INVERSE GRAPHICS

MODULE 4:

Geometric Deep Learning

Project Proposals

Project Midterm Details

- **5 min per team:**
  - **Problem definition:** What are you trying to do?
  - How is it done today, and what are the limits of current practice?
  - What is new in your approach and why do you think it will be successful?
  - Who cares? If you are successful, what difference will it make?
  - Description of the first, simplest experiment that will show you if your idea has merit
  - Description of final experiment - best case, what will you be able to show?

Project Midterm Details

- **5 mi**
  - **Pr**
  - Ho ?
  - Wh
    suc
  - Wh
  - De f
    you
  - Description of final experiment - best case, what will you be able to show?

Don’t stress - we’re just there to give you feedback and help you along!

Assignment 3


3D Scene
Image Formation
Inference
Neural Scene Representation
Downstream Tasks
Graphics
“inverse graphics”...?

Image Formation and Multi-View Geometry

3D Scene

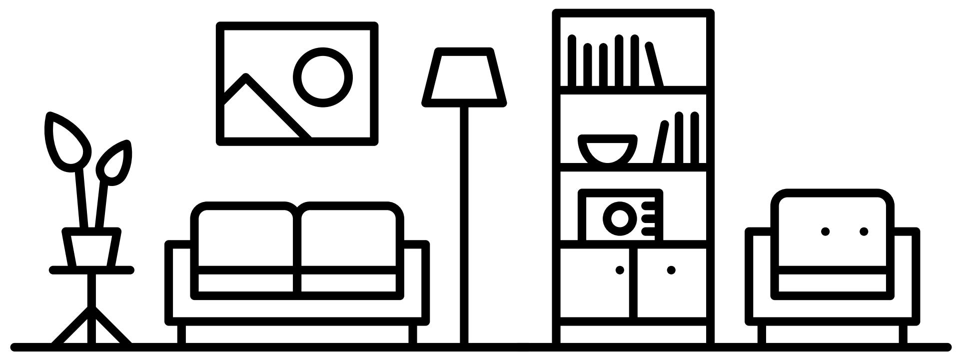


Image Formation

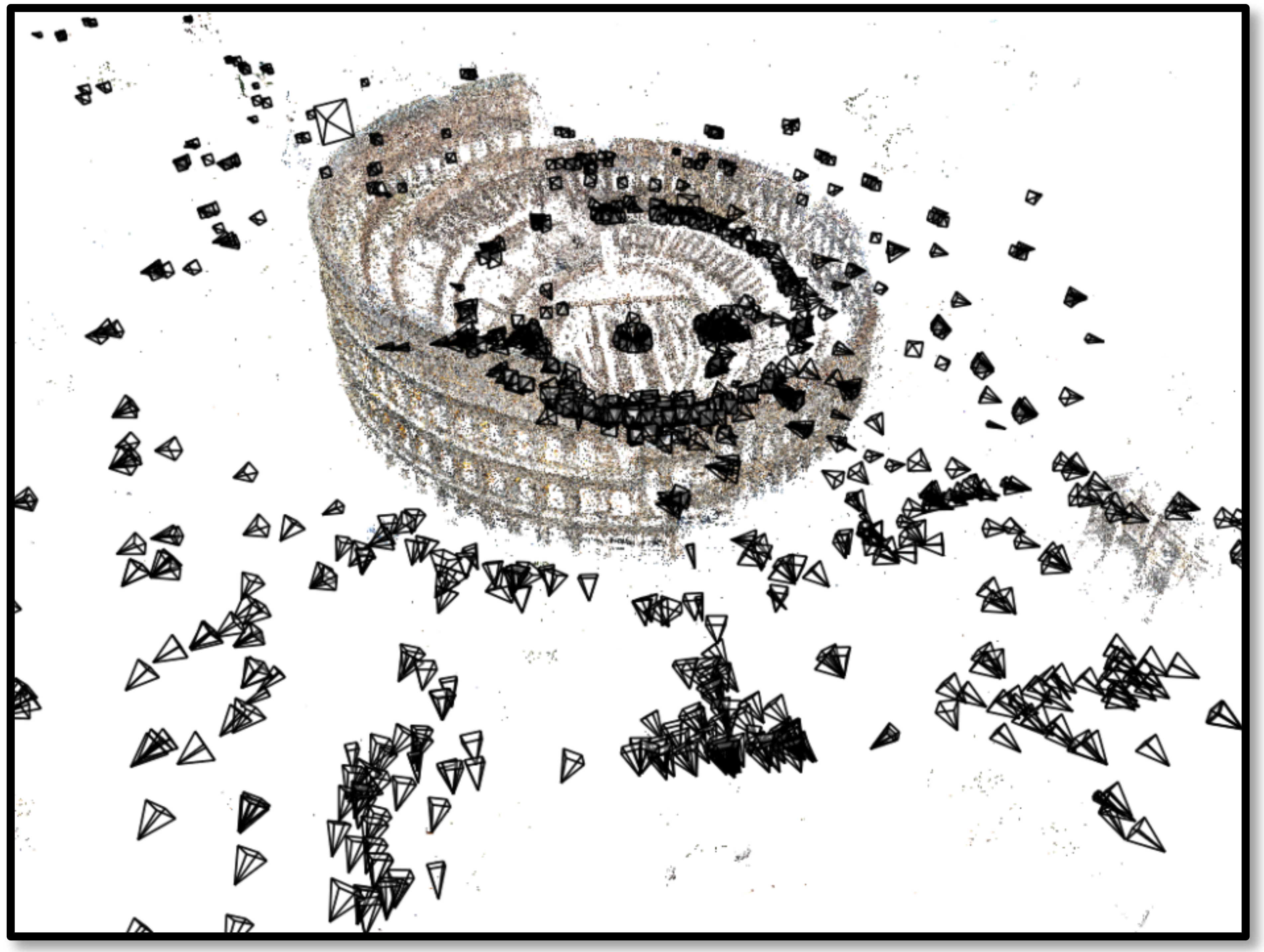
Inference

Neural Scene Representation

Downstream Tasks

Why?	We want to understand 3D world only from 2D observations (images). For that, we need to have a mathematical understanding of how they are connected.
What you'll learn.	Mathematical model of cameras. Reconstruct camera poses, approximate geometry, and camera parameters from 2D images of a scene.

Image Formation and Multi-View Geometry



Why?	We want to understand 3D world only from 2D observations (images). For that, we need to have a mathematical understanding of how they are connected.
What you'll learn.	Mathematical model of cameras. Reconstruct camera poses, approximate geometry, and camera parameters from 2D images of a scene.

Image credits: *Noah Snavely*

3D Representations

3D Scene

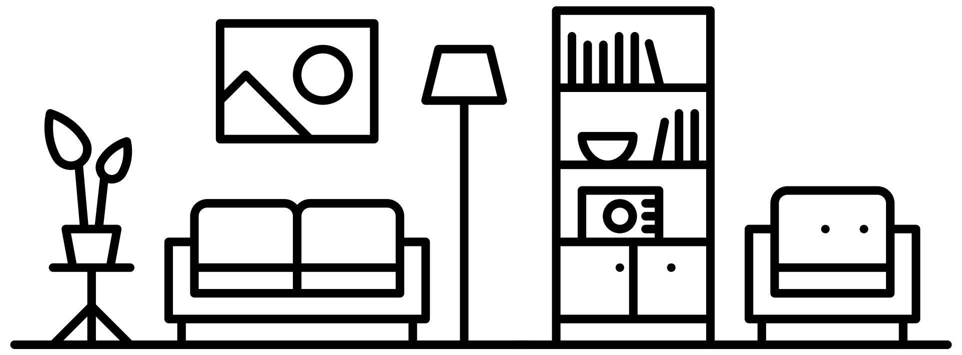


Image Formation

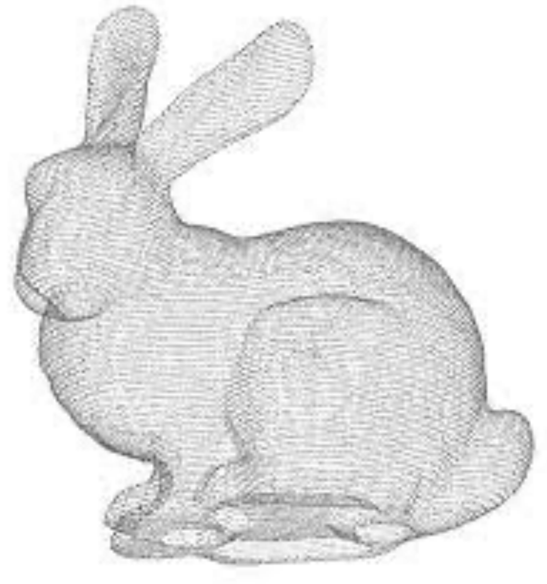
Inference

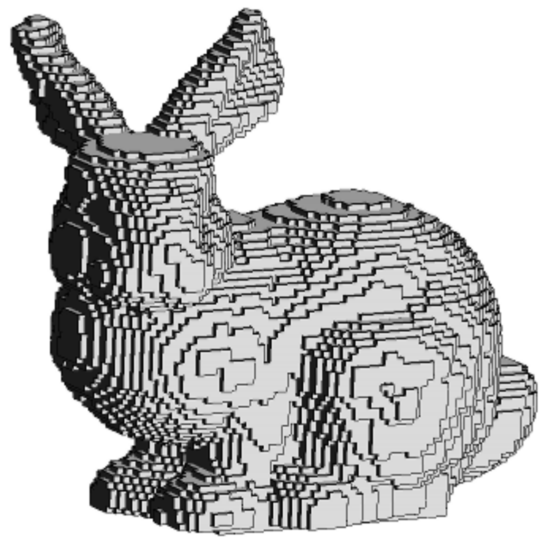
Neural Scene Representation

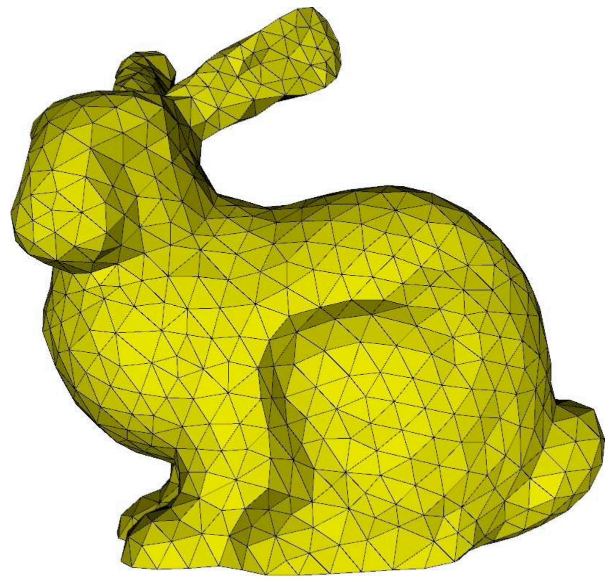
Downstream Tasks

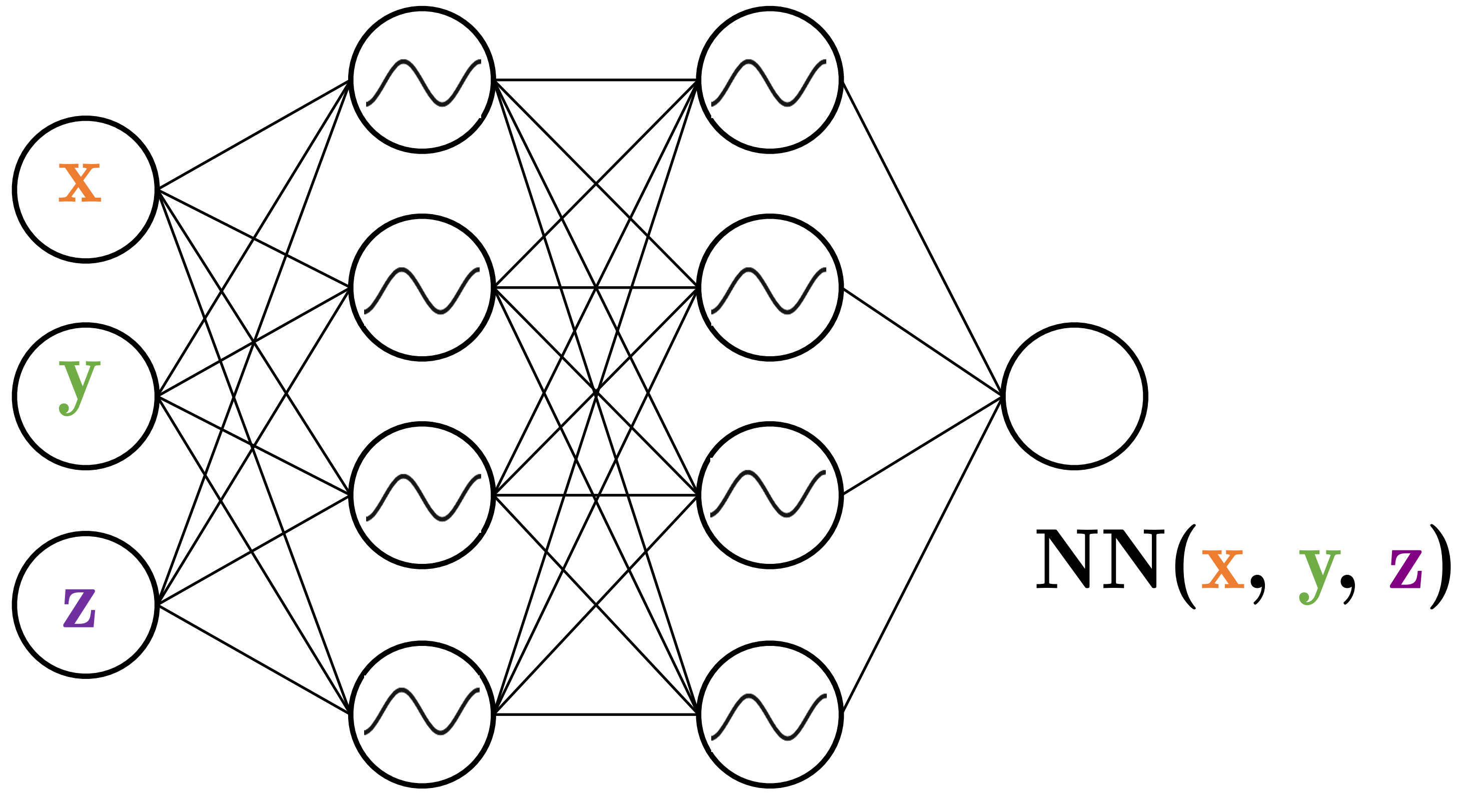
Why?	At the end of the day, we want to make predictions about 3D scenes. For that, we need to know how we can represent 3D scenes computationally.
What you'll learn.	Surface-based representations, volumetric representations, discrete representations, continuous representations.

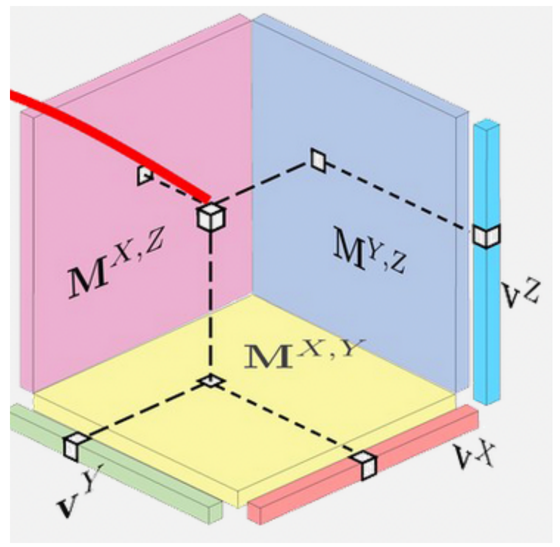
3D Representations





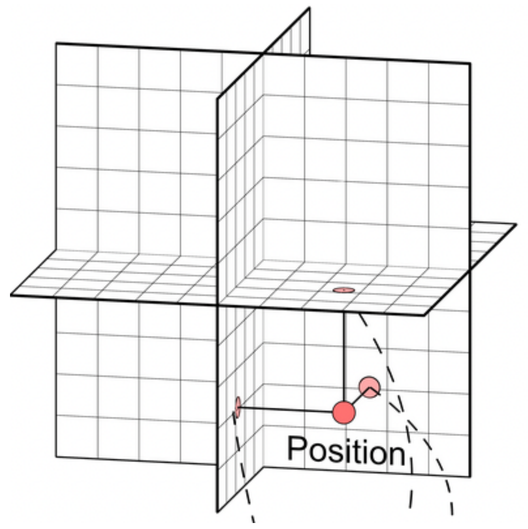


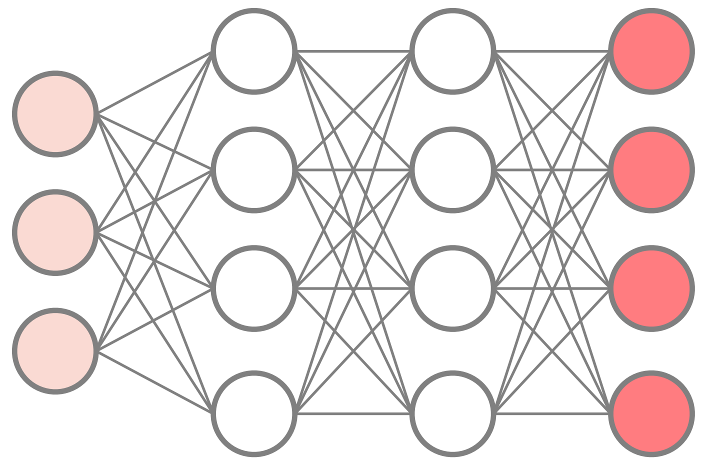




TensoRF, Chen & Xu et al. 2022

EG3D, Chan et al. 2022





Why?	At the end of the day, we want to make predictions about 3D scenes. For that, we need to know how we can represent 3D scenes computationally.
What you'll learn.	Surface-based representations, volumetric representations, discrete representations, continuous representations.

Today: Light Transport & High-level of Physics-based Rendering

3D Scene

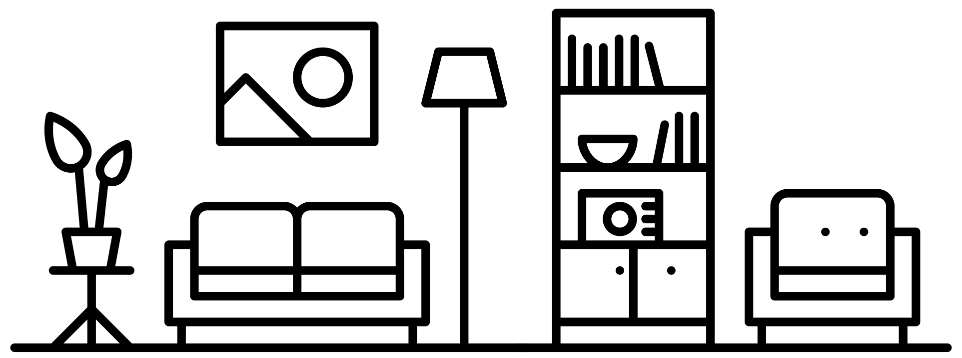


Image Formation

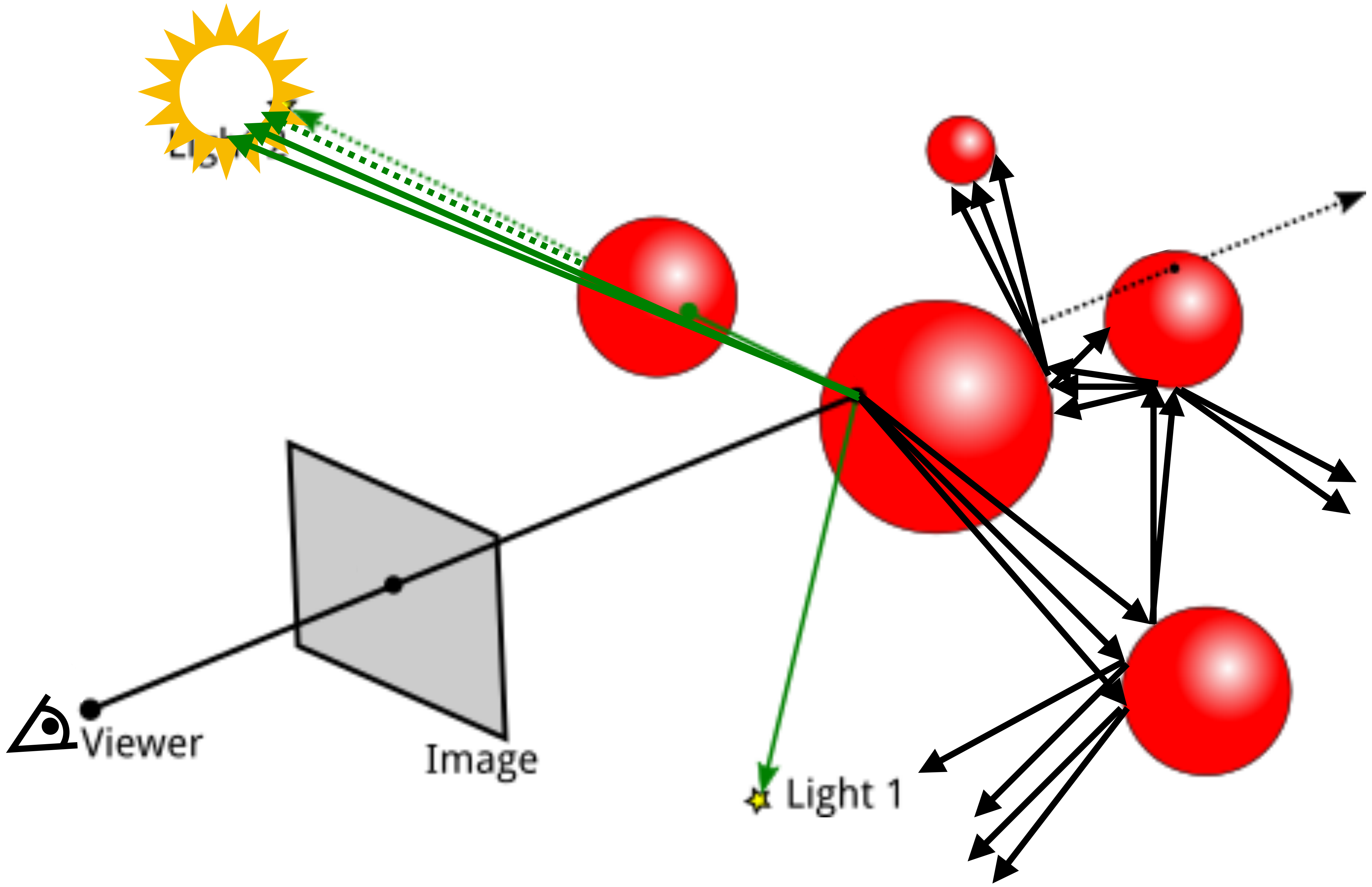
Inference

Neural Scene Representation

Downstream Tasks

Why?	In order to reason about the 3D world from images, we need to understand how 3D properties such as materials, lighting, etc. relate to the measurements observed by a camera.
What you'll learn.	The rendering equation, simulating light transport via multi-bounce ray-tracing, bidirectional radiance distribution functions, rasterization, rendering.

Models of Light Transport



From "Computer Graphics in the Age of AI", C. Karen Liu & Jiajun Wu

Why?	In order to reason about the 3D world from images, we need to understand how 3D properties such as materials, lighting, etc. relate to the measurements observed by a camera.
What you'll learn.	The rendering equation, simulating light transport via multi-bounce ray-tracing, bidirectional radiance distribution functions, rasterization, rendering.

