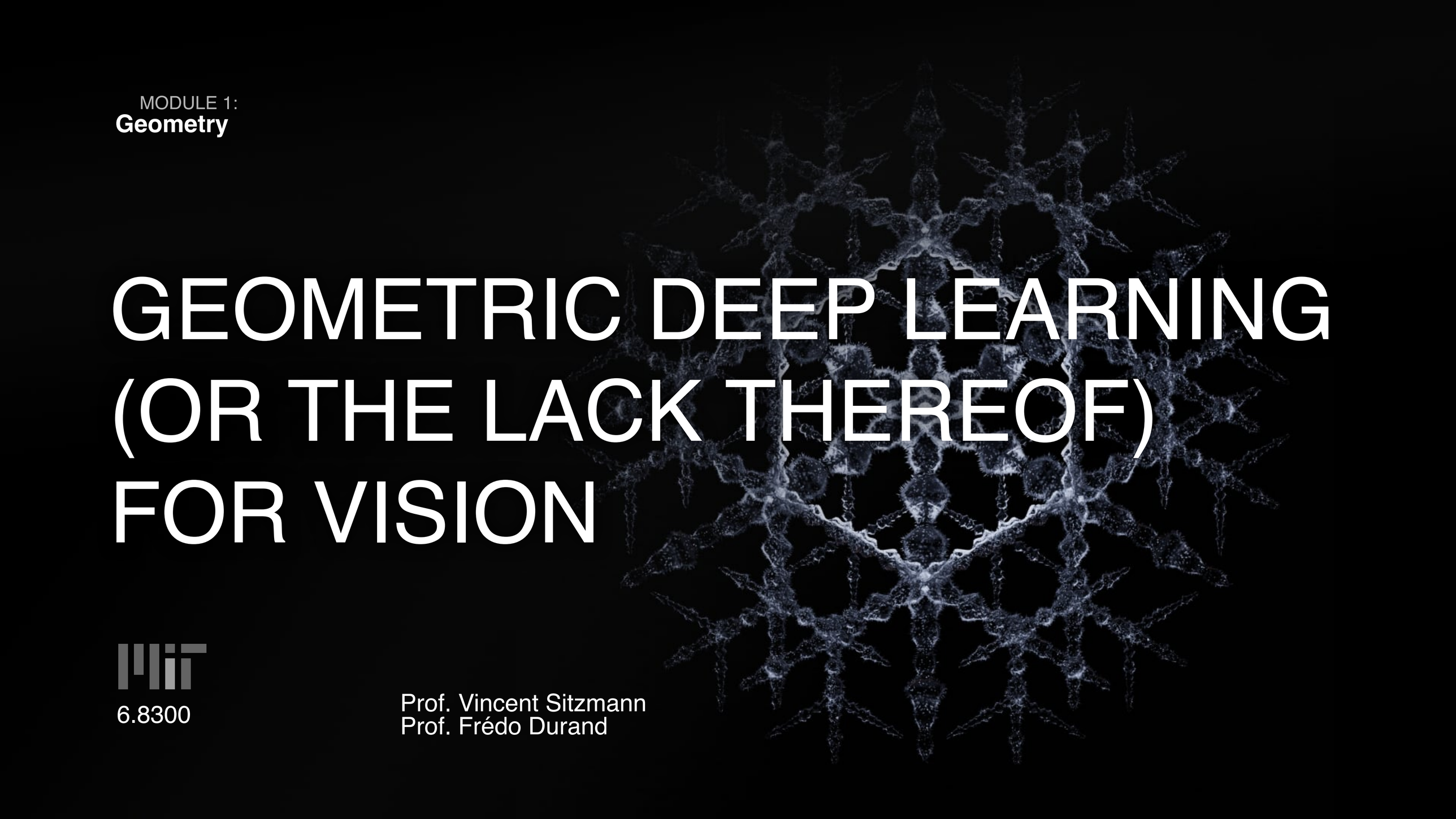
MODULE 1:
Geometry

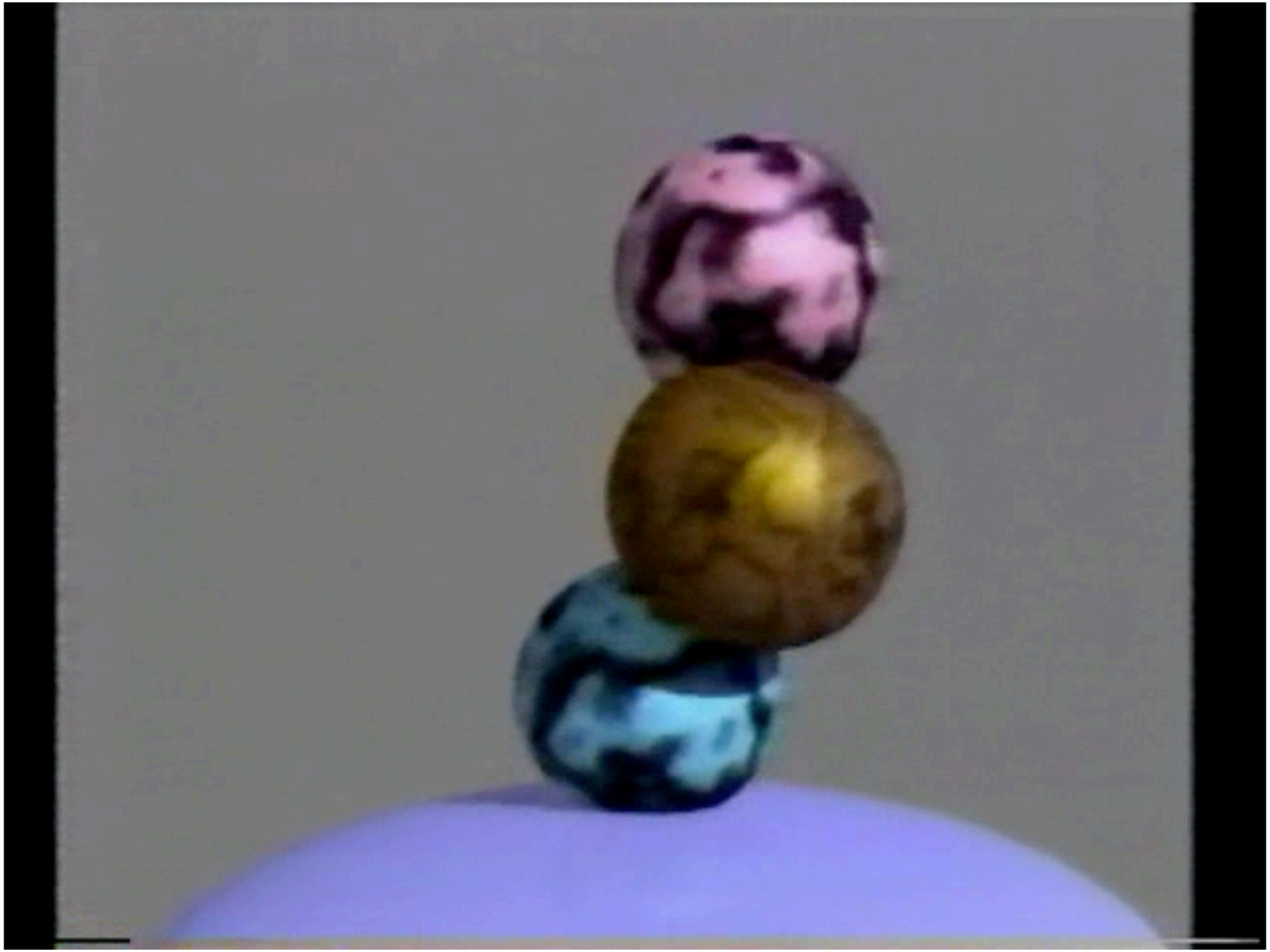
GEOMETRIC DEEP LEARNING (OR THE LACK THEREOF) FOR VISION



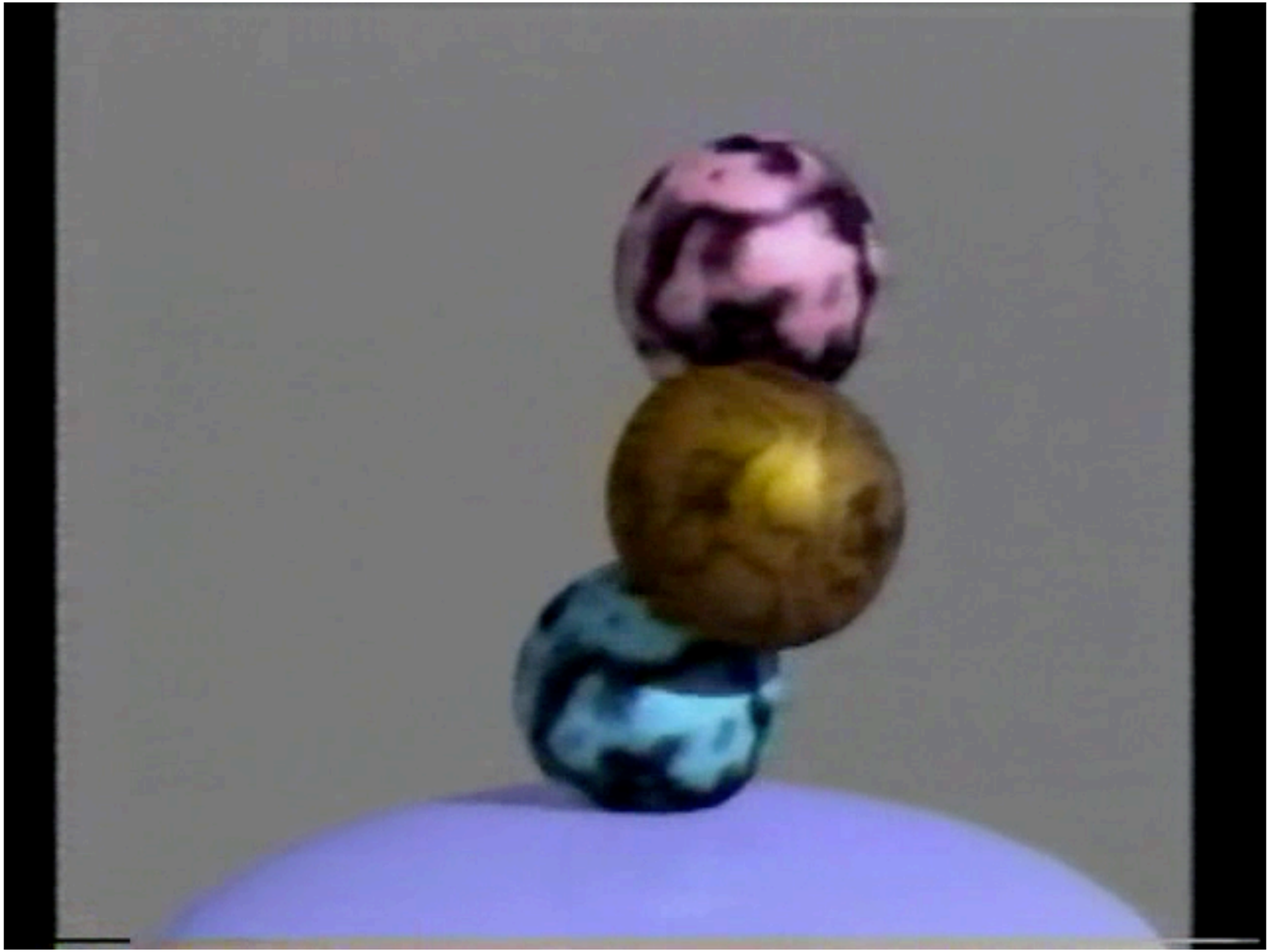
6.8300

Prof. Vincent Sitzmann
Prof. Frédo Durand

Motion without movement



Motion without movement



The Visual Microphone: Passive Recovery of Sound from Video

Abe Davis[1] Michael Rubinstein[2,1] Neal Wadhwa[1] Gautham J. Mysore[3] Frédo Durand[1] William T. Freeman[1]

