Research
m.welling@uva.nl

Wouter Boomsma
University of Copenhagen
wb@di.ku.dk

Taco Cohen
Qualcomm AI Research
taco.cohen@gmail.com

Abstract

We present a convolutional network that is equivariant to rigid body motions. The model uses scalar-, vector-, and tensor fields over 3D Euclidean space to represent data, and equivariant convolutions to map between such representations. These SE(3)-equivariant convolutions utilize kernels which are parameterized as a linear combination of a complete steerable kernel basis, which is derived analytically in this paper. We prove that equivariant convolutions are the most general equivariant linear maps between fields over $\mathbb{R}^3$. Our experimental results confirm the effectiveness of 3D Steerable CNNs for the problem of amino acid propensity prediction and protein structure classification, both of which have inherent SE(3) symmetry.

1 Introduction

Increasingly, machine learning techniques are being applied in the natural sciences. Many problems in this domain, such as the analysis of protein structure, exhibit exact or approximate symmetries. It has long been understood that the equations that define a model or natural law should respect the symmetries of the system under study, and that knowledge of symmetries provides a powerful constraint on the space of admissible models. Indeed, in theoretical physics, this idea is enshrined as a fundamental principle, known as Einstein's principle of general covariance. Machine learning, which is, like physics, concerned with the induction of predictive models, is no different: our models must respect known symmetries in order to produce physically meaningful results.

A lot of recent work, reviewed in Sec. 2, has focused on the problem of developing equivariant networks, which respect some known symmetry. In this paper, we develop the theory of SE(3)-equivariant networks. This is far from trivial, because SE(3) is both non-commutative and non-compact. Nevertheless, at run-time, all that is required to make a 3D convolution equivariant using our method, is to parameterize the convolution kernel as a linear combination of pre-computed steerable basis kernels. Hence, the 3D Steerable CNN incorporates equivariance to symmetry transformations without deviating far from current engineering best practices.

The architectures presented here fall within the framework of Steerable G-CNNs [8, 10, 40, 45], which represent their input as fields over a homogeneous space ($\mathbb{R}^3$ in this case), and use steerable

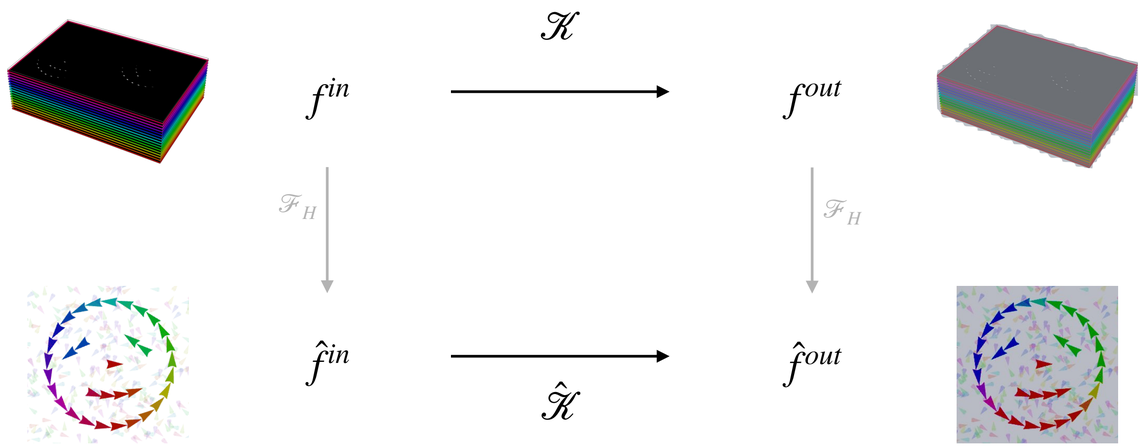
* Equal Contribution. MG initiated the project, derived the kernel space constraint, wrote the first network implementation and ran the Shrec17 experiment. MW solved the kernel constraint analytically, designed the anti-aliased kernel sampling in discrete space and coded / ran many of the CATH experiments.
Source code is available at https://github.com/mariogeiger/se3cnn

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

Slide credit: Erik Bekkers, University of Amsterdam

Steerable group convolutions

Group convolution $\mathscr{K}[f](g) = \int_G k(g^{-1}g')f(g)\mathrm{d}\mathbf{x}'\mathrm{d}g$



Normal convolution $\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$\forall_{h \in H}\ \forall_{\mathbf{x} \in \mathbb{R}^d}$: $k(g\,\mathbf{x}) = \rho_Y(h)k(\mathbf{x})\rho_X(h^{-1})$

General E(2) - Equivariant Steerable CNNs

Maurice Weiler*
University of Amsterdam, QUVA Lab
m.weiler@uva.nl

Gabriele Cesa*†
University of Amsterdam
cesa.gabriele@gmail.com

Abstract

The big empirical success of group equivariant networks has led in recent years to the sprouting of a great variety of equivariant network architectures. A particular focus has thereby been on rotation and reflection equivariant CNNs for planar images. Here we give a general description of E(2)-equivariant convolutions in the framework of *Steerable CNNs*. The theory of Steerable CNNs thereby yields constraints on the convolution kernels which depend on group representations describing the transformation laws of feature spaces. We show that these constraints for arbitrary group representations can be reduced to constraints under irreducible representations. A general solution of the kernel space constraint is given for arbitrary representations of the Euclidean group E(2) and its subgroups. We implement a wide range of previously proposed and entirely new equivariant network architectures and extensively compare their performances. E(2)-steerable convolutions are further shown to yield remarkable gains on CIFAR-10, CIFAR-100 and STL-10 when used as drop in replacement for non-equivariant convolutions.

1 Introduction

The equivariance of neural networks under symmetry group actions has in the recent years proven to be a fruitful prior in network design. By guaranteeing a desired transformation behavior of convolutional features under transformations of the network input, equivariant networks achieve improved generalization capabilities and sample complexities compared to their non-equivariant counterparts. Due to their great practical relevance, a big pool of rotation- and reflection- equivariant models for planar images has been proposed by now. Unfortunately, an empirical survey, reproducing and comparing all these different approaches, is still missing.

An important step in this direction is given by the theory of *Steerable CNNs* [1, 2, 3, 4, 5] which defines a very general notion of equivariant convolutions on homogeneous spaces. In particular, steerable CNNs describe E(2)-equivariant (i.e. rotation- and reflection-equivariant) convolutions on the image plane $\mathbb{R}^2$. The feature spaces of steerable CNNs are thereby defined as spaces of *feature fields*, characterized by a group representation which determines their transformation behavior under transformations of the input. In order to preserve the specified transformation law of feature spaces, the convolutional kernels are subject to a linear constraint, depending on the corresponding group representations. While this constraint has been solved for specific groups and representations [1, 2], no general solution strategy has been proposed so far. In this work we give a general strategy which reduces the solution of the kernel space constraint under arbitrary representations to much simpler constraints under single, *irreducible* representations.

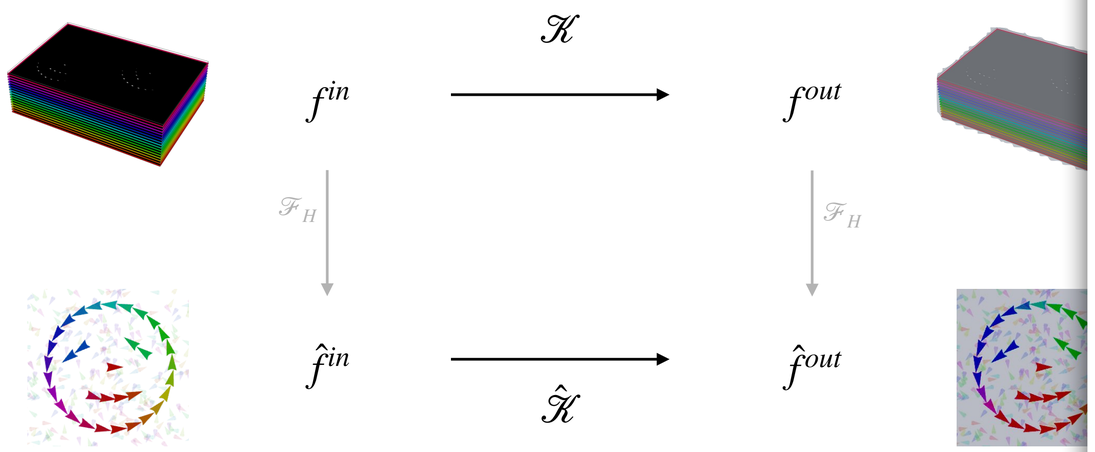
Specifically for the Euclidean group E(2) and its subgroups, we give a general solution of this kernel space constraint. As a result, we are able to implement a wide range of equivariant models, covering regular GCNNs [6, 7, 8, 9, 10, 11], classical Steerable CNNs [1], Harmonic Networks [12], gated Harmonic Networks [2], Vector Field Networks [13], Scattering Transforms [14, 15, 16, 17, 18] and entirely new architectures, in one unified framework. In addition, we are able to build hybrid models, mixing different field types (representations) of these networks both over layers and within layers.

* Equal contribution, author ordering determined by random number generator.
† This research has been conducted during an internship at QUVA lab, University of Amsterdam.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Steerable group convolutions

Group convolution $\mathscr{K}[f](g) = \int_G k(g^{-1}g')f(g)\mathrm{d}\mathbf{x}'\mathrm{d}g$



Normal convolution $\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$\forall_{h\in H}\ \forall_{\mathbf{x}\in\mathbb{R}^d}$: $\quad k(g\,\mathbf{x}) = \rho_Y(h)k(\mathbf{x})\rho_X(h^{-1})$

Published as a conference paper at ICLR 2022

A PROGRAM TO BUILD E(*n*)-EQUIVARIANT STEERABLE CNNS

Gabriele Cesa
Qualcomm AI Research*
University of Amsterdam
gcesa@qti.qualcomm.com

Leon Lang
University of Amsterdam
l.lang@uva.nl

Maurice Weiler
University of Amsterdam
m.weiler.ml@gmail.com

ABSTRACT

Equivariance is becoming an increasingly popular design choice to build data efficient neural networks by exploiting prior knowledge about the symmetries of the problem at hand. Euclidean steerable CNNs are one of the most common classes of equivariant networks. While the constraints these architectures need to satisfy are understood, existing approaches are tailored to specific (classes of) groups. No generally applicable method that is *practical* for implementation has been described so far. In this work, we generalize the Wigner-Eckart theorem proposed in Lang & Weiler (2020), which characterizes general G-steerable kernel spaces for compact groups G over their homogeneous spaces, to arbitrary G-spaces. This enables us to directly parameterize filters in terms of a band-limited basis on the whole space rather than on G's orbits, but also to easily implement steerable CNNs equivariant to a large number of groups. To demonstrate its generality, we instantiate our method on a variety of isometry groups acting on the Euclidean space $\mathbb{R}^3$. Our framework allows us to build E(3) and SE(3)-steerable CNNs like previous works, but also CNNs with arbitrary $G \leq \mathrm{O}(3)$-steerable kernels. For example, we build 3D CNNs equivariant to the symmetries of platonic solids or choose $G = \mathrm{SO}(2)$ when working with 3D data having only azimuthal symmetries. We compare these models on 3D shapes and molecular datasets, observing improved performance by matching the model's symmetries to the ones of the data.

1 INTRODUCTION

In machine learning, it is common for learning tasks to present a number of *symmetries*. A symmetry in the data occurs, for example, when some property (e.g., the label) does not change if a set of transformations is applied to the data itself, e.g. translations or rotations of images. Symmetries are algebraically described by *groups*. If prior knowledge about the symmetries of a task is available, it is usually beneficial to encode them in the models used (Shawe-Taylor, 1989; Cohen & Welling, 2016a). The property of such models is referred to as *equivariance* and is obtained by introducing some *equivariance constraints* in the architecture (see Eq. 2). A classical example are convolutional neural networks (CNNs), which achieve translation equivariance by constraining linear layers to be convolution operators. A wider class of equivariant models are Euclidean steerable CNNs (Cohen & Welling, 2016b; Weiler et al., 2018a; Weiler & Cesa, 2019; Jenner & Weiler, 2022), which guarantee equivariance to isometries $\mathbb{R}^n \rtimes G$ of a Euclidean space $\mathbb{R}^n$, i.e., to translations and a group G of origin-preserving transformations, such as rotations and reflections. As proven in Weiler et al. (2018a; 2021); Jenner & Weiler (2022), this requires convolutions with *G-steerable* (equivariant) kernels.