Differentiable Rendering

3D Scene

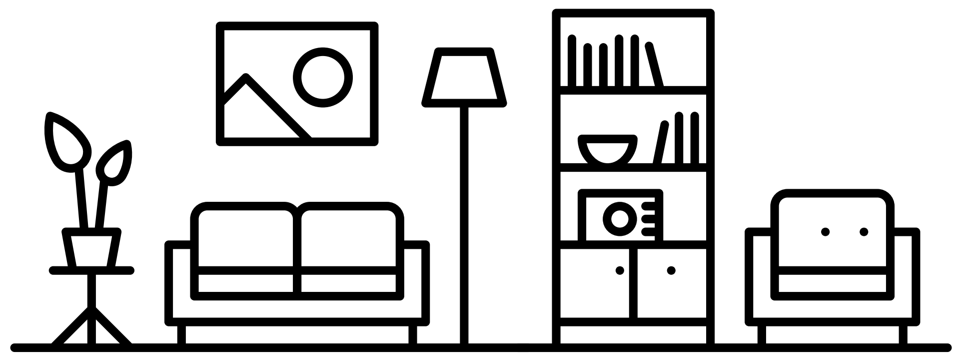


Image Formation

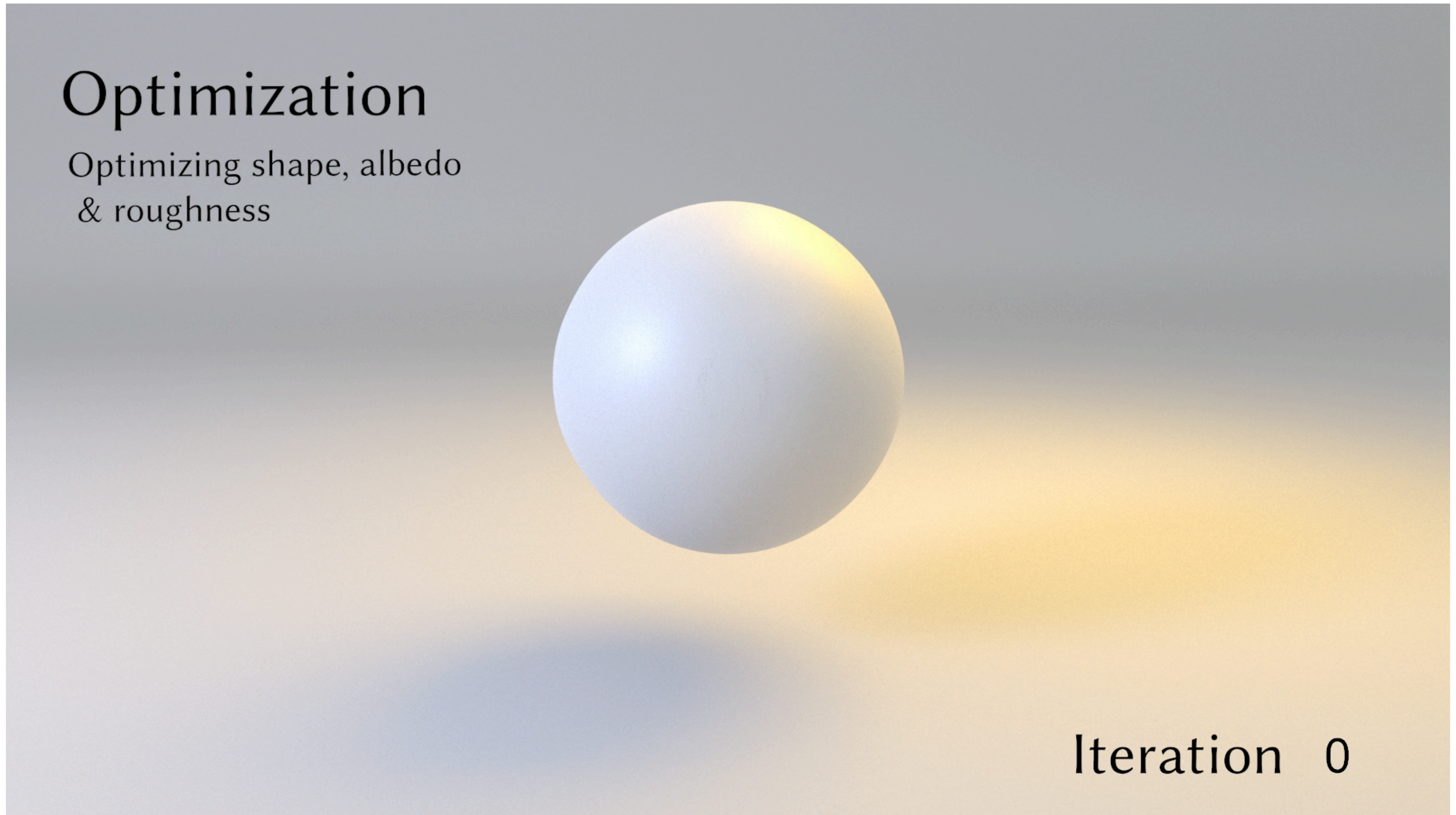
Inference

Neural Scene Representation

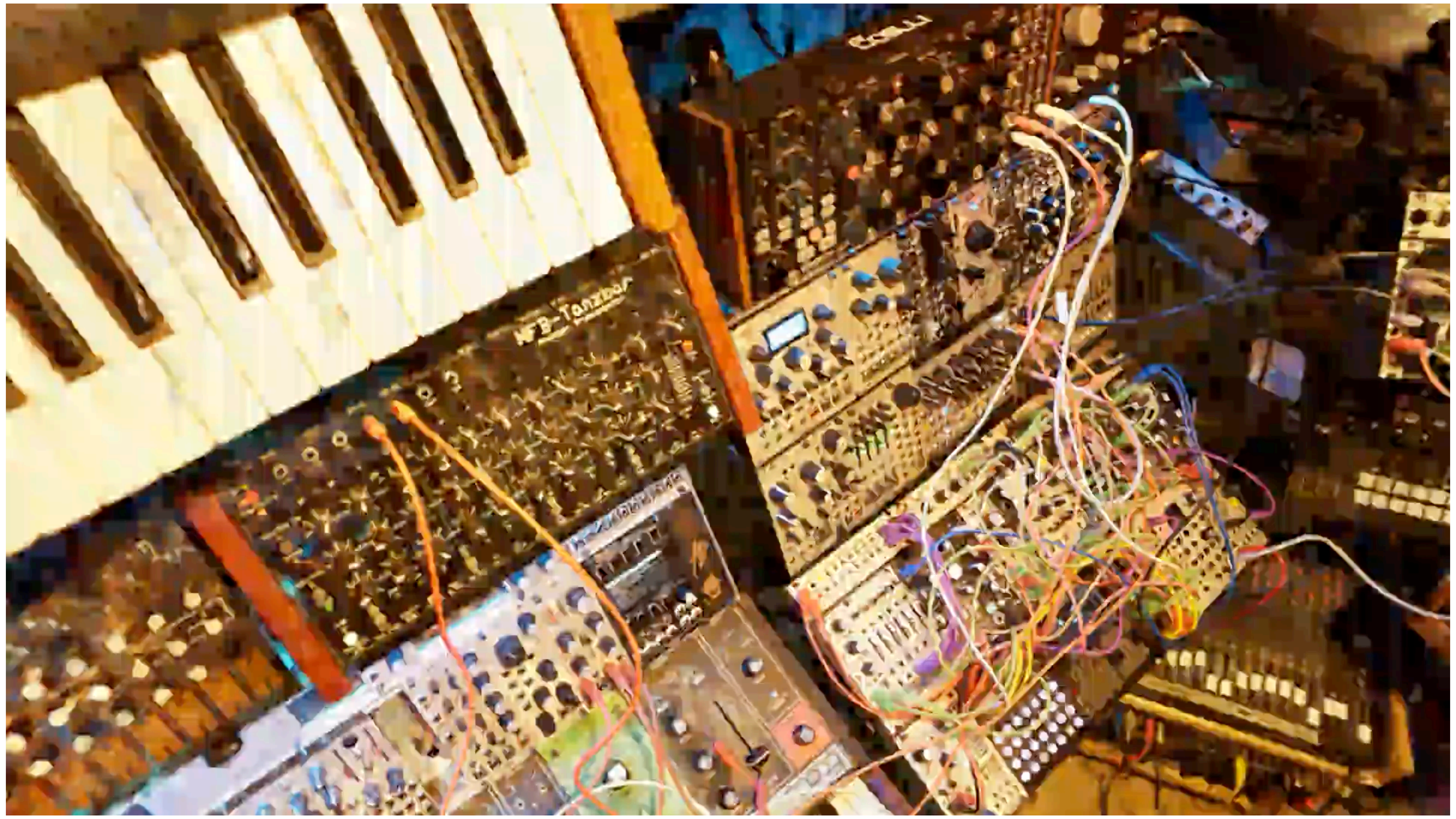
Downstream Tasks

Why?	Part of our problem relates to **inverting** the rendering process: Given 2D images, we want to reconstruct 3D scenes. Differentiable rendering is one way of exactly inverting the rendering process.
What you'll learn.	The structure of differentiable renderers, pros and cons and assumptions of different algorithms.

Differentiable Rendering



Differentiable Signed Distance Function Rendering,
Vicini et al. 2022



InstantNGP, Müller et al. 2022

Why?	Part of our problem relates to **inverting** the rendering process: Given 2D images, we want to reconstruct 3D scenes. Differentiable rendering is one way of exactly inverting the rendering process.
What you'll learn.	The structure of differentiable renderers, pros and cons and assumptions of different algorithms.

Prior-based Reconstruction

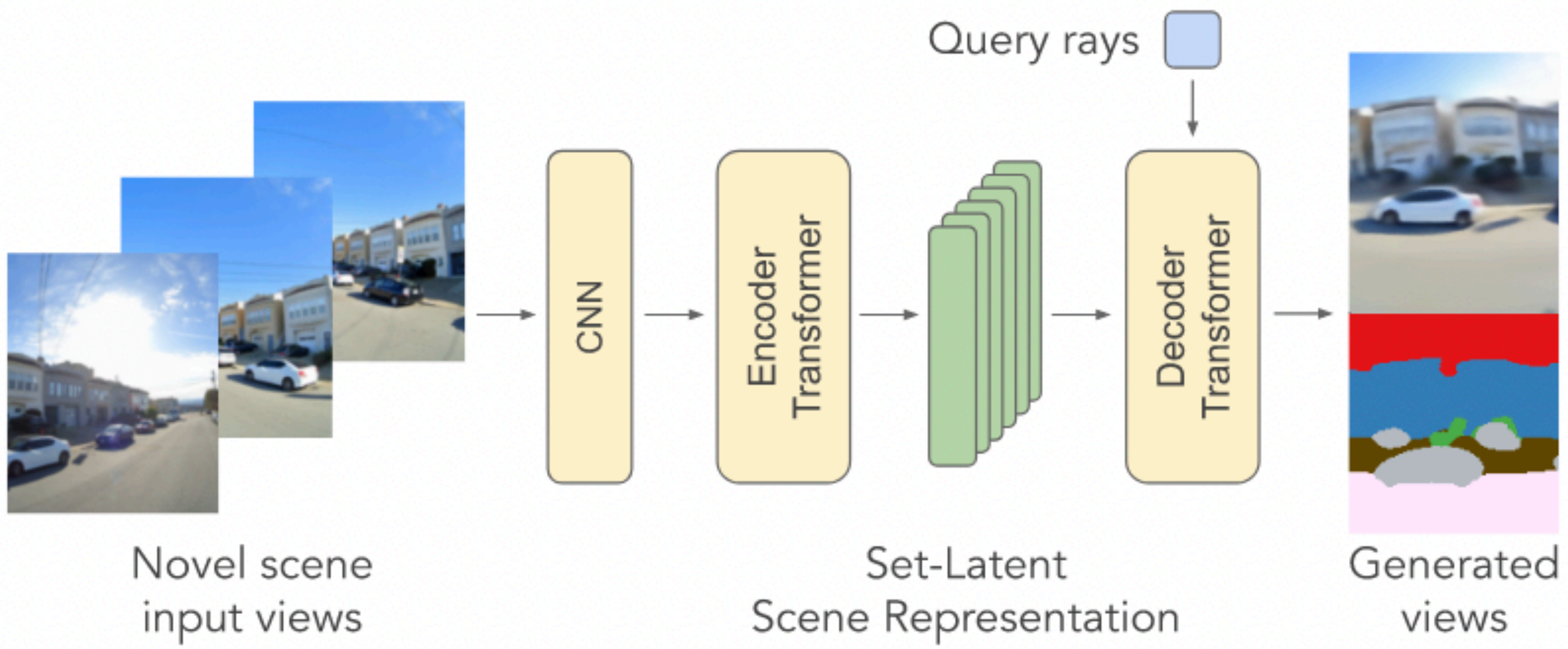
Why?	We humans can reconstruct 3D from **incomplete** observations, by using knowledge that we have learned about the world. Deep learning is the best way we know to date to learn such priors from data.
What you'll learn.	How to express priors over 3D scenes using deep learning, different ways of doing inference (encoding, auto-decoding)

Prior-based Reconstruction

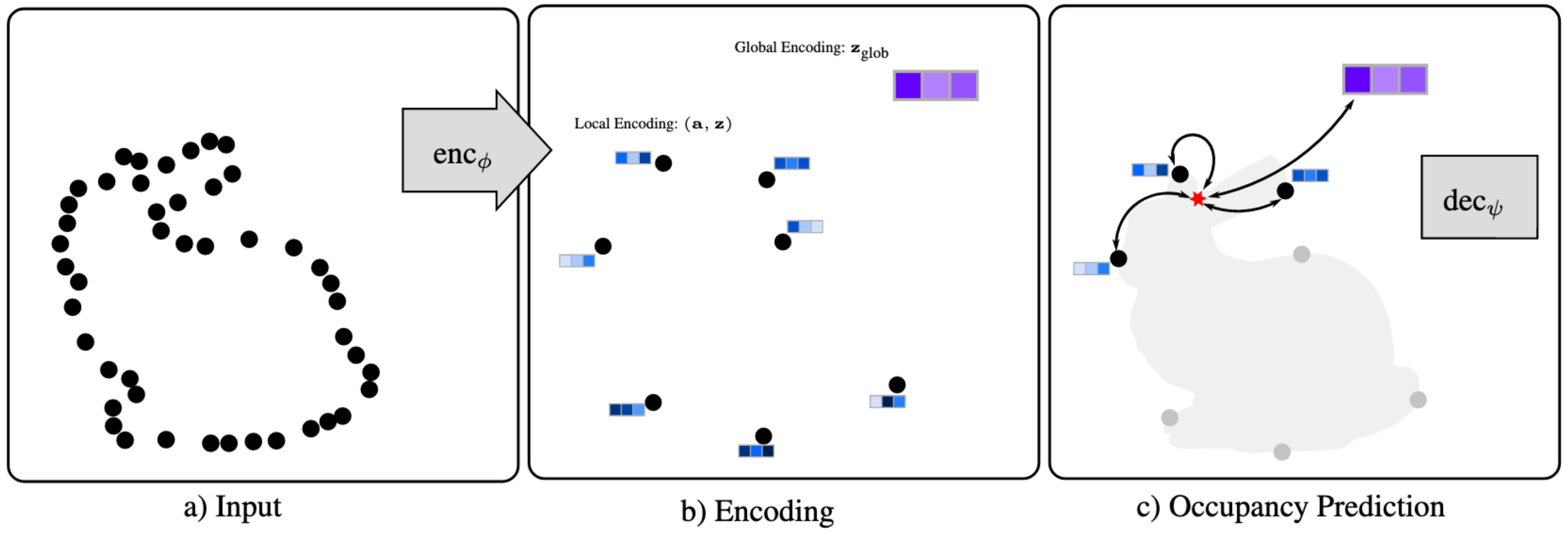
Seeing 3D Objects in a Single Image via Self-Supervised Static-Dynamic Disentanglement, Sharma et al. 2022

Why?	We humans can reconstruct 3D from **incomplete** observations, by using knowledge that we have learned about the world. Deep learning is the best way we know to date to learn such priors from data.
What you'll learn.	How to express priors over 3D scenes using deep learning, different ways of doing inference (encoding, auto-decoding)

Advanced Inference Topics



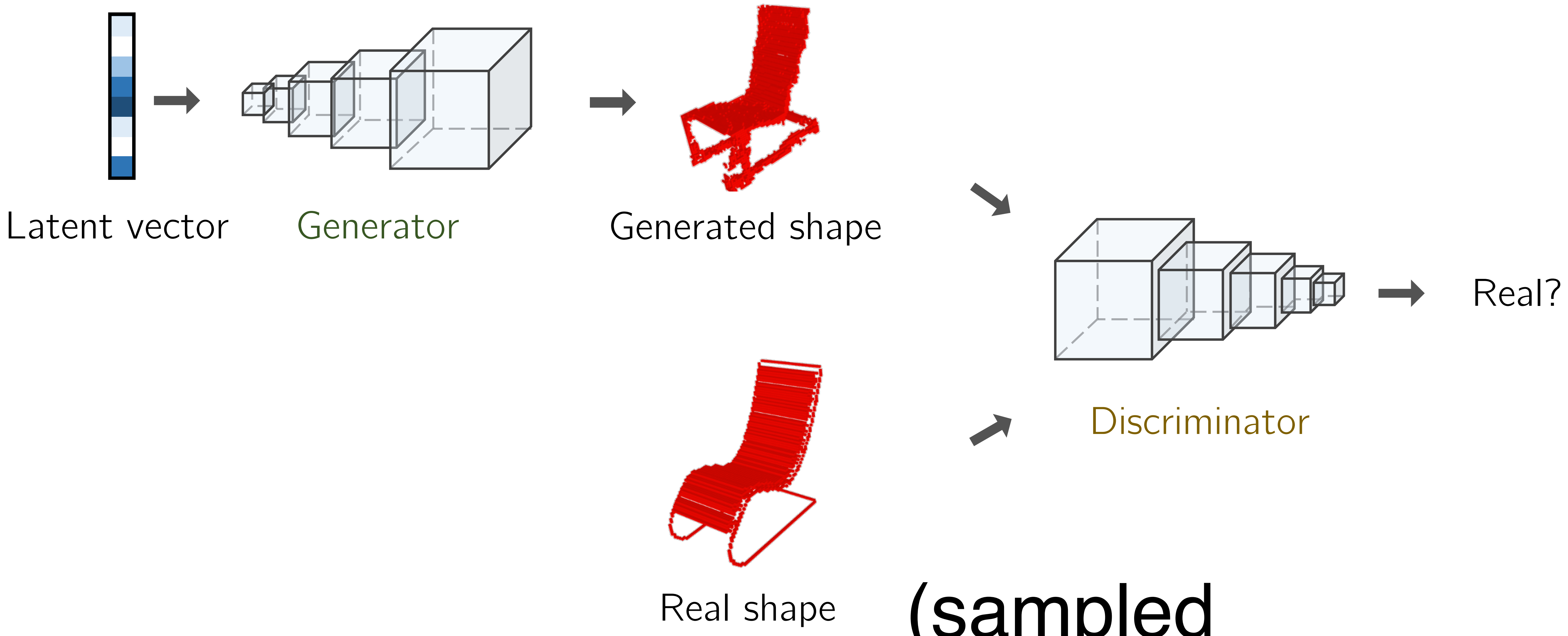
Scene Representation Transformer, Sajjadi et al. 2022



AIR-Nets, Giebenhain et al. 2021

Why?	**Deep learning affords us alternative ways of formulating 3D reconstruction that don't involve exact forward models (such as multi-bounce ray-marching). This is potentially more tractable & biologically plausible.**
What you'll learn.	**Transformer-based inference, attention-based conditioning, gradient-based meta-learning, light field neural scene representations.**

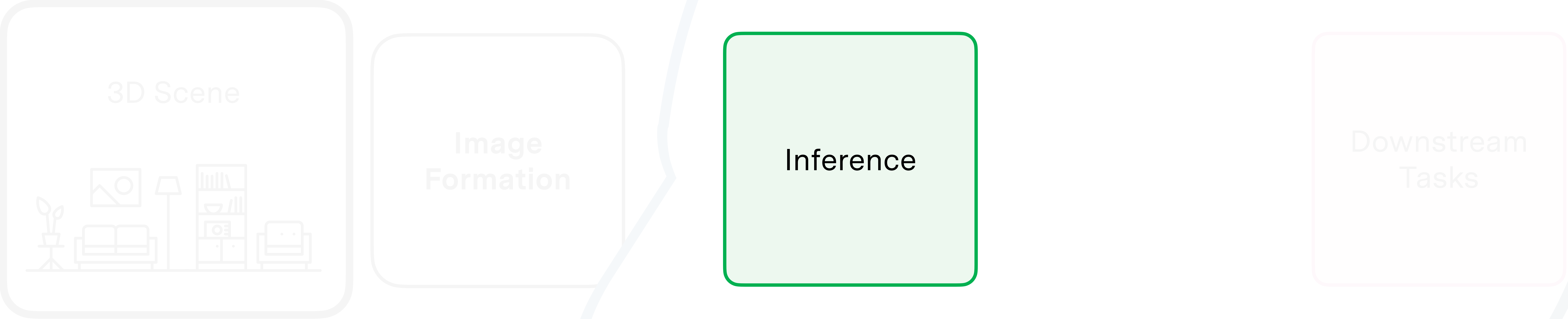
Unconditional Generative 3D Modeling



(sampled

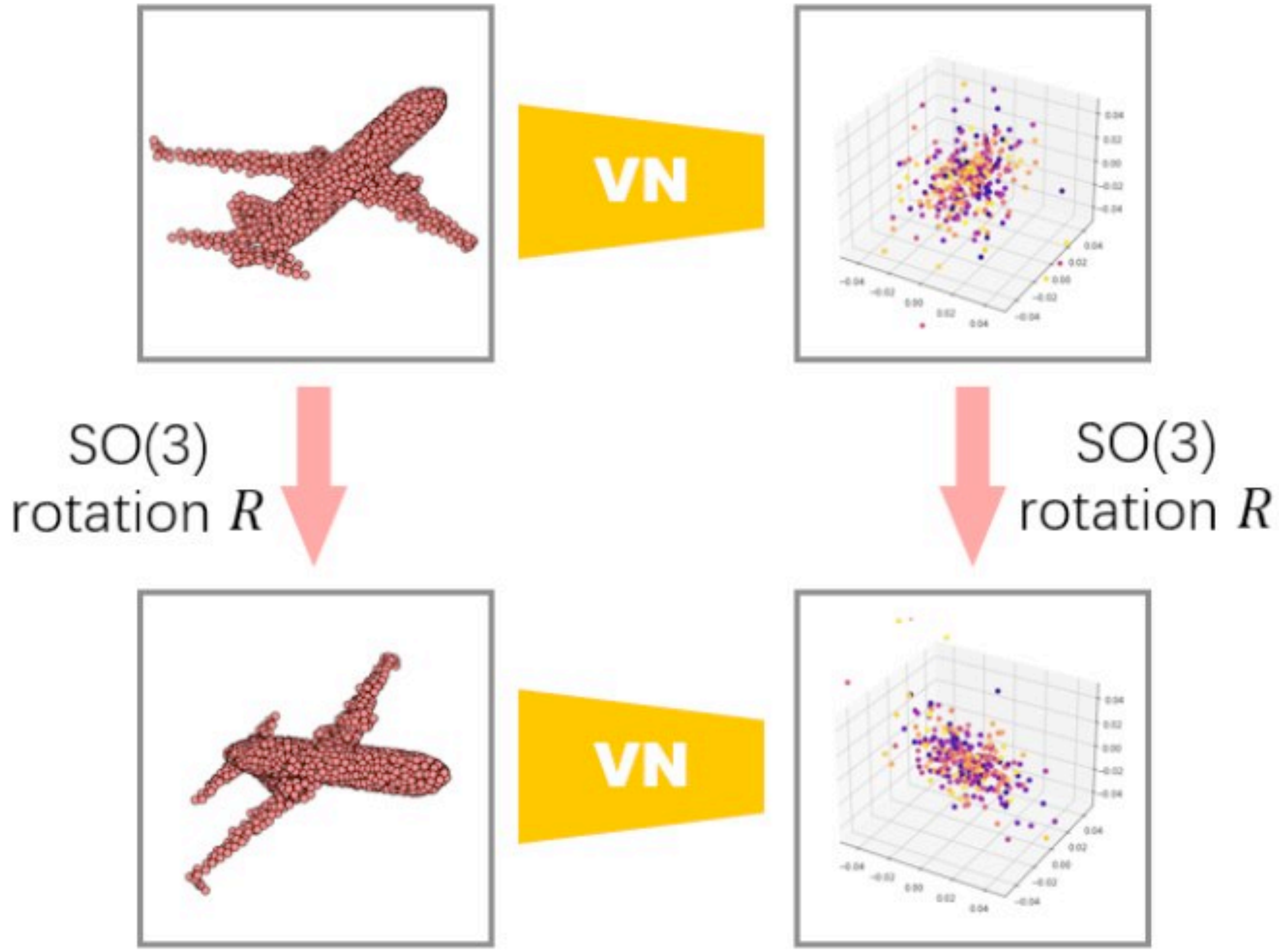
Why?	We humans can reconstruct 3D from **incomplete** observations, by using knowledge that we have learned about the world. Deep learning is the best way we know to date to learn such priors from data.
What you'll learn.	How to express priors over 3D scenes using deep learning, different ways of doing inference (encoding, auto-decoding)

Today: Geometric Deep Learning

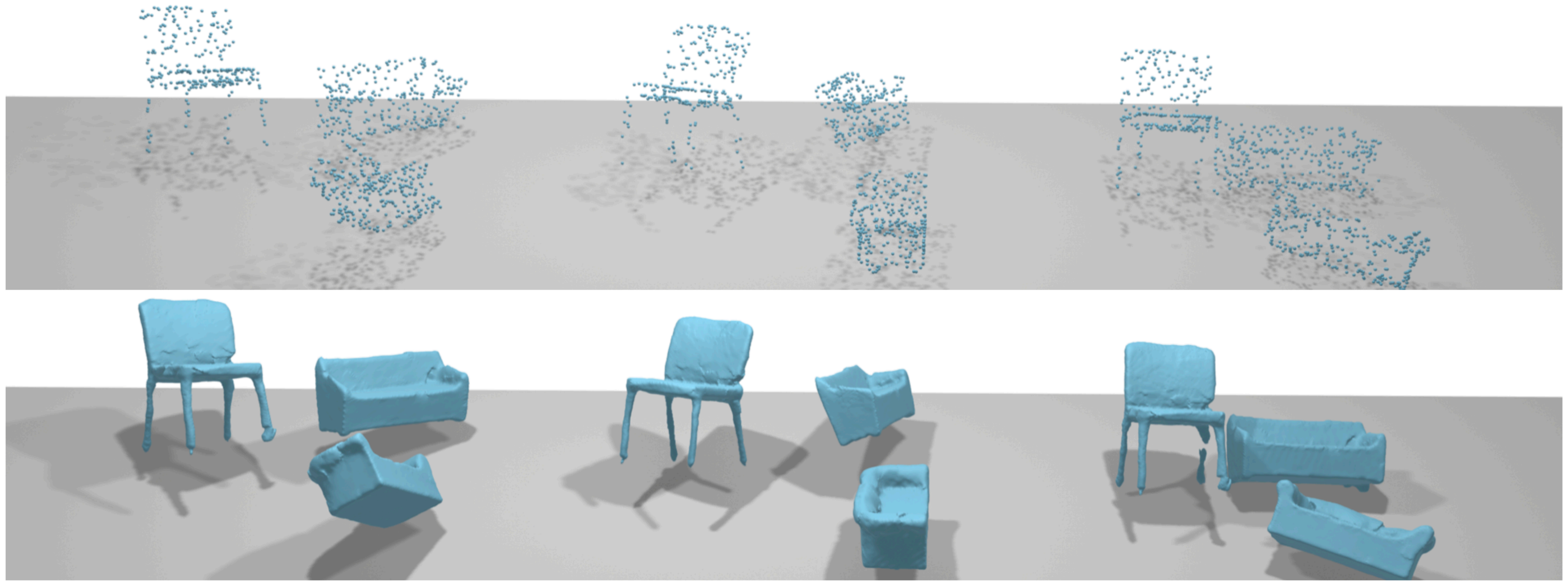


Why?	To guarantee generalization to certain transformations (such as 3D translations and rotations), we need to build special neural network architectures.
What you'll learn.	Basics of group theory, neural network architectures that respect symmetries (shift, rotation, scale equivariance)

Today: Geometric Deep Learning



Vector Neurons, Deng et al. 2021



SE(3)-Equivariant Attention Networks for Shape Reconstruction in Function Space, Chatzipantazis & Pertigkiozoglou et al.

Why?	**To guarantee generalization to certain transformations (such as 3D translations and rotations), we need to build special neural network architectures.**
What you'll learn.	**Basics of group theory, neural network architectures that respect symmetries (shift, rotation, scale equivariance)**

Many 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!

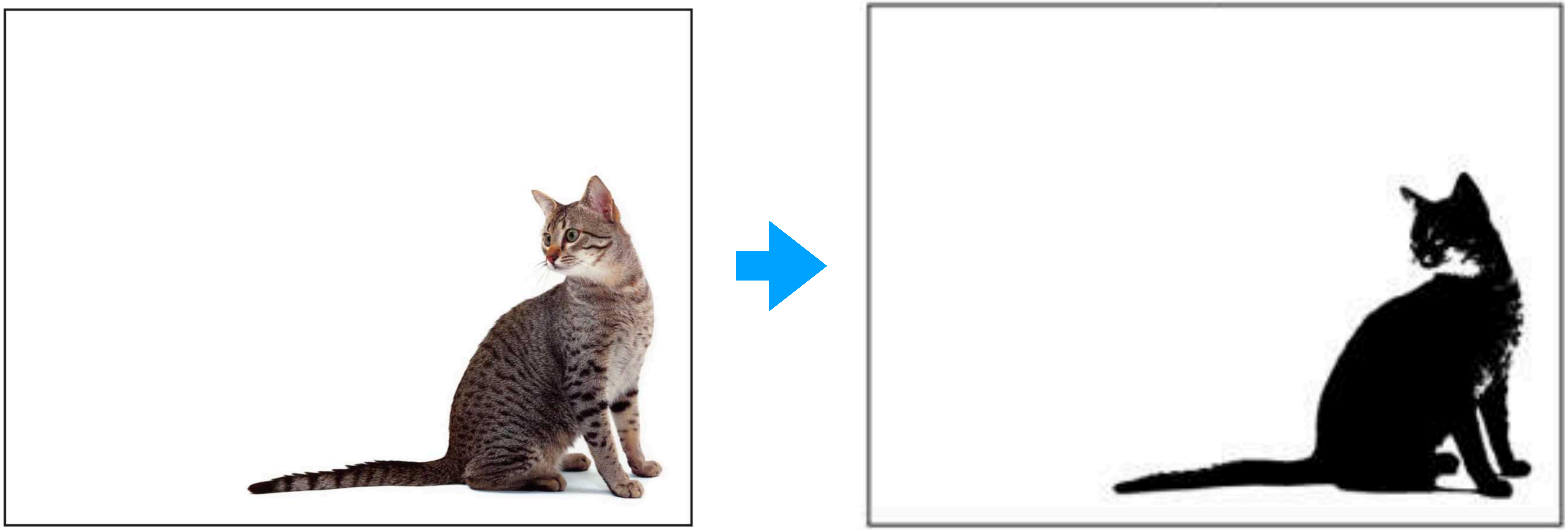
INVERSE GRAPHICS

MODULE 4:
Geometric Deep Learning

Representation Theory & **SYMMETRIES**

6.S980

A fundamental question:



What Neural Network to Use?