[1]MIT CSAIL [2]Microsoft Research [3]Adobe Research

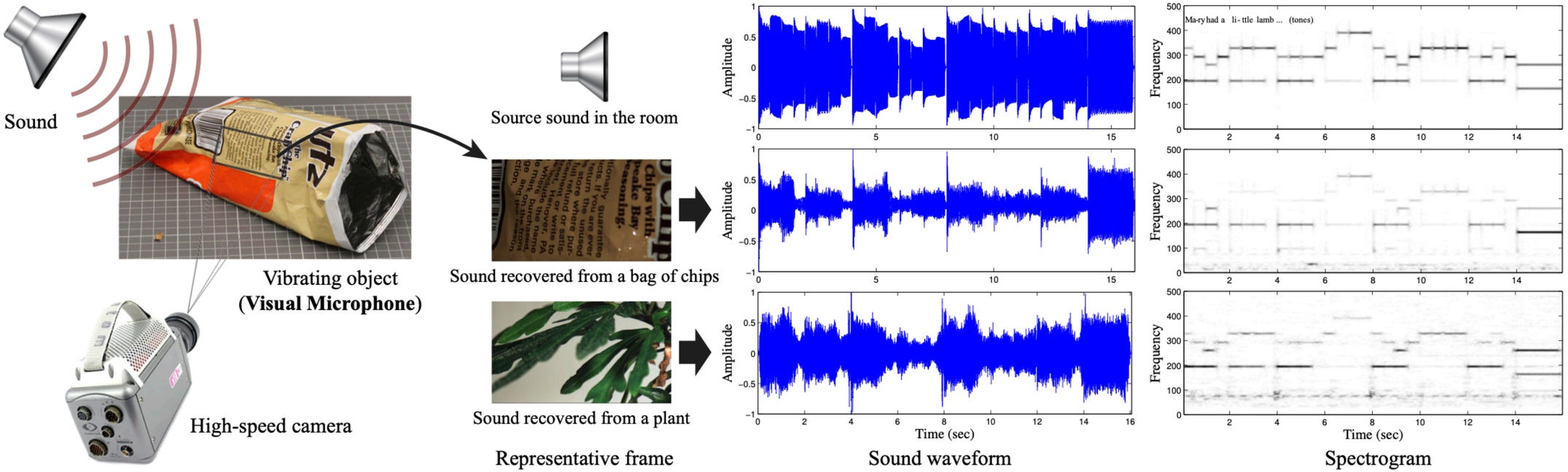
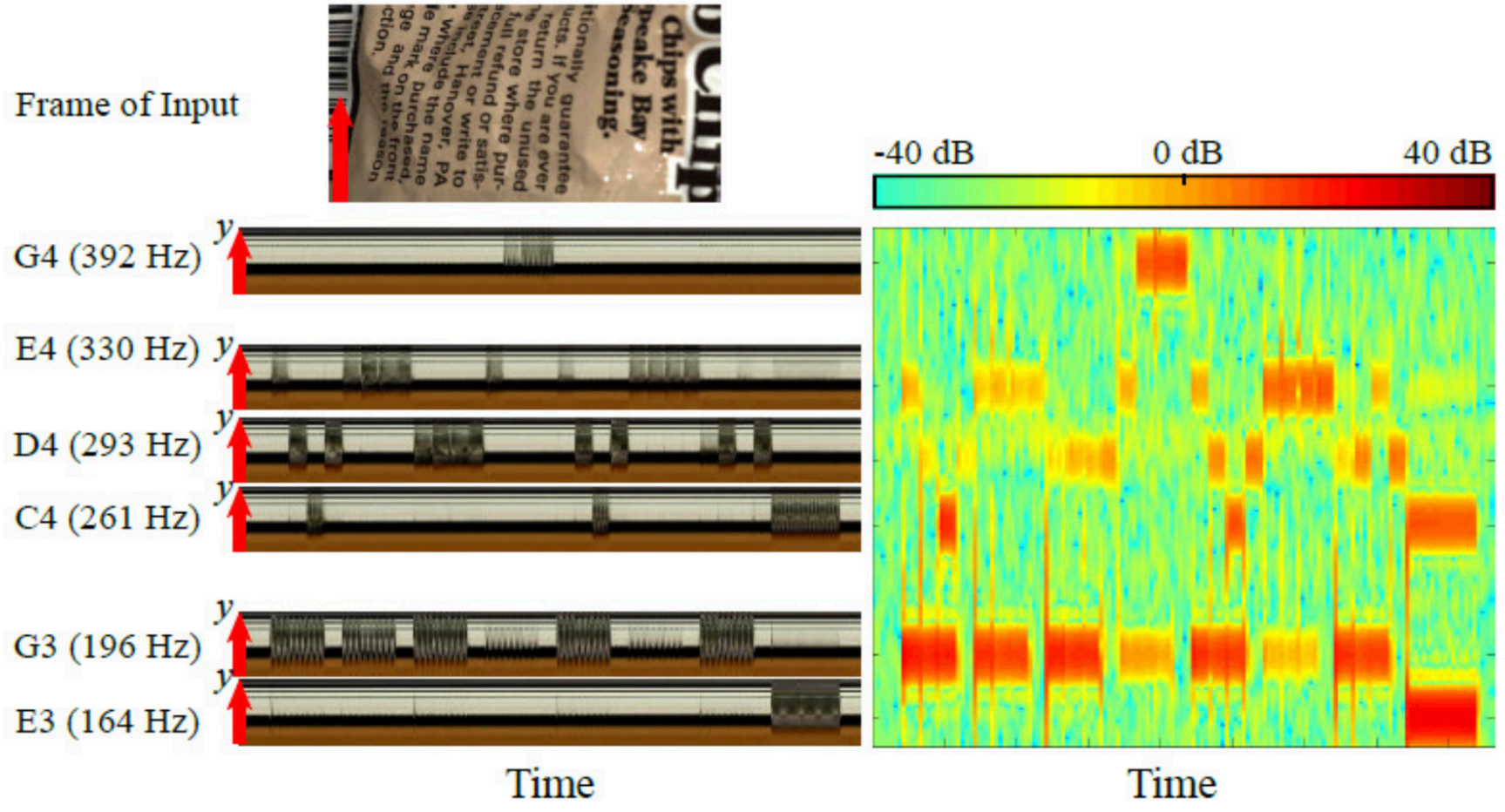
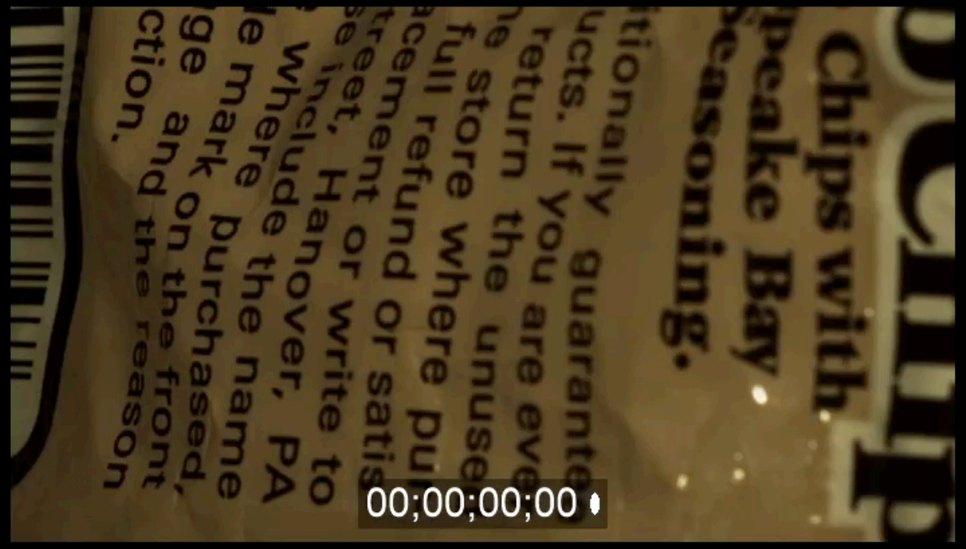


Figure 1: *Recovering sound from video. Left: when sound hits an object (in this case, an empty bag of chips) it causes extremely small surface vibrations in that object. We are able to extract these small vibrations from high speed video and reconstruct the sound that produced them - using the object as a visual microphone from a distance. Right: an instrumental recording of "Mary Had a Little Lamb" (top row) is played through a loudspeaker, then recovered from video of different objects: a bag of chips (middle row), and the leaves of a potted plant (bottom row). For the source and each recovered sound we show the waveform and spectrogram (the magnitude of the signal across different frequencies over time, shown in linear scale with darker colors representing higher energy). The input and recovered sounds for all of the experiments in the paper can be found on the project web page.*

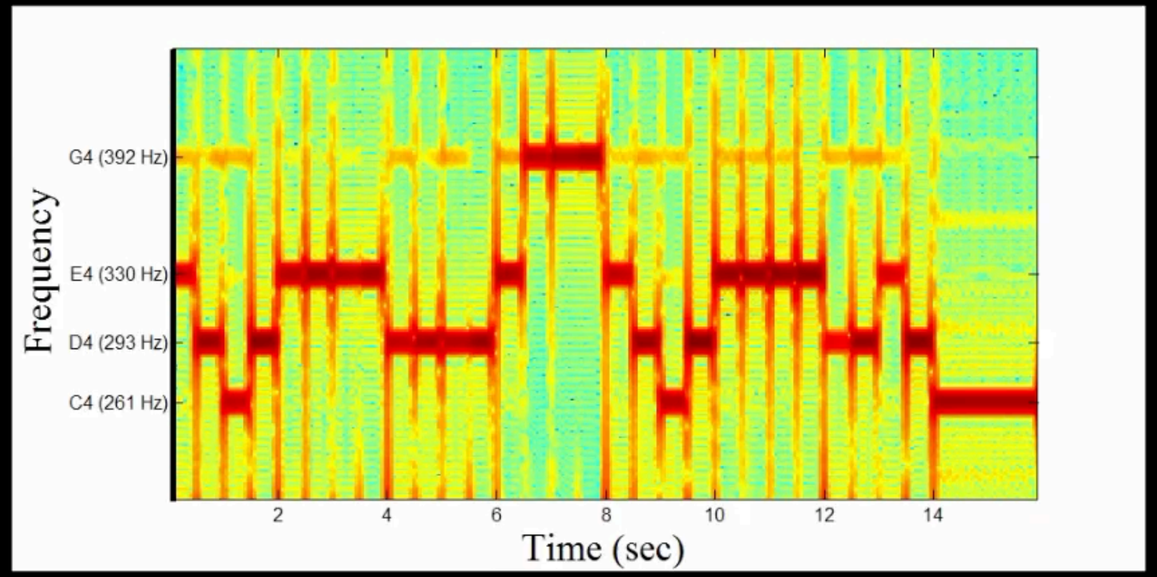
https://people.csail.mit.edu/mrub/VisualMic/



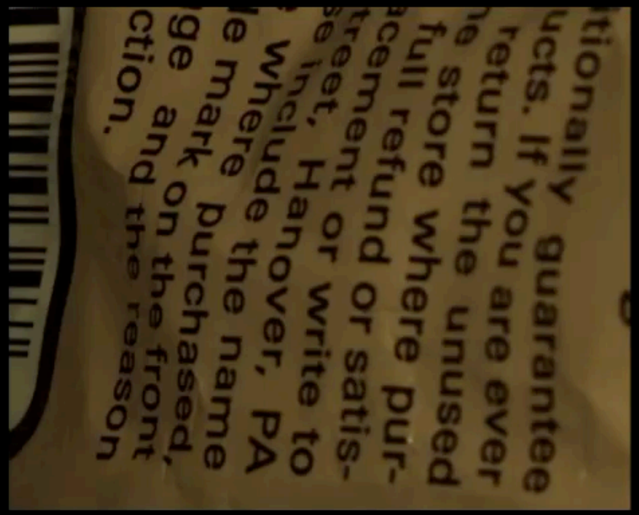
A pure-tone version of "Mary Had a Little Lamb" is played at a chip bag and the motions corresponding to each note are magnified separately. Time slices of the resulting videos, in which the vertical dimension is space and the horizontal dimension is time, are used to produce a visual spectrogram (a), which closely matches the spectrogram of the recovered audio (b).