Our goal is developing a program to parameterize with minimal requirements arbitrary G-steerable kernel spaces, with compact G, which are required to implement $\mathbb{R}^n \rtimes G$ equivariant CNNs. Lang & Weiler (2020) provides a first step in this direction by generalizing the *Wigner-Eckart theorem* from quantum mechanics to obtain a general technique to parametrize G-steerable kernel spaces over *orbits* of a compact G. The theorem reduces the task of building steerable kernel bases to that of finding some pure representation theoretic ingredients. Since the equivariance constraint only relates points $g.x \in \mathbb{R}^n$ in the same *orbit* $G.x \subset \mathbb{R}^n$, a kernel can take independent values on different orbits. Fig. 1 shows

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

https://github.com/QUVA-Lab/escnn

up convolutions

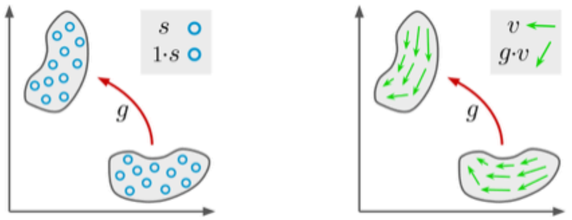
E(n)-Equivariant Steerable CNNs

Documentation | **Paper ICLR 22** | **Paper NeurIPS 19** | **e2cnn library** | **e2cnn experiments** | **Thesis**

escnn is a PyTorch extension for equivariant deep learning. *escnn* is the successor of the e2cnn library, which only supported planar isometries.

Equivariant neural networks guarantee a specified transformation behavior of their feature spaces under transformations of their input. For instance, classical convolutional neural networks (*CNNs*) are by design equivariant to translations of their input. This means that a translation of an image leads to a corresponding translation of the network's feature maps. This package provides implementations of neural network modules which are equivariant under all *isometries* E(2) of the image plane $\mathbb{R}^2$ and all *isometries* E(3) of the 3D space $\mathbb{R}^3$, that is, under *translations*, *rotations* and *reflections* (and can, potentially, be extended to all isometries E(n) of $\mathbb{R}^n$). In contrast to conventional CNNs, E(n)-equivariant models are guaranteed to generalize over such transformations, and are therefore more data efficient.

The feature spaces of E(n)-Equivariant Steerable CNNs are defined as spaces of *feature fields*, being characterized by their transformation law under rotations and reflections. Typical examples are scalar fields (e.g. gray-scale images or temperature fields) or vector fields (e.g. optical flow or electromagnetic fields).



scalar field

vector field

Instead of a number of channels, the user has to specify the field *types* and their *multiplicities* in order to define a feature space. Given a specified input- and output feature space, our `R2conv` and `R3conv` modules instantiate

Published as a conference paper at ICLR 2022

A PROGRAM TO BUILD E(*n*)-EQUIVARIANT STEERABLE CNNS

Gabriele Cesa
Qualcomm AI Research*
University of Amsterdam
gcesa@qti.qualcomm.com

Leon Lang
University of Amsterdam
l.lang@uva.nl

Maurice Weiler
University of Amsterdam
m.weiler.ml@gmail.com

ABSTRACT

Equivariance is becoming an increasingly popular design choice to build data efficient neural networks by exploiting prior knowledge about the symmetries of the problem at hand. Euclidean steerable CNNs are one of the most common classes of equivariant networks. While the constraints these architectures need to satisfy are understood, existing approaches are tailored to specific (classes of) groups. No generally applicable method that is *practical* for implementation has been described so far. In this work, we generalize the Wigner-Eckart theorem proposed in Lang & Weiler (2020), which characterizes general G-steerable kernel spaces for compact groups G over their homogeneous spaces, to arbitrary G-spaces. This enables us to directly parameterize filters in terms of a band-limited basis on the whole space rather than on G's orbits, but also to easily implement steerable CNNs equivariant to a large number of groups. To demonstrate its generality, we instantiate our method on a variety of isometry groups acting on the Euclidean space $\mathbb{R}^3$. Our framework allows us to build E(3) and SE(3)-steerable CNNs like previous works, but also CNNs with arbitrary $G \leq \mathrm{O}(3)$-steerable kernels. For example, we build 3D CNNs equivariant to the symmetries of platonic solids or choose $G = \mathrm{SO}(2)$ when working with 3D data having only azimuthal symmetries. We compare these models on 3D shapes and molecular datasets, observing improved performance by matching the model's symmetries to the ones of the data.

1 INTRODUCTION

In machine learning, it is common for learning tasks to present a number of *symmetries*. A symmetry in the data occurs, for example, when some property (e.g., the label) does not change if a set of transformations is applied to the data itself, e.g. translations or rotations of images. Symmetries are algebraically described by *groups*. If prior knowledge about the symmetries of a task is available, it is usually beneficial to encode them in the models used (Shawe-Taylor, 1989; Cohen & Welling, 2016a). The property of such models is referred to as *equivariance* and is obtained by introducing some *equivariance constraints* in the architecture (see Eq. 2). A classical example are convolutional neural networks (CNNs), which achieve translation equivariance by constraining linear layers to be convolution operators. A wider class of equivariant models are Euclidean steerable CNNs (Cohen & Welling, 2016b; Weiler et al., 2018a; Weiler & Cesa, 2019; Jenner & Weiler, 2022), which guarantee equivariance to isometries $\mathbb{R}^n \rtimes G$ of a Euclidean space $\mathbb{R}^n$, i.e., to translations and a group G of origin-preserving transformations, such as rotations and reflections. As proven in Weiler et al. (2018a; 2021); Jenner & Weiler (2022), this requires convolutions with G-*steerable* (equivariant) kernels.

Our goal is developing a program to parameterize with minimal requirements arbitrary G-steerable kernel spaces, with compact G, which are required to implement $\mathbb{R}^n \rtimes G$ equivariant CNNs. Lang & Weiler (2020) provides a first step in this direction by generalizing the *Wigner-Eckart theorem* from quantum mechanics to obtain a general technique to parametrize G-steerable kernel spaces over *orbits* of a compact G. The theorem reduces the task of building steerable kernel bases to that of finding some pure representation theoretic ingredients. Since the equivariance constraint only relates points $g.x \in \mathbb{R}^n$ in the same *orbit* $G.x \subset \mathbb{R}^n$, a kernel can take independent values on different orbits. Fig. 1 shows

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Normal convolution $\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$\forall_{h \in H} \; \forall_{\mathbf{x} \in \mathbb{R}^d} :$ $\quad k(g\,\mathbf{x}) = \rho_Y(h) k(\mathbf{x}) \rho_X(h^{-1})$

https://github.com/QUVA-Lab/escnn

up convolutions

E(n)-Equivariant Steerable CNNs

Documentation | **Paper ICLR 22** | **Paper NeurIPS 19** | **e2cnn library** | **e2cnn experiments** | **Thesis**

escnn is a PyTorch extension for equivariant deep learning. *escnn* is the successor of the e2cnn library, which only supported planar isometries.

Equivariant neural networks guarantee a specified transformation behavior of their feature spaces under transformations of their input. For instance, classical convolutional neural networks (*CNNs*) are by design equivariant to translations of their input. This means that a translation of an image leads to a corresponding translation of the network's feature maps. This package provides implementations of neural network modules which are equivariant under all *isometries* E(2) of the image plane $\mathbb{R}^2$ and all *isometries* E(3) of the 3D space $\mathbb{R}^3$, that is, under *translations*, *rotations* and *reflections* (and can, potentially, be extended to all isometries E(n) of $\mathbb{R}^n$). In contrast to conventional CNNs, E(n)-equivariant models are guaranteed to generalize over such transforma

The feature
characteriz
gray-scale

Instead of a
feature spa

Getting Started

escnn is easy to use since it provides a high level user interface which abstracts most intricacies of group and representation theory away. The following code snippet shows how to perform an equivariant convolution from an RGB-image to 10 *regular* feature fields (corresponding to a group convolution).

```
from escnn import gspaces                                    #  1
from escnn import nn                                         #  2
import torch                                                 #  3
                                                             #  4
r2_act = gspaces.rot2dOnR2(N=8)                              #  5
feat_type_in  = nn.FieldType(r2_act,  3*[r2_act.trivial_repr])   #  6
feat_type_out = nn.FieldType(r2_act, 10*[r2_act.regular_repr])   #  7
                                                             #  8
conv = nn.R2Conv(feat_type_in, feat_type_out, kernel_size=5)     #  9
relu = nn.ReLU(feat_type_out)                                # 10
                                                             # 11
x = torch.randn(16, 3, 32, 32)                               # 12
x = feat_type_in(x)                                          # 13
                                                             # 14
y = relu(conv(x))                                            # 15
```

Line 5 specifies the symmetry group action on the image plane $\mathbb{R}^2$ under which the network should be equivariant. We choose the *cyclic group* C_8, which describes discrete rotations by multiples of 2π/8. Line 6 specifies the input feature field types. The three color channels of an RGB image are thereby to be identified as three independent scalar fields, which transform under the *trivial representation* of C_8. Similarly, the output feature space is in line 7 specified to consist of 10 feature fields which transform under the *regular representation* of C_8. The C_8-equivariant convolution is then instantiated by passing the input and output type as well as the kernel size to the constructor (line 9). Line 10 instantiates an equivariant ReLU nonlinearity which will operate on the output field and is therefore passed the output field type.

Published as a conference paper at ICLR 2022

A Program to Build E(*n*)-Equivariant Steerable CNNs

Gabriele Cesa
Qualcomm AI Research*
University of Amsterdam
gcesa@qti.qualcomm.com

Leon Lang
University of Amsterdam
l.lang@uva.nl

Maurice Weiler
University of Amsterdam
m.weiler.ml@gmail.com

Abstract

Equivariance is becoming an increasingly popular design choice to build data efficient neural networks by exploiting prior knowledge about the symmetries of the problem at hand. Euclidean steerable CNNs are one of the most common classes of equivariant networks. While the constraints these architectures need to satisfy are understood, existing approaches are tailored to specific (classes of) groups. No generally applicable method that is *practical* for implementation has been described so far. In this work, we generalize the Wigner-Eckart theorem proposed in Lang & Weiler (2020), which characterizes general G-steerable kernel spaces for compact groups G over their homogeneous spaces, to arbitrary G-spaces. This enables us to directly parameterize filters in terms of a band-limited basis on the whole space rather than on G's orbits, but also to easily implement steerable CNNs equivariant to a large number of groups. To demonstrate its generality, we instantiate our method on a variety of isometry groups acting on the Euclidean space $\mathbb{R}^3$. Our framework allows us to build E(3) and SE(3)-steerable CNNs like previous works, but also CNNs with arbitrary $G \leq \mathrm{O}(3)$-steerable kernels. For example, we build 3D CNNs equivariant to the symmetries of platonic solids or choose $G = \mathrm{SO}(2)$ when working with 3D data having only azimuthal symmetries. We compare these models on 3D shapes and molecular datasets, observing improved performance by matching the model's symmetries to the ones of the data.

1 Introduction

In machine learning, it is common for learning tasks to present a number of *symmetries*. A symmetry in the data occurs, for example, when some property (e.g., the label) does not change if a set of transformations is applied to the data itself, e.g. translations or rotations of images. Symmetries are algebraically described by *groups*. If prior knowledge about the symmetries of a task is available, it is usually beneficial to encode them in the models used (Shawe-Taylor, 1989; Cohen & Welling, 2016a). The property of such models is referred to as *equivariance* and is obtained by introducing some *equivariance constraints* in the architecture (see Eq. 2). A classical example are convolutional neural networks (CNNs), which achieve translation equivariance by constraining linear layers to be convolution operators. A wider class of equivariant models are Euclidean steerable CNNs (Cohen & Welling, 2016b; Weiler et al., 2018a; Weiler & Cesa, 2019; Jenner & Weiler, 2022), which guarantee equivariance to isometries $\mathbb{R}^n \rtimes G$ of a Euclidean space $\mathbb{R}^n$, i.e., to translations and a group G of origin-preserving transformations, such as rotations and reflections. As proven in Weiler et al. (2018a; 2021); Jenner & Weiler (2022), this requires convolutions with *G-steerable* (equivariant) kernels.