Image: Bernhard Kainz

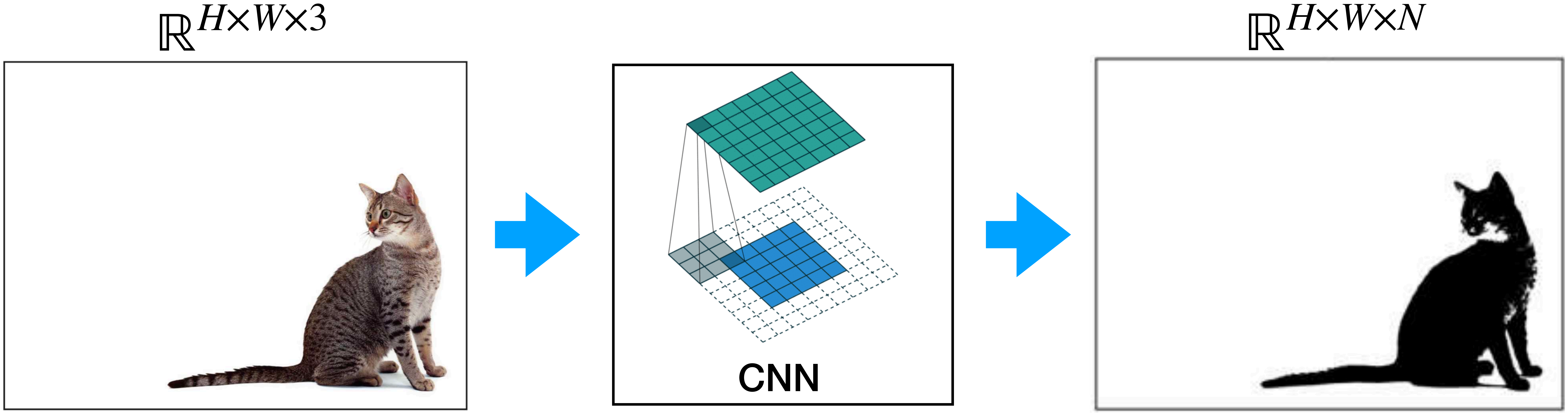


Image: Bernhard Kainz

Image: Bernhard Kainz

Why is this a bad idea?

Image: Bernhard Kainz

Why is this a bad idea?

Image: Bernhard Kainz

“Shift Equivariance”

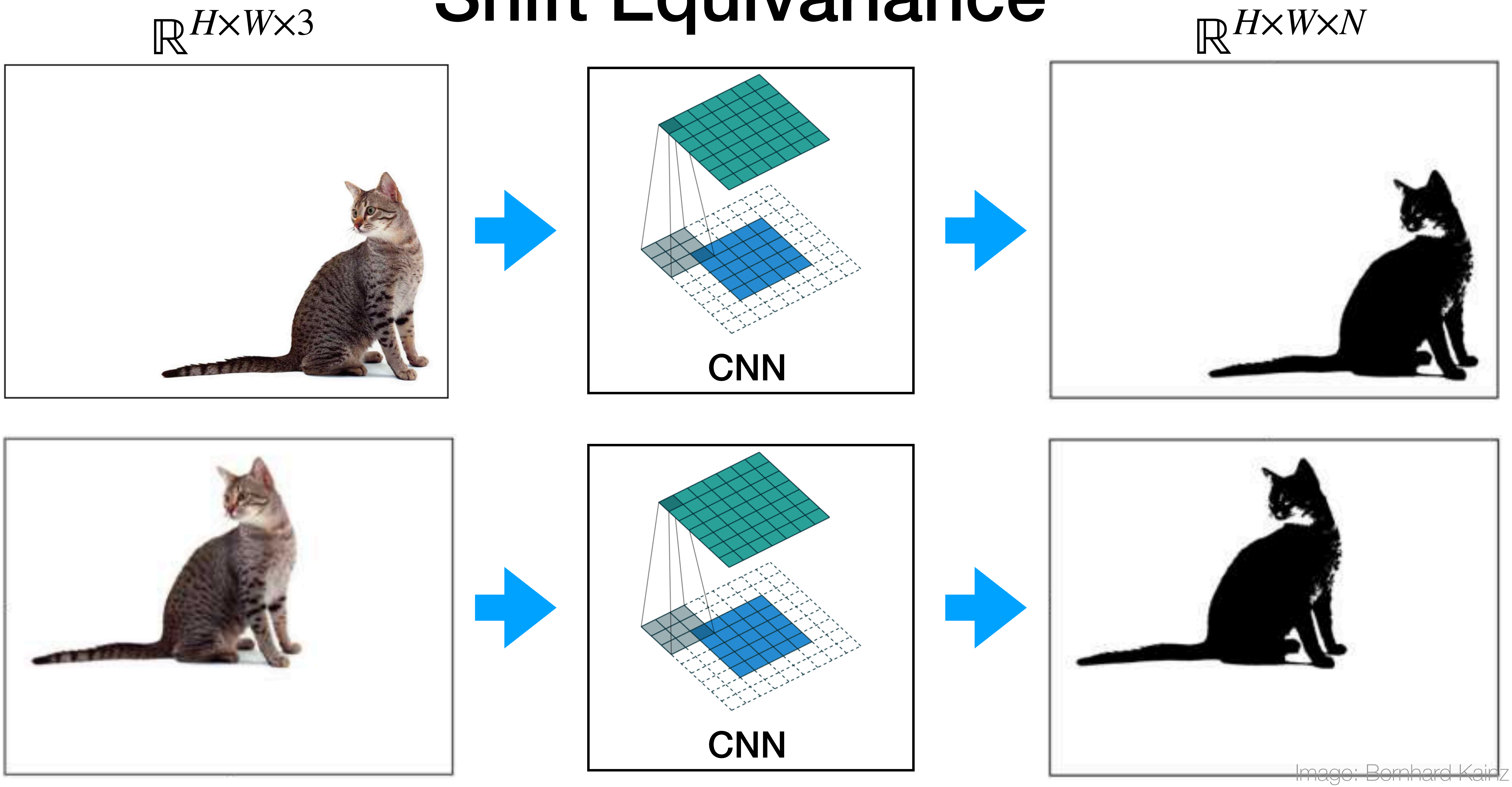


Image: Bernhard Kainz

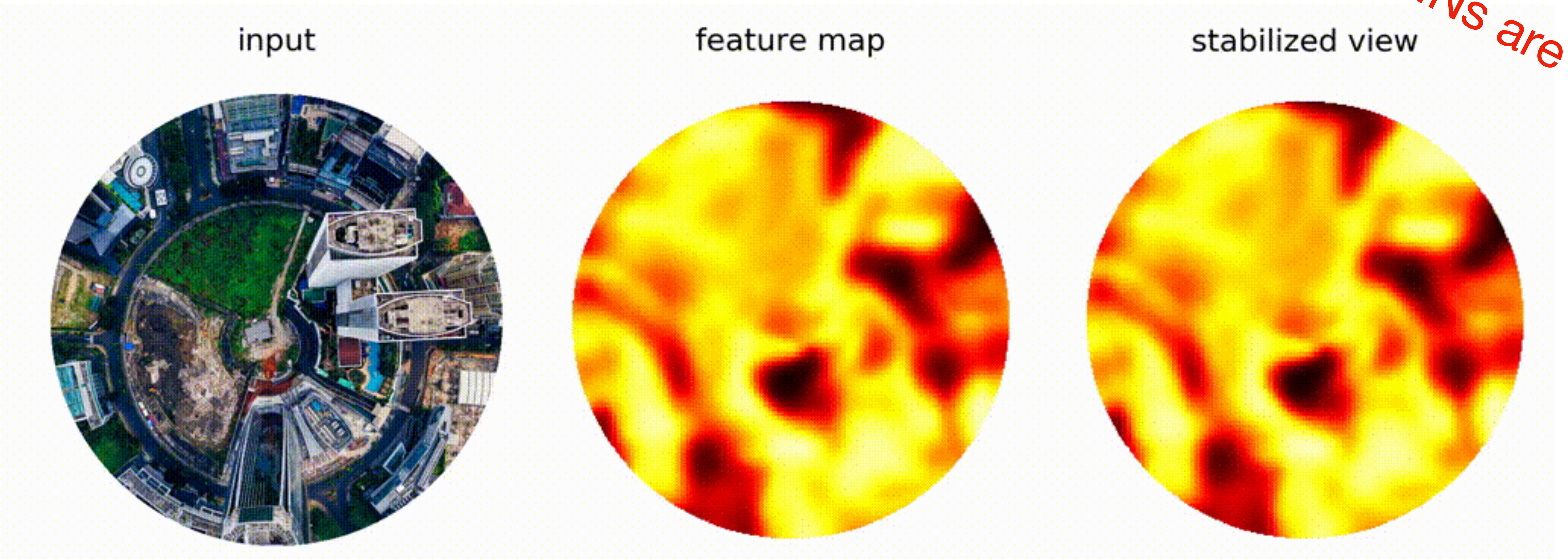
Geometric guarantees (equivariance)

CNNs are translation equivariant

Via convolutions

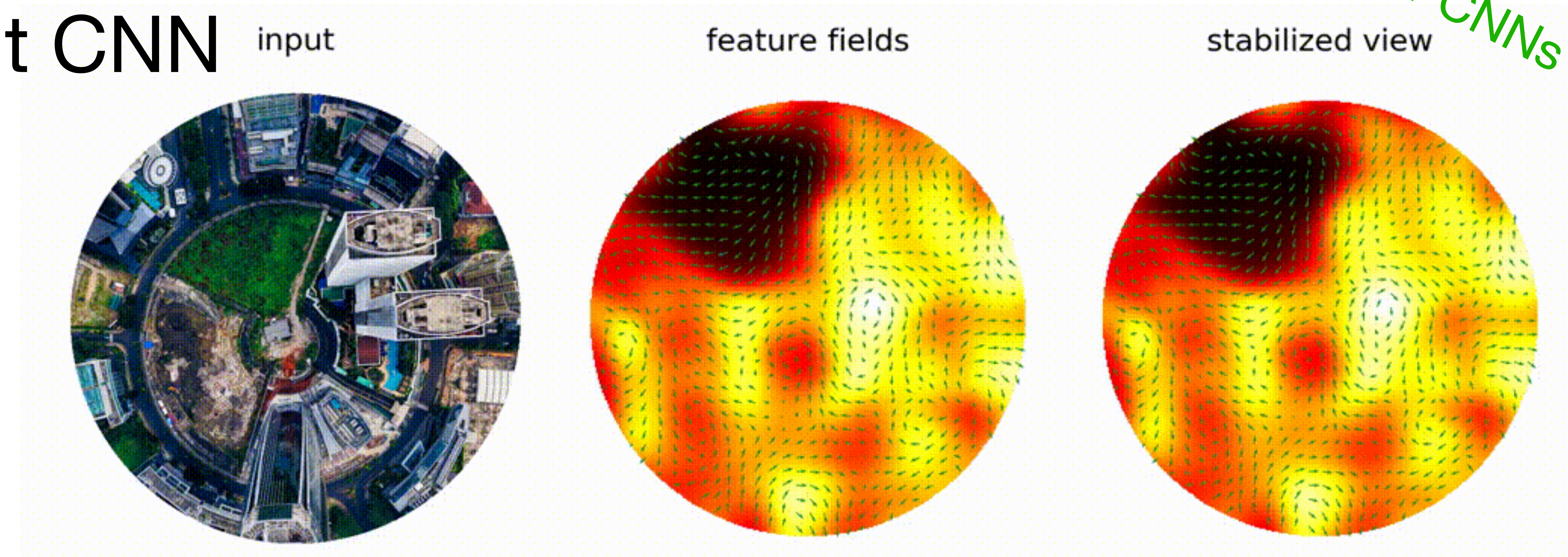
Geometric guarantees (equivariance)

Normal CNN



CNNs are not rotation equivariant

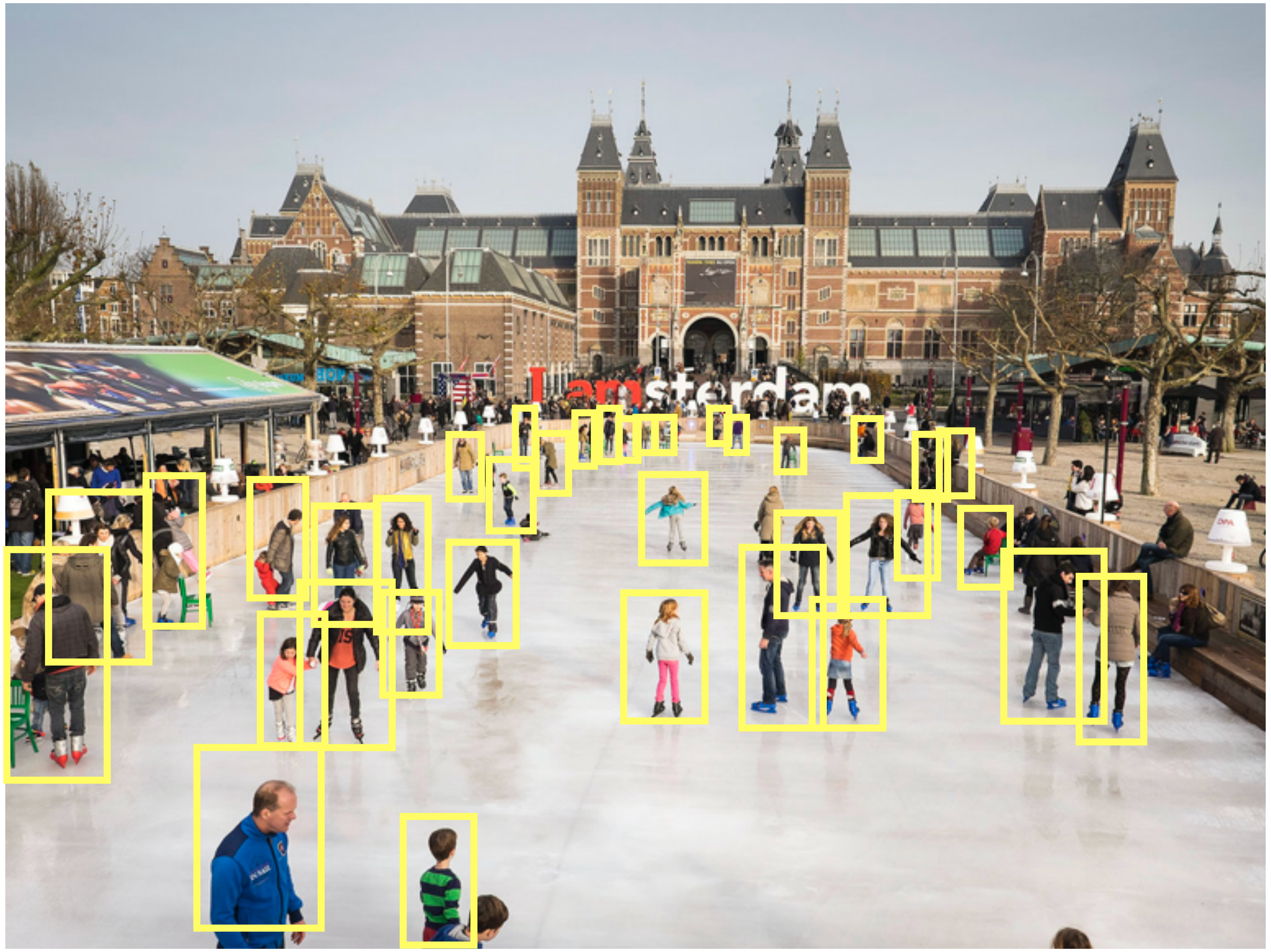
Group equivariant CNN



G-CNNs are rotation equivariant!

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

Geometric guarantees (equivariance)



Importance of equivariance:

- No information is lost when the input is transformed
- Guaranteed stability to (local + global) transformations

Group convolutions:

- Equivariance beyond translations
- Geometric guarantees
- Increased weight sharing

G-CNNs are not only relevant for invariant problems but for any type of structured data!

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

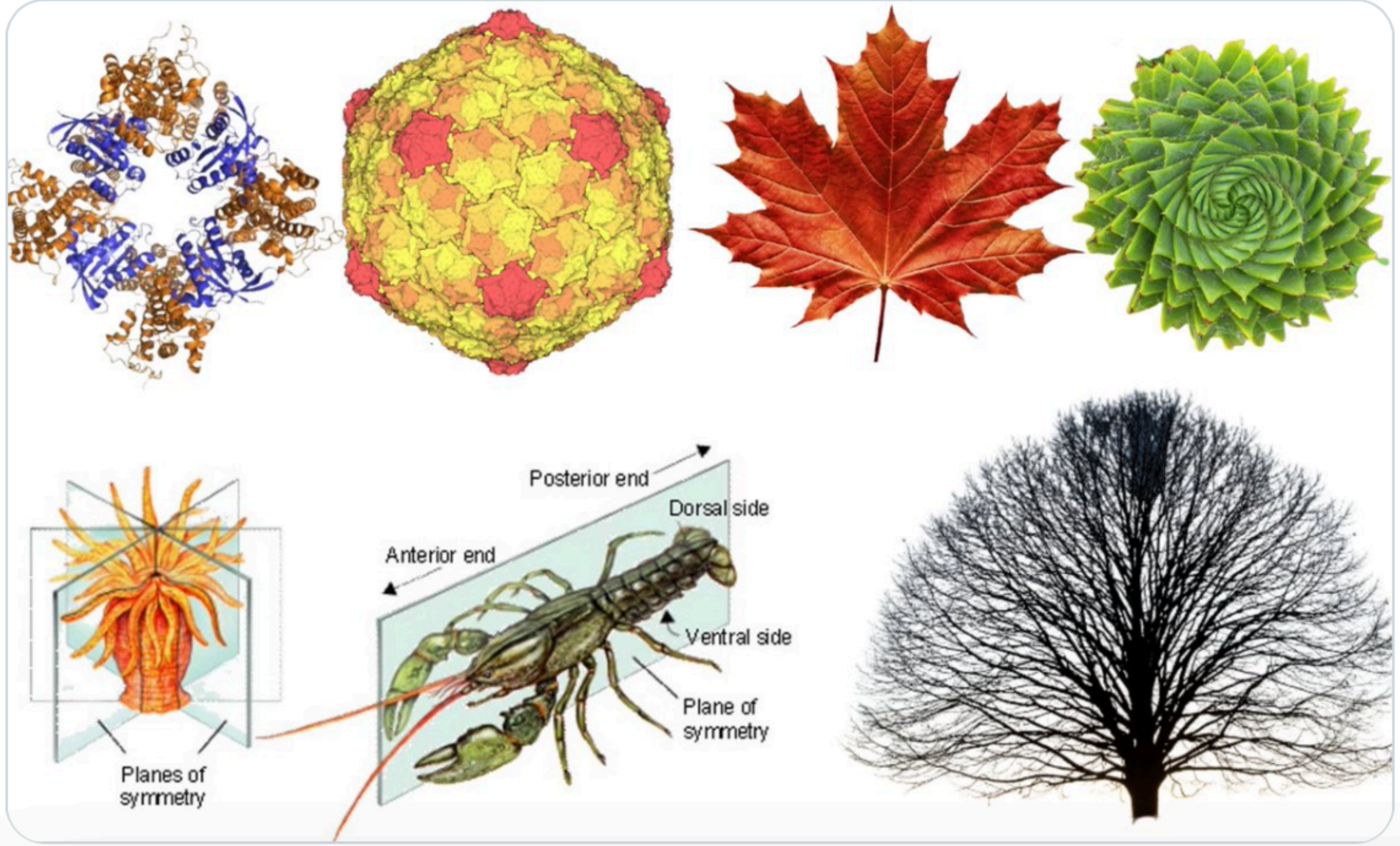
Symmetries in nature

Chico Camargo @evoluchico

Have you ever noticed how nature seems to love symmetry? 🔺🟥🔴

Evolution has literally trillions of shapes to pick from, and yet, biological structures often show symmetry and simplicity.

This is the story of the discovery that completely changed how I see biology. 🧵



Chaitanya K. Joshi @chaitjo

"Why does evolution favor symmetric structures when they only represent a minute subset of all possible forms? ...$\boxed{\text{Since symmetric structures need less information to encode}}$, they are much more likely to appear as potential variation."

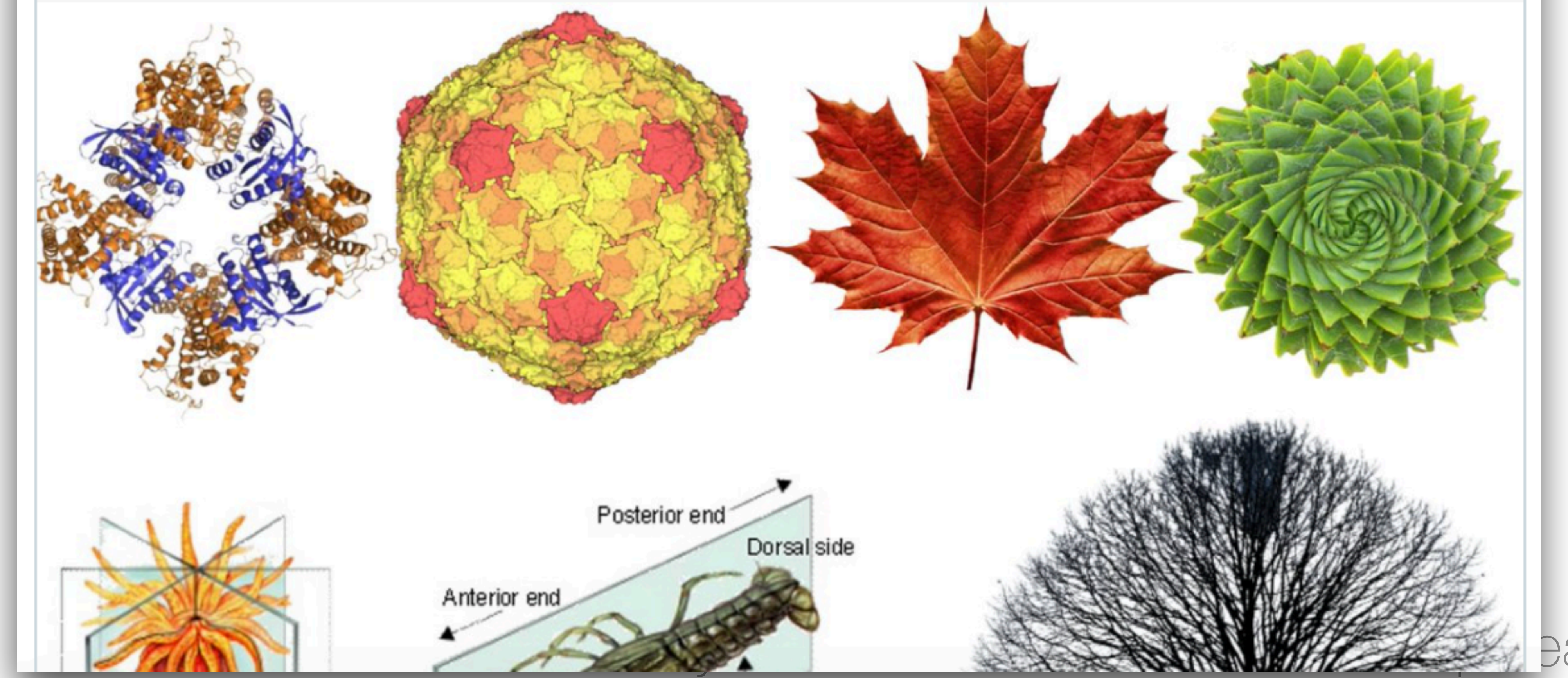
Chico Camargo @evoluchico · Mar 14

Have you ever noticed how nature seems to love symmetry? 🔺🟥🔴

Evolution has literally trillions of shapes to pick from, and yet, biological structures often show symmetry and simplicity.

This is the story of the discovery that completely changed how I see biology. 🧵

Show this thread



What is a group?

A group $(G, \cdot\ )$ is a set of elements G equipped with a group product $\cdot$, a binary operator, that satisfies the following four axioms:

- **Closure**: Given two elements g and h of G, the product $g \cdot h$ is also in G.
- **Associativity**: For $g, h, i \in G$ the product $\cdot$ is associative, i.e., $g \cdot (h \cdot i) = (g \cdot h) \cdot i$.
- **Identity element**: There exists an identity element $e \in G$ such that $e \cdot g = g \cdot e = g$ for any $g \in G$.
- **Inverse element**: For each $g \in G$ there exists an inverse element $g^{-1} \in G$ s.t. $g^{-1} \cdot g = g \cdot g^{-1} = e$.

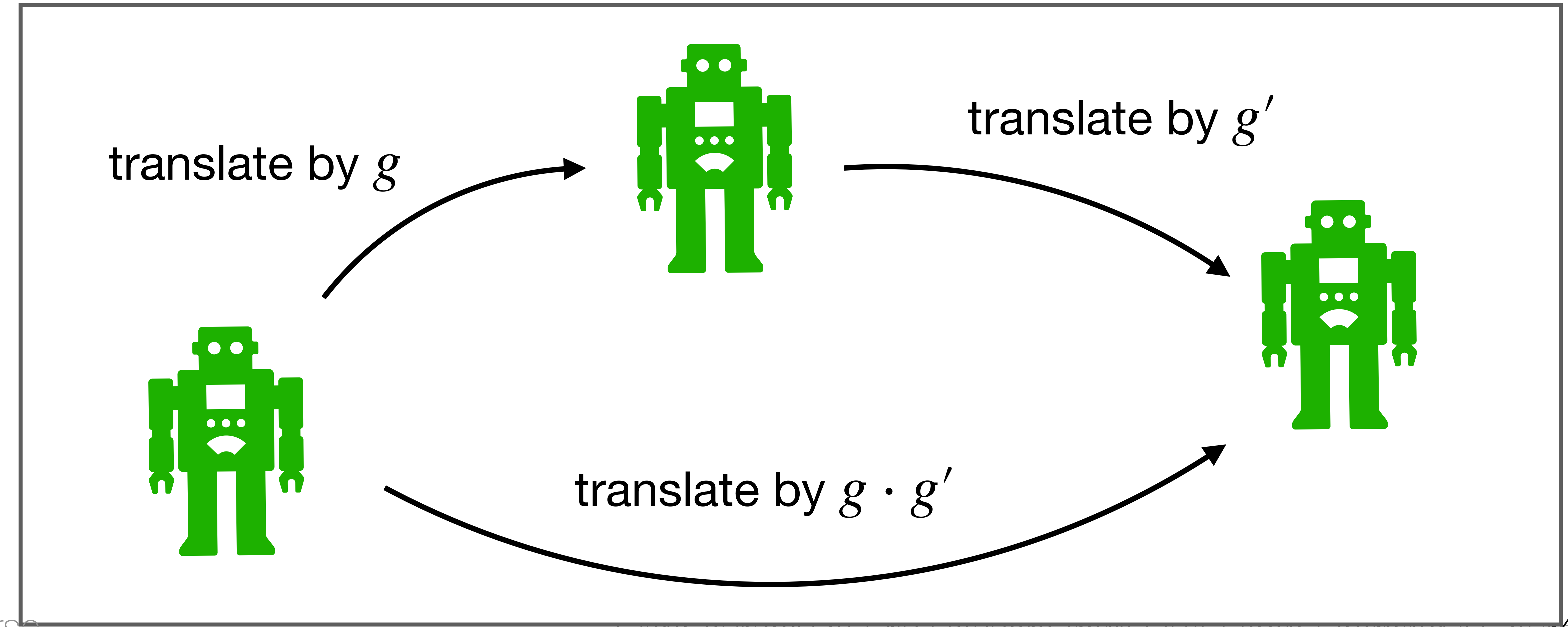
Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Translation group $(\mathbb{R}^2, +)$

The translation group consists of all possible translations in $\mathbb{R}^2$ and is equipped with the group product and group inverse:

$$g \cdot g' = (\mathbf{x} + \mathbf{x}')$$
$$g^{-1} = (-\mathbf{x})$$

with $g = (\mathbf{x}), g' = (\mathbf{x}')$ and $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^2$.