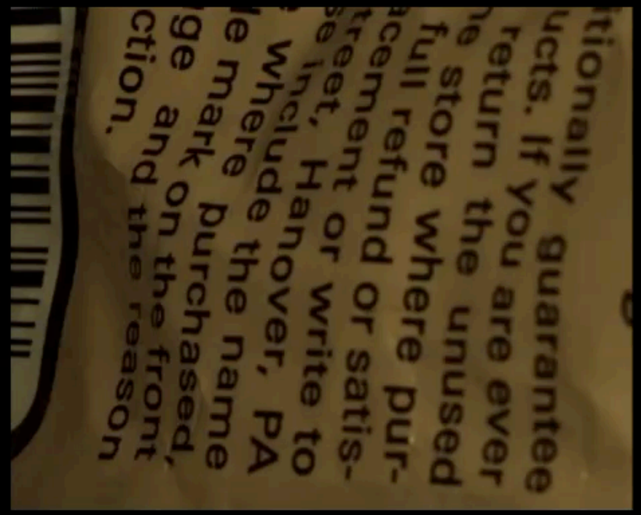
Source video (2200 fps)



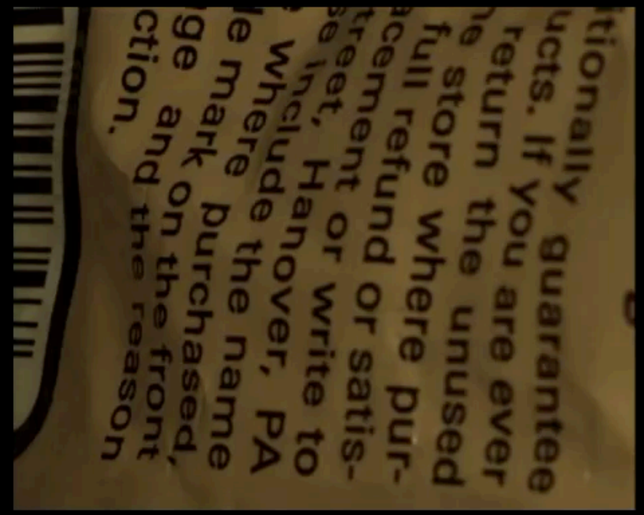
Sound spectrogram



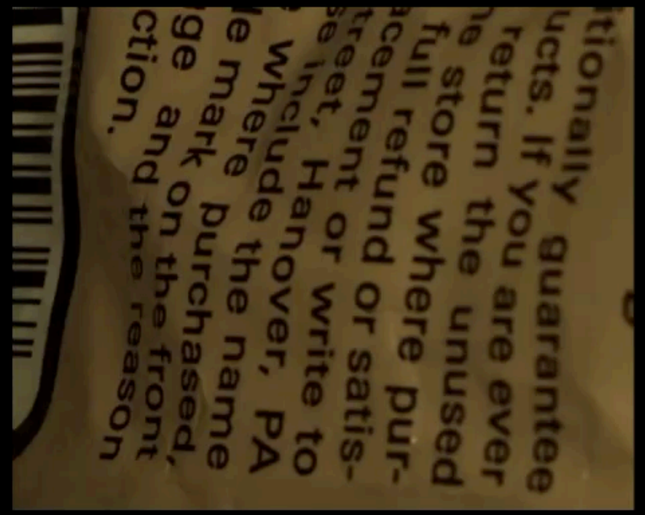
C4 (261 Hz) x50



D4 (293 Hz) x150

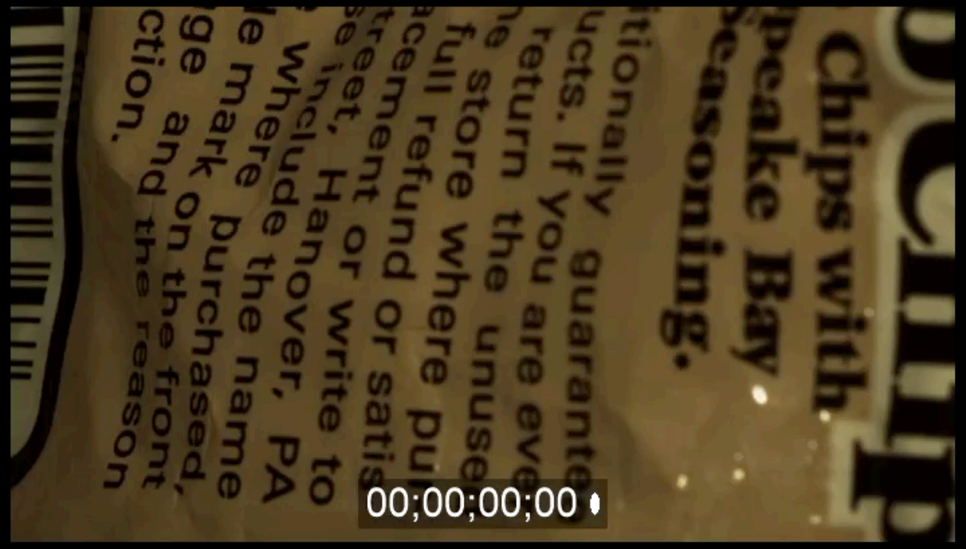


E4 (330 Hz) x150

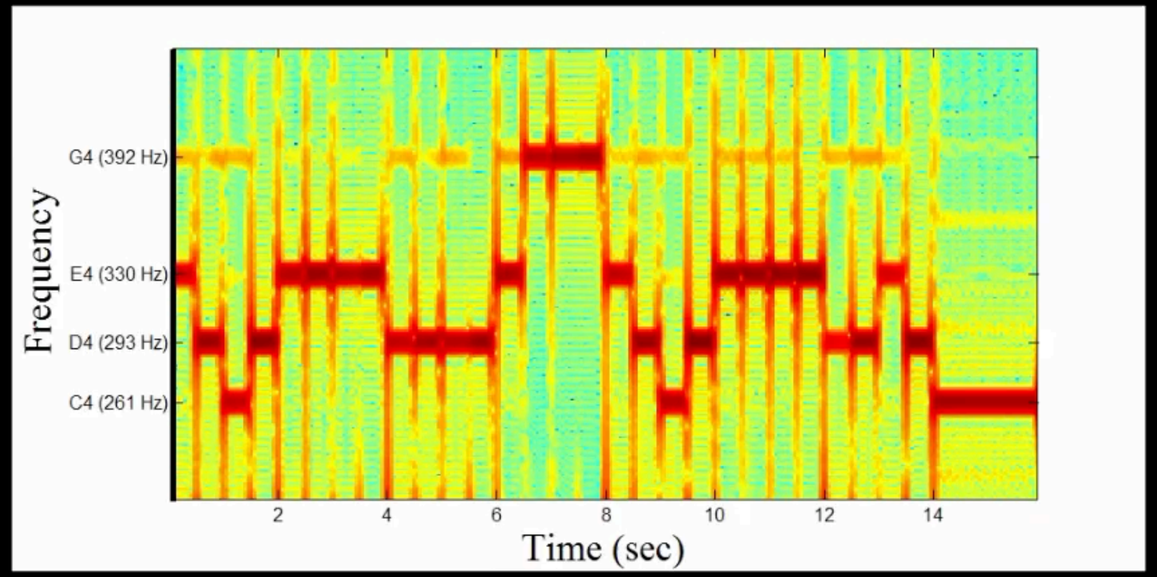


G4 (392 Hz) x120

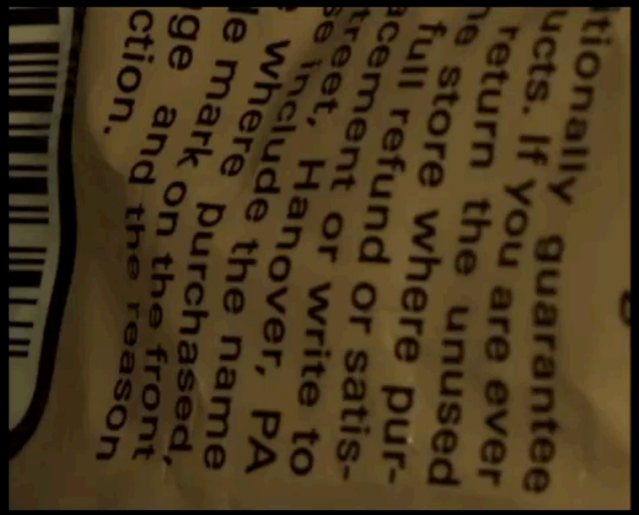
Motion-magnified videos



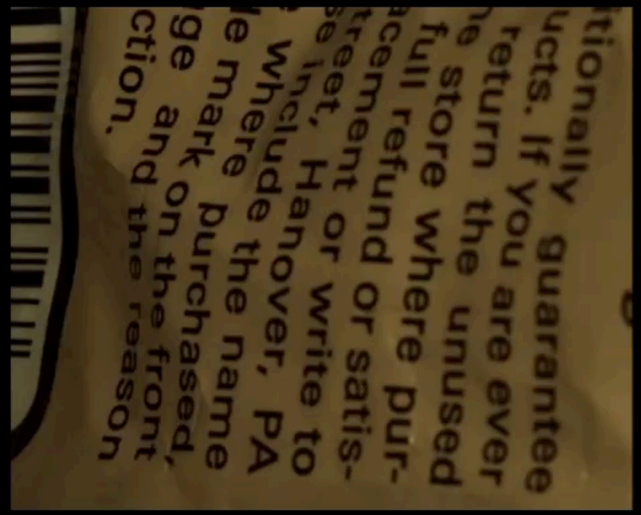
Source video (2200 fps)



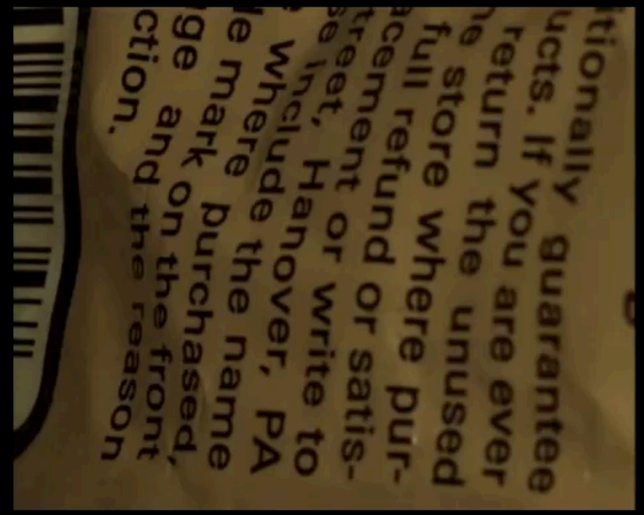
Sound spectrogram



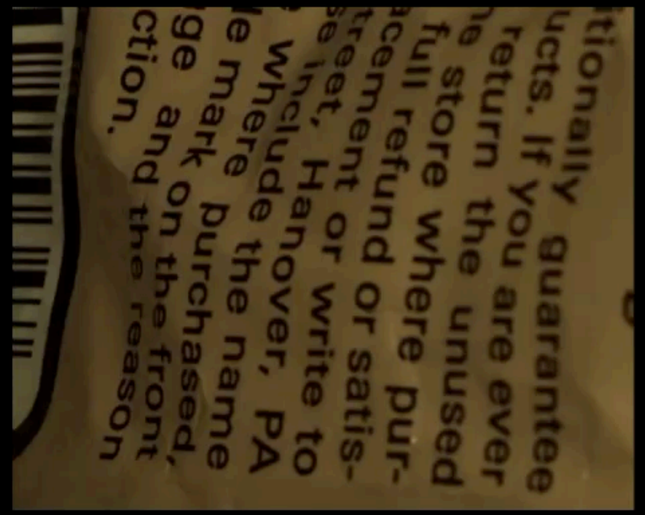
C4 (261 Hz) x50



D4 (293 Hz) x150



E4 (330 Hz) x150



G4 (392 Hz) x120

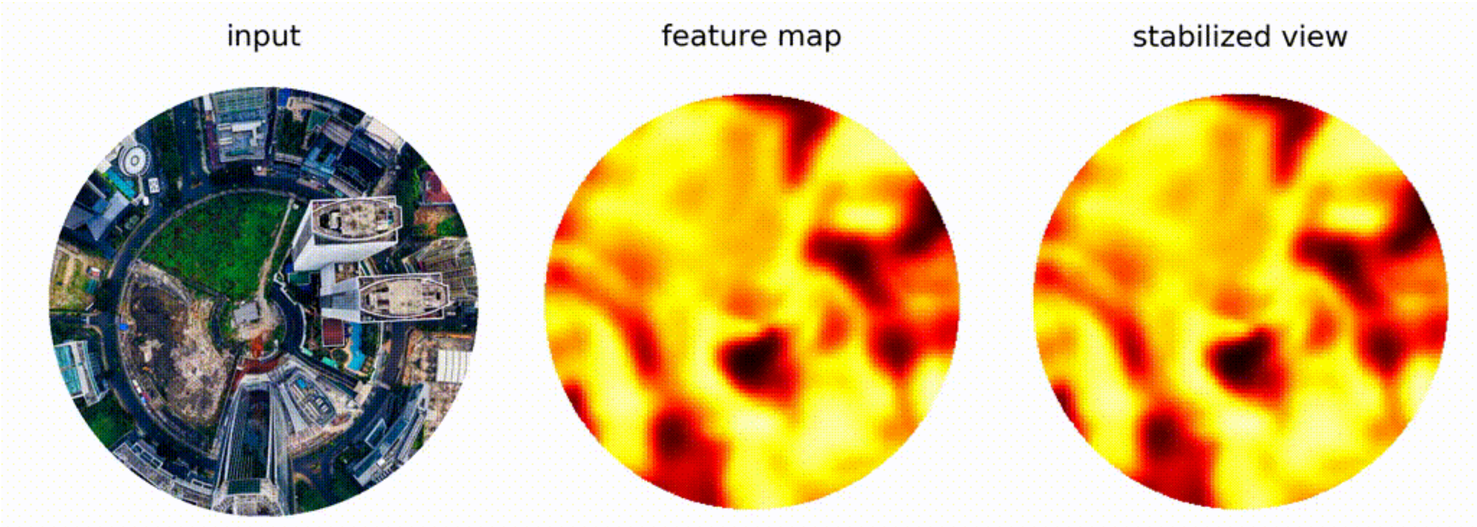
Motion-magnified videos

What are we trying to achieve?