Our goal is developing a program to parameterize with minimal requirements arbitrary G-steerable kernel spaces, with compact G, which are required to implement $\mathbb{R}^n \rtimes G$ equivariant CNNs. Lang & Weiler (2020) provides a first step in this direction by generalizing the *Wigner-Eckart theorem* from quantum mechanics to obtain a general technique to parametrize G-steerable kernel spaces over *orbits* of a compact G. The theorem reduces the task of building steerable kernel bases to that of finding some pure representation theoretic ingredients. Since the equivariance constraint only relates points $g.x \in \mathbb{R}^n$ in the same *orbit* $G.x \subset \mathbb{R}^n$, a kernel can take independent values on different orbits. Fig. 1 shows

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

1

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Applications in 3D Computer Vision

Vector Neurons: A General Framework for SO(3)-Equivariant Networks

Congyue Deng[1] Or Litany[2] Yueqi Duan[1] Adrien Poulenard[1]
Andrea Tagliasacchi[3,4] Leonidas Guibas[1]
[1]Stanford University [2]NVIDIA [3]Google Research [4]University of Toronto

Abstract

Invariance and equivariance to the rotation group have been widely discussed in the 3D deep learning community for pointclouds. Yet most proposed methods either use complex mathematical tools that may limit their accessibility, or are tied to specific input data types and network architectures. In this paper, we introduce a general framework built on top of what we call Vector Neuron *representations for creating SO(3)-equivariant neural networks for pointcloud processing. Extending neurons from 1D scalars to 3D vectors, our vector neurons enable a simple mapping of SO(3) actions to latent spaces thereby providing a framework for building equivariance in common neural operations – including linear layers, non-linearities, pooling, and normalizations. Due to their simplicity, vector neurons are versatile and, as we demonstrate, can be incorporated into diverse network architecture backbones, allowing them to process geometry inputs in arbitrary poses. Despite its simplicity, our method performs comparably well in accuracy and generalization with other more complex and specialized state-of-the-art methods on classification and segmentation tasks. We also show for the first time a rotation equivariant reconstruction network. Source code is available at* https://github.com/FlyingGiraffe/vnn.

1. Introduction

With the proliferation of lower-cost depth sensors, learning on 3D data has seen rapid progress in recent years. Of particular interest are pointcloud networks, such as PointNet [27] or ACNe [33] that fully respect the inherent set symmetry – that point sets are not ordered – by incorporating order-invariant and/or order-equivariant layers. Yet, there are other important symmetries that have been less perfectly addressed in the context of pointcloud processing, with 3D rotations being a prime example. Consider a scenario where one scans an object using their LIDAR-equipped phone to retrieve similar objects. Clearly, the global object pose should not affect the query result. PointNet uses spatial transformer layers [16], which only attain approximate pose invariance while also requiring extensive augmentation at train time.

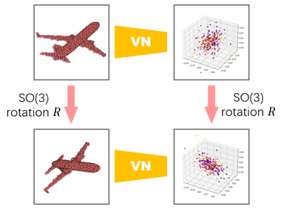


Figure 1: By lifting latent representations from vectors of scalar entries to vectors of 3D points (i.e., matrices) we facilitate the creation of a simple rotation equivariant toolbox allowing the implementation of fully equivariant pointcloud networks.

To avoid an exhaustive data augmentation with all possible rotations, there is a need for network layers that are equivariant to both order and SO(3) symmetries. Recently, two approaches have been introduced to tackle this setting: Tensor Field Networks [35] and SE(3)-Transformers [14]. While guaranteeing equivariance by construction, both frameworks involve an intricate formulation and are hard to incorporate into existing pipelines as they are restricted to convolutions and rely on relative positions of adjacent points.

In this work, we address these issues by proposing a simple, lightweight framework to build SO(3) equivariant and invariant pointcloud networks. A core ingredient in our framework is a *Vector Neuron* (VN) representation, extending classical scalar neurons to 3D vectors. Consequently, instead of latent vector representations which can be views as ordered sequences of scalars, we deploy latent matrix representations which can be viewed as (ordered) sequences of 3-vectors. Such a representation supports a direct map-

Published as a conference paper at ICLR 2023

SE(3)-Equivariant Attention Networks for Shape Reconstruction in Function Space

Evangelos Chatzipantazis*, Stefanos Pertigkiozoglou* **Edgar Dobriban**
University of Pennsylvania University of Pennsylvania
{vaghat,pstefano}@seas.upenn.edu dobriban@wharton.upenn.edu

Kostas Daniilidis
University of Pennsylvania
kostas@cis.upenn.edu

Abstract

We propose a method for 3D shape reconstruction from unoriented point clouds. Our method consists of a novel SE(3)-equivariant coordinate-based network (TFONet), that parametrizes the occupancy field of the shape and respects the inherent symmetries of the problem. In contrast to previous shape reconstruction methods that align the input to a regular grid, we operate directly on the irregular point cloud. Our architecture leverages equivariant attention layers that operate on local tokens. This mechanism enables local shape modelling, a crucial property for scalability to large scenes. Given an unoriented, sparse, noisy point cloud as input, we produce equivariant features for each point. These serve as keys and values for the subsequent equivariant cross-attention blocks that parametrize the occupancy field. By querying an arbitrary point in space, we predict its occupancy score. We show that our method outperforms previous SO(3)-equivariant methods, as well as non-equivariant methods trained on SO(3)-augmented datasets. More importantly, local modelling together with SE(3)-equivariance create an ideal setting for SE(3) scene reconstruction. We show that by training only on single, aligned objects and without any pre-segmentation, we can reconstruct novel scenes containing arbitrarily many objects in random poses without any performance loss.

1 Introduction

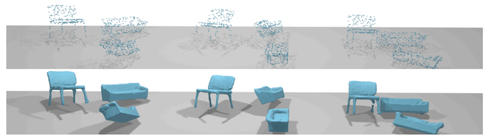


Figure 1: (Above): A scene-level point cloud produced by individual SE(3)-transformations of three sparse object point clouds.(Below): Our equivariant reconstruction. *The whole scene* is given as input to our network. The network, trained only on *single objects in canonical poses* and agnostic to the number, position and orientation of the objects is able to reconstruct the scene accurately.

With the advent of range sensors in robotics and in medical applications, research in shape reconstruction from point clouds has seen an increasing activity (Berger et al., 2017). The performance of classical optimization methods tends to degrade when point clouds become sparser, noisier, unoriented, or untextured. Deep learning methods have been proven useful in encoding shape priors, and solving the reconstruction problem end to end (Riegler et al., 2017). Many of these deep learning methods operate on meshes (Wang & Zhang, 2022; Gong et al., 2019), voxels (Riegler et al.,

*Equal Contribution

Neural Descriptor Fields: SE(3)-Equivariant Object Representations for Manipulation

Anthony Simeonov*,[1], Yilun Du*,[1], Andrea Tagliasacchi[2,3],
Joshua B. Tenenbaum[1], Alberto Rodriguez[1], Pulkit Agrawal†,[1], Vincent Sitzmann†,[1]
[1]Massachusetts Institute of Technology [2]Google Research [3]University of Toronto
*Authors contributed equally, order determined by coin flip. †Equal Advising.

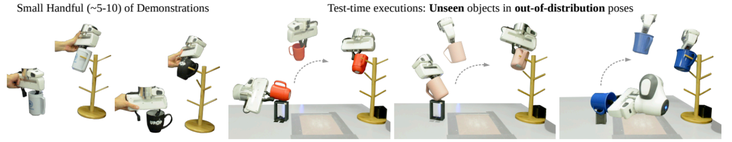


Fig. 1: Given a few (∼5-10) demonstrations of a manipulation task (left), Neural Descriptor Fields (NDFs) generalize the task to novel object instances in any 6-DoF configuration, *including those unobserved at training time*, such as mugs with arbitrary 3D translation and rotation (right). NDFs are continuous functions that map 3D spatial coordinates to spatial descriptors. We generalize this to functions which encode SE(3) poses, such as those used for grasping and placing. NDFs are trained self-supervised for the surrogate task of 3D reconstruction, do not require labeled keypoints, and are SE(3)-equivariant, guaranteeing generalization to unseen object configurations.

***Abstract*— We present Neural Descriptor Fields (NDFs), an object representation that encodes both points and relative poses between an object and a target (such as a robot gripper or a rack used for hanging) via category-level descriptors. We employ this representation for object manipulation, where given a task demonstration, we want to repeat the same task on a *new* object instance from the *same* category. We propose to achieve this objective by searching (via optimization) for the pose whose descriptor matches that observed in the demonstration. NDFs are conveniently trained in a self-supervised fashion via a 3D auto-encoding task that does not rely on expert-labeled keypoints. Further, NDFs are SE(3)-equivariant, guaranteeing performance that generalizes across all possible 3D object translations *and* rotations. We demonstrate learning of manipulation tasks from few (∼5-10) demonstrations both in simulation and on a real robot. Our performance generalizes across both object instances and 6-DoF object poses, and significantly outperforms a recent baseline that relies on 2D descriptors. Project website: https://yilundu.github.io/ndf/**

I. Introduction

Task demonstrations are an intuitive and a powerful mechanism for communicating complex tasks to a robot [1, 30, 35]. However, the ability of current methods to learn from demonstrations is severely limited. Consider the task of teaching the robot to pick up a mug and place it on a rack. After learning, if we want the robot to place a novel instance of a mug from any starting location and orientation, state-of-the-art systems would require a large number of demonstrations spanning the space of different initial positions, orientations and mug instances. This requirement makes it extremely tedious to communicate tasks using demonstrations. Moreover, this approach based on data augmentation comes with no algorithmic guarantees to generalization to out-of-distribution object configurations.

Our goal is to build a robotic system that can learn such pick-and-place tasks for unseen objects in a data-efficient manner. In particular, we desire to construct a system which can manipulate objects from the same category into target configurations, irrespective of the object's 3D location and orientation (see Figure 1) from just a few training demonstrations ($\sim 5-10$).

Consider the task of picking a mug. When task and demonstration objects are identical, the robot can pick up the object by transferring the demonstrated grasp to the new object configuration. For this it suffices to attach a coordinate frame to the demonstration mug, estimate the pose of this frame on the new mug, and move the robot to the relative grasp pose that was recorded in the demonstration with respect to the coordinate frame. Let us now consider mugs that vary in shape and size, wherein grasping requires aligning the gripper to a *local* geometric feature whose location *varies* depending on the shape of the mug. In this case, estimating the coordinate frame on the new mug and moving to the relative grasp pose recorded in the demonstration will fail, *unless* the frame is attached to the specific geometric feature that is used for grasping. However, the choice of which geometric feature to use is *under-specified* unless we consider the task, and different tasks require alignment to *different* features.

For example, to imitate *grasping* along the rim, we may require to define a local frame such that identical gripper poses expressed in this frame all lead to grasping along the rim, irrespective of the height of the mug. On the other hand, imitating a demonstration of object *placing* may require a *new* coordinate frame that can align a placing target (e.g., a shelf)

Applications in “2D Computer Vision”

2D shift-equivariance of CNNs and particular positional embeddings is useful: better generation, better img2img tasks… Nobody uses MLPs for image processing.

Other forms of equivariance have only found applications in niche fields. No mainstream neural net for image processing has any kind of hard-coded equivariance AFAIK.

Applications in “2D Computer Vision”

2D shift-equivariance of CNNs and particular positional embeddings is useful: better generation, better img2img tasks… Nobody uses MLPs for image processing.

Other forms of equivariance have only found applications in niche fields. No mainstream neural net for image processing has any kind of hard-coded equivariance AFAIK.

Why? For simple groups (like rotations etc), we can simply perform data augmentation…

Applications in “2D C

2D shift-equivariance of CNNs and par
useful: better generation, better img2im
image proce

Other forms of equivariance have only f
No mainstream neural net for image p
coded equivarian

Why? For simple groups (like rotation
data augment

DOES EQUIVARIANCE MATTER AT SCALE?