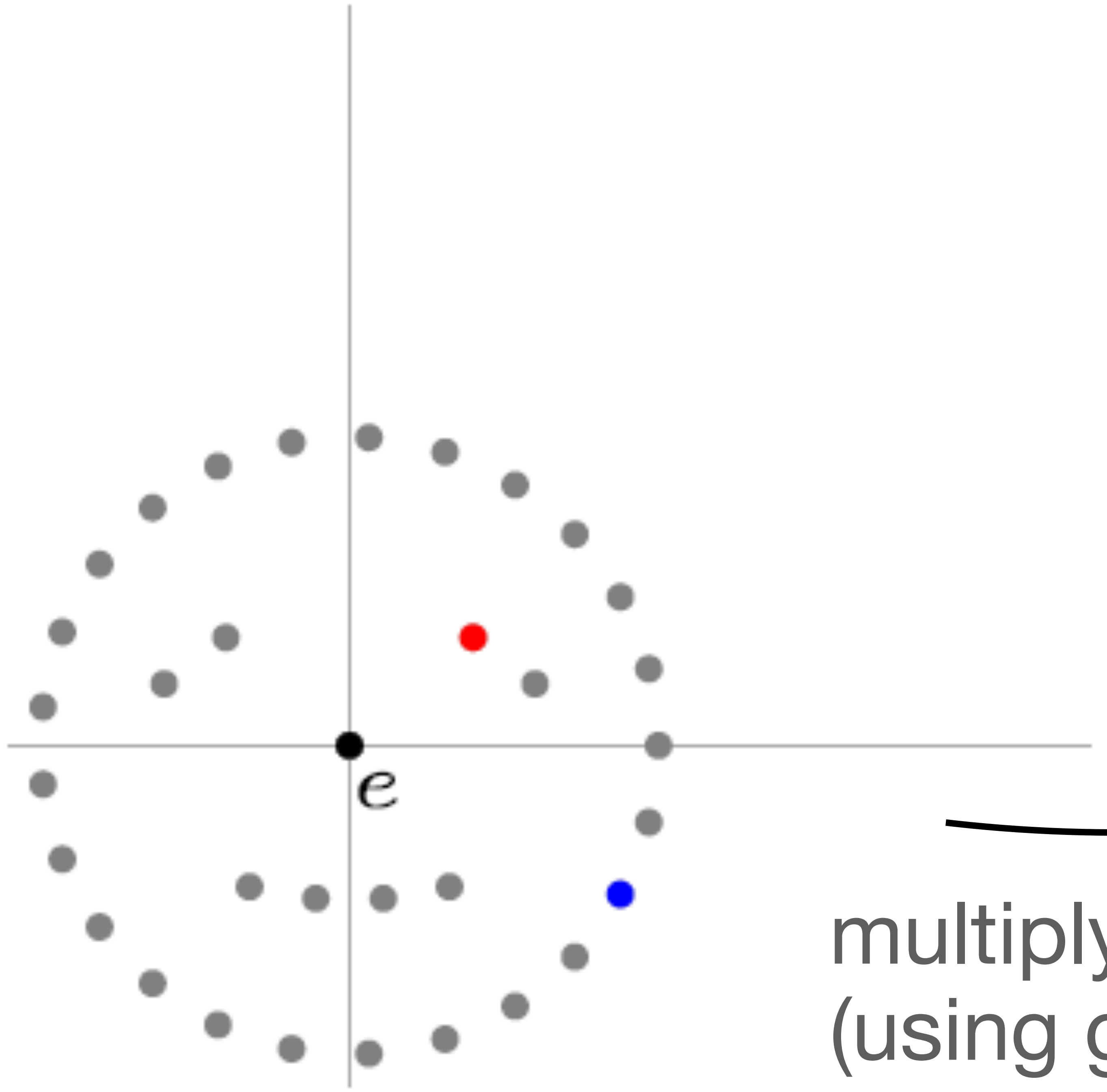
Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Translation group $(\mathbb{R}^2, +)$

The translation group consists of all possible translations in $\mathbb{R}^2$ and is equipped with the group product and group inverse:

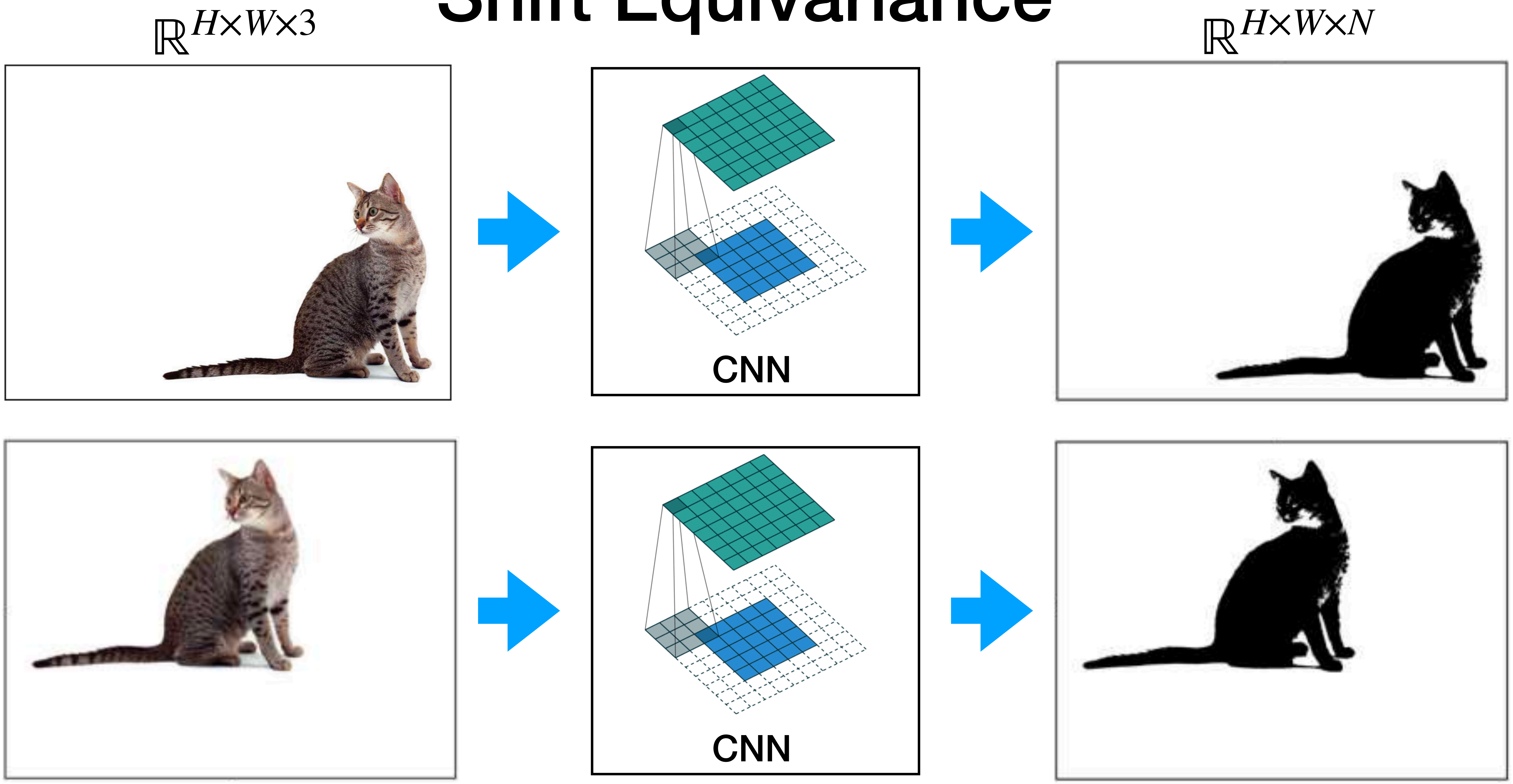
$$g \cdot g' = (\mathbf{x} + \mathbf{x}')$$
$$g^{-1} = (-\mathbf{x})$$

with $g = (\mathbf{x})$, $g' = (\mathbf{x}')$ and $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^2$.



multiply each element with g
(using group prod)

“Shift Equivariance”



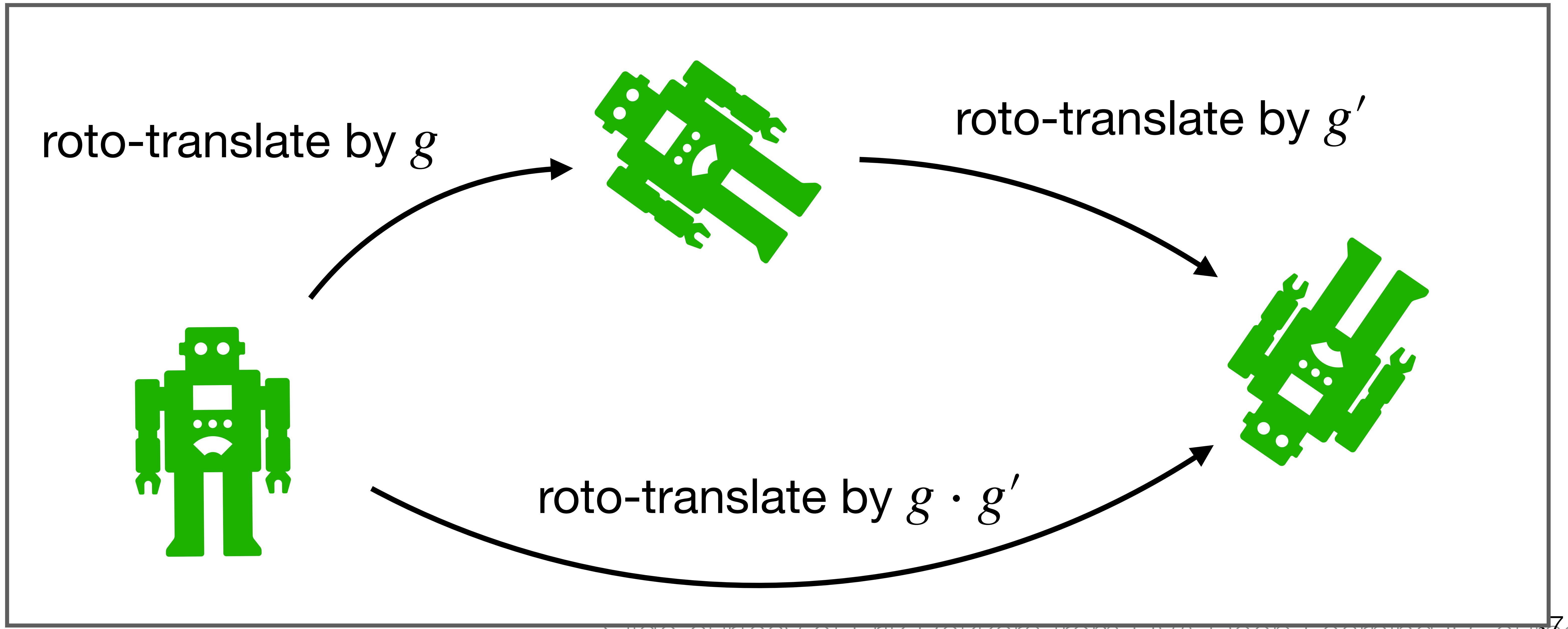
Roto-ranslation group $SE(2)$

*2D **S**pecial **E**uclidean motion group*

The group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$ consists of the coupled space $\mathbb{R}^2 \times S^1$ of translations vectors in $\mathbb{R}^2$, and rotations in $SO(2)$ (or equivalently orientations in S^1), and is equipped with the group product and group inverse:

$$g \cdot g' = (\mathbf{x}, \mathbf{R}_\theta) \cdot (\mathbf{x}', \mathbf{R}_{\theta'}) = (\mathbf{R}_\theta \mathbf{x}' + \mathbf{x}, \mathbf{R}_{\theta+\theta'})$$
$$g^{-1} = (-\mathbf{R}_\theta^{-1}\mathbf{x}, \mathbf{R}_\theta^{-1})$$

with $g = (\mathbf{x}, \mathbf{R}_\theta)$, $g' = (\mathbf{x}', \mathbf{R}_{\theta'})$.



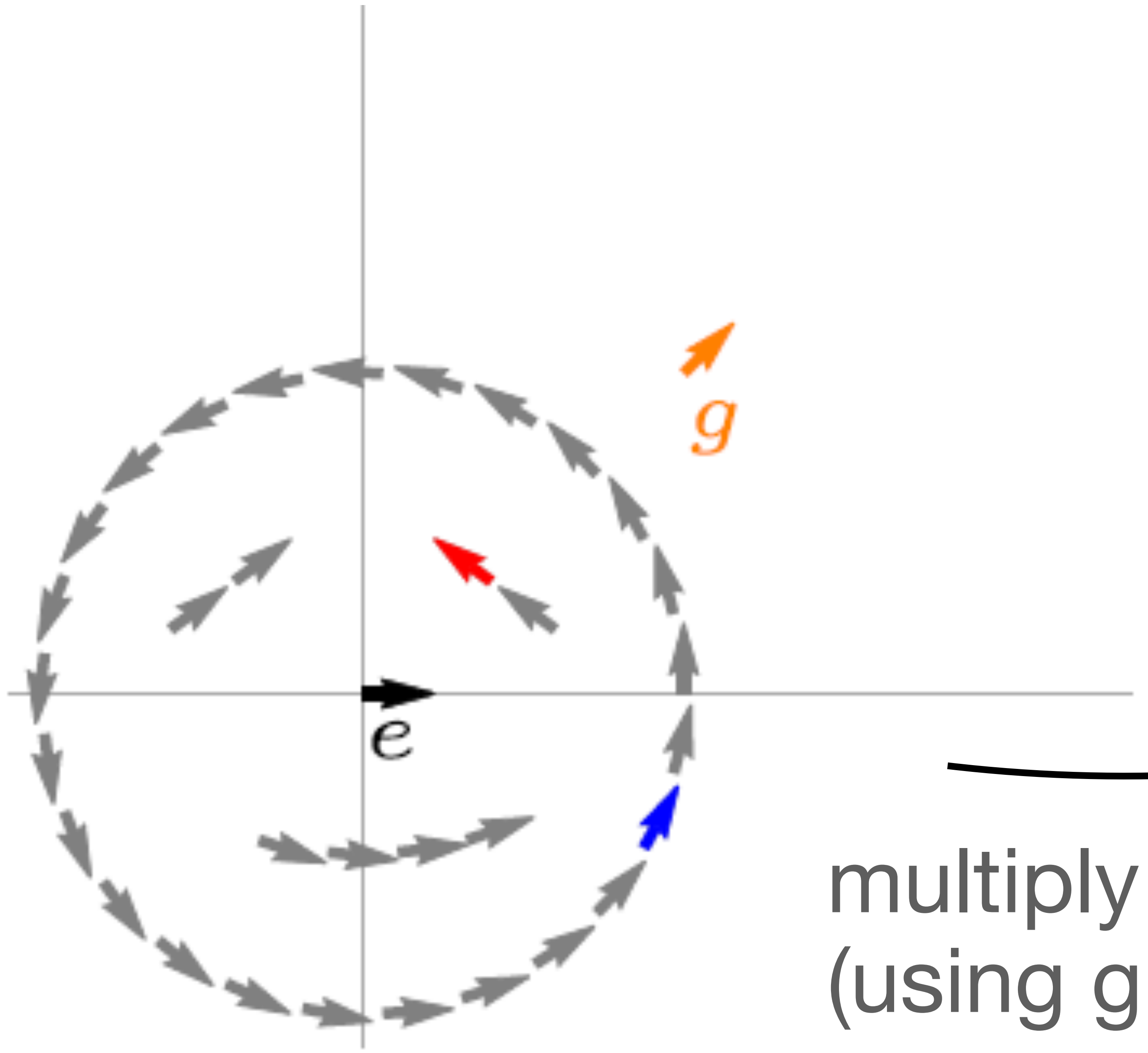
Roto-ranslation group $SE(2)$

*2D **S**pecial **E**uclidean motion group*

The group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$ consists of the coupled space $\mathbb{R}^2 \times S^1$ of translations vectors in $\mathbb{R}^2$, and rotations in $SO(2)$ (or equivalently orientations in S^1), and is equipped with the group product and group inverse:

$$g \cdot g' = (\mathbf{x}, \mathbf{R}_\theta) \cdot (\mathbf{x}', \mathbf{R}_{\theta'}) = (\mathbf{R}_\theta \mathbf{x}' + \mathbf{x}, \mathbf{R}_{\theta+\theta'})$$
$$g^{-1} = (-\mathbf{R}_\theta^{-1}\mathbf{x}, \mathbf{R}_\theta^{-1})$$

with $g = (\mathbf{x}, \mathbf{R}_\theta)$, $g' = (\mathbf{x}', \mathbf{R}_{\theta'})$.



multiply each element with g
(using group prod)

Roto-ranslation group $SE(2)$

*2D **S**pecial **E**uclidean motion group*

Matrix representation: The group can also be represented by **invertible matrices** $\mathbf{G} \in GL(V)$

$$g = (\mathbf{x}, \mathbf{R}_\theta) \quad \leftrightarrow \quad \mathbf{G} = \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_\theta & \mathbf{x} \\ \mathbf{0}^T & 1 \end{pmatrix}$$

with the group product and inverse simply given by the matrix product and matrix inverse.

In parametric form: $(\mathbf{x}, \theta) \cdot (\mathbf{x}', \theta') = (\mathbf{R}_\theta \mathbf{x}' + \mathbf{x},\ \theta + \theta' \bmod 2\pi)$

$$\leftrightarrow$$

In matrix form: $\begin{pmatrix} \mathbf{R}_\theta & \mathbf{x} \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}'_\theta & \mathbf{x}' \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_{\theta+\theta'} & \mathbf{R}_\theta \mathbf{x}' + \mathbf{x} \\ \mathbf{0}^T & 1 \end{pmatrix}$

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Scale-translation group $R^2 \rtimes \mathbb{R}^+$

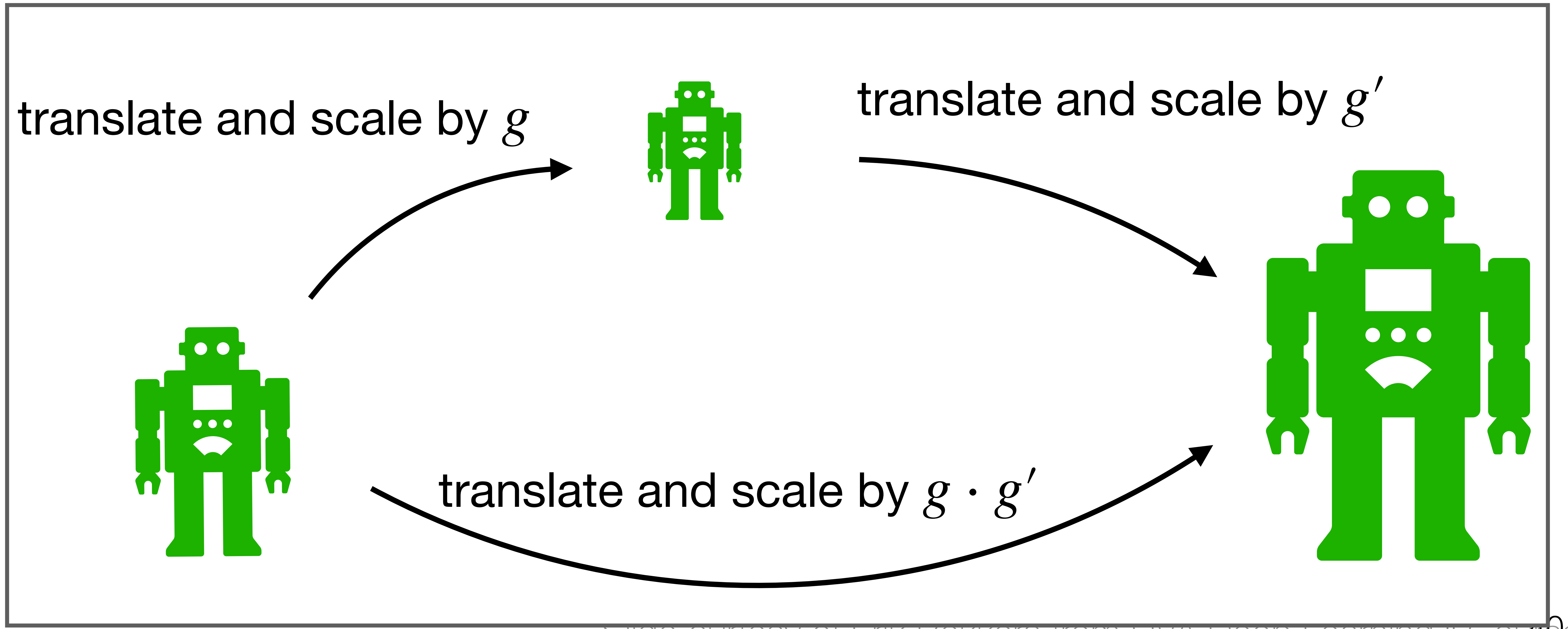
The scale-translation group of space $\mathbb{R}^2 \times \mathbb{R}^+$ of translations vectors in $\mathbb{R}^2$ and scale/dilation factors in $\mathbb{R}^+$, and is equipped with the group product and group inverse:

$$g \cdot g' = (\mathbf{x}, s) \cdot (\mathbf{x}', s') = (s\mathbf{x}' + \mathbf{x}, ss')$$
$$g^{-1} = \left(-\frac{1}{s}\mathbf{x}, \frac{1}{s}\right)$$

with $g = (\mathbf{x}, s), g' = (\mathbf{x}', s')$.

with $g \cdot g^{-1} = e = (\mathbf{0},1)$

matrix repr: $\mathbf{G} = \begin{pmatrix} \mathbf{I}_s & \mathbf{x} \\ \mathbf{0}^T & 1 \end{pmatrix}$



So… How to translate this to CNNs?

Set of points (group elements)

Convolution kernel

Groups are separate from the vector spaces on which they act. In this case, we have a group (translation) that can do something to $\mathbb{R}^2$. And we have functions over $\mathbb{R}^2$.

$\{g_1, g_2,$

ace of

ctions on $\mathbb{R}^2$

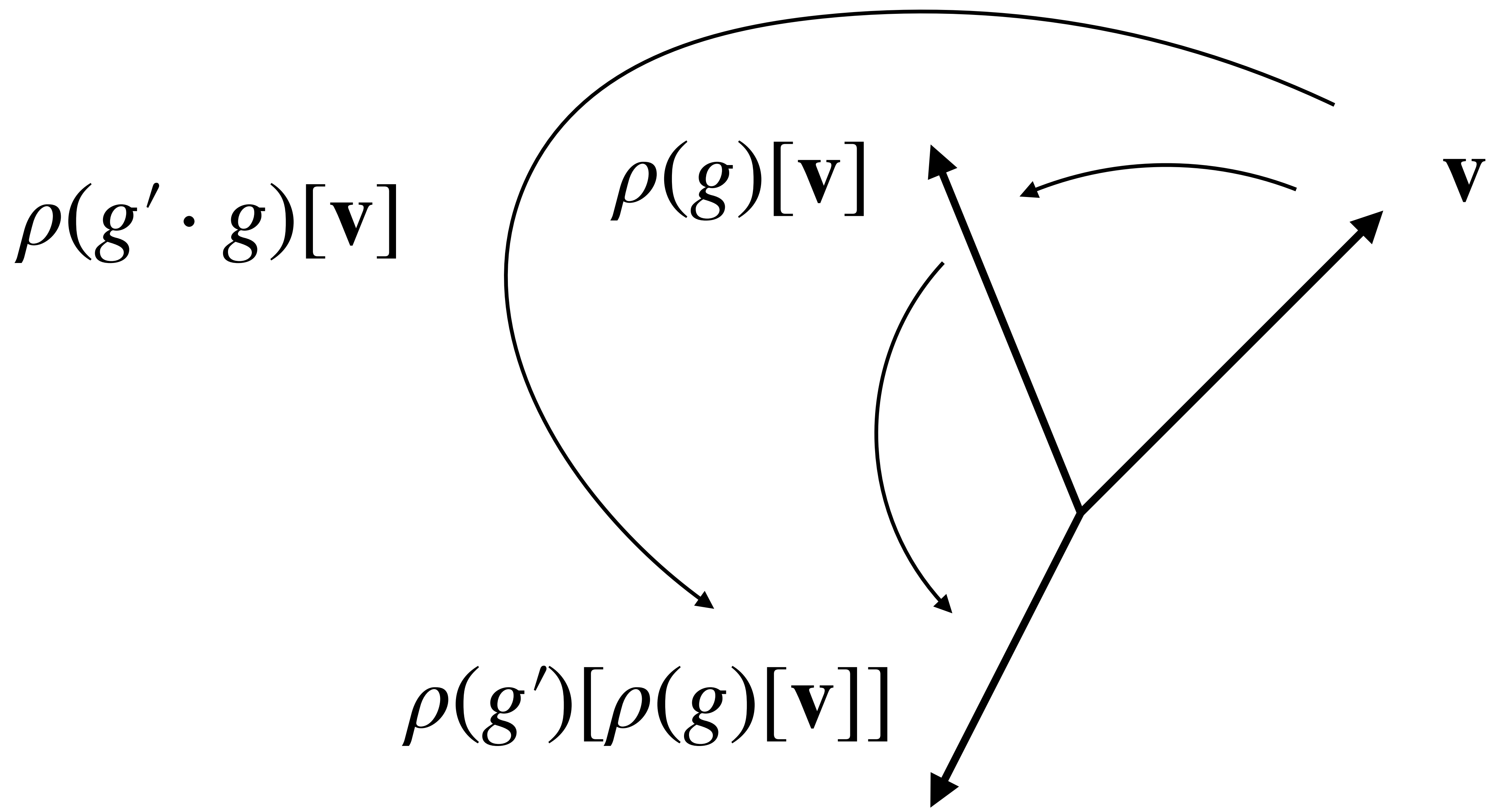
"A collection of parts in certain poses"

Assigning weights to relative poses"

Transforms via group product

Transforms via group representations

Representations



A representation $\rho : G \rightarrow GL(V)$ is a group homomorphism from G to the general linear group $GL(V)$ (the space of invertible matrices!)