- Expand tools from 2D lattice-based image processing to more complex domains (sphere, 3D surfaces)
    - We will have good success !
- Build in symmetries and regularities about the world into our solutions
    - e.g. invariance to shifts, rotation, scale, 3D transformations
    - With the hope that it helps when data is scarce
    - The jury is still out on this one.

Geometric guarantees (equivariance)

CNN

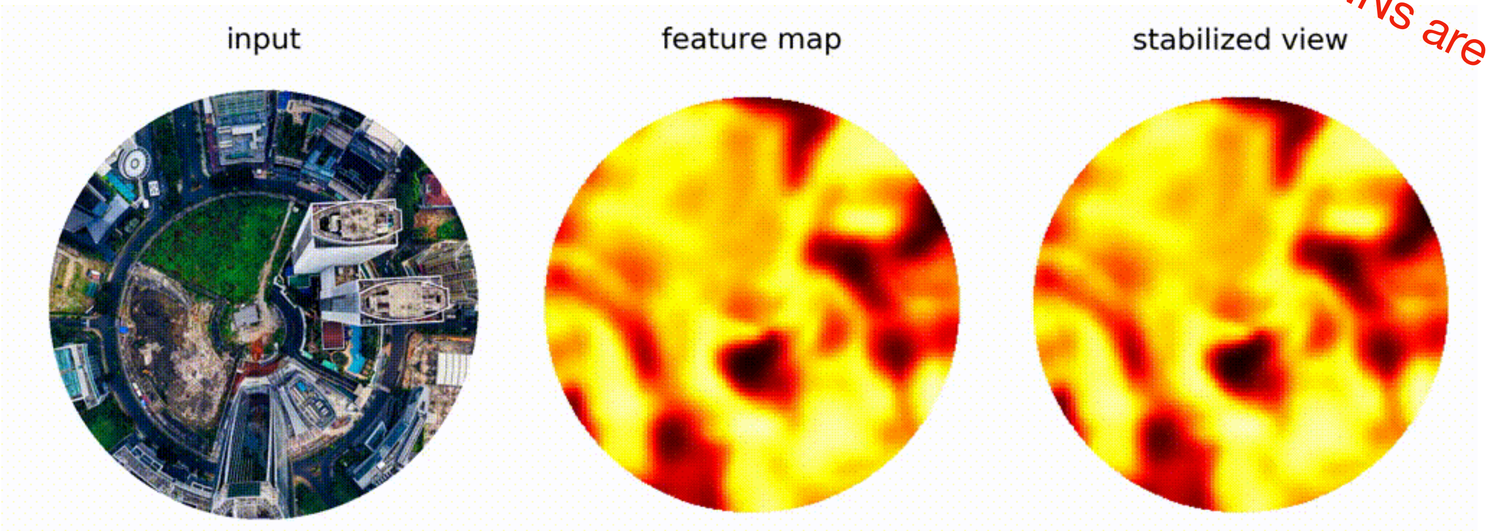


Figures source:
https://github.com/QUVA-Lab/e2cnn

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Geometric guarantees (equivariance)

CNN



CNNs are not rotation equivariant

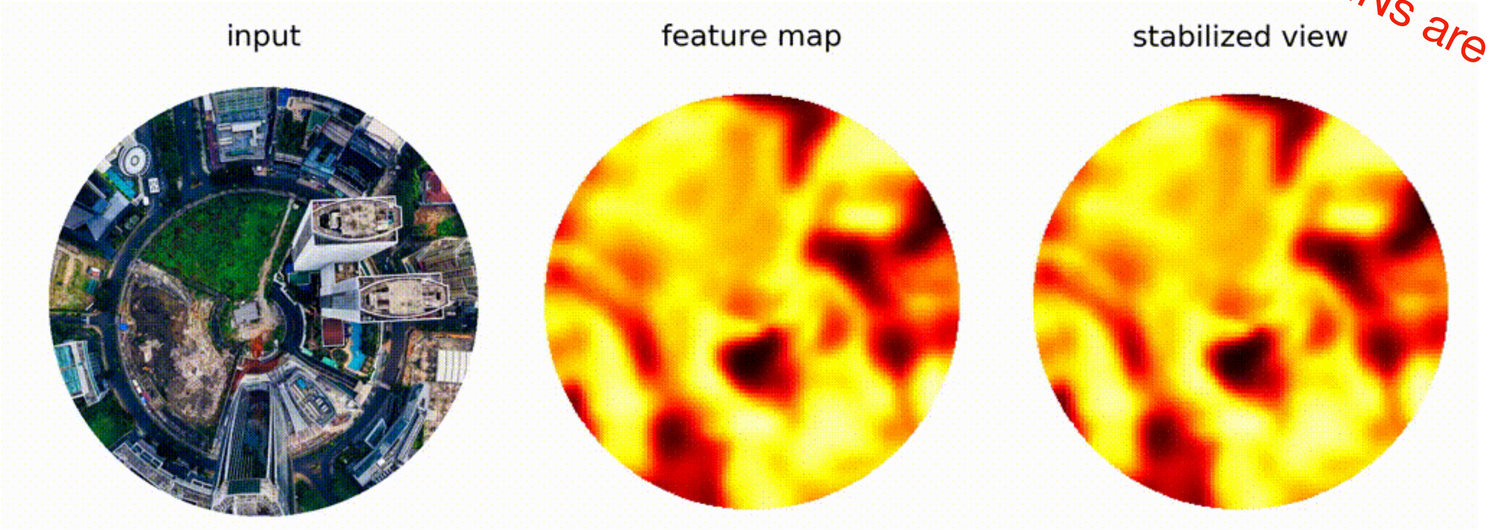


Figures source:
https://github.com/QUVA-Lab/e2cnn

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Geometric guarantees (equivariance)

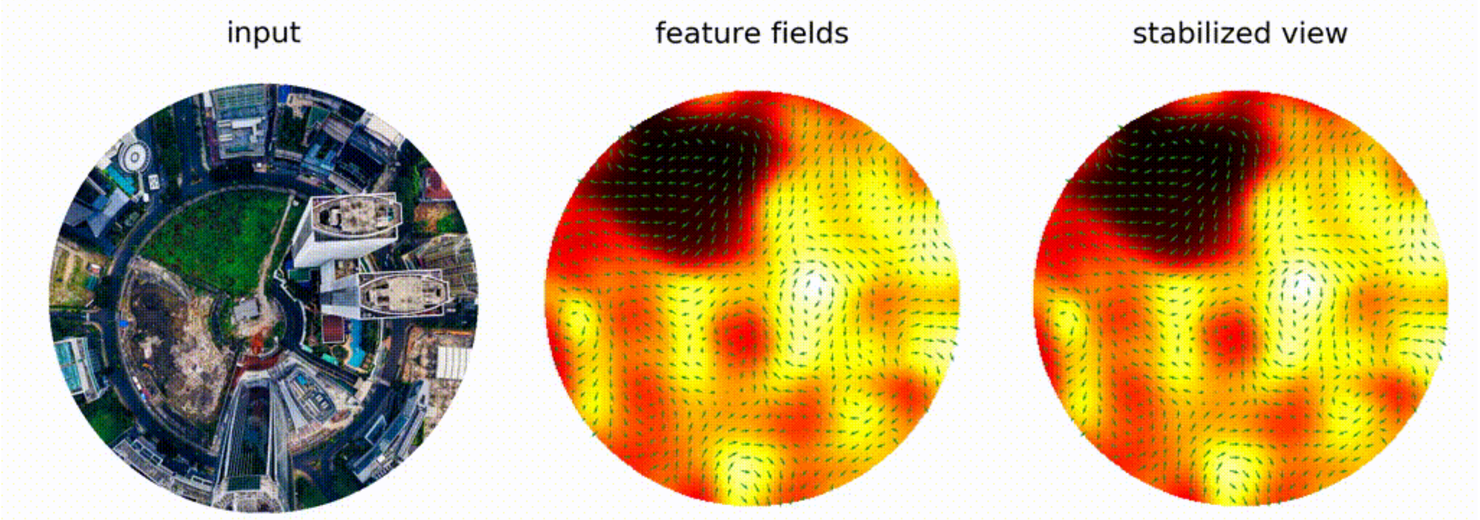
CNN



CNNs are not rotation equivariant



Equivariant NN

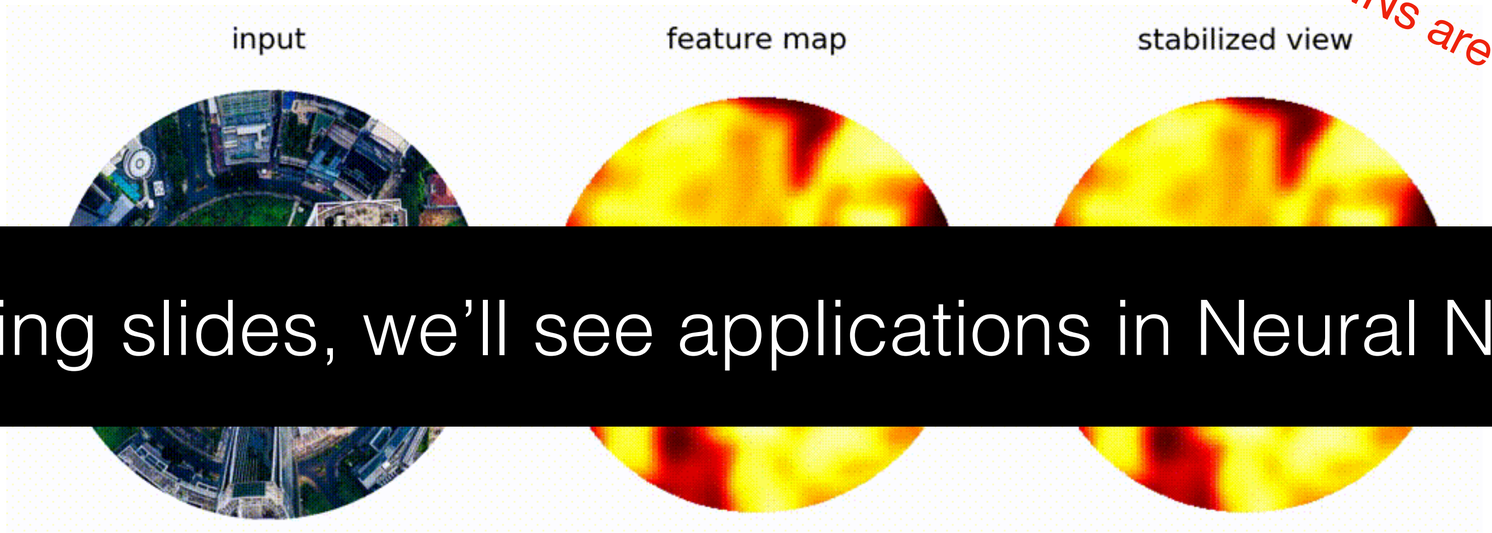




Figures source:
https://github.com/QUVA-Lab/e2cnn

Geometric guarantees (equivariance)