Johann Brehmer* **Sönke Behrends** **Pim de Haan*** **Taco Cohen***
Qualcomm AI Research†
mail@johannbrehmer.de

ABSTRACT

Given large data sets and sufficient compute, is it beneficial to design neural architectures for the structure and symmetries of each problem? Or is it more efficient to learn them from data? We study empirically how equivariant and non-equivariant networks scale with compute and training samples. Focusing on a benchmark problem of rigid-body interactions and on general-purpose transformer architectures, we perform a series of experiments, varying the model size, training steps, and dataset size. We find evidence for three conclusions. First, equivariance improves data efficiency, but training non-equivariant models with data augmentation can close this gap given sufficient epochs. Second, scaling with compute follows a power law, with equivariant models outperforming non-equivariant ones at each tested compute budget. Finally, the optimal allocation of a compute budget onto model size and training duration differs between equivariant and non-equivariant models.

1 INTRODUCTION

In a time of big data and abundant compute, how important are strong inductive biases? Consider problems governed by known symmetries: should one take these into account by designing and using

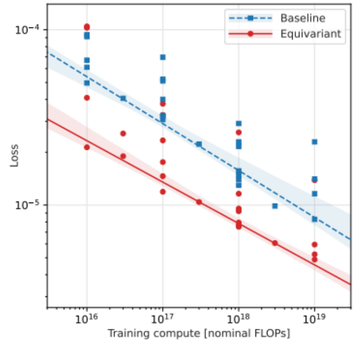


Figure 1: **Scaling with compute**. The dots show the training compute budget and test loss in our experiments, the lines indicate the compute-optimal performance according to the scaling laws we find. The test losses of both non-equivariant (– –) and equivariant (——) transformers scale as a power law with compute, and the equivariant model outperforms the non-equivariant model by a similar factor at all tested compute budgets.

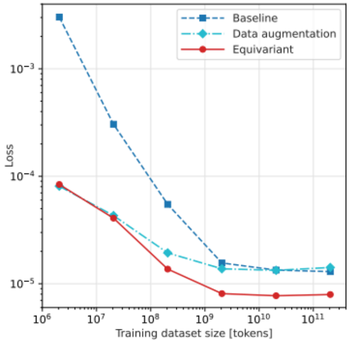


Figure 2: **Scaling with training data**. We show the performance of the non-equivariant transformer (– –), non-equivariant transformer trained with data augmentation (—·—), and equivariant transformer (——) as a function of the number of unique tokens in the training dataset. All experiments use the same training compute budget. Equivariance improves data efficiency compared to the baseline, but data augmentation can close this gap.

*Work was completed while an employee at Qualcomm Technologies Netherlands B.V.
†Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc. and/or its subsidiaries.

What about *3D Geometry in 2D Video?*

SE(3) at work…

What about *3D Geometry in 2D Video?*

SE(3) at work…

What about *3D Geometry in 2D Video?*

The techniques we have discussed do not work for this case.

The problem: 3D symmetry groups *do not act on images*.

Getting neural nets that are provably robust to 3D transforms is an *open problem.*

What about *3D Geometry in 2D Video?*

The techniques we have discussed do not work for this case.

The problem: 3D symmetry groups *do not act on images*.

Getting neural nets that are provably robust to 3D transforms is an *open problem.*

What about *3D Geometry in 2D Video?*

This leads to problems: We need to see any object of interest from all camera poses at training time. Not always possible: think robotics…

Data augmentation is tricky: would need photo-realistic 3D models for that…

We need massive datasets for pretty much anything.

What about *3D Geometry in 2D Video?*

This leads to problems: We need to see any object of interest from all camera poses at training time. Not always possible: think robotics…

Data augmentation is tricky: would need photo-realistic 3D models for that…

We need massive datasets for pretty much anything.

Exciting Research Direction: “Symmetry Discovery”

Neural Fourier Transform: A General Approach to Equivariant Representation Learning

Masanori Koyama[1] **Kenji Fukumizu**[2,1] **Kohei Hayashi**[1] **Takeru Miyato**[3,1]
[1]Preferred Networks, Inc. [2]The Institute of Statistical Mathematics [3]University of Tübingen

Abstract

Symmetry learning has proven to be an effective approach for extracting the hidden structure of data, with the concept of equivariance relation playing the central role. However, most of the current studies are built on architectural theory and corresponding assumptions on the form of data. We propose Neural Fourier Transform (NFT), a general framework of learning the latent linear action of the group without assuming explicit knowledge of how the group acts on data. We present the theoretical foundations of NFT and show that the existence of a linear equivariant feature, which has been assumed ubiquitously in equivariance learning, is equivalent to the existence of a group invariant kernel on the dataspace. We also provide experimental results to demonstrate the application of NFT in typical scenarios with varying levels of knowledge about the acting group.

1 Introduction

Various types of data admit symmetric structure explicitly or implicitly, and such symmetry is often formalized with *action* of a *group*. As a typical example, an RGB image can be regarded as a function defined on the set of 2D coordinates $\mathbb{R}^2 \to \mathbb{R}^3$, and this image admits the standard shift and rotation on the coordinates. Data of 3D object/scene accepts SO(3) action [5, 46], and molecular data accepts permutations [36] as well. To leverage the symmetrical structure for various tasks, equivariant features are used in many applications in hope that such features extract essential information of data.

Fourier transform (FT) is one of the most classic tools in science that utilizes an equivariance relation to investigate the symmetry in data. Originally, FT was developed to study the symmetry induced by the shift action $a \circ f(\cdot) := f(\cdot - a)$ on a square-integrable function $f \in L_2(\mathbb{R})$. FT maps f invertibly to another function $\Phi(f) = \hat{f} \in L_2(\mathbb{R})$. It is well known that FT satisfies $(a \circ \hat{f})(\omega) = e^{-ia\omega}\hat{f}(\omega)$ and hence the equivariant relation $\Phi(a \circ f) = e^{ia\omega}\Phi(f)$. By this equivariant mapping, FT achieves the decomposition of $L_2(\mathbb{R})$ into shift-equivariant subspaces (also called *frequencies/irreducible representations*). This idea has been extended to the actions of general groups, and extensively studied as *harmonic analysis on groups* [38]. In recent studies of deep learning, group convolution (GC) is a popular approach to equivariance [12, 19, 9, 10, 41], and the theory of FT also provides its mathematical foundation [27, 13, 11].

One practical limitation of FT and GC is that they can be applied only when we know how the group acts on the data. Moreover, FT and GC also assume that the group acts linearly on the constituent unit of input (e.g. pixel). In many cases, however, the group action on the dataset may not be linear or explicit. For example, when the observation process involves an unknown nonlinear deformation such as fisheye transformation, the effect of the action on the observation is also intractable and nonlinear (Fig. 1, left). Another such instance may occur for the 2D pictures of a rotating 3D object rendered with some camera pose (Fig. 1, right). In both examples, any two consecutive frames are implicitly generated as $(x, g \circ x)$, where g is a group action of *shift/rotation*. They are clearly not linear transformations in the 2D pixel space. To learn the hidden equivariance relation describing the

Preprint. Under review.

Neural Isometries: Taming Transformations for Equivariant ML

Thomas W. Mitchel
PlayStation
tommy.mitchel@sony.com

Michael Taylor
PlayStation
mike.taylor@sony.com

Vincent Sitzmann
MIT
sitzmann@mit.edu

Abstract

Real-world geometry and 3D vision tasks are replete with challenging symmetries that defy tractable analytical expression. In this paper, we introduce Neural Isometries, an autoencoder framework which learns to map the observation space to a general-purpose latent space wherein encodings are related by *isometries* whenever their corresponding observations are geometrically related in world space. Specifically, we regularize the latent space such that maps between encodings preserve a learned inner product and commute with a learned functional operator, in the same manner as rigid-body transformations commute with the Laplacian. This approach forms an effective backbone for self-supervised representation learning, and we demonstrate that a simple off-the-shelf equivariant network operating in the pre-trained latent space can achieve results on par with meticulously-engineered, handcrafted networks designed to handle complex, nonlinear symmetries. Furthermore, isometric maps capture information about the respective transformations in world space, and we show that this allows us to regress camera poses directly from the coefficients of the maps between encodings of adjacent views of a scene.

1 Introduction

We constantly capture lossy observations of our world – images, for instance, are 2D projections of the 3D world. Observations captured consecutively in time are often related by transformations which are easily described in world space, but are intractable in the space of observations. For instance, video frames captured by a camera moving through a static scene are fully described by a combination of the 3D scene geometry and SE(3) camera poses. In contrast, the image space transformations between these frames can only be characterized by optical flow, a high-dimensional vector field that does not itself have any easily tractable low-dimensional representation.

Geometric deep learning seeks to build neural network architectures that are provably robust to transformations acting on their inputs, such as rotations [1–3], dilations [4, 5], and projective transformations [6, 7]. However, such approaches are only tractable for transformations that have group structure, and, even in those cases, still require meticulously handcrafted and complex architectures. Yet, many real-world transformations of interest, for instance in vision and geometry processing, altogether lack identifiable group structure, such as the effect of camera motion in image space—see Fig. 1 to the right. Even when group-structured, they are often non-linear and non-compact, such as is the case of image

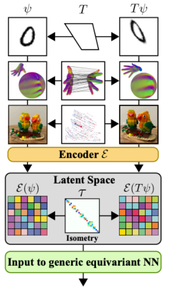
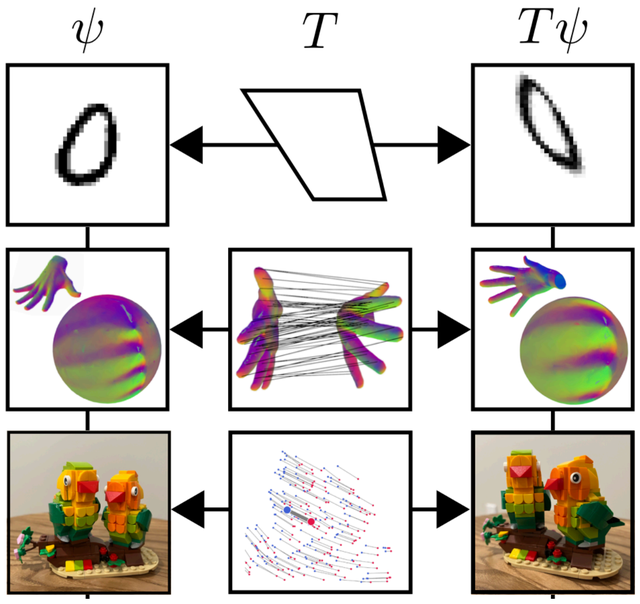


Figure 1: Neural Isometries find latent spaces where complex transformations become tractable.

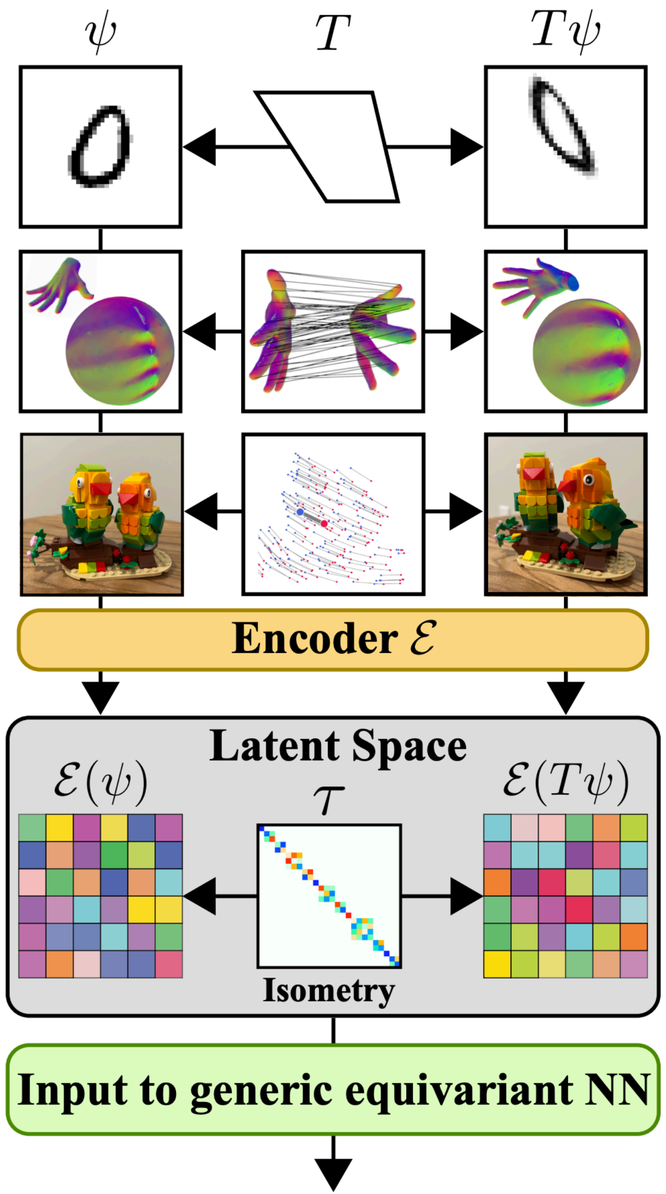
38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Neural Isometries



Real-World Transformations are difficult in observation space…

Neural Isometries

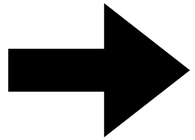


Real-World Transformations are difficult in observation space…

…but there must exist a latent space where they are simple

Toy example: 2D Shifts!