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 curtesy of Erik Bekkers from UVA Deep Learning II Course

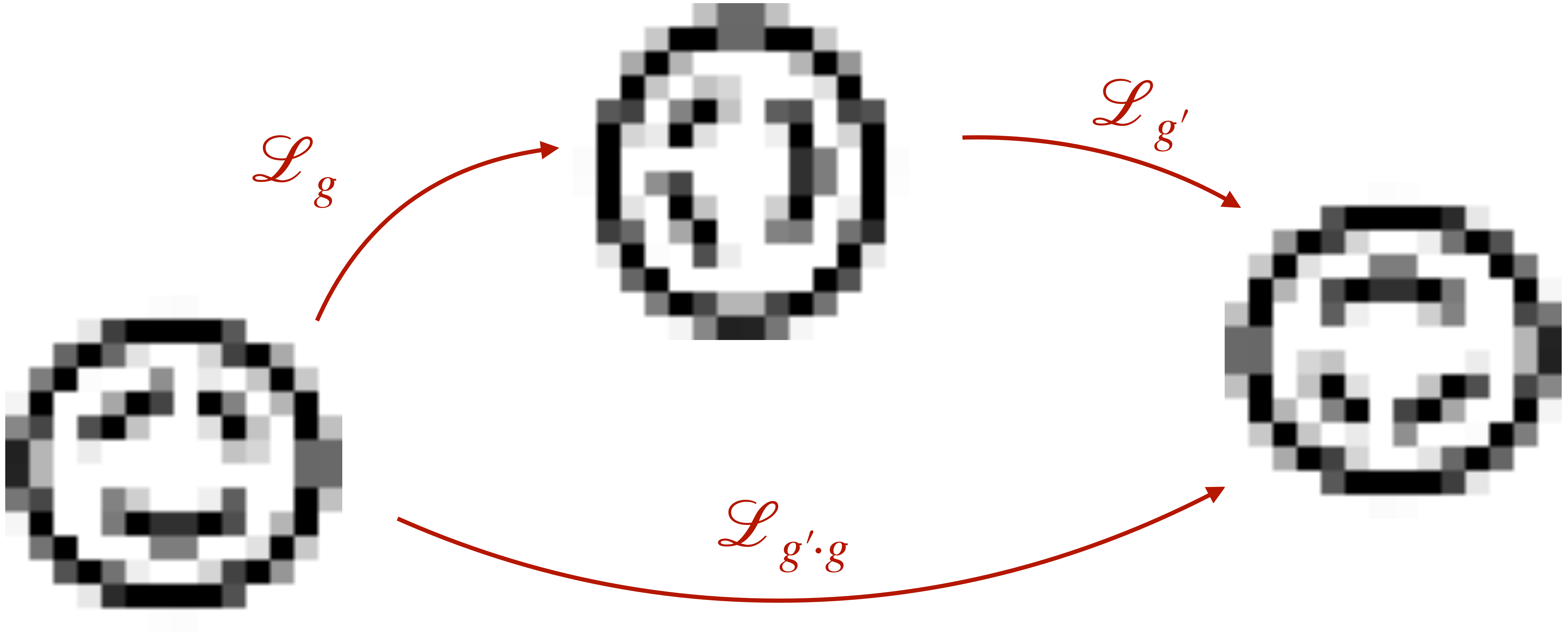
Left-regular Representations

Example:

$f \in \mathbb{L}_2(\mathbb{R}^2)$
- a 2D image

$G = SE(2)$
- the roto-translation group

$\mathscr{L}_g(f)(\mathbf{y}) = f(\mathbf{R}_\theta^{-1}(\mathbf{y} - \mathbf{x}))$
- a roto-translation of the image



A left-regular representation $\mathscr{L}_g$ is a representation that transforms functions f by transforming their domains via the inverse group action

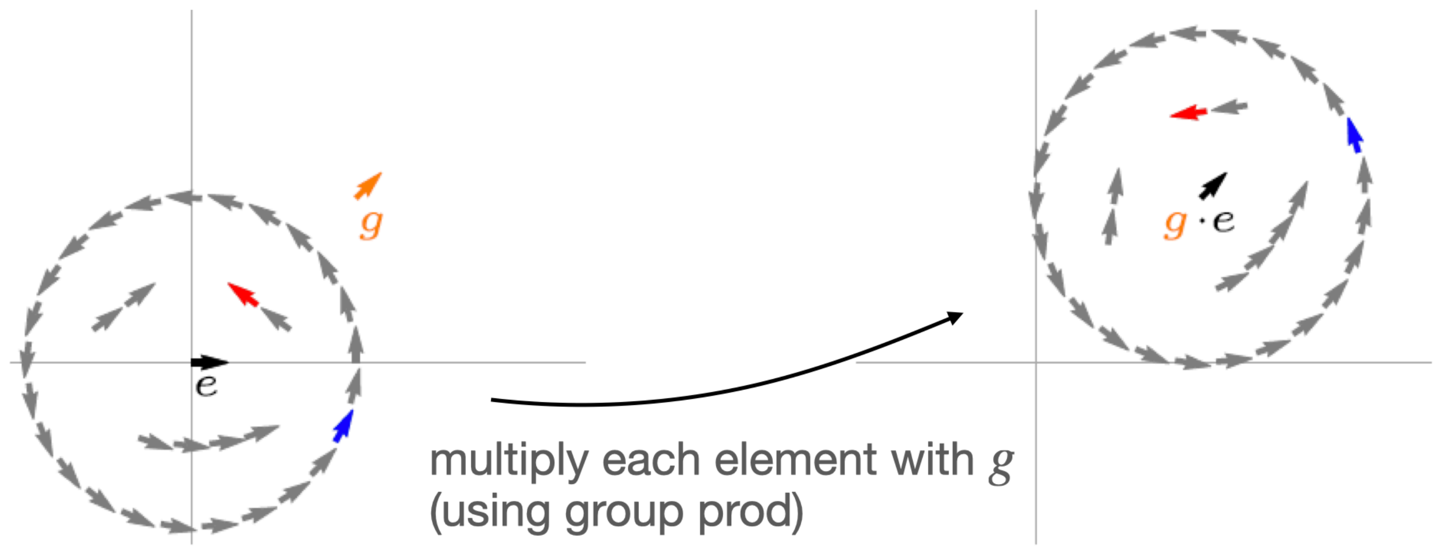
$$\mathscr{L}_g[f](x) := f(g^{-1} \cdot x)$$

"group action" equals group product when domain is G

Group actions

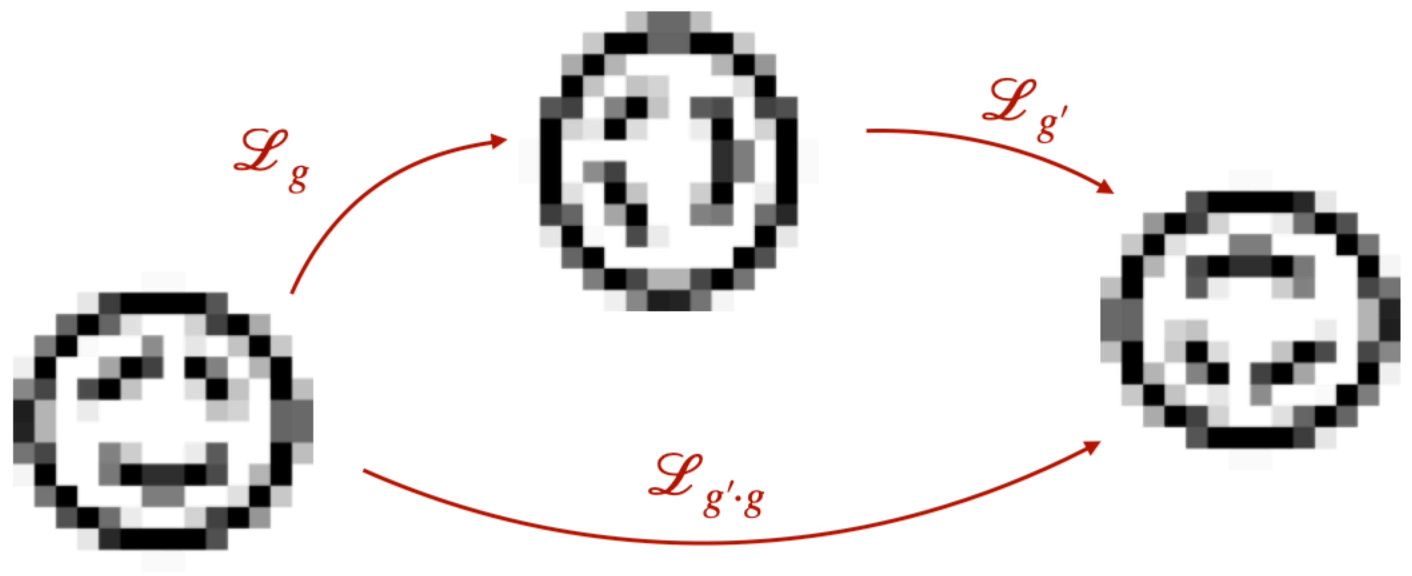
Group product (the action on G)

$g \cdot g'$ $\qquad$ gg'



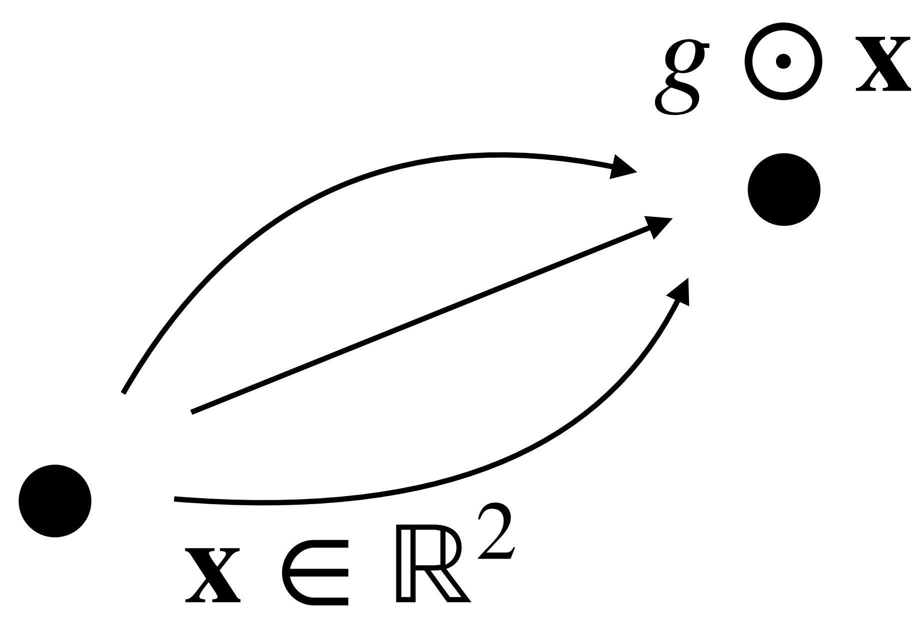
Left regular representation (the action on $\mathbb{L}_2(X)$)

$\mathscr{L}_g f$ $\qquad$ gf



Group action (the action on $\mathbb{R}^d$)

$g \odot \mathbf{x}$ $\qquad$ $g\mathbf{x}$

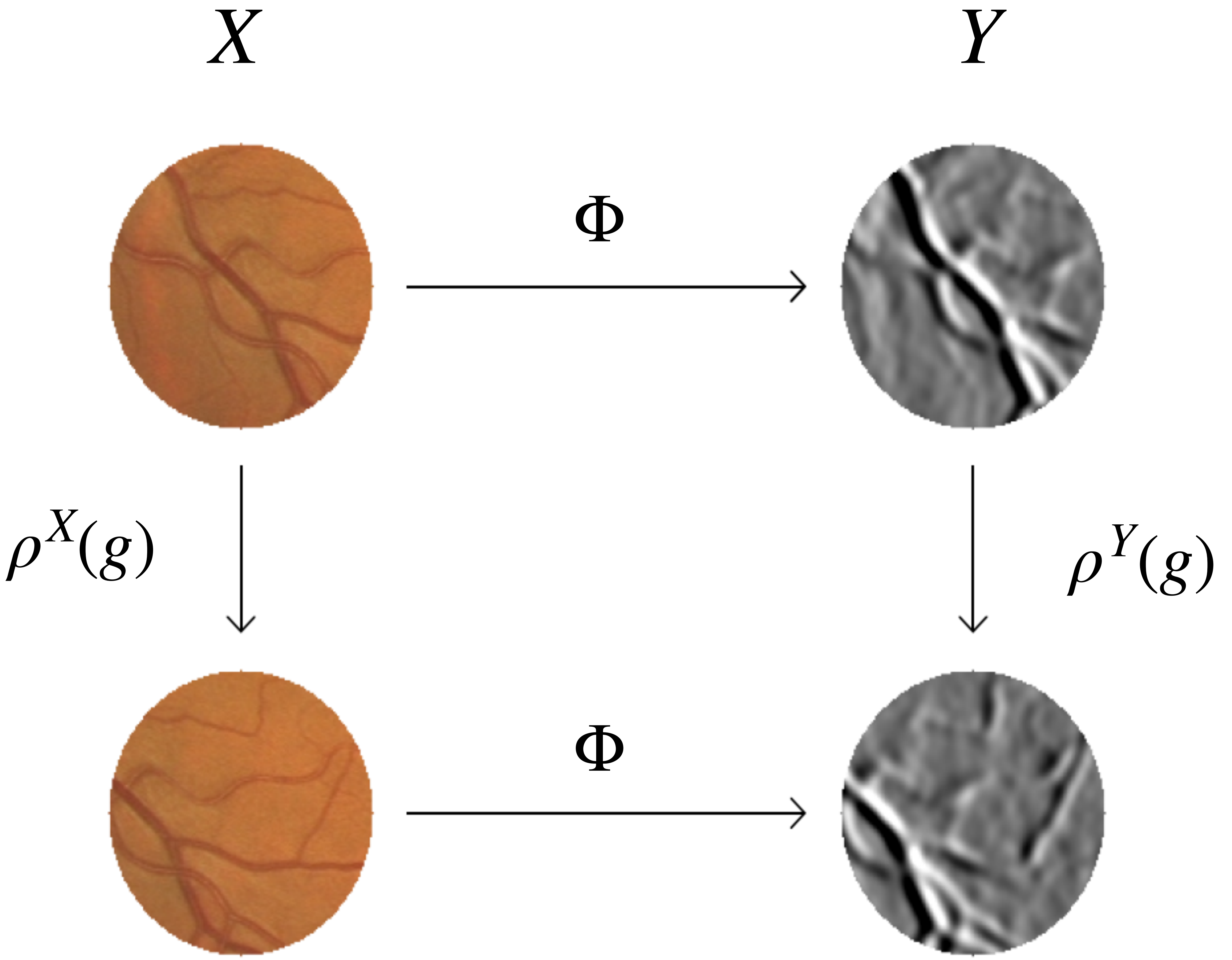


Equivariance

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

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

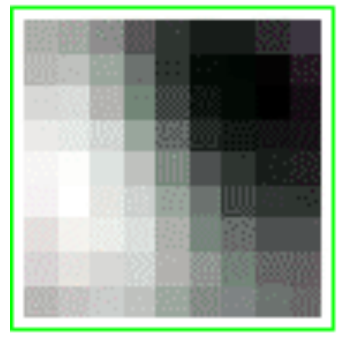
group representation action on X



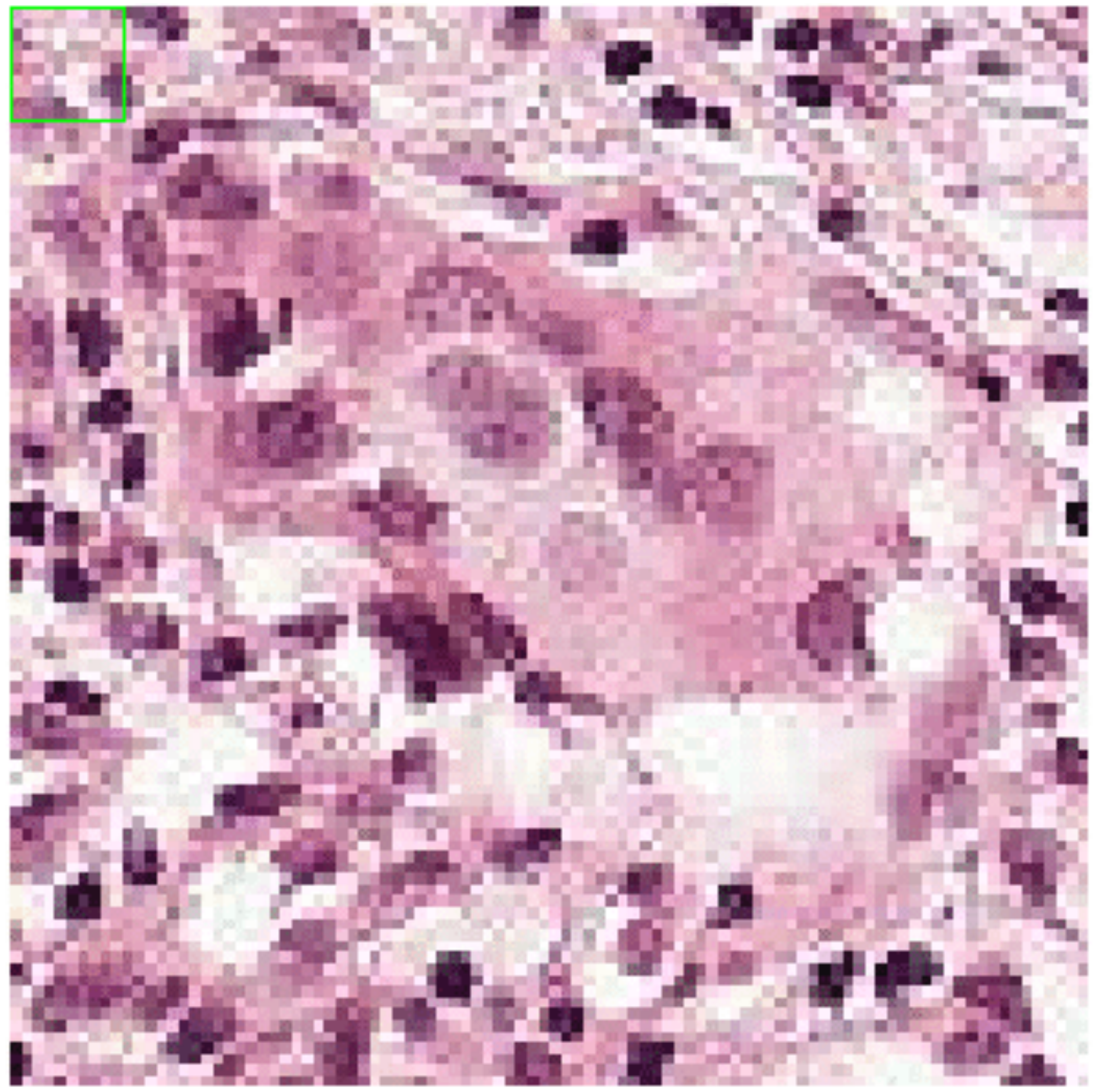
Are convolutions with reflected conv kernels (and vice versa)

Cross-correlations

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



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



$=$

k
2D convolution kernel

f^{in}
2D feature map

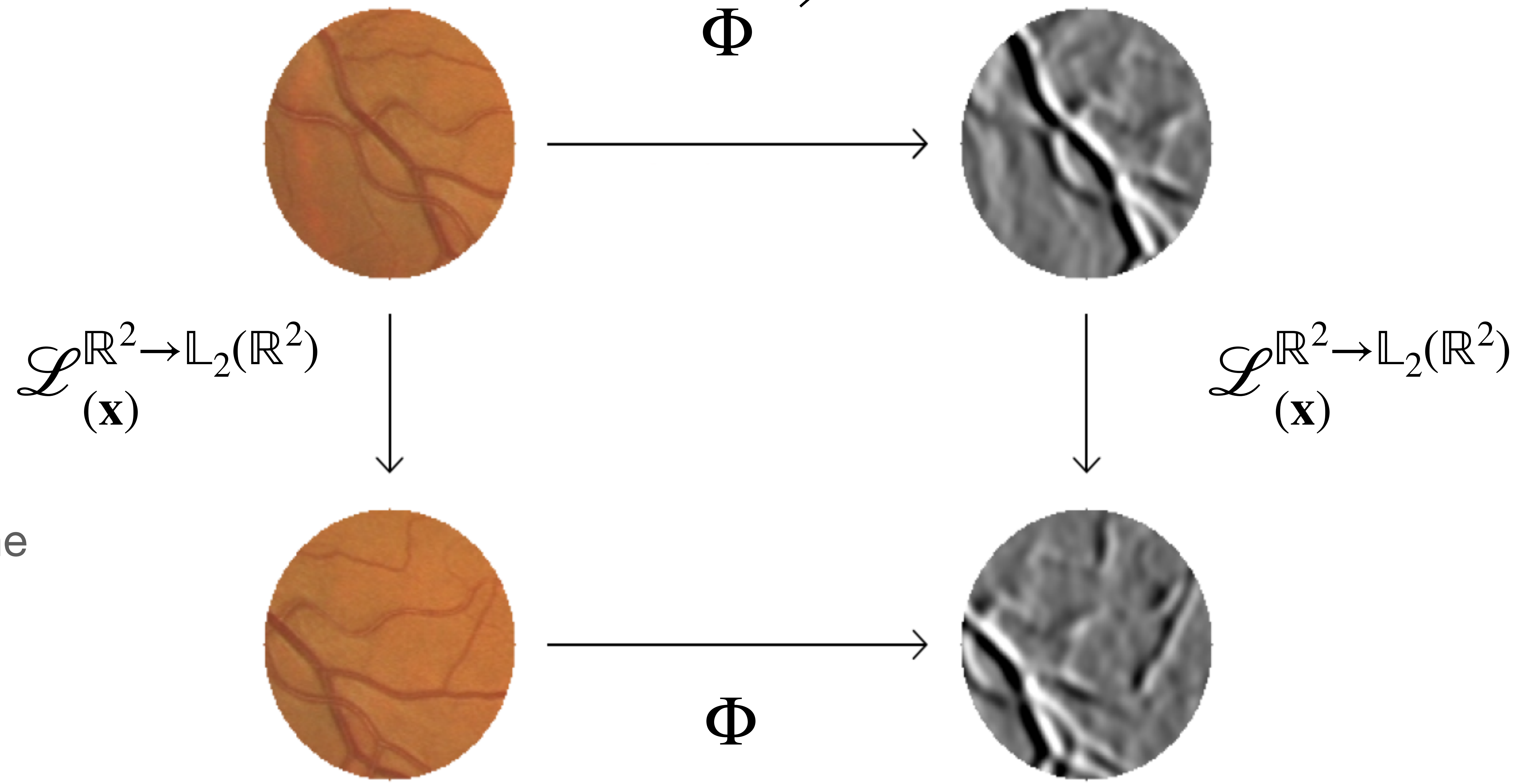
f^{out}
2D feature map (after ReLU)

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Equivariance

Convolutions/cross-correlations are translation equivariant

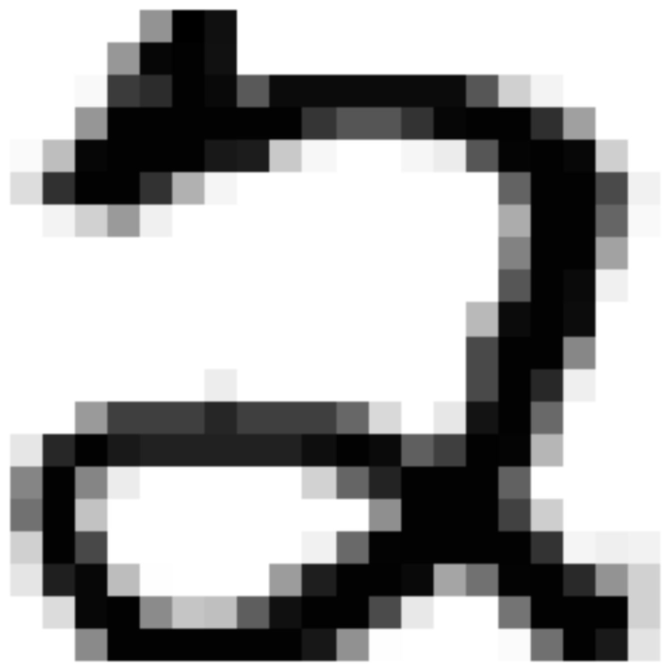
$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = (\mathscr{L}_{(\mathbf{x})}^{\mathbb{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)} k\ ,\ f)_{\mathbb{L}_2(\mathbb{R}^2)}$$



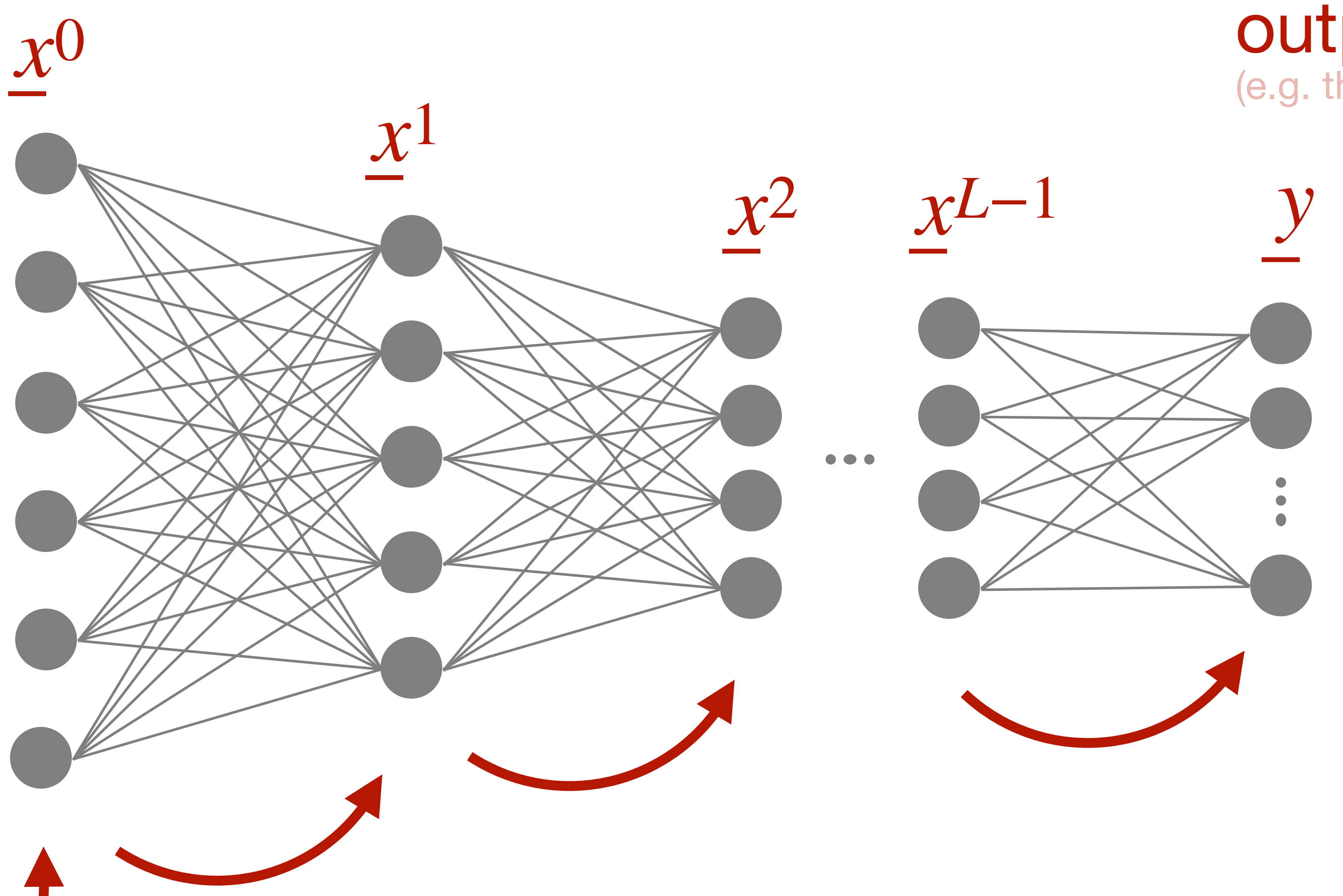
Representation of the translation group

MLPs

What's my input? $\underline{x}^0 \in \mathcal{X} = ?$



input vector



output probability vector

(e.g. through soft max)

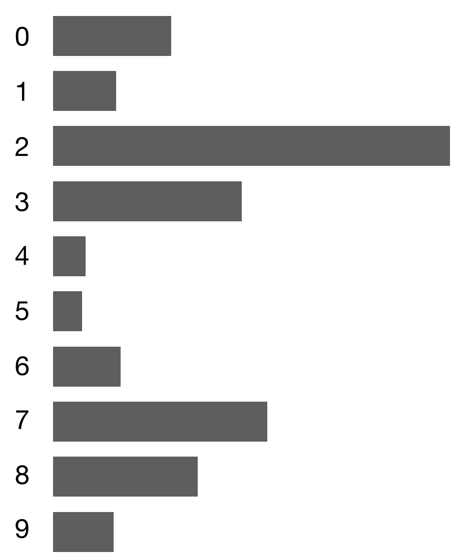


Image analyst: $\underline{x}^0 \in \mathcal{X} = \mathbb{L}_2(\mathbb{R}^2)$

Naive deep learner: $\underline{x}^0 \in \mathcal{X} = \mathbb{R}^{784}$

Iteratively transform the vector via

$$\underline{x}^l = \varphi(K_{\mathbf{w}_l}\underline{x}^{l-1} + \underline{b}^l)$$