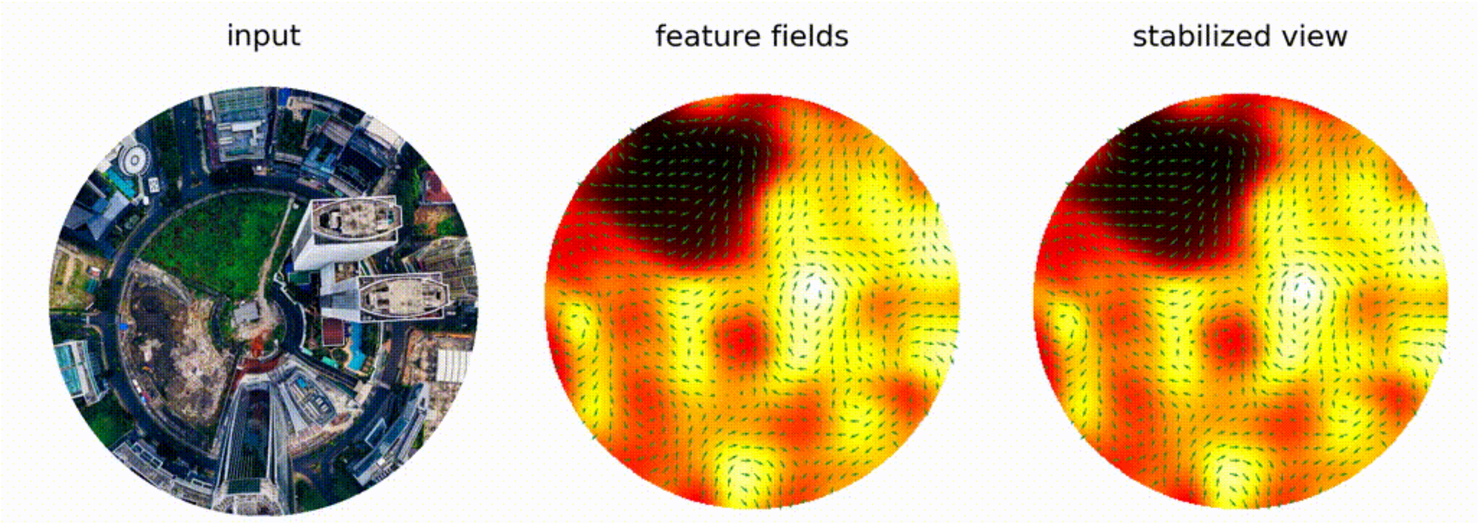
CNN



CNNs are not rotation equivariant

In following slides, we'll see applications in Neural Nets!

Equivariant NN



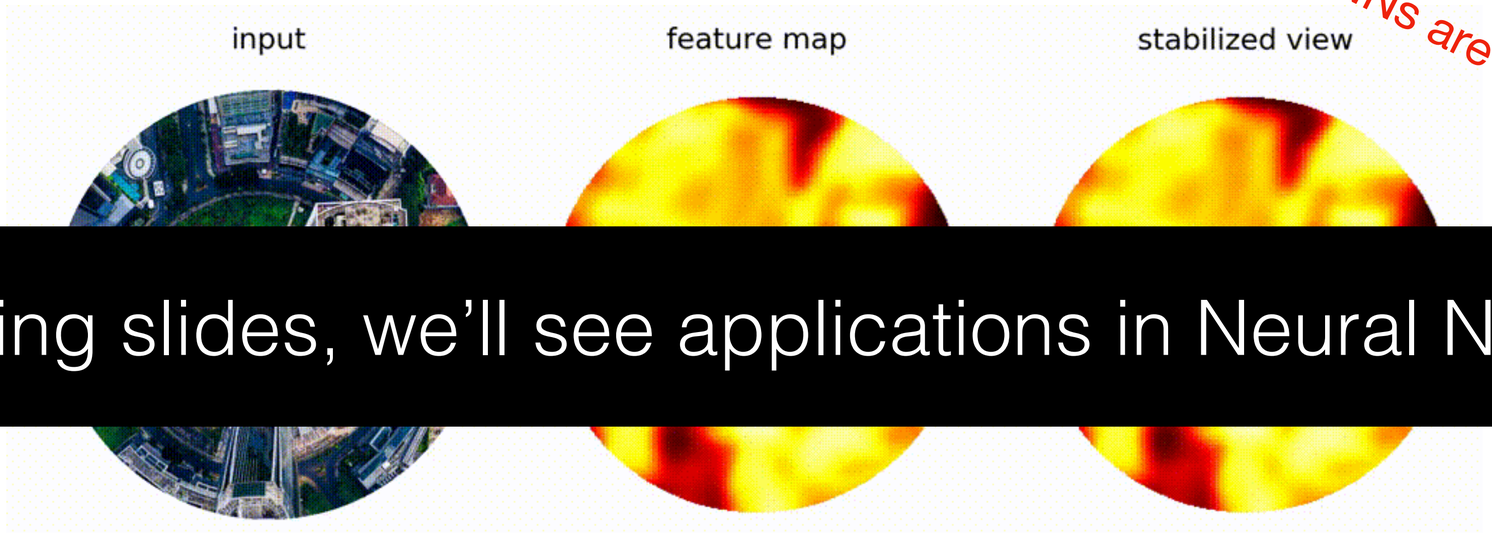


Figures source:
https://github.com/QUVA-Lab/e2cnn

Geometric guarantees (equivariance)

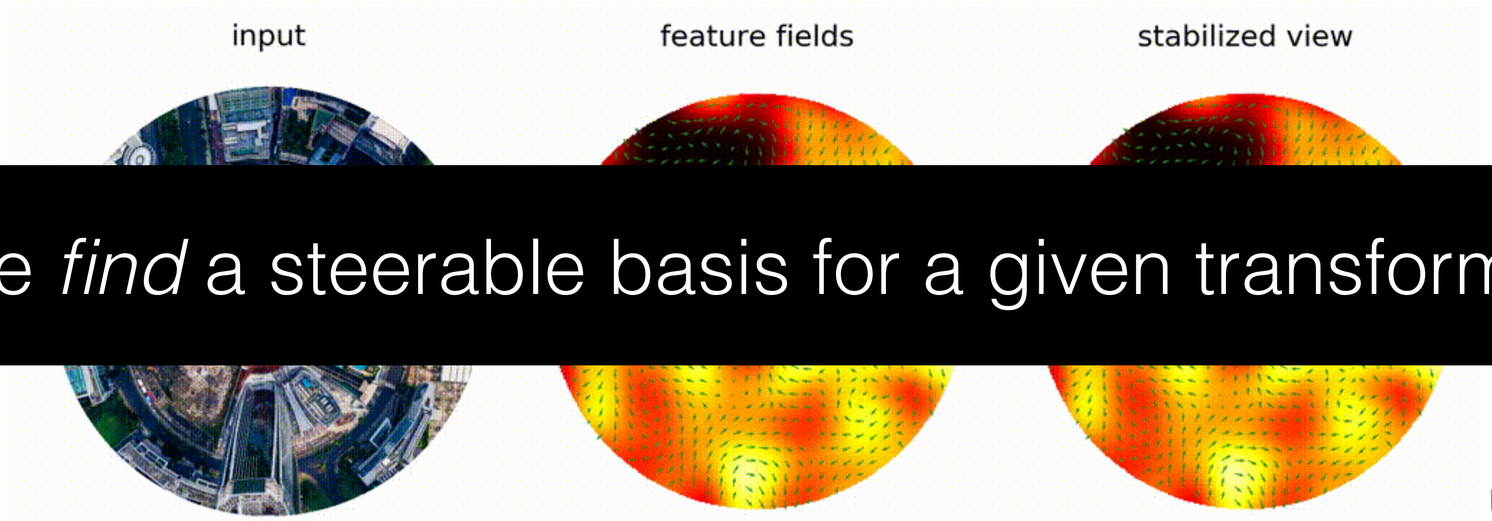
CNNs are not rotation equivariant

CNN



In following slides, we'll see applications in Neural Nets!

Equivariant NN



How can we *find* a steerable basis for a given transformation?

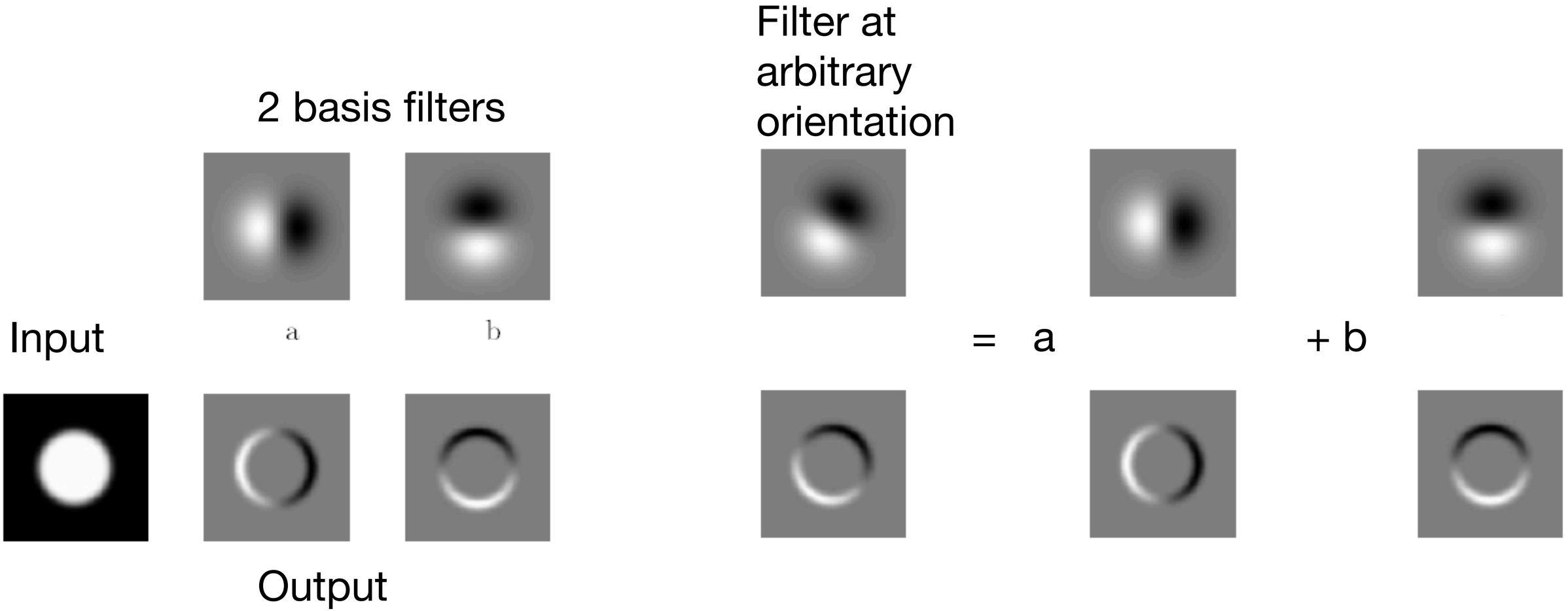
Figures source:
https://github.com/QUVA-Lab/e2cnn

Steerability at a high level

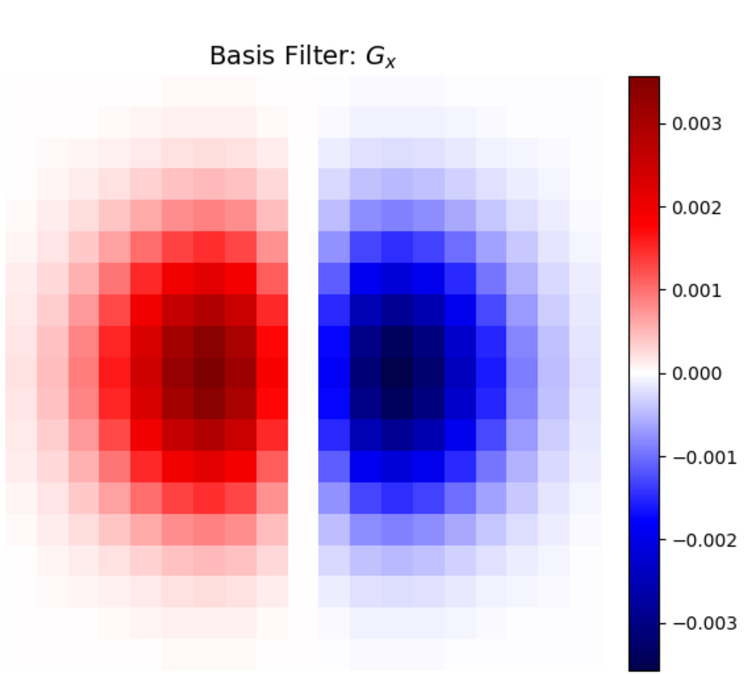
- Group of transformations G (e.g. 2D rotations SO2, (circular) shifts)
- For a given vector/signal/image/function
  - Discrete, finite, basis of outputs
- Can reconstitute the full group output