ψ



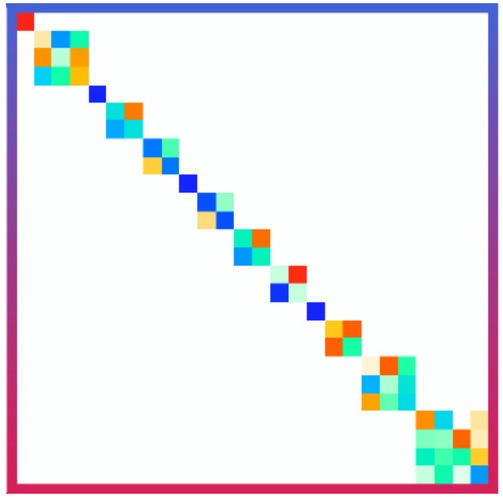
$T\psi$

How to parameterize the space of
all maps between these images?

Toy example: 2D Shifts!

ψ

Attention Map /
“Functional Map”



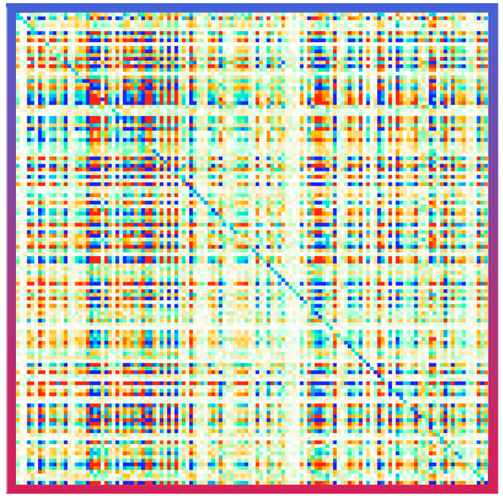
$T\psi$

Express every pixel as weighted sum of pixels in other image.

Toy example: 2D Shifts!

ψ

Attention Map /
“Functional Map”



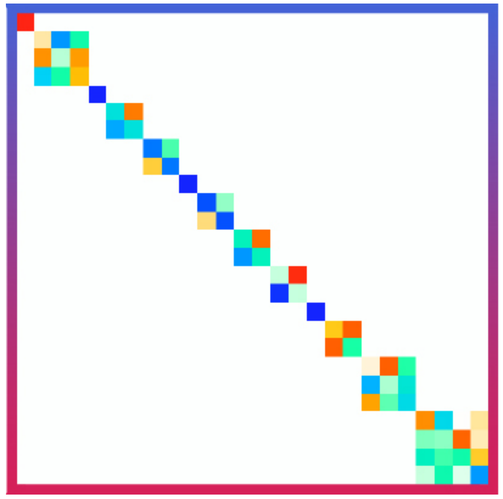
$T\psi$

Express every pixel as weighted sum of pixels in other image.

Toy example: 2D Shifts!

ψ

Functional Map



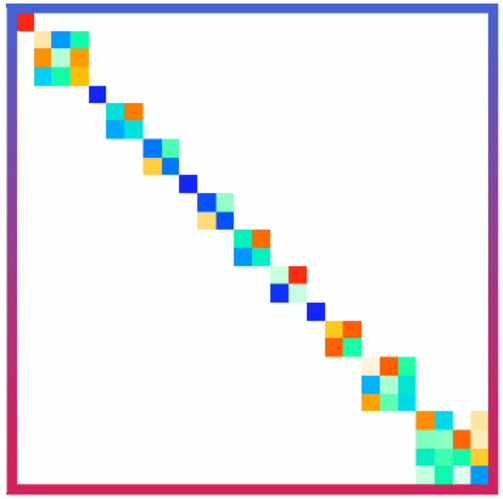
$T\psi$

How to find the *correct* map?

Toy example: 2D Shifts!

ψ

Functional Map



$T\psi$

How to find the *correct* map?

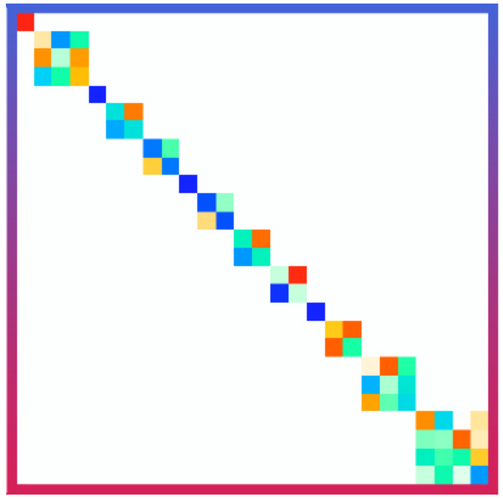
Assumption 1: Our transformations form a *group*.

Assumption 2: Our group can be well-approximated by a *compact group.*

Toy example: 2D Shifts!

ψ

Functional Map



$T\psi$

How to find the *correct* map?

Assumption 1: Our transformations form a *group*.

Assumption 2: Our group can be well-approximated by a *compact group.*

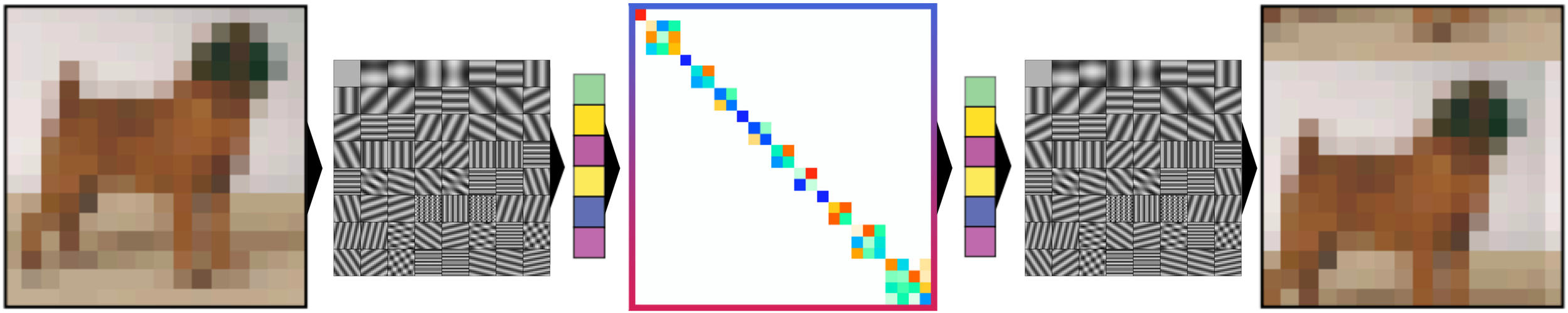
Peter-Weyl Theorem: There exists a basis for images in which our transformations will be block-diagonal!

Learn Function Basis and Solve For Transformation in that Basis

ψ

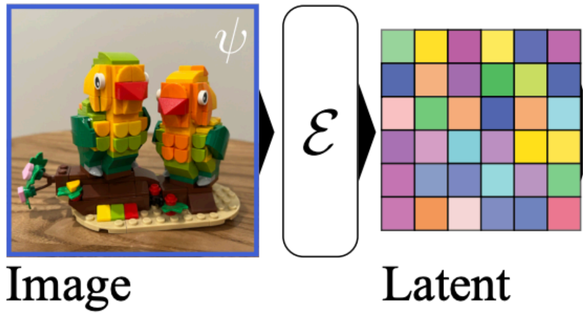
Functional Map

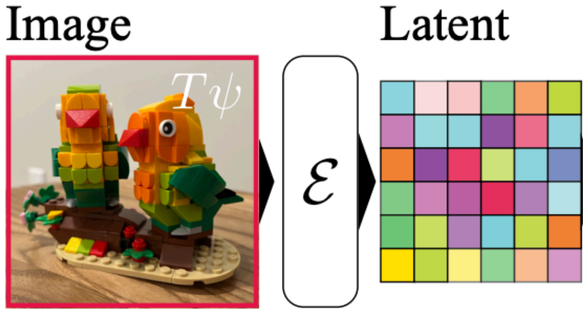
$T\psi$



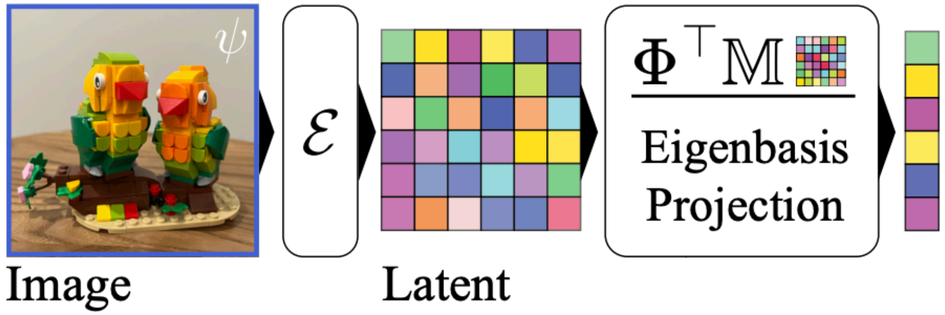
For shifts, you all know what function basis it is: The Fourier basis!