Linear map: matrix-vector multiplication with $K_{\mathbf{w}_l} \in \mathbb{R}^{N^l \times N^{l-1}}$

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Convolution: special case of fully connected layer

- Way too many degrees of freedom!
- Does not leverage/preserve structure in data

$$\underline{y} \quad = \varphi(\qquad {\color{red} K_{\mathbf{w}}} \qquad \underline{x} \quad + \quad \underline{b} \quad)$$

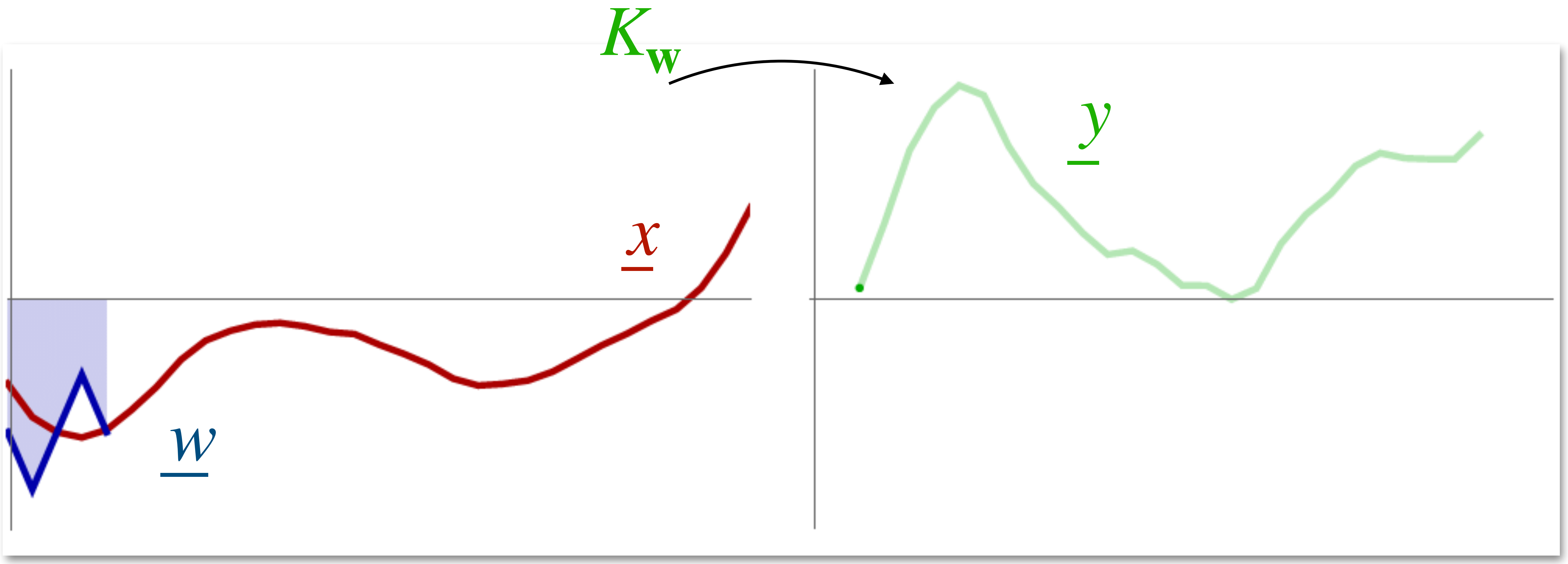
$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{pmatrix} = \varphi \left(\begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} & \cdots \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} & \cdots \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \end{pmatrix} \right)$$

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Convolution: special case of fully connected layer

Convolution as linear operator

+ Localized transformations
+ Shift equivariance
+ Sparsification of the linear operator
+ Weightsharing



$$\underline{y} = \varphi(K_{\mathbf{w}} \underline{x} + \underline{b})$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{pmatrix} = \varphi \left(\begin{pmatrix} w_1 & w_2 & w_3 & 0 & 0 & \dots \\ 0 & w_1 & w_2 & w_3 & 0 & \dots \\ 0 & 0 & w_1 & w_2 & w_3 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \end{pmatrix} \right)$$

Equivariance

Convolutions are generally **not equivariant to roto-translations**

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

Representation of the translation group

$\xrightarrow{\Phi}$

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

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

Representation of the rotation group

$\xrightarrow{\Phi}$

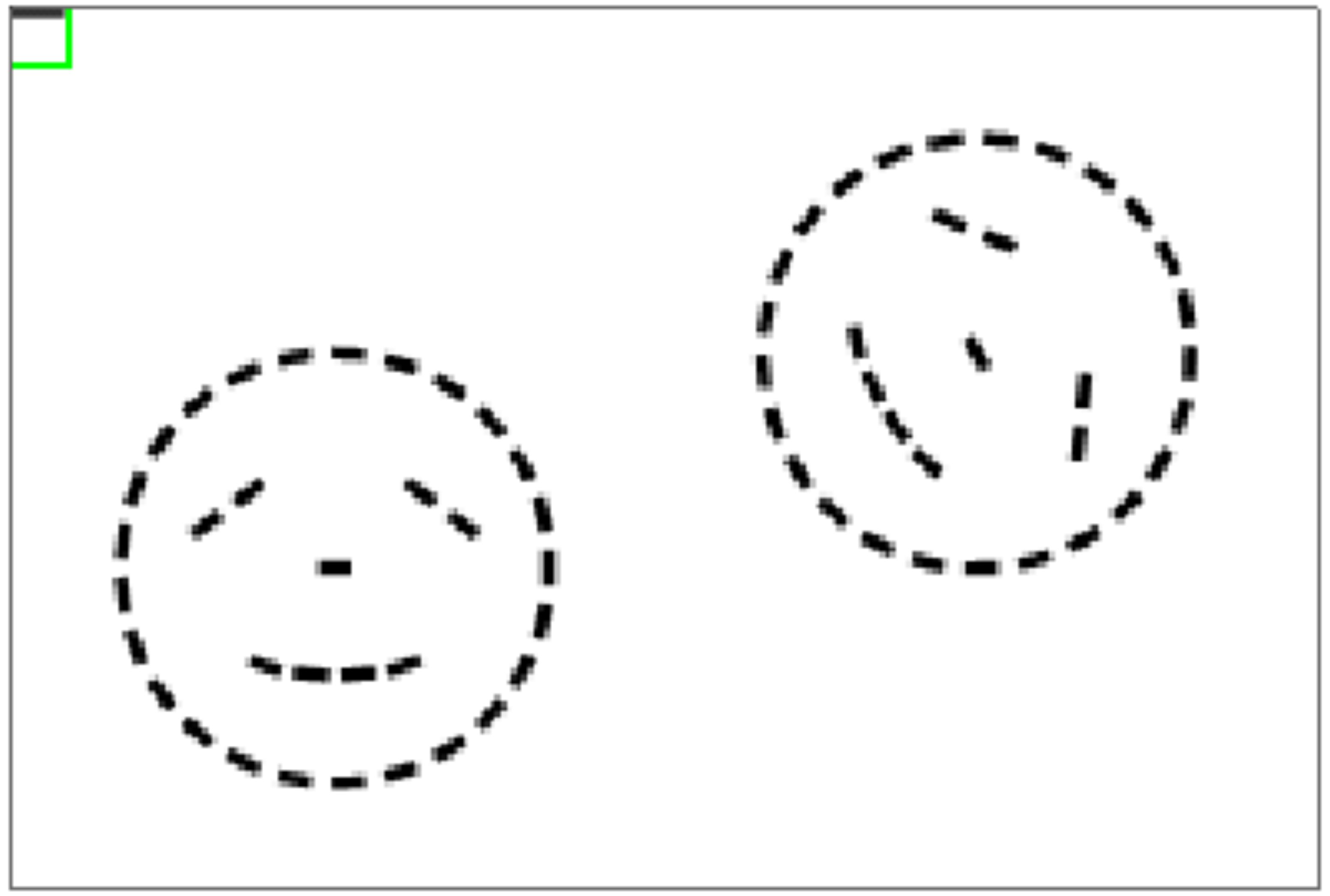
SE(2) equivariant cross-correlations

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

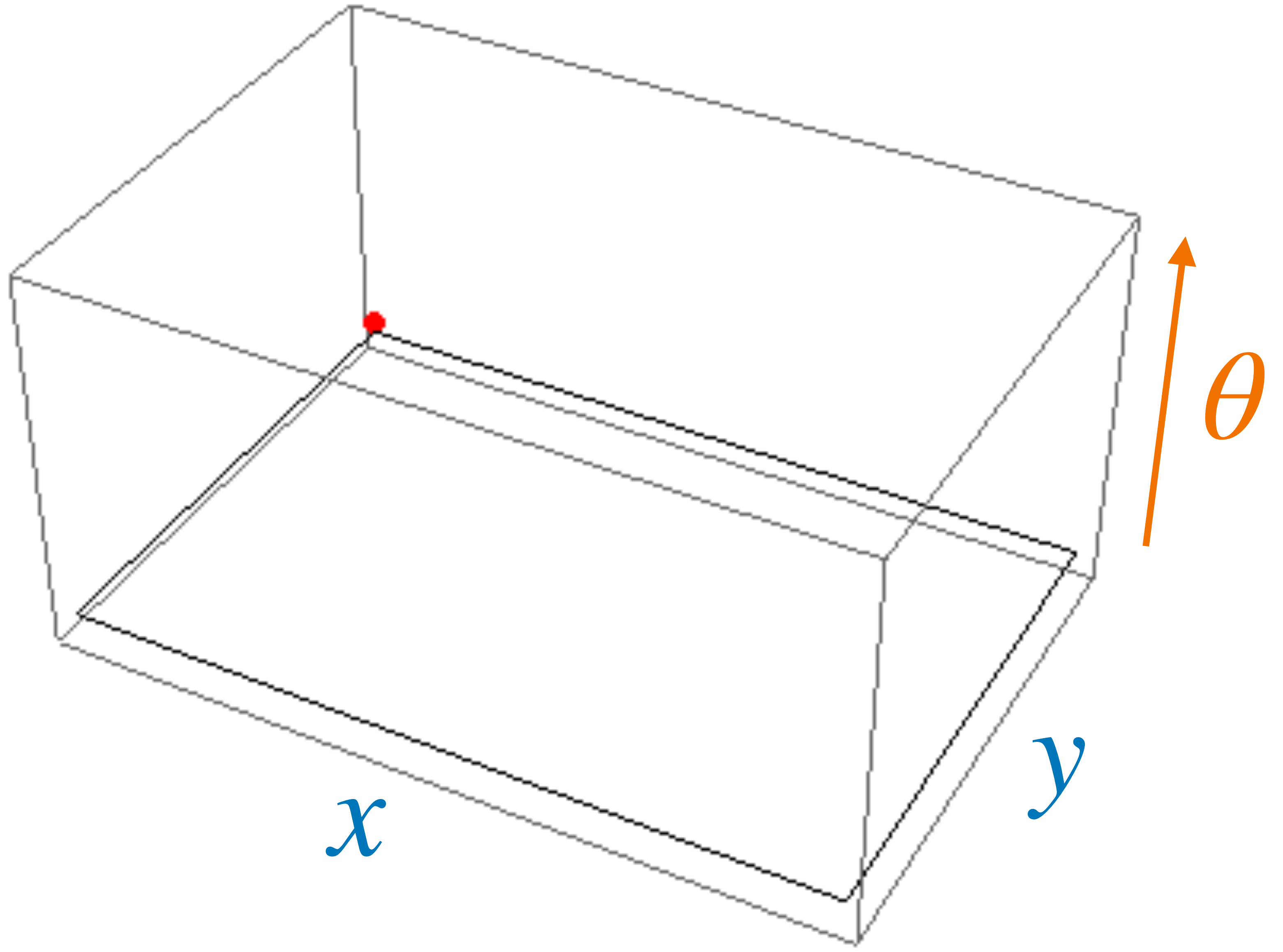
Representation of the roto-translation group!

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

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



$=$



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

Rotated 2D convolution kernel

f^{in}

2D feature map

f^{out}

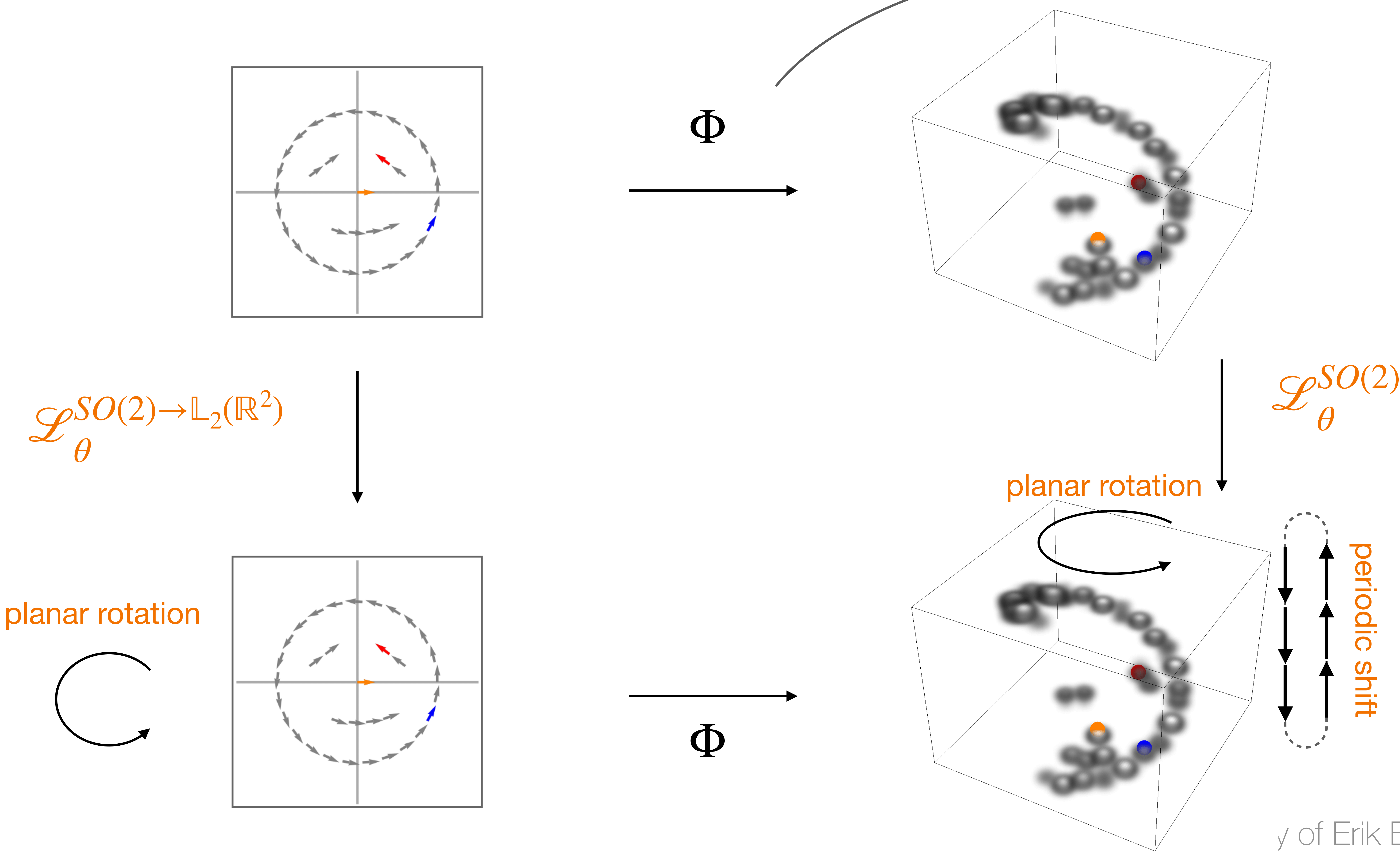
3D (*SE(2)*) feature map (after ReLU)

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Equivariance

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

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



What about subsequent layers?

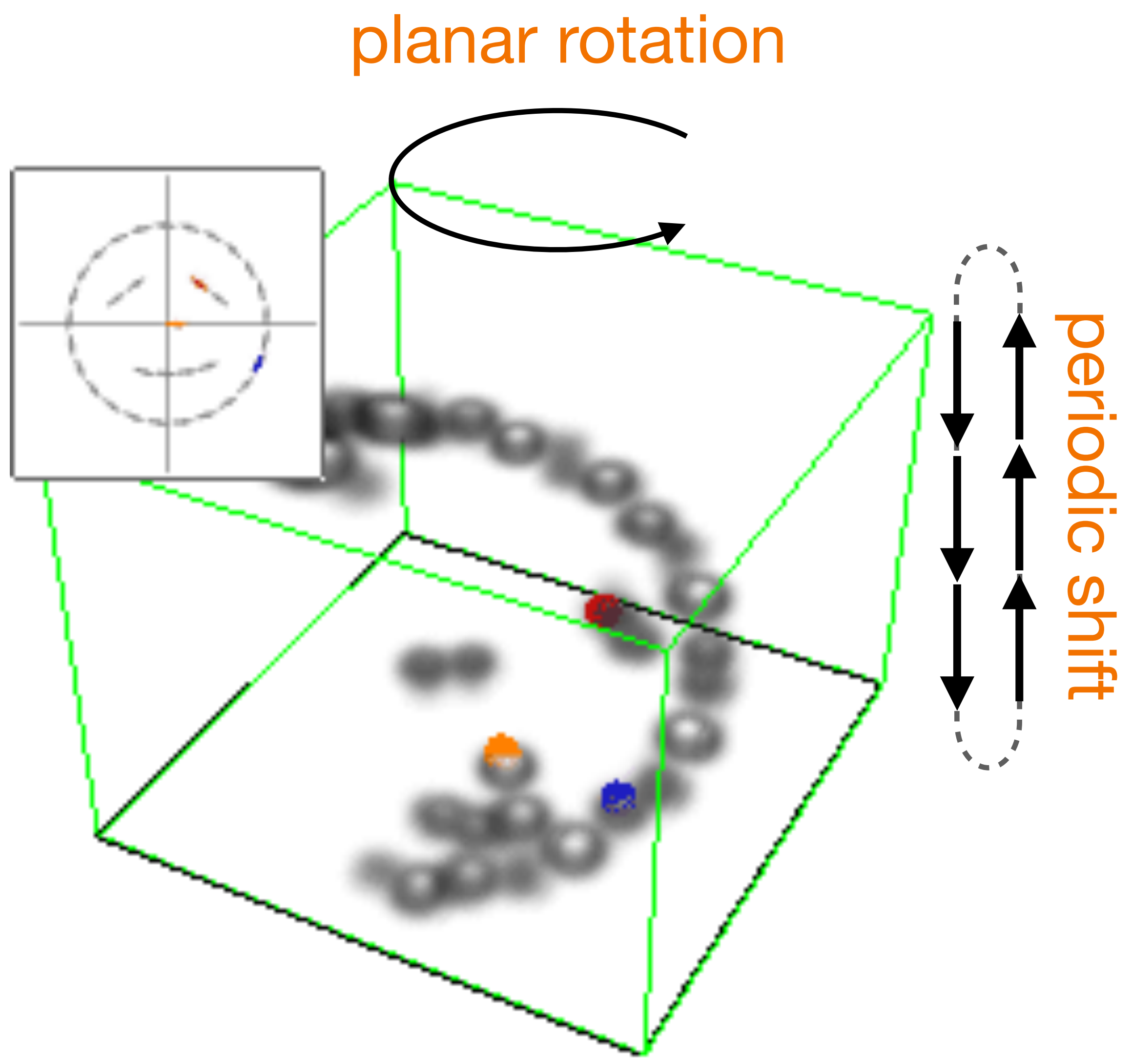
SE(2) equivariant cross-correlations

Group correlations:

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

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

translation *rotation*



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

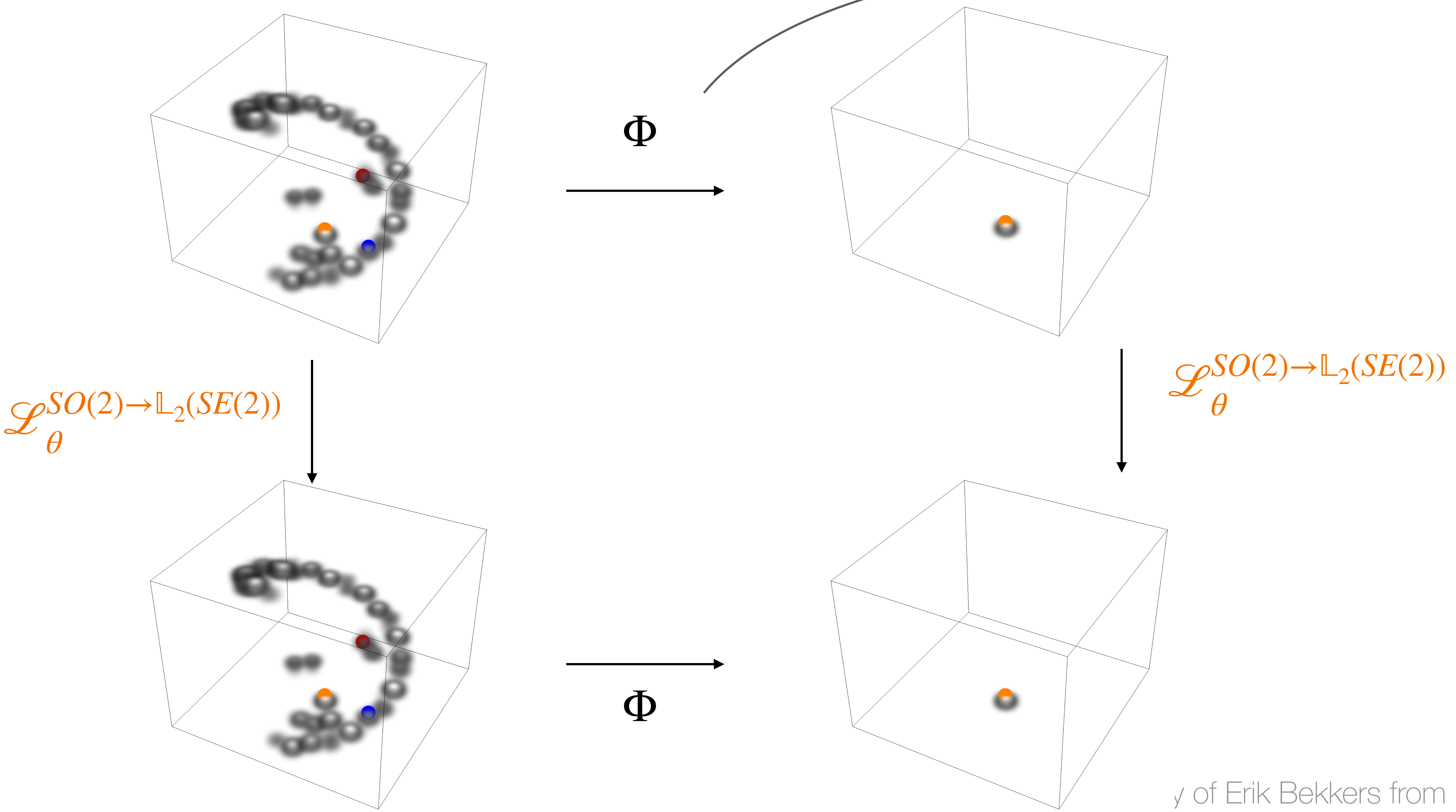
Rotated SE(2) convolution kernel

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

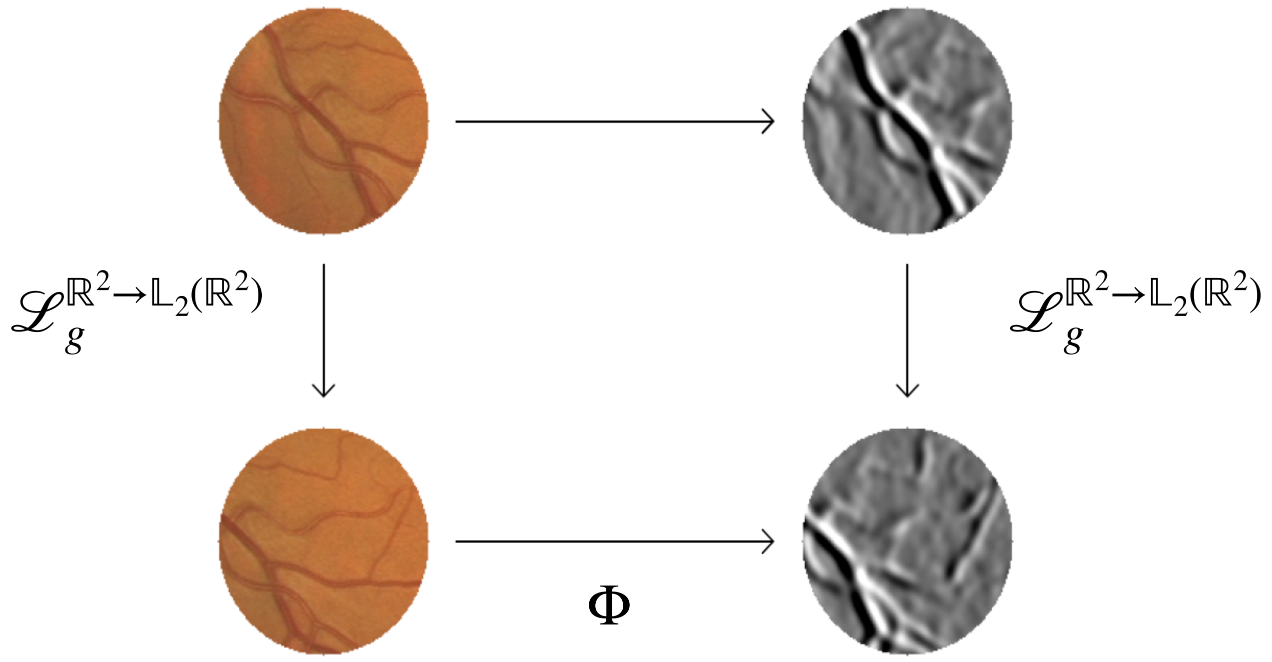
Equivariance

SE(2) group convolutions are roto-translation equivariant

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



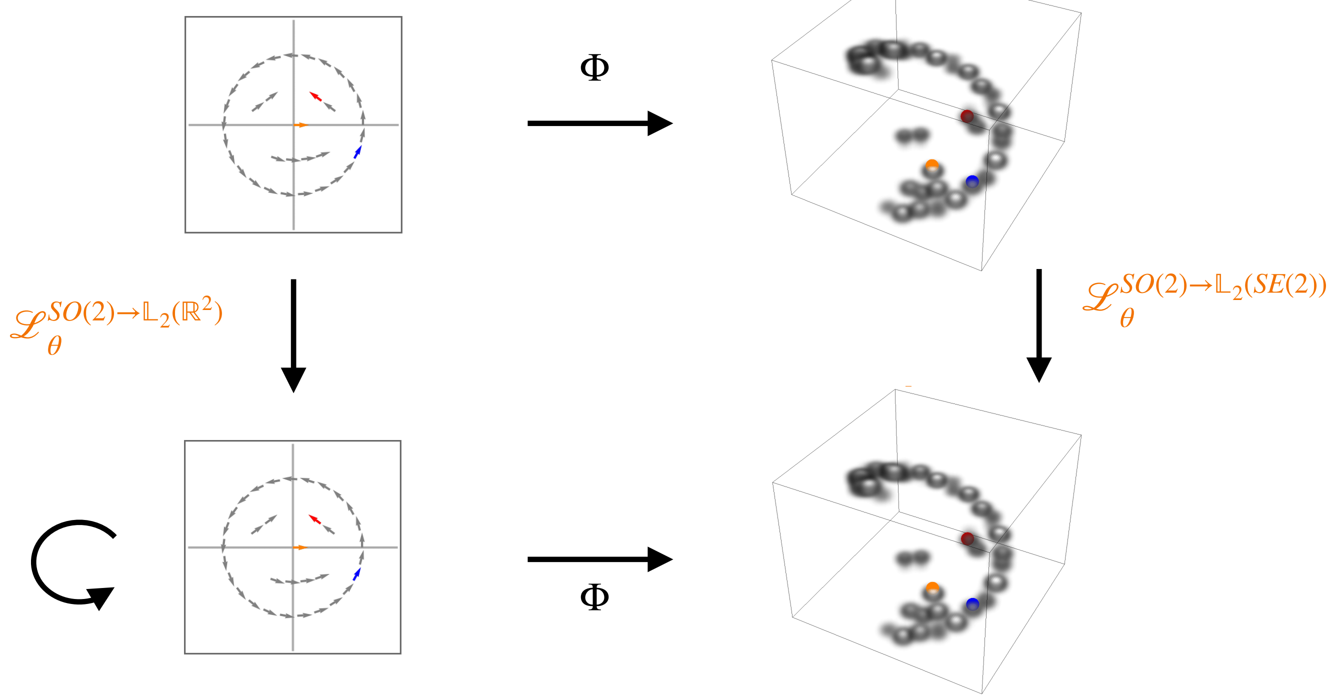
2D cross-correlation (translation equivariant)