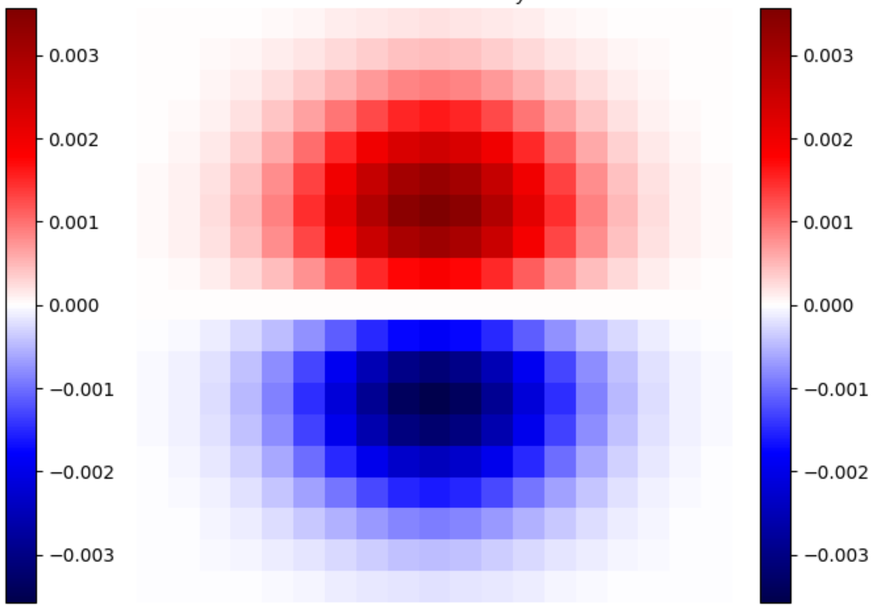
- If we are lucky (shifts), the basis is the same for all vectors
  - Globally steerable
- Warning: tends to require some form of compactness