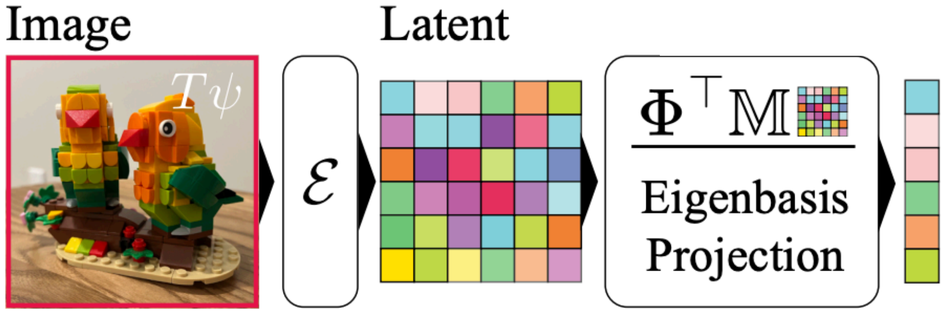
Neural Isometries



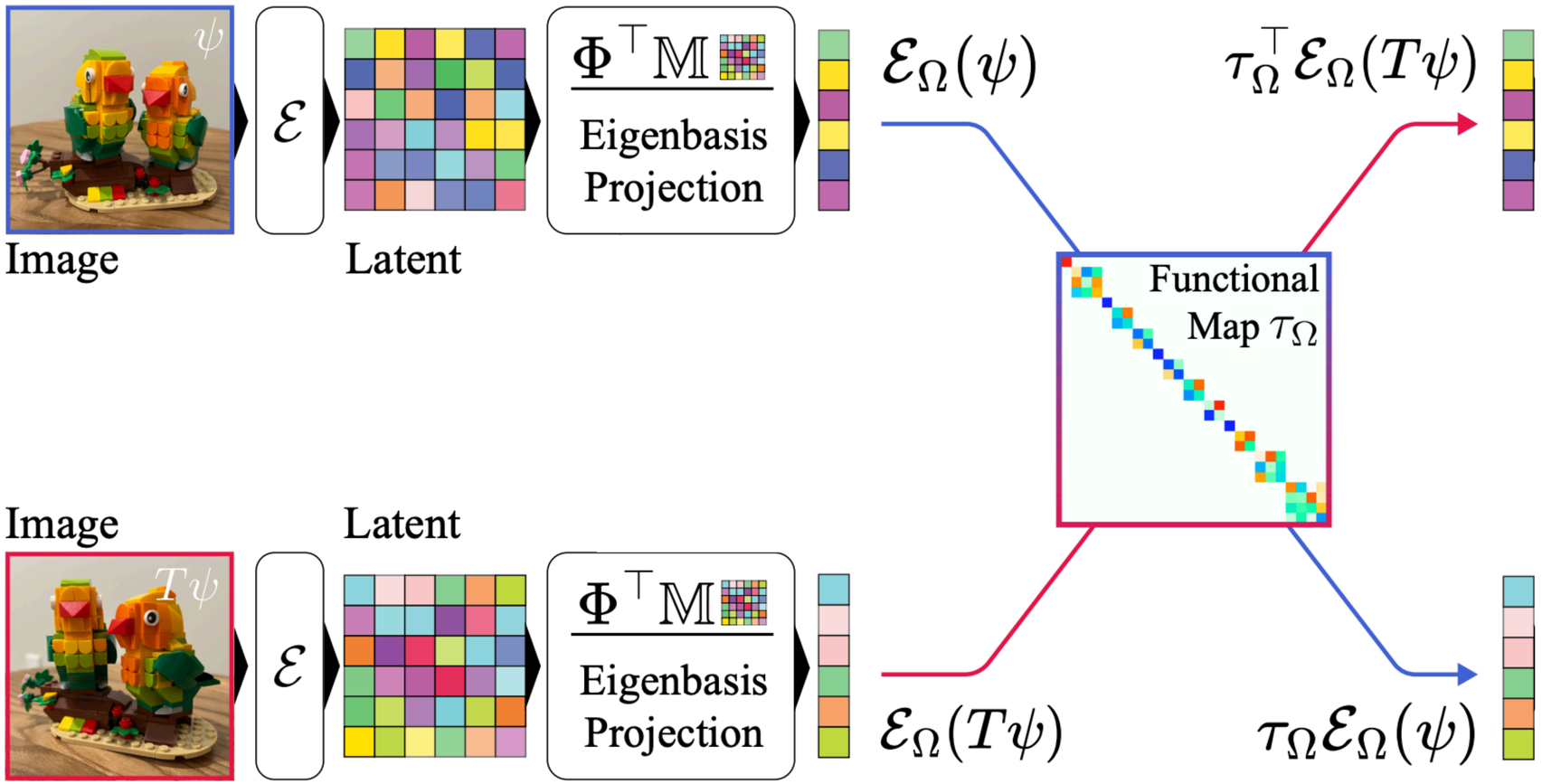


Neural Isometries

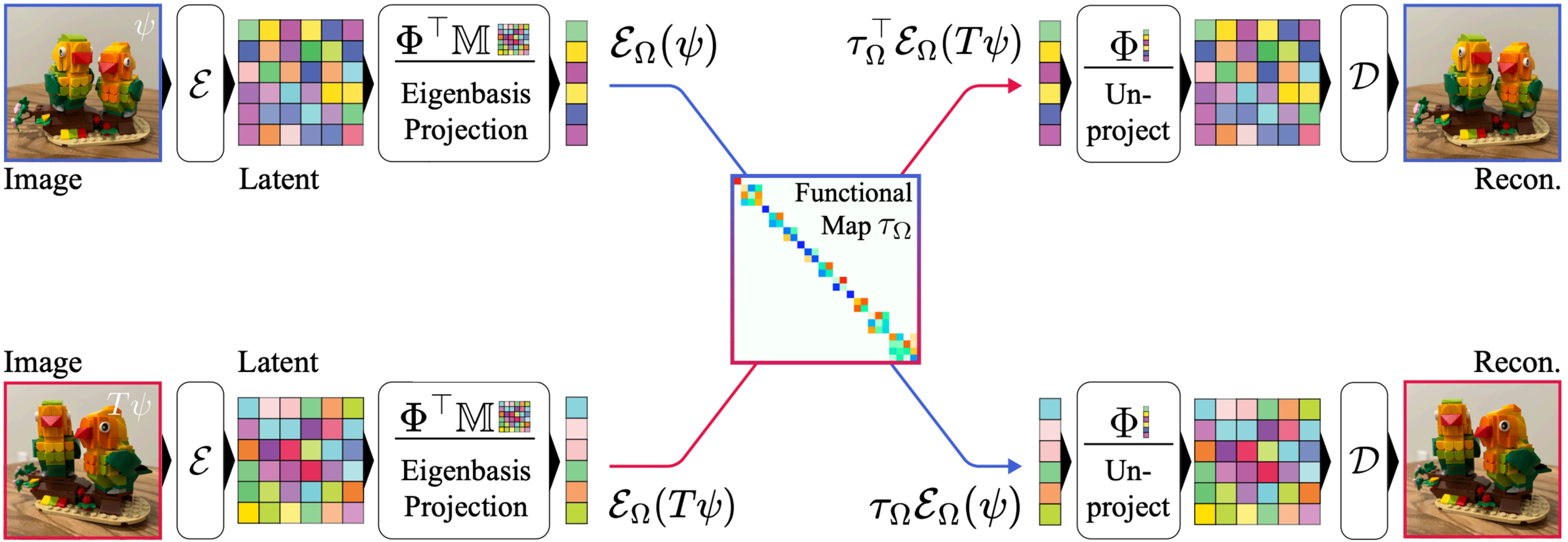




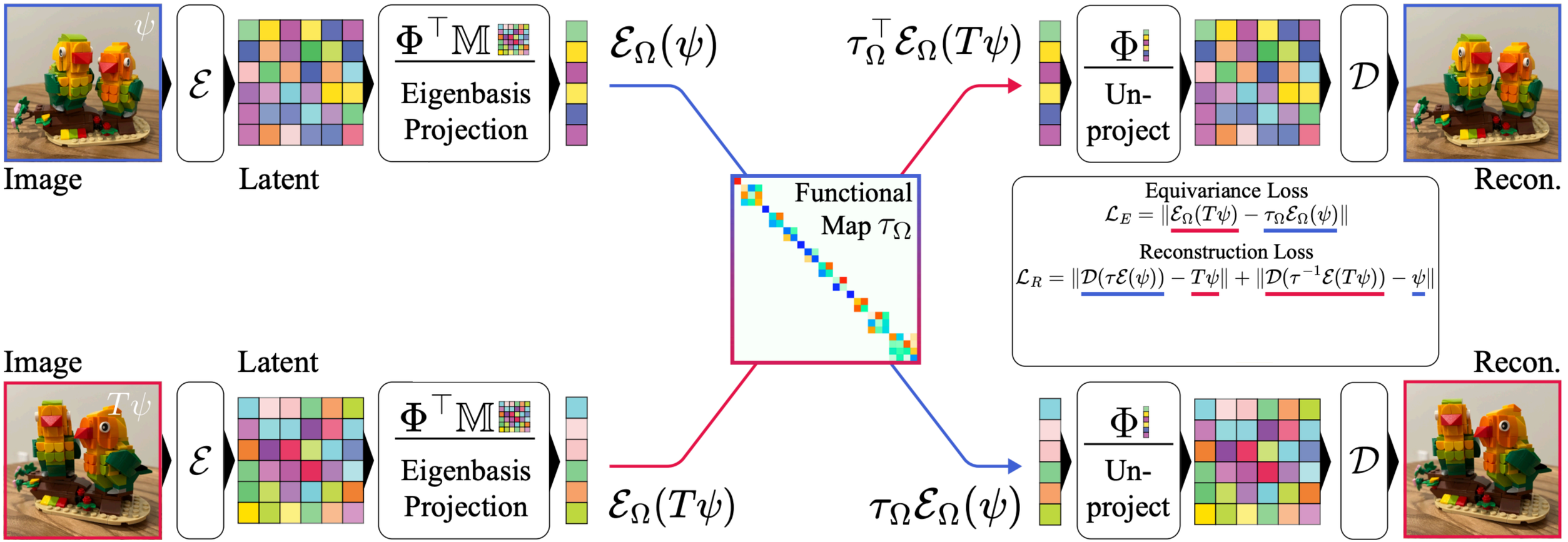
Neural Isometries



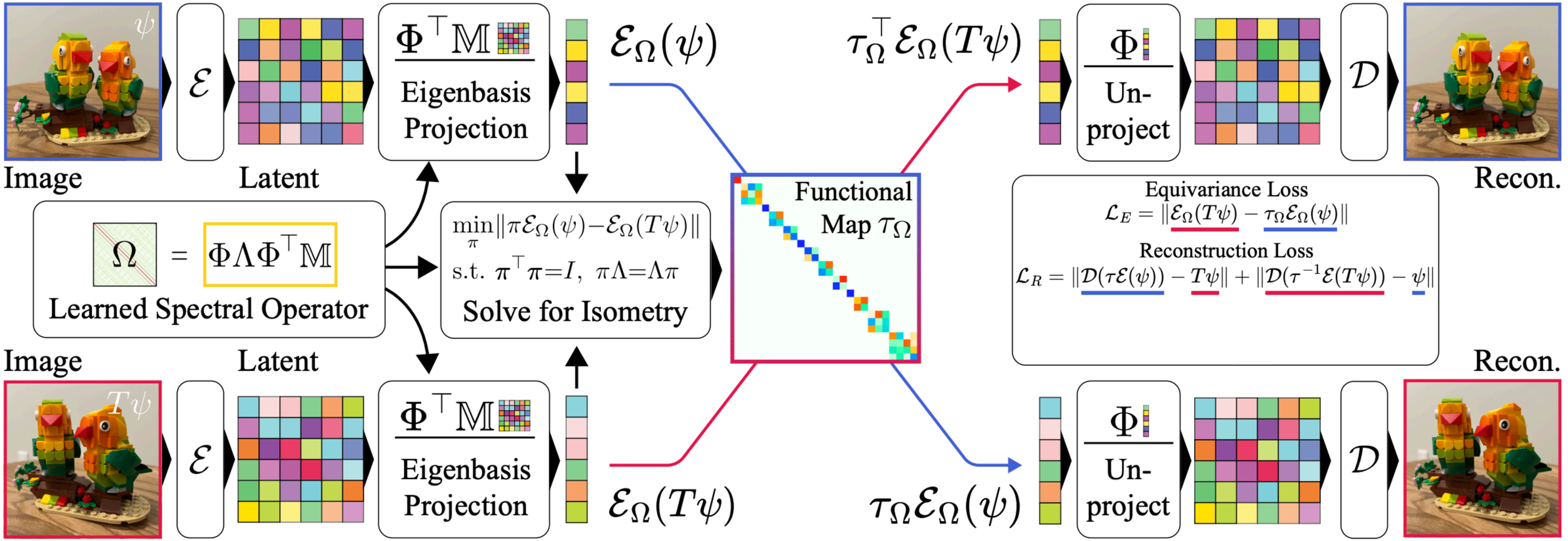
Neural Isometries



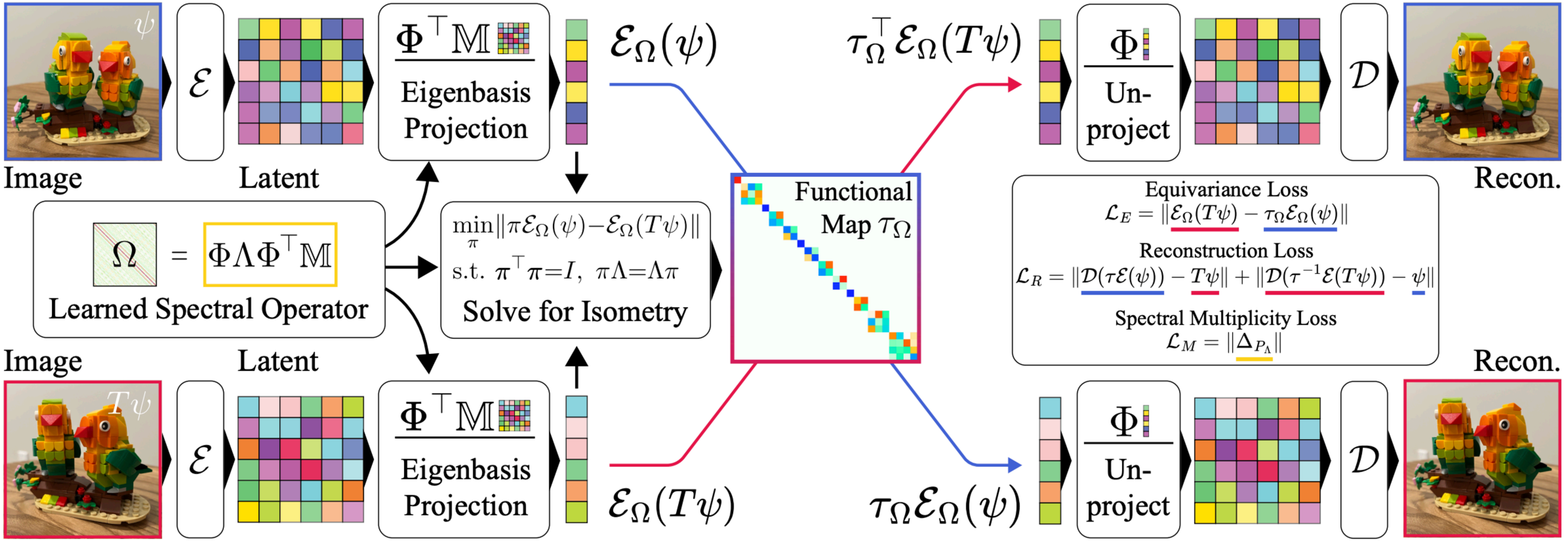
Neural Isometries



Neural Isometries



Neural Isometries



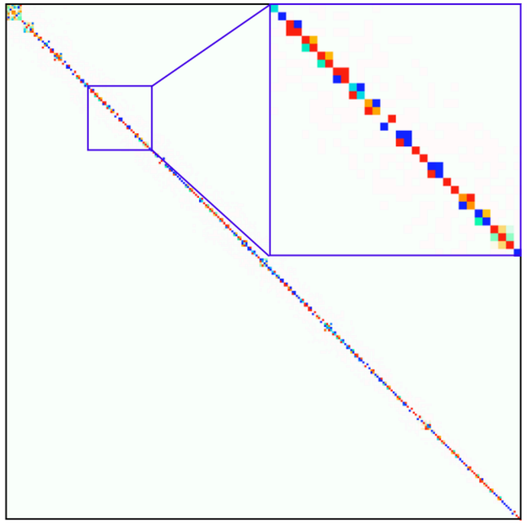
Toy example: Discovering the Fourier Transform

Data

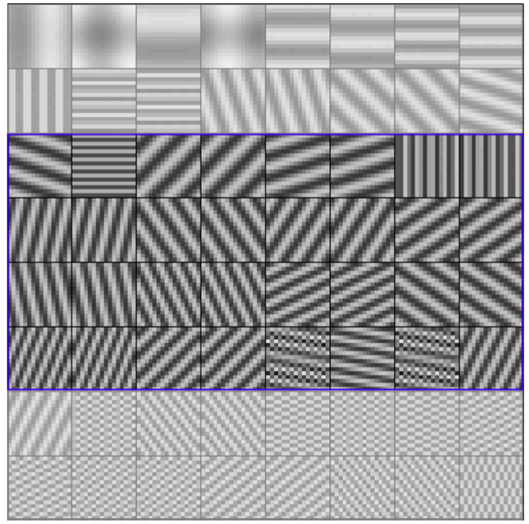