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

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

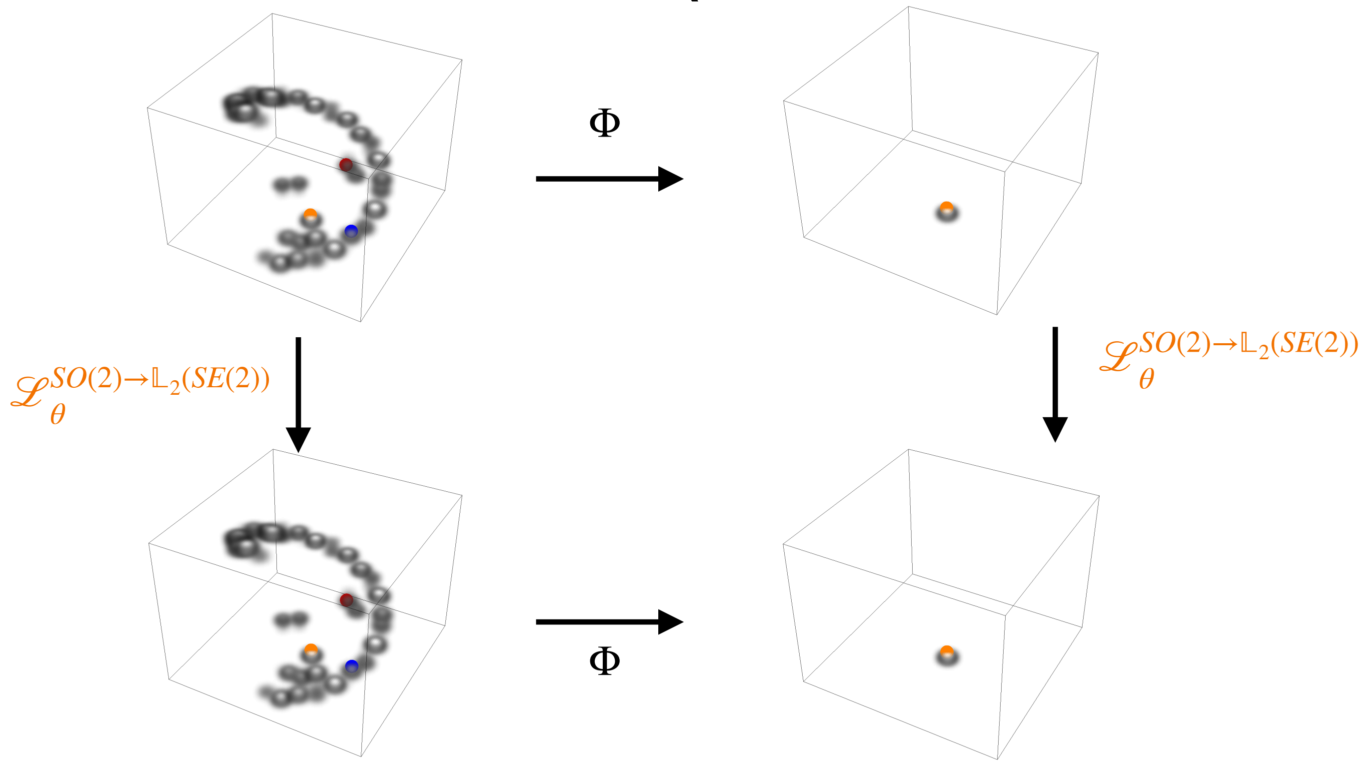
SE(2) lifting correlations (roto-translation equivariant)



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

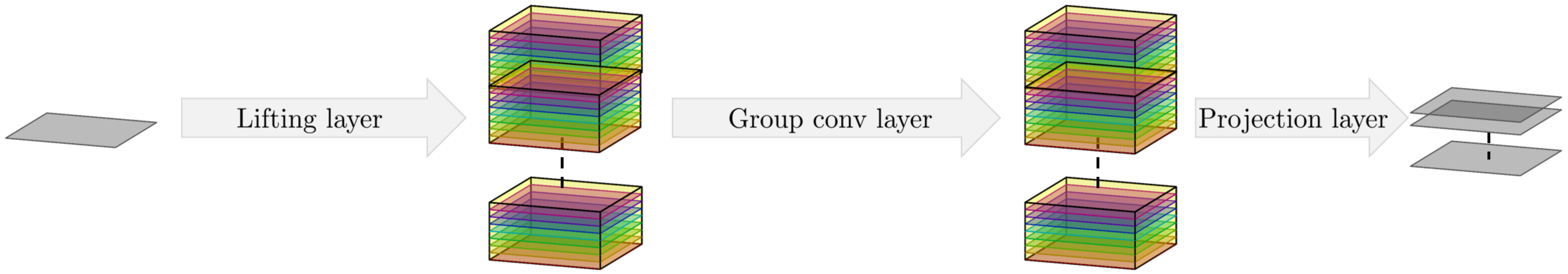
$$= \int_{\mathbb{R}^2} k(\mathbf{R}_\theta^{-1}(\mathbf{x}' - \mathbf{x})) f(\mathbf{x}') d\mathbf{x}'$$

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



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

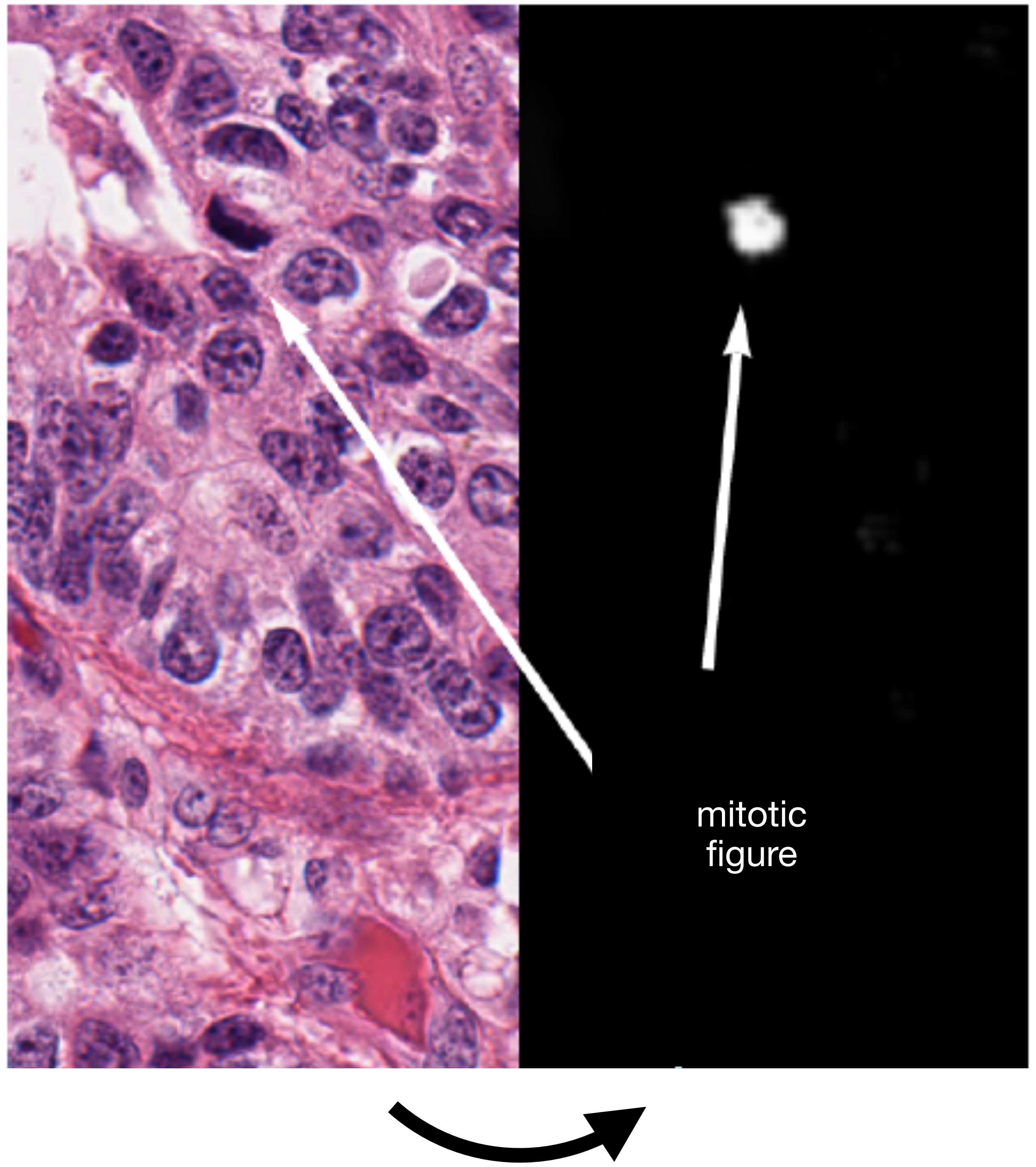
$$= \int_{\mathbb{R}^2} \int_{S^1} k(\mathbf{R}_\theta^{-1}(\mathbf{x}' - \mathbf{x}), \theta' - \theta \mod 2\pi) f(\mathbf{x}', \theta') d\mathbf{x}' d\theta'$$



Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Aren’t there other kernels with this property?

Neural Networks for Signal Data



$$\mathcal{K} : \mathbb{L}_2(X)^{N_l} \rightarrow \mathbb{L}_2(Y)^{N_{l+1}}$$

Let's build neural networks for signal data via the layers of the form:

$$\underline{f}^{l+1} = \sigma(\mathcal{K}\,\underline{f}^{l} + \mathbf{b}^{l})$$

The linear map has to be an integral transform with a two-argument kernel
(Dunford-Pettis theorem)

$$(\mathcal{K}f)(y) = \int_X \mathbf{k}(y,x)\underline{f}(x)\mathrm{d}x$$

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

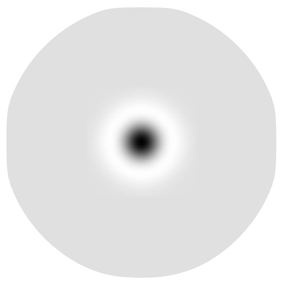
Types of layers $\quad K : \mathbb{L}_2(\mathbb{R}^2) \rightarrow \mathbb{L}_2(\mathbb{R}^2)$

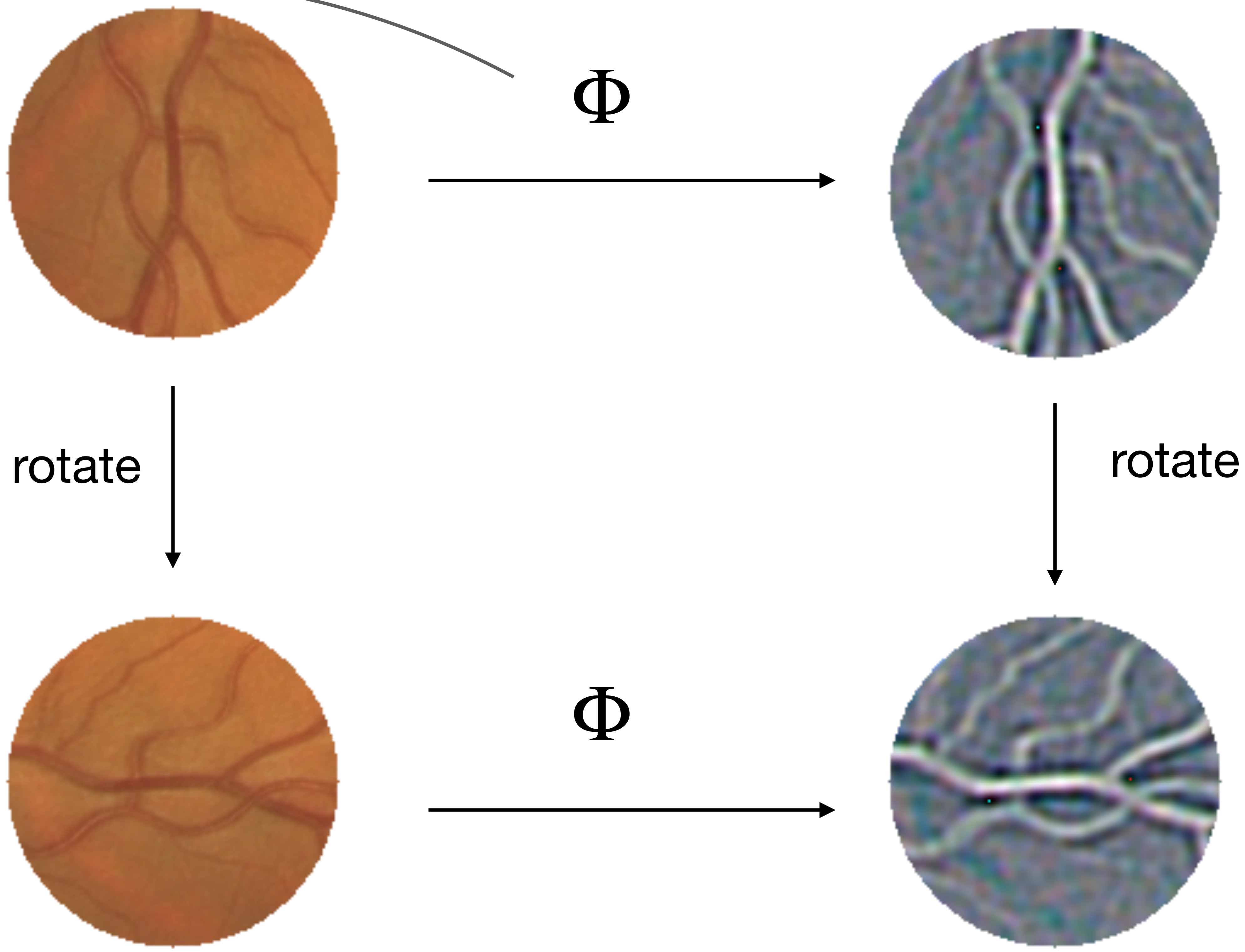
$(X = Y = G/H)$

Isotropic/Constraint convolutions on spaces of lower dimension than G, $\forall_{h \in H} : k(hx) = k(x)$

Example 2D CNN

$X = \mathbb{R}^2 \equiv SE(2)/SO(2)$

Conv2D(input , )

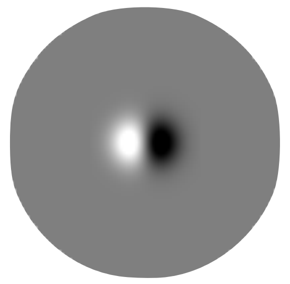


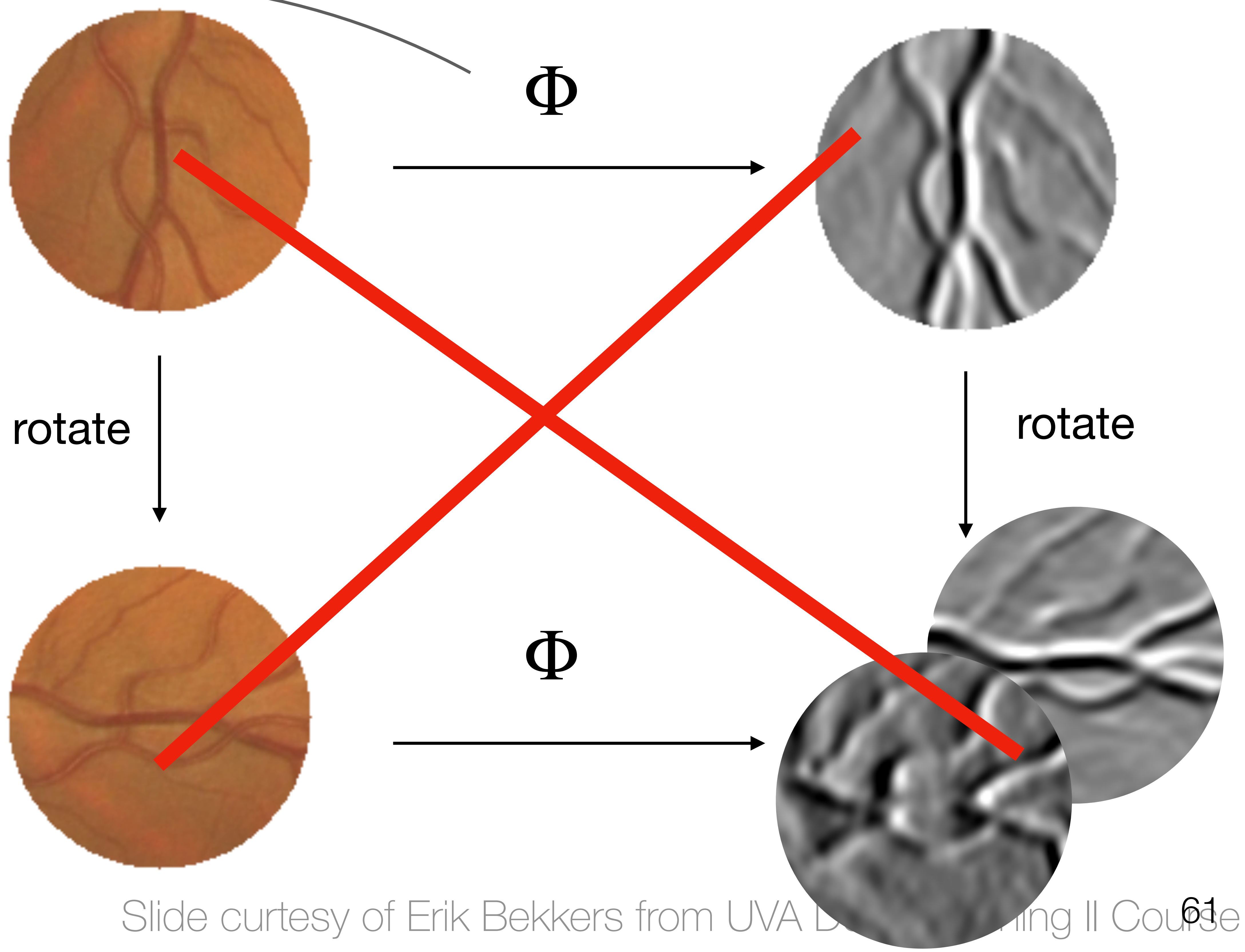
Types of layers $K : \mathbb{L}_2(\mathbb{R}^2) \rightarrow \mathbb{L}_2(\mathbb{R}^2)$

$(X = Y = G/H)$

~~**Isotropic/Constraint convolutions**~~ on spaces of lower dimension ~~than G, $\forall_{h \in H} : k(hx) = k(x)$~~

Example 2D CNN
$X = \mathbb{R}^2 \equiv SE(2)/SO(2)$

Conv2D(input , )



Types of layers $K : \mathbb{L}_2(\mathbb{R}^2) \rightarrow \mathbb{L}_2(SE(2))$

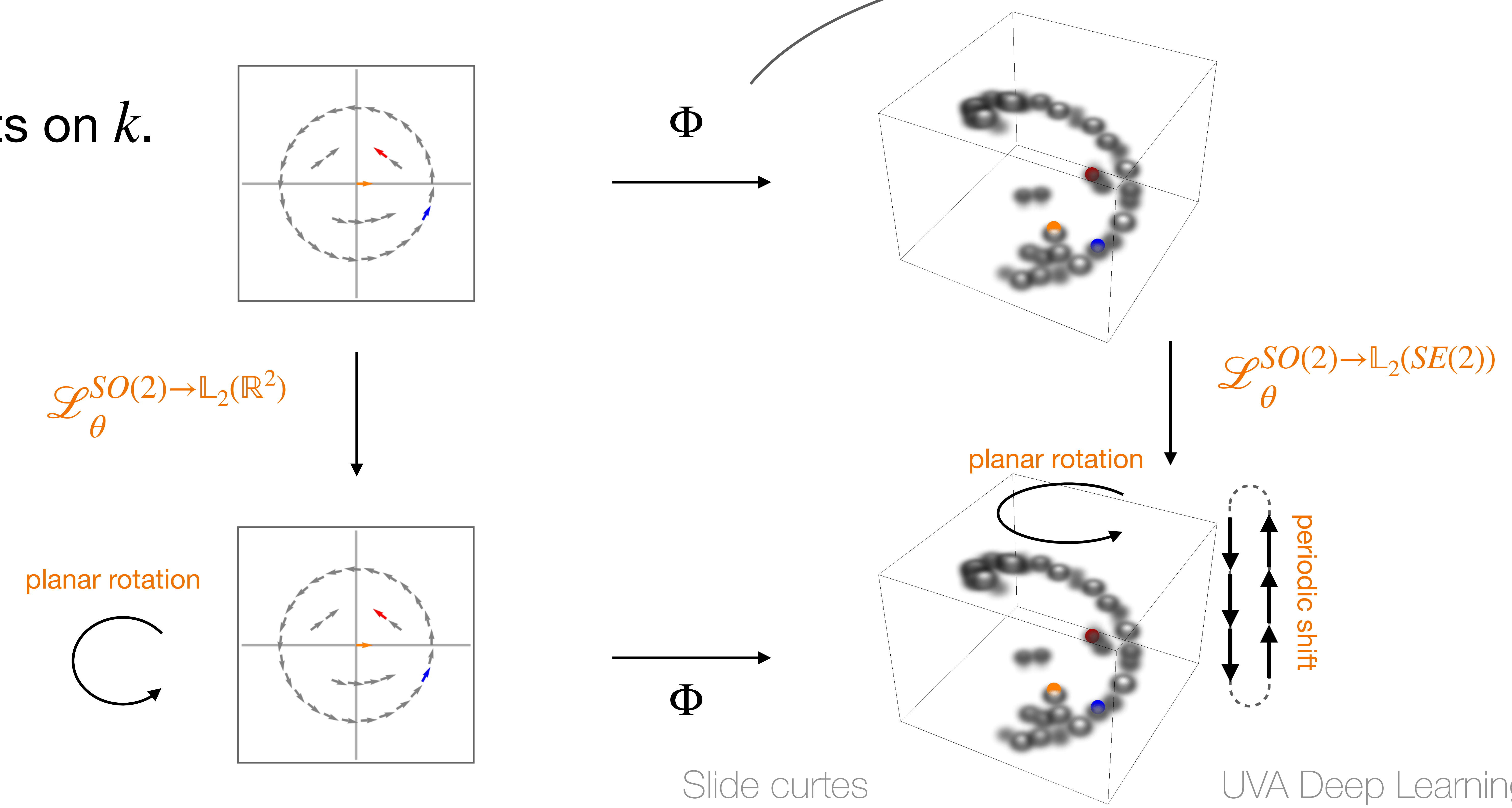
$(X = Y = G/H)$

Isotropic/Constraint convolutions on spaces of lower dimension than G, $\forall_{h \in H} : k(hx) = k(x)$

$(X = G/H, Y = G)$

Lifting convolution. No constraints on k.

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



Types of layers $K : \mathbb{L}_2(SE(2)) \rightarrow \mathbb{L}_2(SE(2))$

$(X = Y = G/H)$

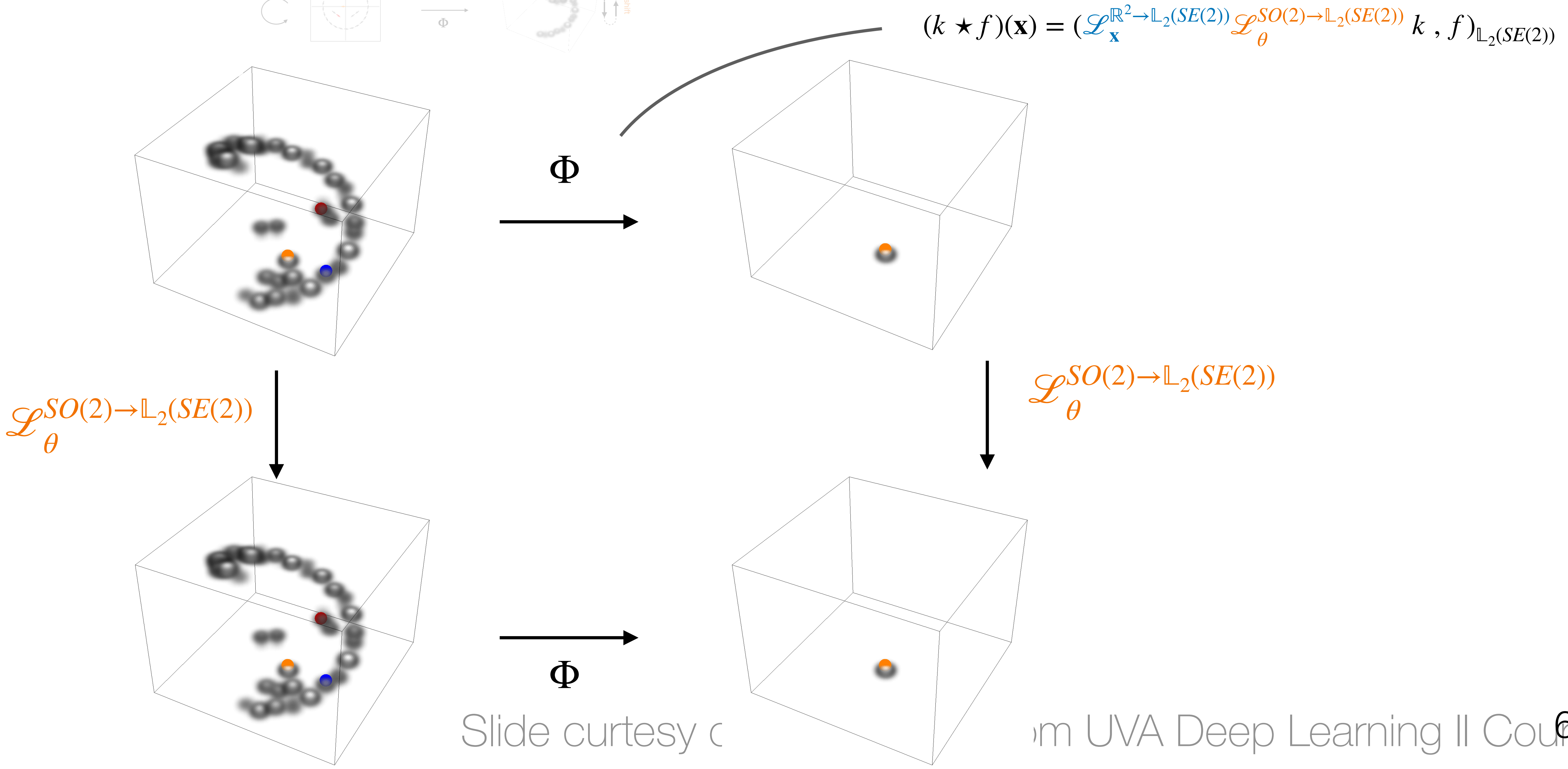
Isotropic/Constraint convolutions on spaces of lower dimension than G, $\forall_{h \in H} : k(hx) = k(x)$

$(X = G/H, Y = G)$

Lifting convolution. No constraints on k.