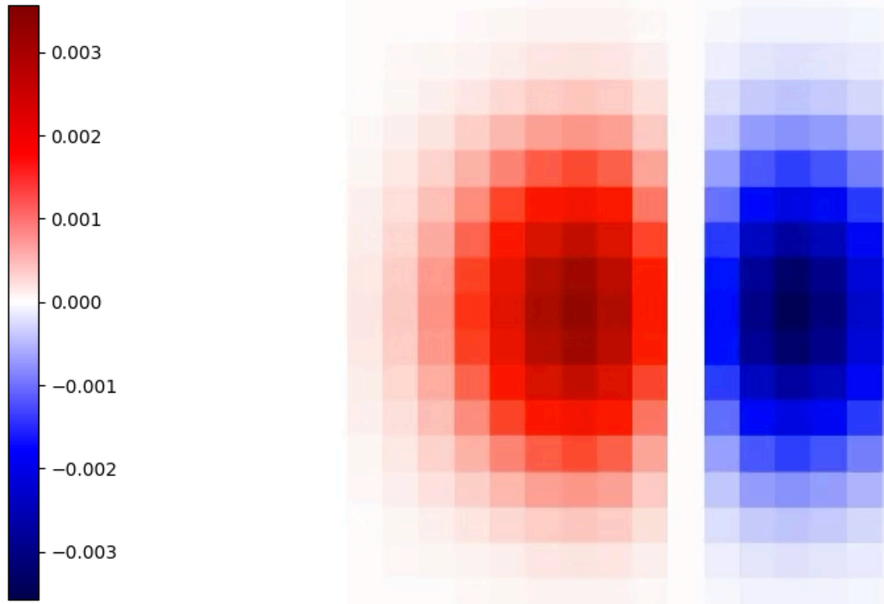
Steerability in orientation



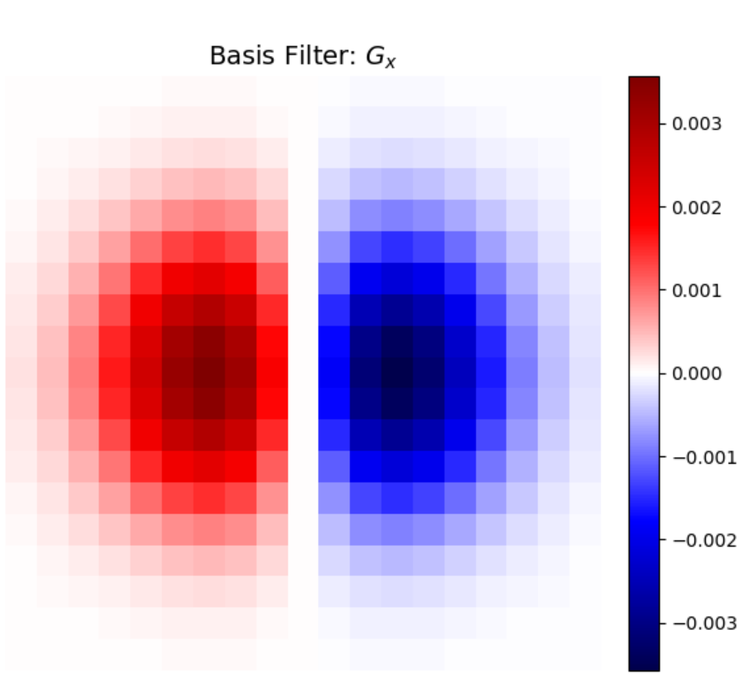
Steerable Filters

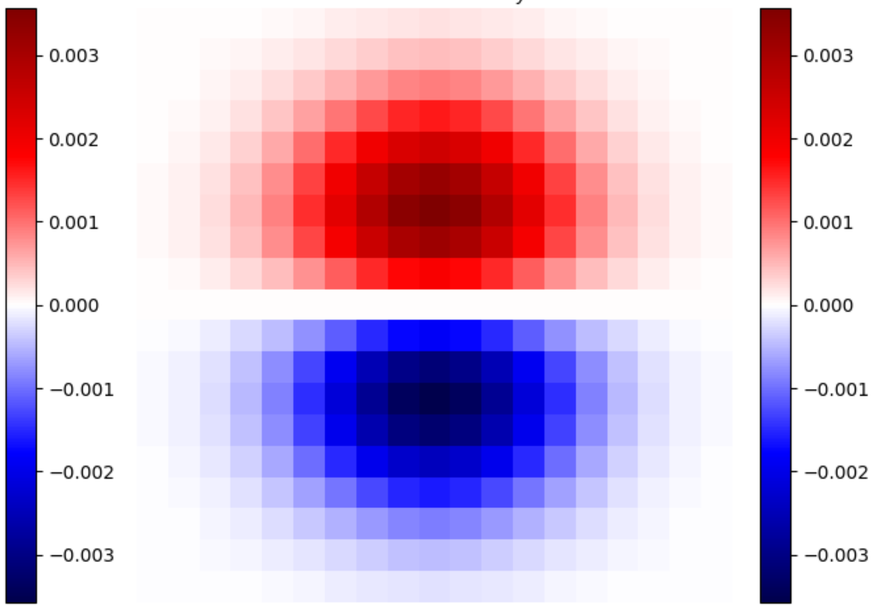


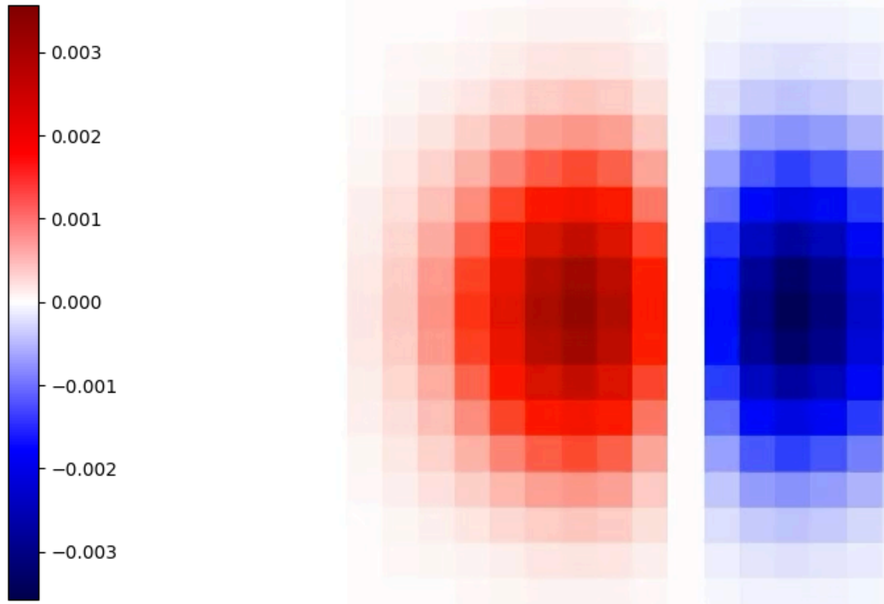




Steerable Filters

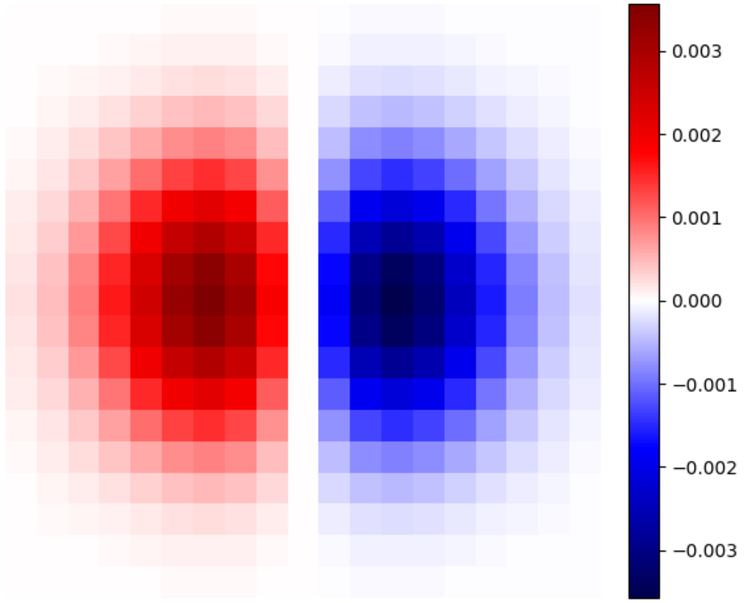




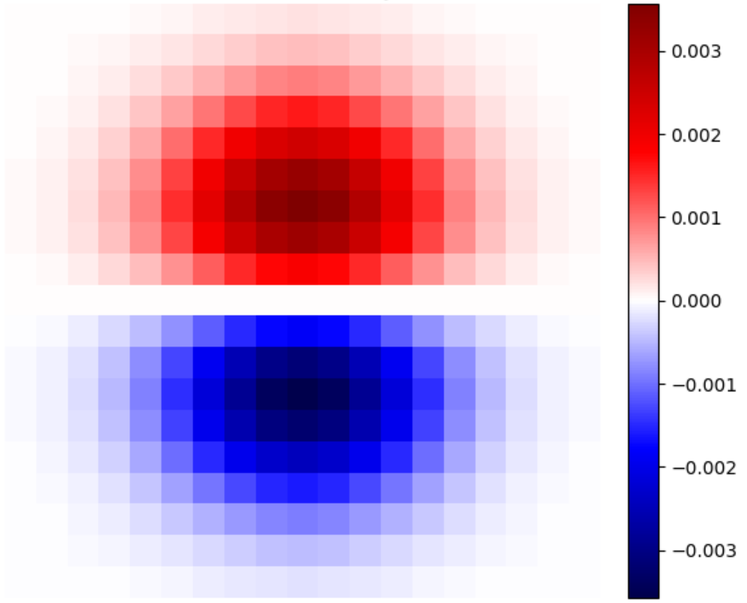


Steerable Filters

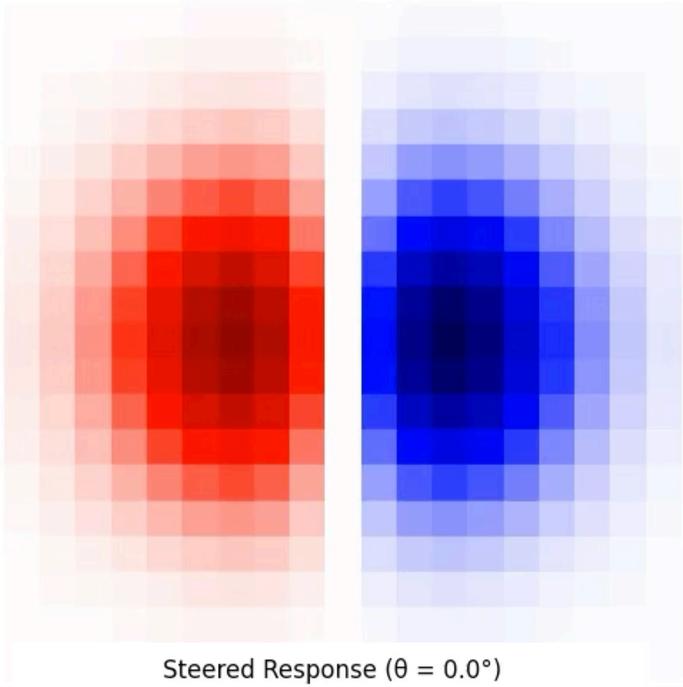
Basis Filter: G_x



Basis Filter: G_y



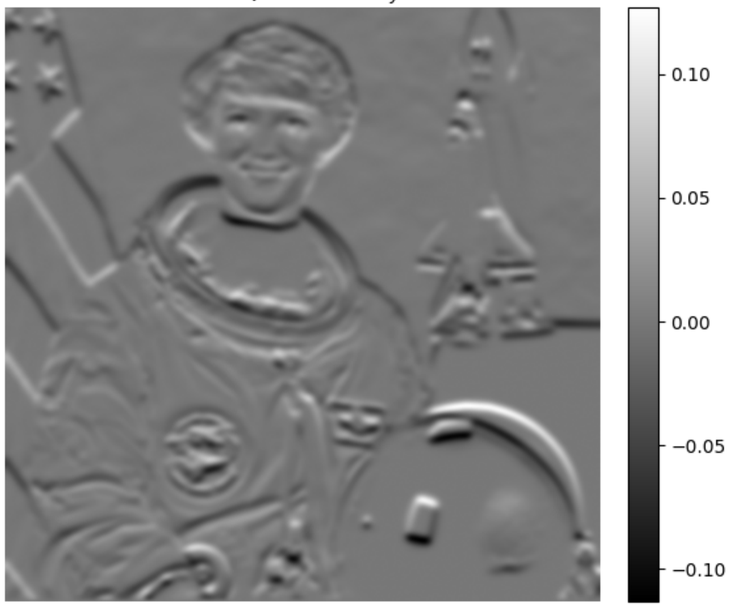
Steered Filter (θ = 0.0°)



Response to G_x



Response to G_y

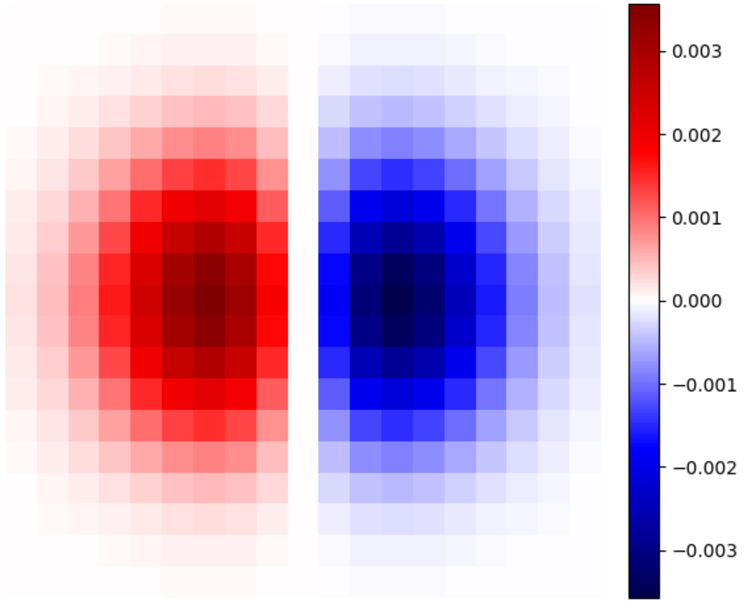


Steered Response (θ = 0.0°)

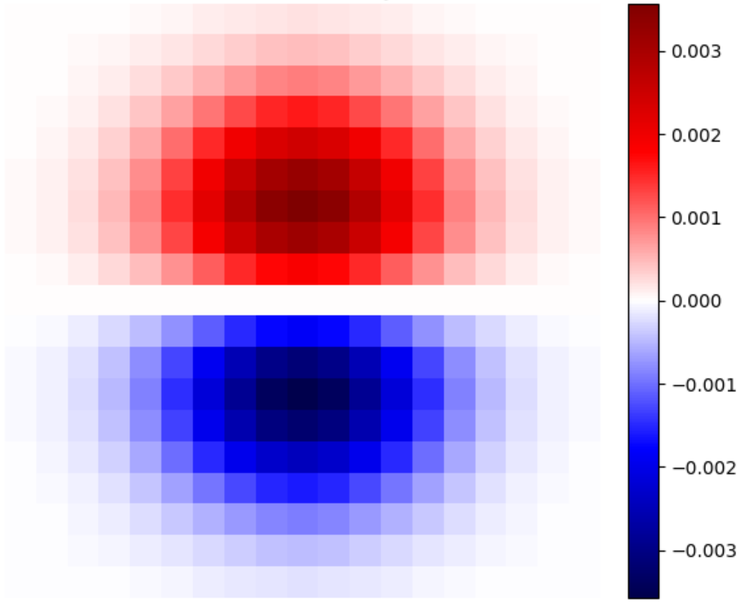


Steerable Filters

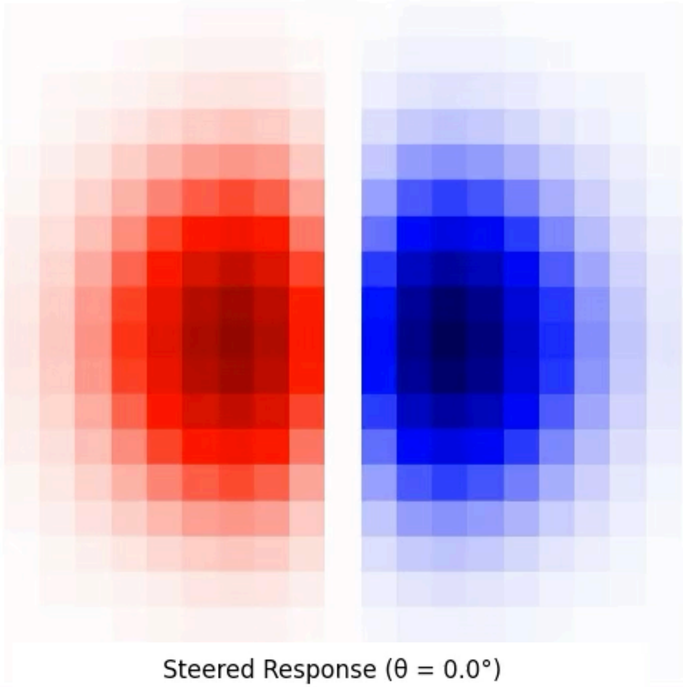
Basis Filter: G_x



Basis Filter: G_y



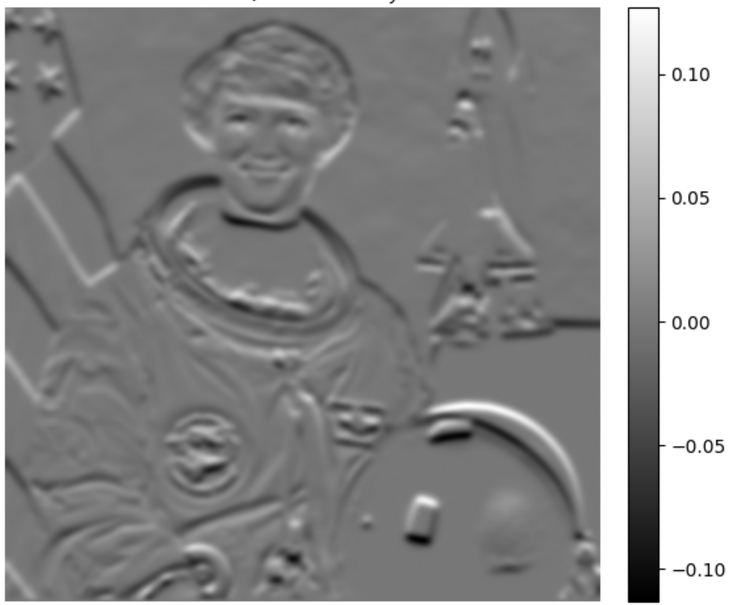
Steered Filter (θ = 0.0°)



Response to G_x



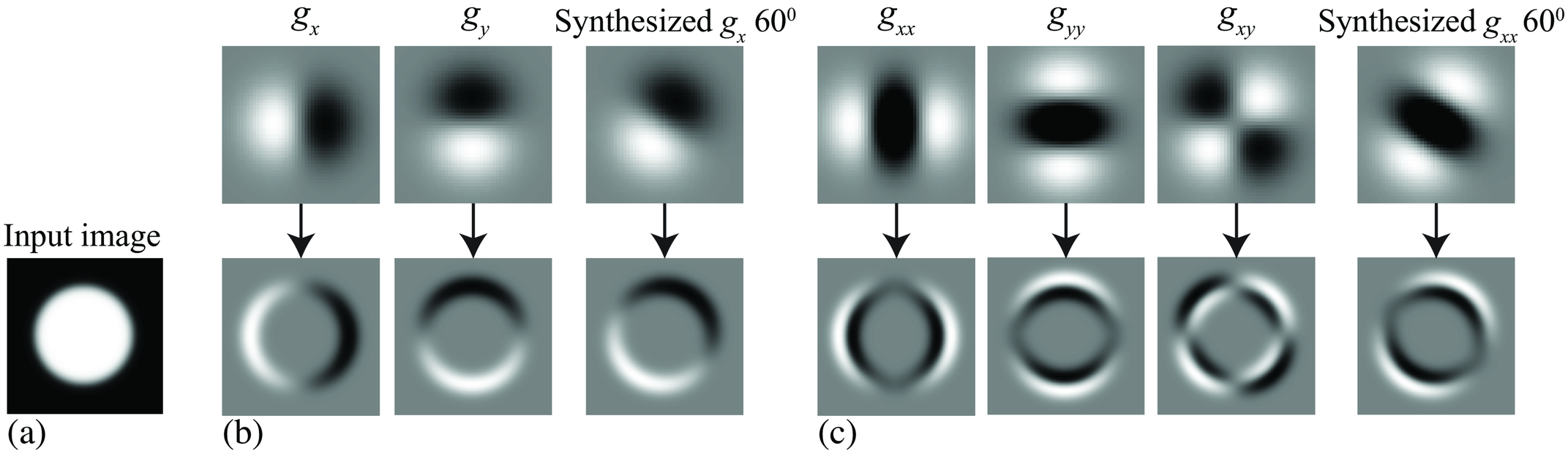
Response to G_y



Steered Response (θ = 0.0°)



Basis not always rotationally symmetric



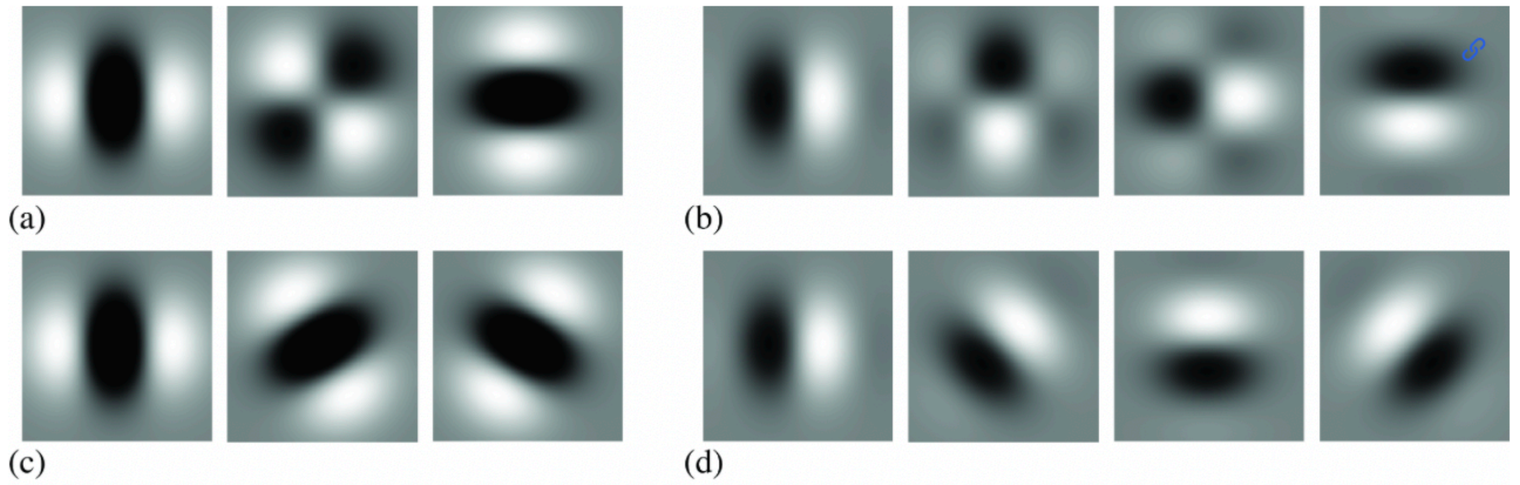


Figure 22.12: (a) Second derivative of Gaussian, $x - y$ separable steerable basis set. b) Approximation to Hilbert transform of second derivative of Gaussian, $x - y$ steerable basis set. (c) Nonseparable basis equivalent to (a). (d) Nonseparable basis set equivalent to (b).