ψ $T\psi$

NIso Outputs

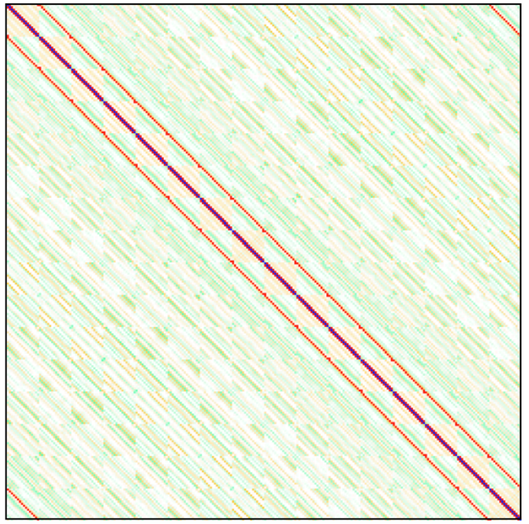
Latent Map τ_Ω



Eigenfunctions Φ

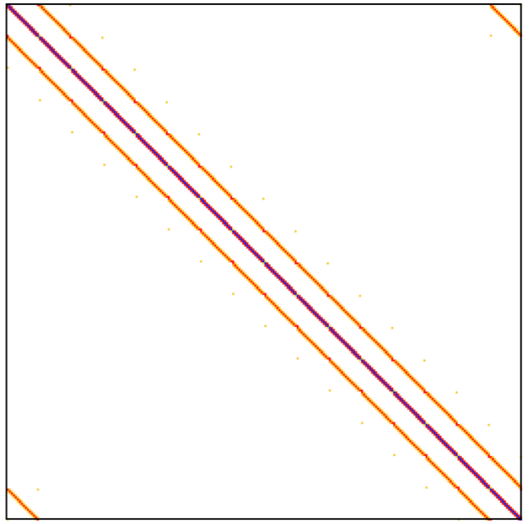


Operator Ω

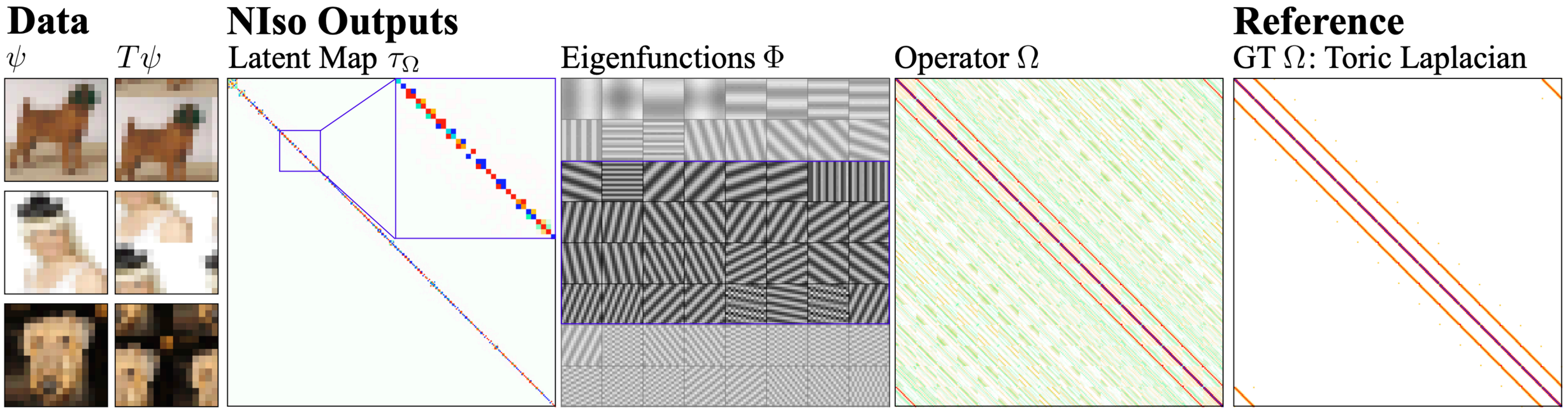


Reference

GT Ω: Toric Laplacian



Toy example: Discovering the Fourier Transform

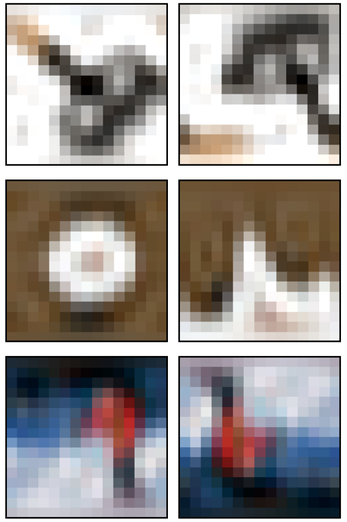


We jointly learned shift-equivariant basis functions (sines and cosines) and the shift operator purely by saying “find me a basis such that the transform is block-diagonal”