$(X = Y = G)$

Group convolutions. No constraints on k.



Types of layers $K : \mathbb{L}_2(SE(2)) \rightarrow \mathbb{L}_2(\mathbb{R}^2)$

$(X = Y = G/H)$

Isotropic/Constraint convolutions on spaces of lower dimension than G, $\forall_{h \in H} : k(hx) = k(x)$

$(X = G/H, Y = G)$

Lifting convolution. No constraints on k.

$(X = Y = G)$

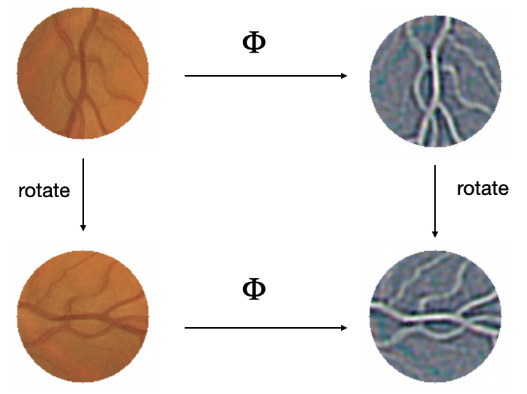
Group convolutions. No constraints on k.

$(X = G, Y = G/H)$

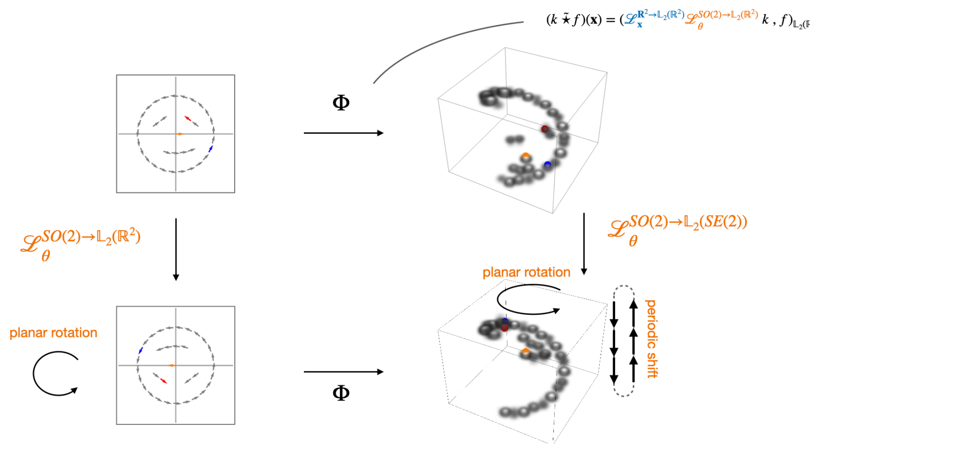
Projection layer. Mean pooling over H.

Types of layers

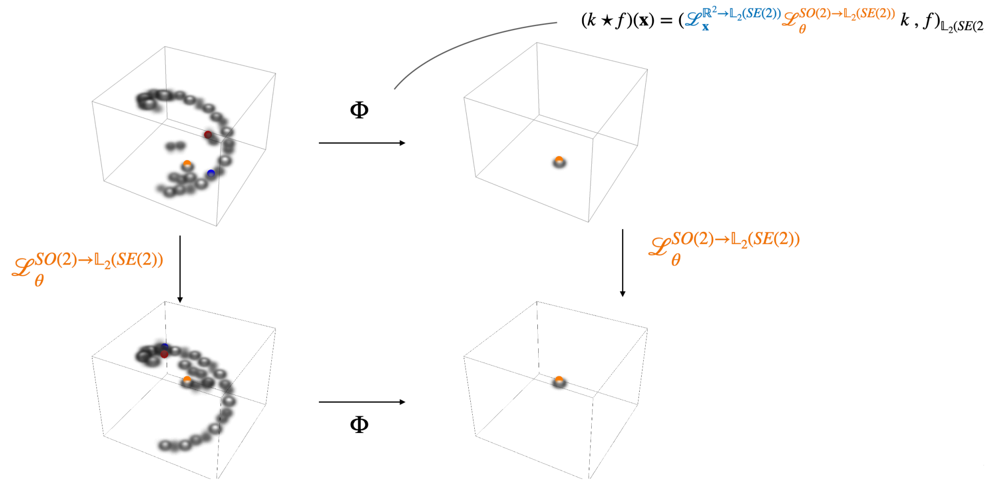
$(X = Y = G/H)$
Isotropic/Constraint convolutions on spaces of lower dimension than G, $\forall_{h \in H} : k(hx) = k(x)$



$(X = G/H, Y = G)$
Lifting convolution. No constraints on k.



$(X = Y = G)$
Group convolutions. No constraints on k.

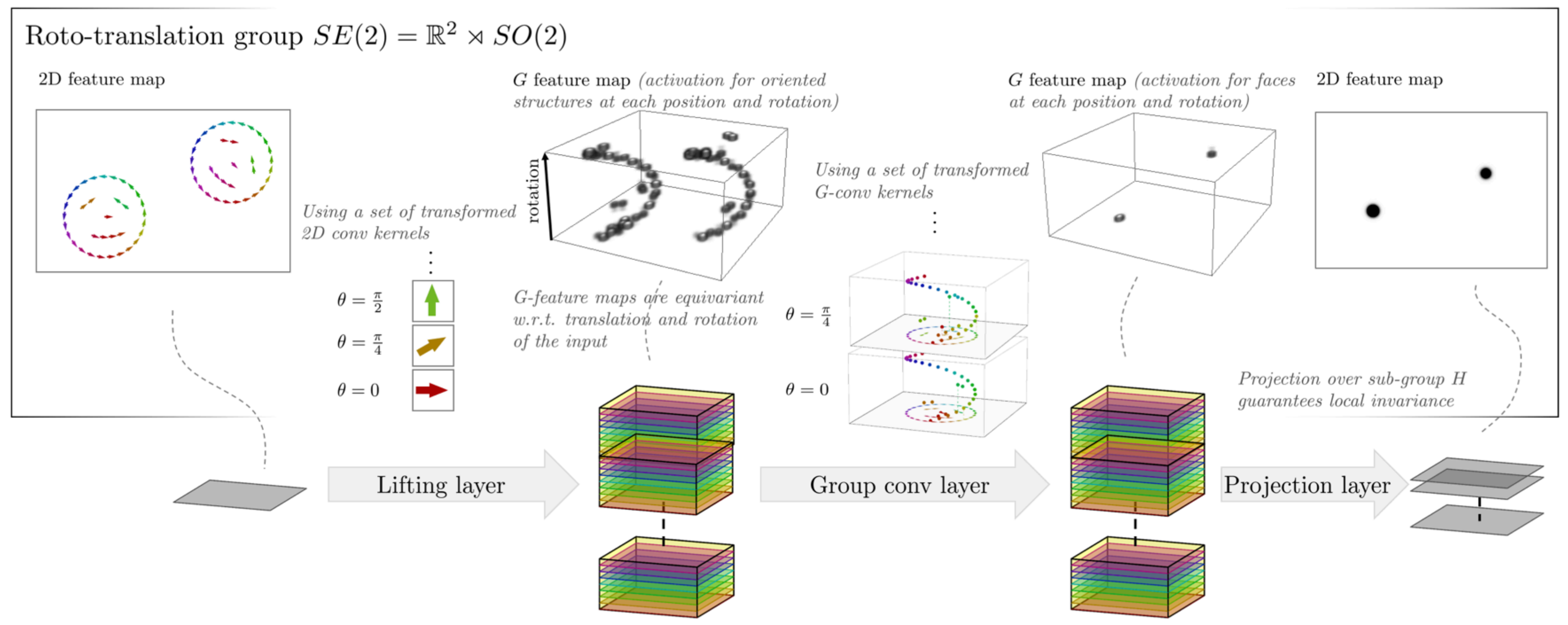


$(X = G, Y = G/H)$
Projection layer. Mean pooling over H.

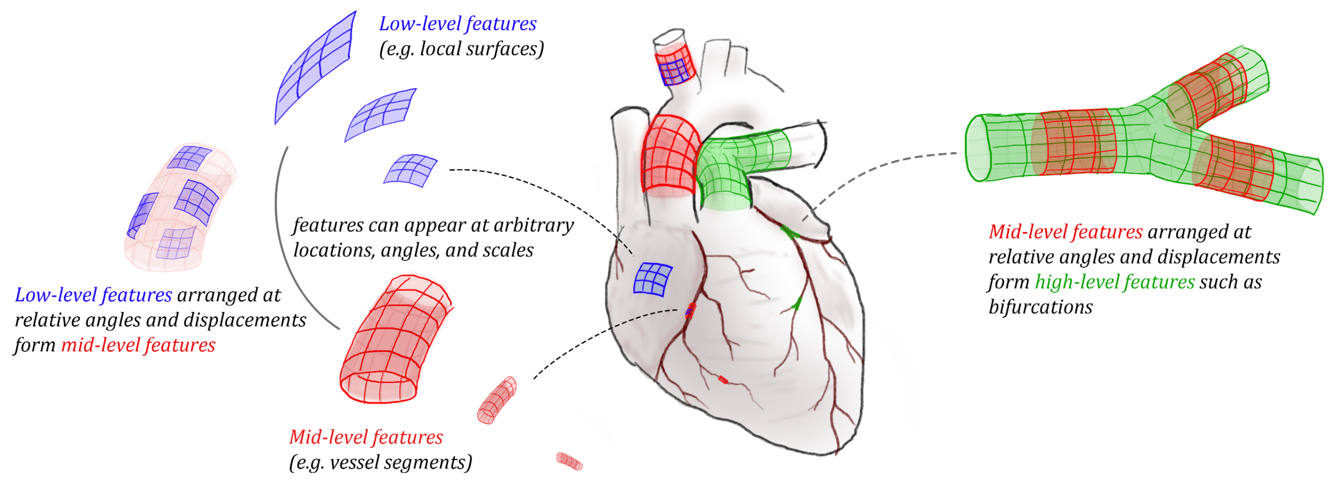
The most expressive group equivariant architectures are obtained by lifting the feature maps to the group

Summary

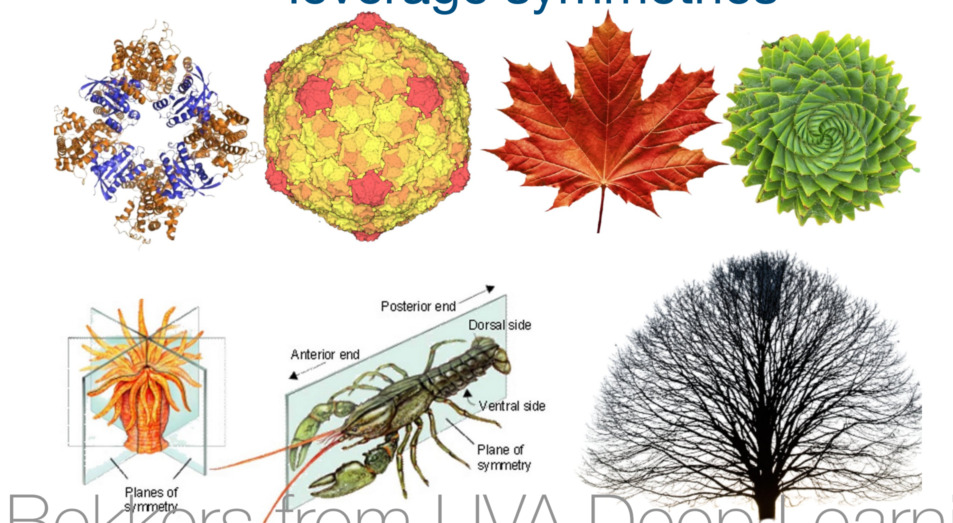
- Group convolutional neural networks intuitively perform template matching
- A template (kernel) is transformed and matched (inner-product) under all possible transformations in the group
- This creates higher-dimensional feature maps (functions on the group) on which we again define template matching via the group action
- In these higher dimensional feature maps we can detect advanced patterns in terms of features at relative poses!
- G-CNNs are based on equivariant layers (thus weight sharing) and guarantee invariance through pooling



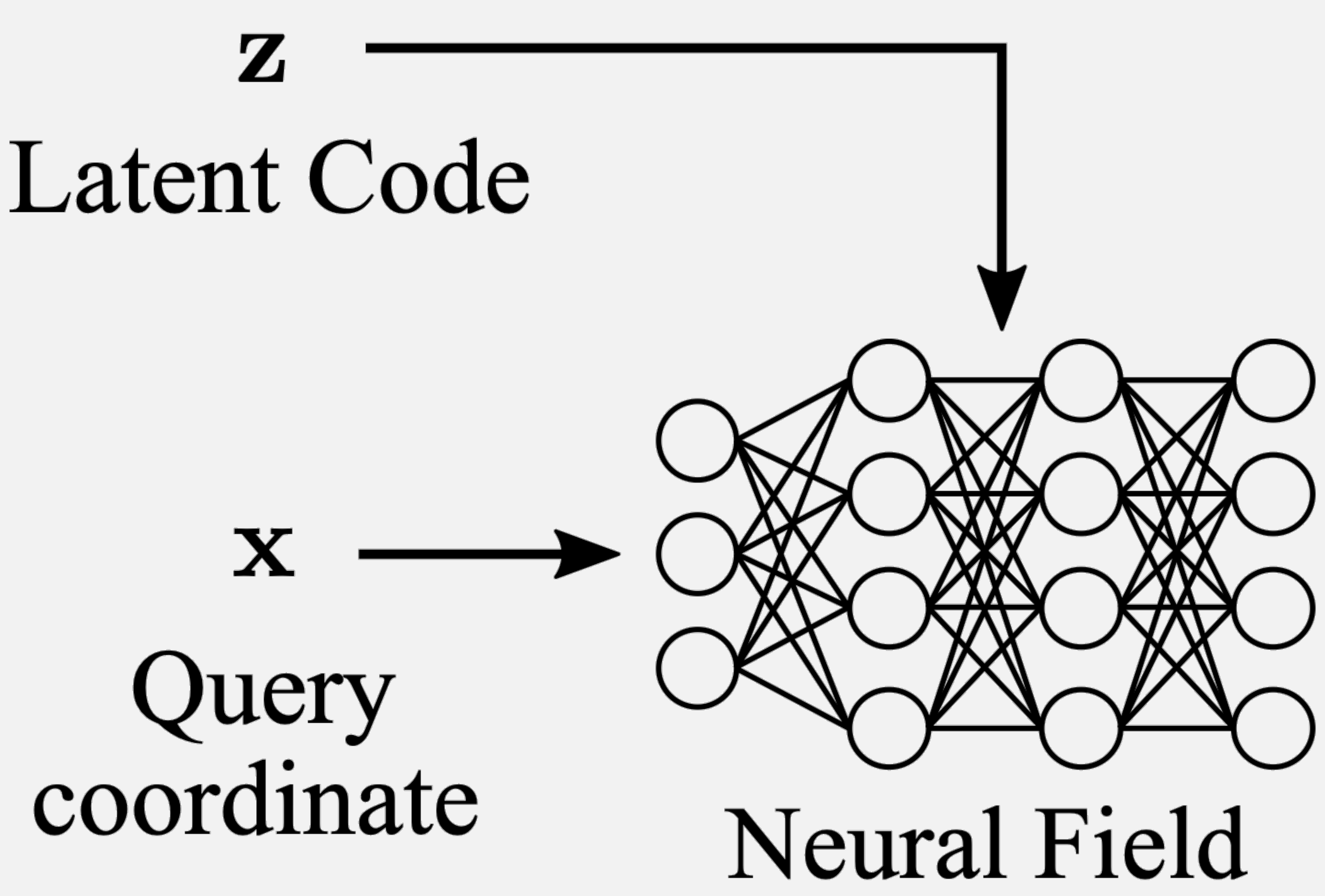
recognition by components



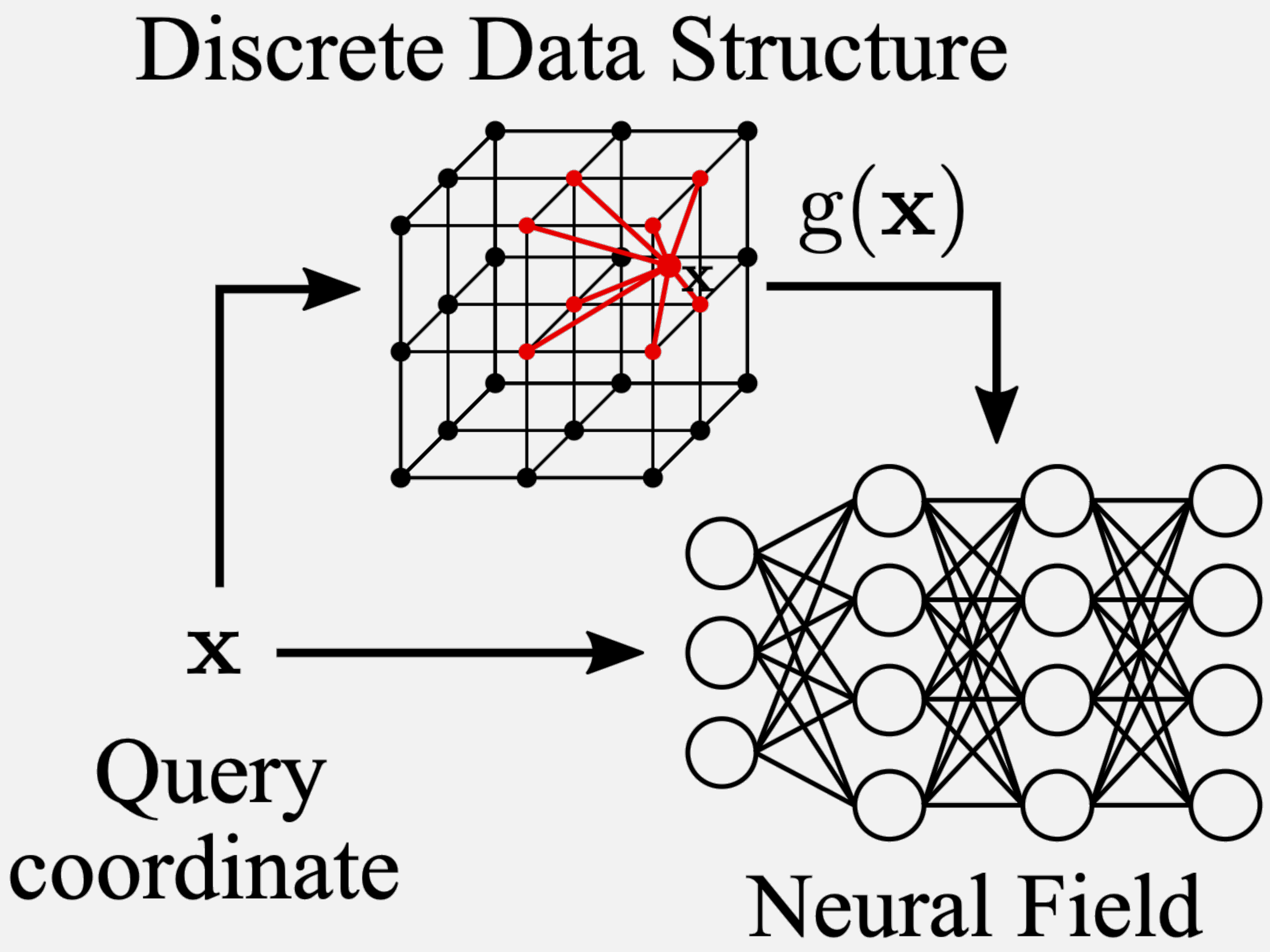
leverage symmetries



Global Conditioning: Single Latent Code for whole 3D Scene



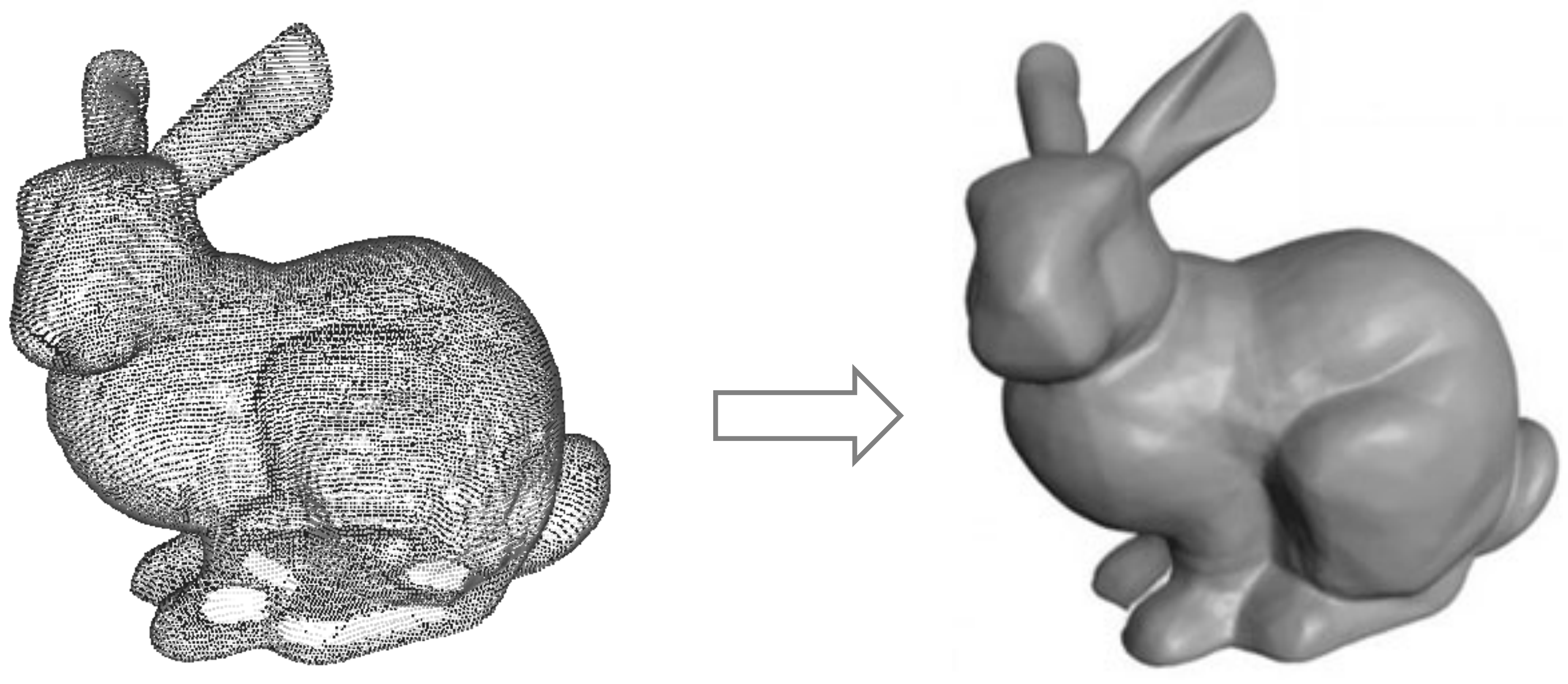
Global Conditioning



Local Conditioning

Neural Fields in Visual Computing and Beyond, Xie et al., EG STAR 2022

Global Latent Codes: Enables reconstruction from partial observations!



DeepSDF, Occupancy Networks, IM-Net

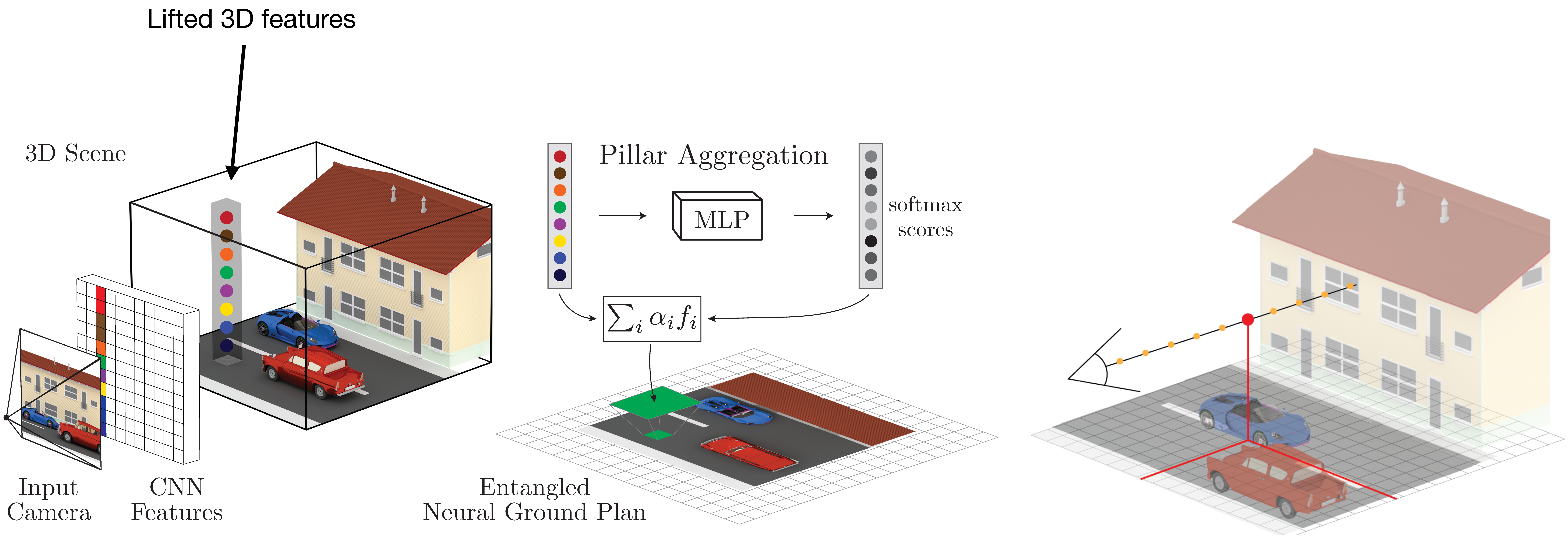
Differential Volumetric Rendering,
Niemeyer et al., CVPR 2020

Scene Representation Networks: Continuous
3D-Structure-Aware Neural Scene Representations, NeurIPS 2019.

Local Conditioning: Pixel-Aligned Features.

PiFU, Saito et al., ICCV 2019.
PixelNeRF, Yu et al., CVPR 2021
Grf: Learning a general radiance field…, Trevithick et al.

Conditional Ground Plans for Single-Image 3D Reconstruction



Seeing 3D Objects in a Single Image via Self-Supervised Static-Dynamic Disentanglement, Sharma et al. 2022

Static-dynamic disentanglement from a **single** image!

Seeing 3D Objects in a Single Image via Self-Supervised Static-Dynamic Disentanglement, Sharma et al. 2022

Discussion

- Geometric Deep Learning and Group Convolutions are great in theory
- In practice, obviously, they are similarly useful: See MLP vs. CNN.
- However, for groups *other* than the translation group, it is *not obvious* that we want perfect equivariance.
- Group convolutions can become expensive b/c of combinatorial growth of feature maps: *for each translation* $\times$ *for each scale* $\times$ *for each rotation* $\times$ *…*
- Nevertheless, there are clear and apparent use-cases - which we will see in the paper session :)