Toy example 2: Discovering the Spherical Harmonics

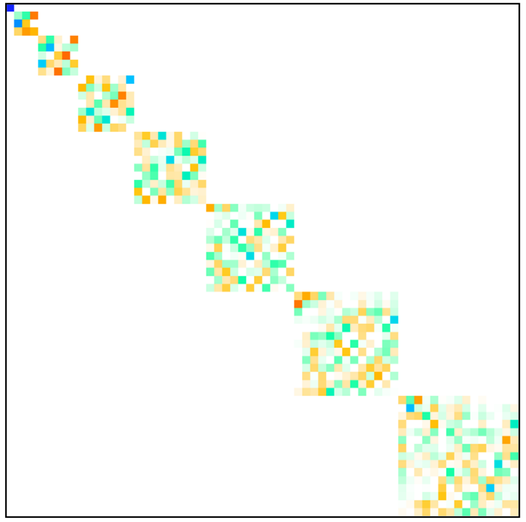
Data

ψ $T\psi$

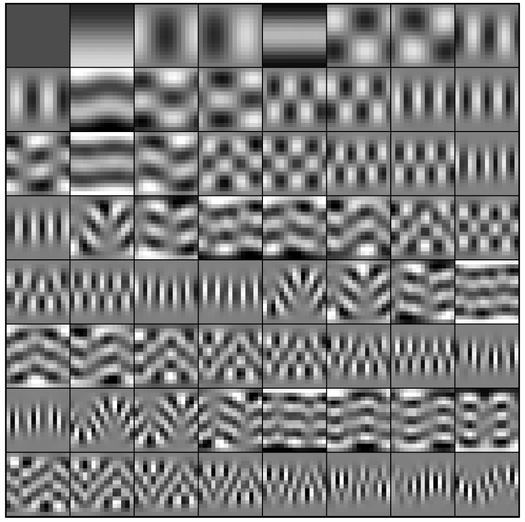


NIso Outputs

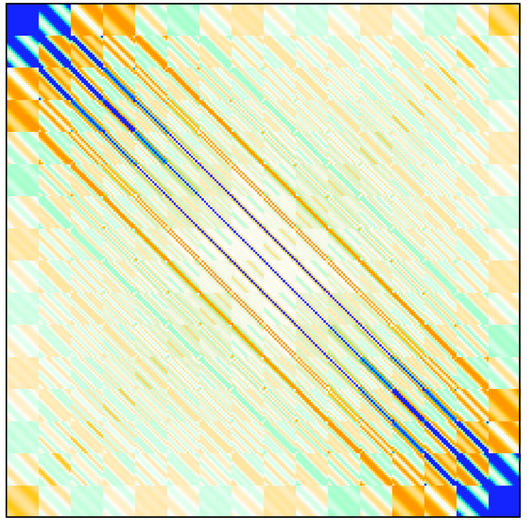
Latent Map τ_Ω



Learned Eigenfunctions Φ

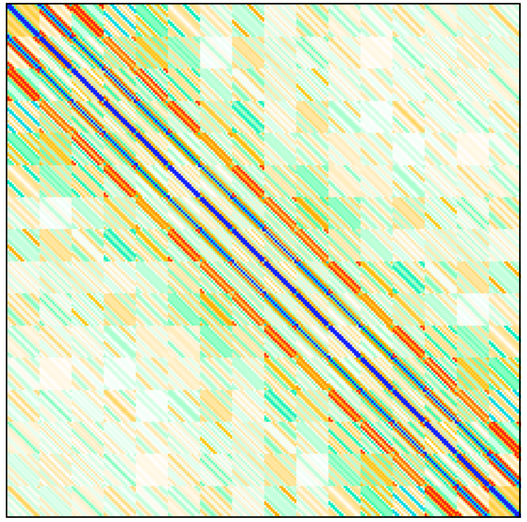


Operator Ω

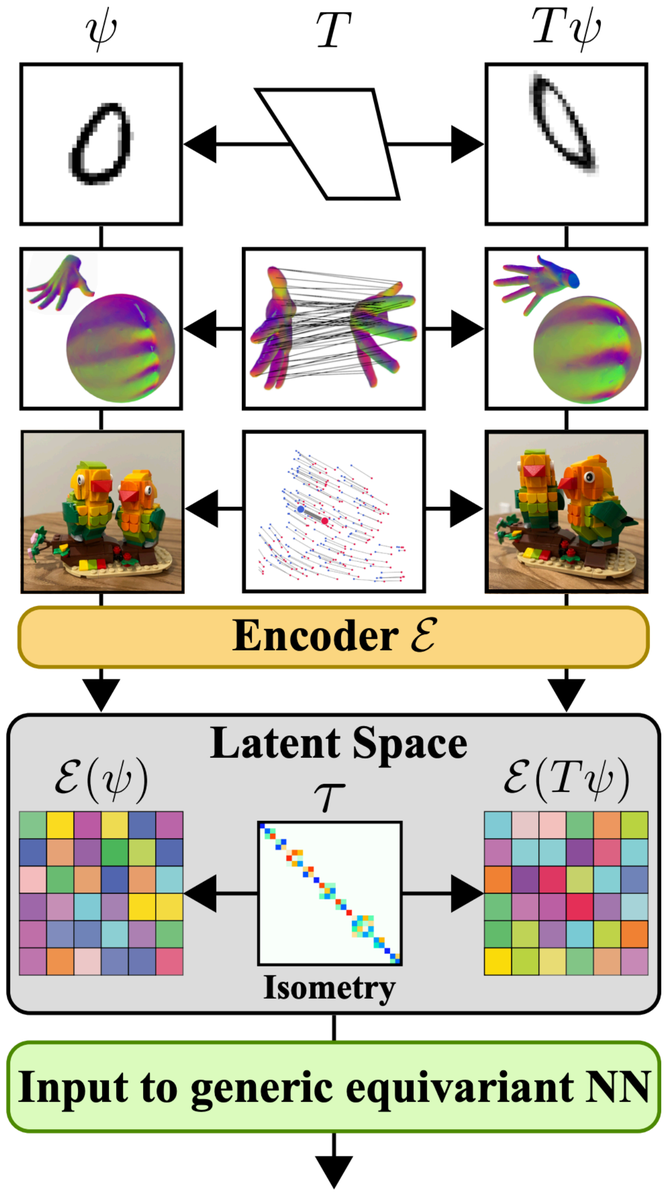


Reference

GTΩ : Spherical Laplacian



Latent Space is Equivariant to Learned Transformation!

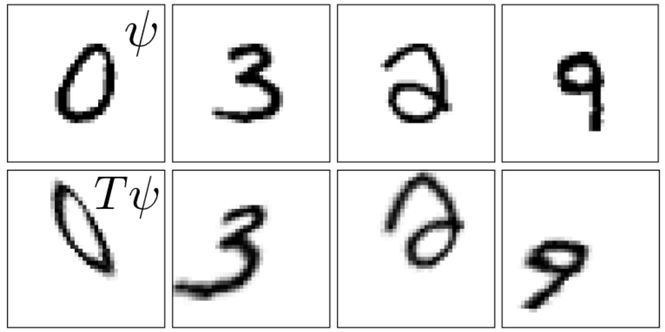


We use a modified version of Vector Neurons [1], which are SO(3) equivariant, but can be modified for our purposes!

[1] C. Deng, et. Al, “Vector Neurons: A General Framework for SO(3)-Equivariant Neural Networks”, ICCV 2021

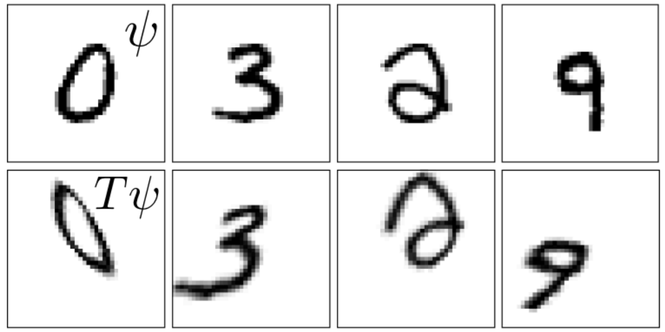
Performs on Par With Hand-Crafted Equivariant Architectures!

Homographic MNIST



Performs on Par With Hand-Crafted Equivariant Architectures!

Homographic MNIST

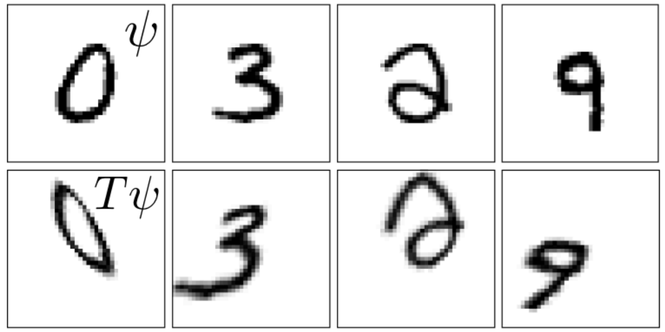


	Acc.
NIso	92.52 ($\pm$ 0.91)
w/ triplet	97.38 ($\pm$ 0.23)
w/o $\mathcal{L}_E$	77.30 ($\pm$ 2.56)
w/o $\mathcal{L}_M$	45.27 ($\pm$ 1.20)
NFT [35]	41.93 ($\pm$ 0.84)
w/ triplet	67.15 ($\pm$ 1.10)
AE Baseline	46.64 ($\pm$ 0.41)
homConv [6]	95.71 ($\pm$ 0.09)
LieDecomp [43]	**98.30 ($\pm$ 0.10)**

Table 1: **Hom. MNIST.**

Performs on Par With Hand-Crafted Equivariant Architectures!

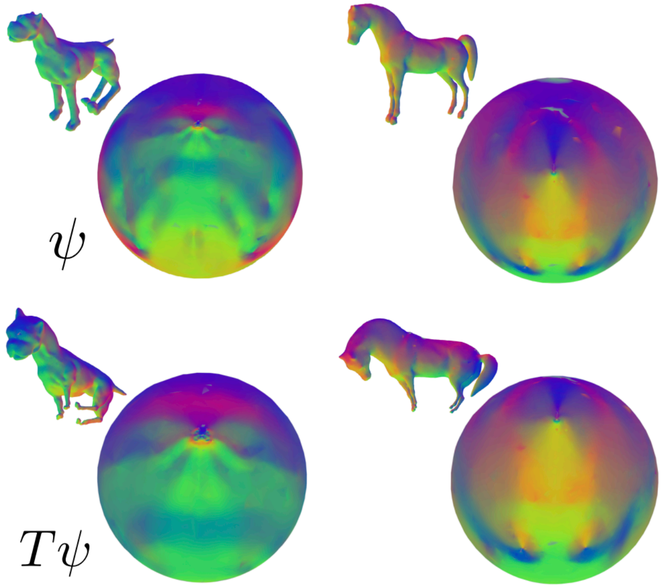
Homographic MNIST



	Acc.
NIso	92.52 ($\pm$ 0.91)
w/ triplet	97.38 ($\pm$ 0.23)
w/o $\mathcal{L}_E$	77.30 ($\pm$ 2.56)
w/o $\mathcal{L}_M$	45.27 ($\pm$ 1.20)
NFT [35]	41.93 ($\pm$ 0.84)
w/ triplet	67.15 ($\pm$ 1.10)
AE Baseline	46.64 ($\pm$ 0.41)
homConv [6]	95.71 ($\pm$ 0.09)
LieDecomp [43]	**98.30 ($\pm$ 0.10)**

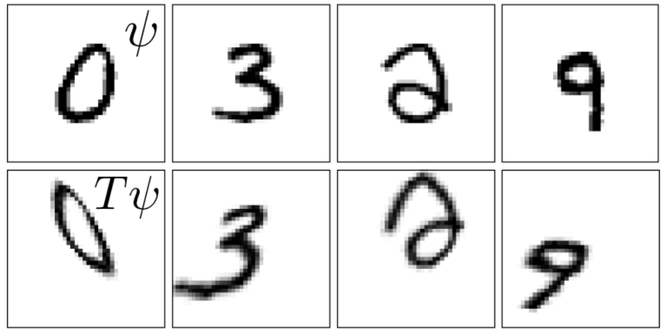
Table 1: **Hom. MNIST.**

Conformal Shape Classification



Performs on Par With Hand-Crafted Equivariant Architectures!

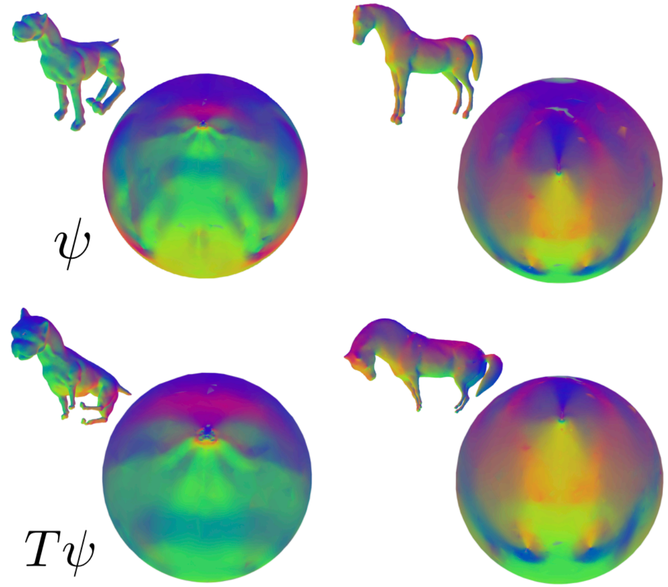
Homographic MNIST



	Acc.
NIso	92.52 ($\pm$ 0.91)
w/ triplet	97.38 ($\pm$ 0.23)
w/o $\mathcal{L}_E$	77.30 ($\pm$ 2.56)
w/o $\mathcal{L}_M$	45.27 ($\pm$ 1.20)
NFT [35]	41.93 ($\pm$ 0.84)
w/ triplet	67.15 ($\pm$ 1.10)
AE Baseline	46.64 ($\pm$ 0.41)
homConv [6]	95.71 ($\pm$ 0.09)
LieDecomp [43]	**98.30 ($\pm$ 0.10)**

Table 1: **Hom. MNIST.**

Conformal Shape Classification



	Acc.
NIso	**90.26 ($\pm$ 1.27)**
NFT [35]	83.24 ($\pm$ 2.03)
AE Baseline	63.76 ($\pm$ 2.47)
MC [7]	86.5

Table 2: **Conf. SHREC '11.**

Performs on Par With Hand-Crafted Equivariant Architectures!

Homographic MNIST

Conformal Shape Classification

Still only works on relatively simple image transformations (homographies etc).
Still far from solving the general 3D problem.

NFT [35]	41.93 ($\pm$ 0.84)
w/ triplet	67.15 ($\pm$ 1.10)
AE Baseline	46.64 ($\pm$ 0.41)
homConv [6]	95.71 ($\pm$ 0.09)
LieDecomp [43]	**98.30 ($\pm$ 0.10)**

Table 1: **Hom. MNIST.**

NIso	**90.26 ($\pm$ 1.27)**
NFT [35]	83.24 ($\pm$ 2.03)
AE Baseline	63.76 ($\pm$ 2.47)
MC [7]	86.5

Table 2: **Conf. SHREC '11.**

Summary

- Have learned a framework for understanding how to build neural networks that are provably robust to geometric transformations
- Discussed two alternatives: in the “pixel domain” (regular group convolutions) and in the “fourier domain” (steerable convolutions)
- Discussed challenges: in computer vision, we are regularly interested in **3D transformations manifesting in 2D images.** That doesn’t fit into existing frameworks of equivariant machine learning.
- Discussed a (budding) new research direction: “Symmetry Discovery”: Can we learn latent spaces that are approximately equivariant to complicated, unknown group transformations?