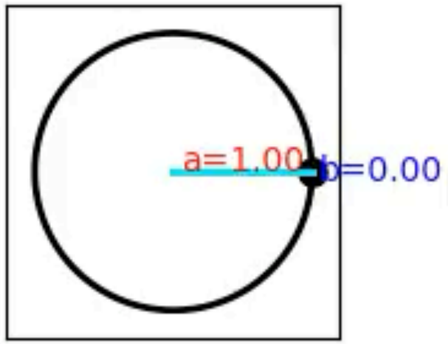
https://visionbook.mit.edu/spatial_filter_sets.html

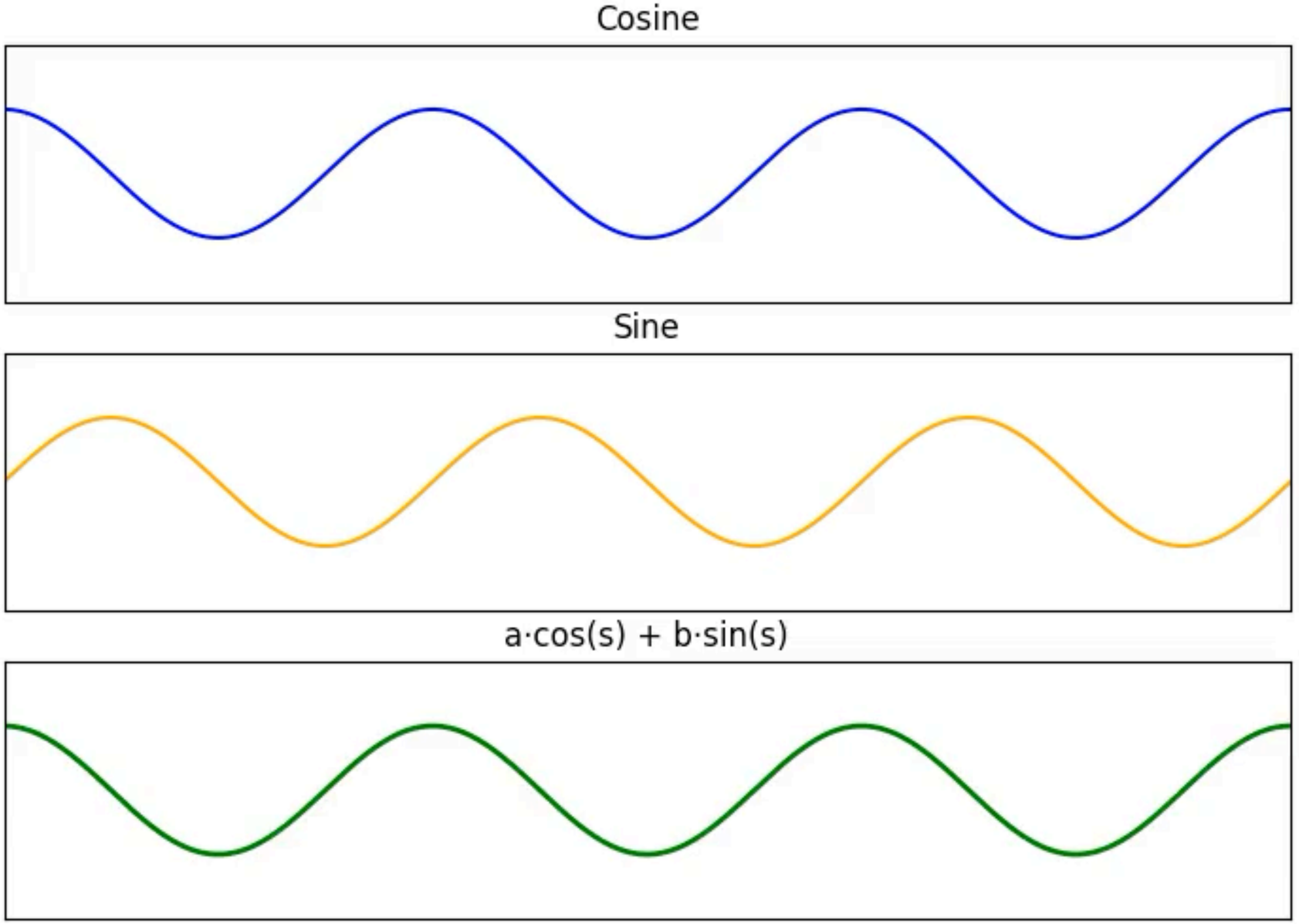
Steerable Bases: shift

Express in
(cos, sin) basis

Translation group acts
via *rotation* of
coefficients!

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



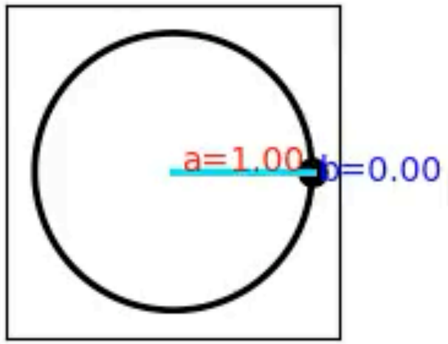


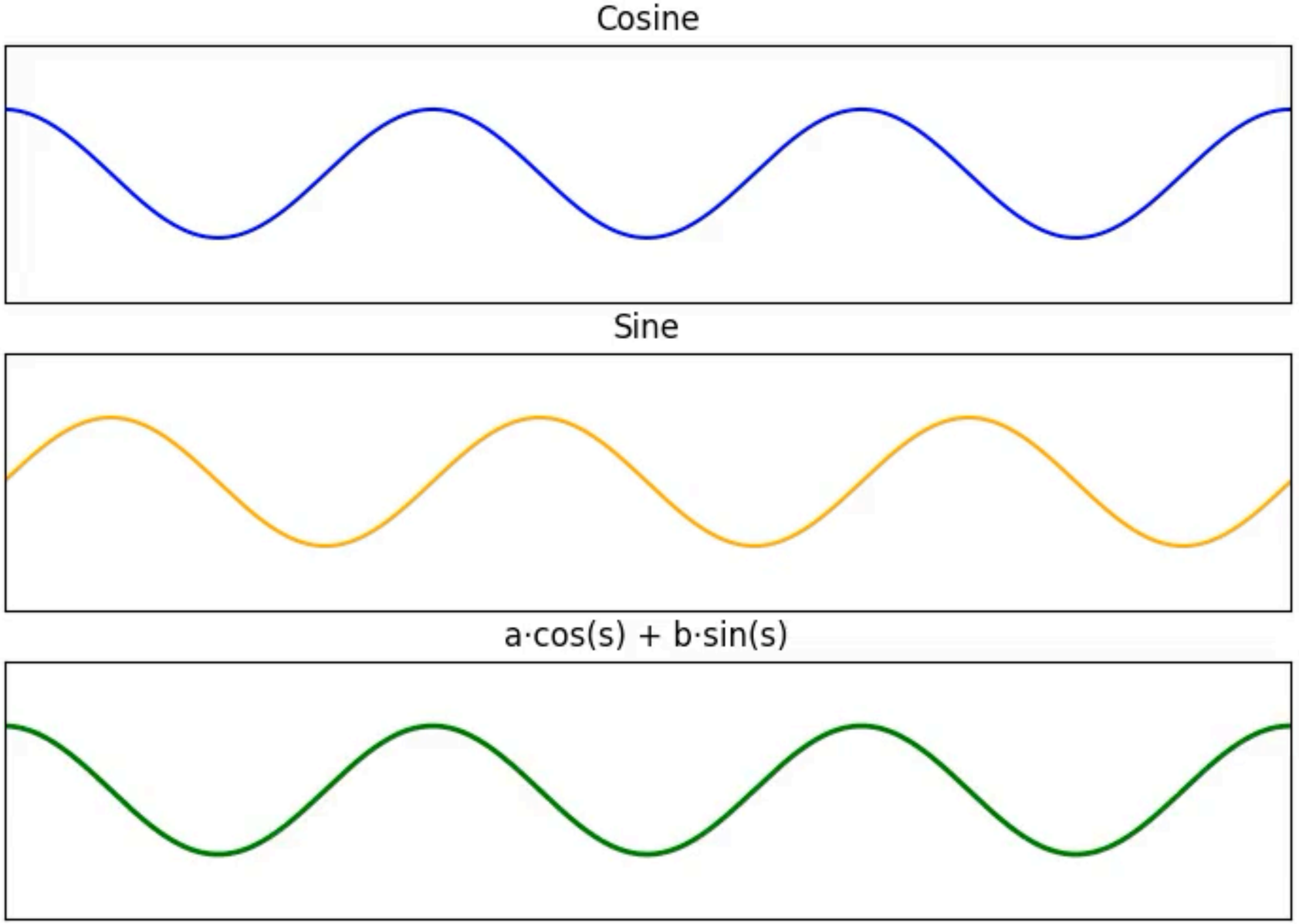
Steerable Bases: shift

Express in
(cos, sin) basis

Translation group acts
via *rotation* of
coefficients!

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





More high level discussion

- Transformations (rotations, translations, etc.)
  and operators (e.g. filter, Laplacian)
- Various good properties:
  - Steerability for one vector
    (for each input vector, there exists discrete basis that gives output for full group)
  - Global steerability (same basis for all vectors)
    - Even better : diagonal steerability (true for shifts but not SO3 rotations)
    - Basis is orthogonal
  - Equivariance : transformation commutes with operator
    (no notion of basis / representation)

Commutativity

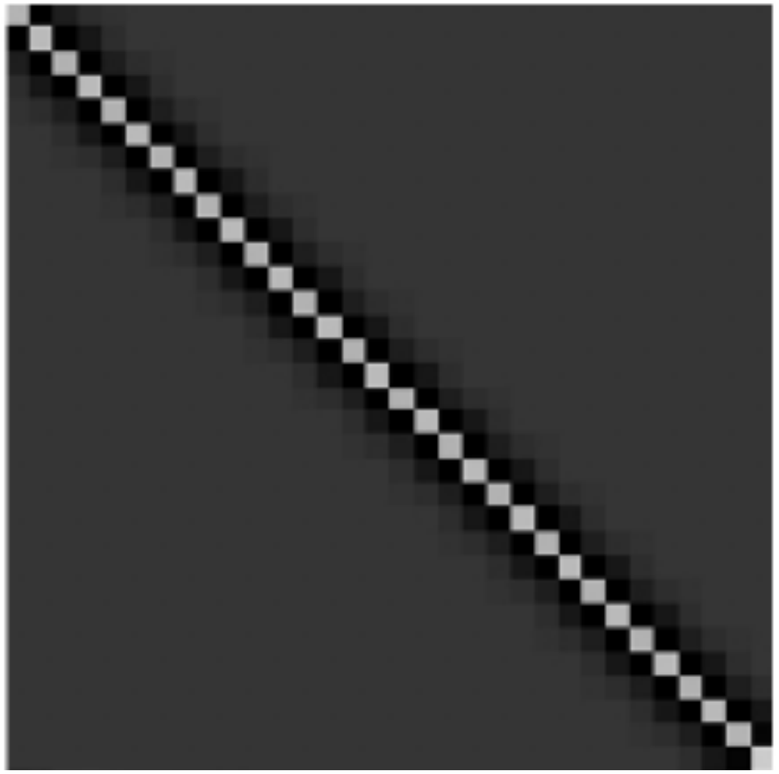
- a b = b a
  - multiplication on scalars (yes!)
    matrices (usually no!)
- Connection to eigenbases
  - If share the same eigenbasis, then commutative
    - Diagonalization brings us back to scalars!
    - But not a requirement
- A group where elements commute is called Abelian
  (another 19th century mathematician!)

Recipe for finding a (globally) 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

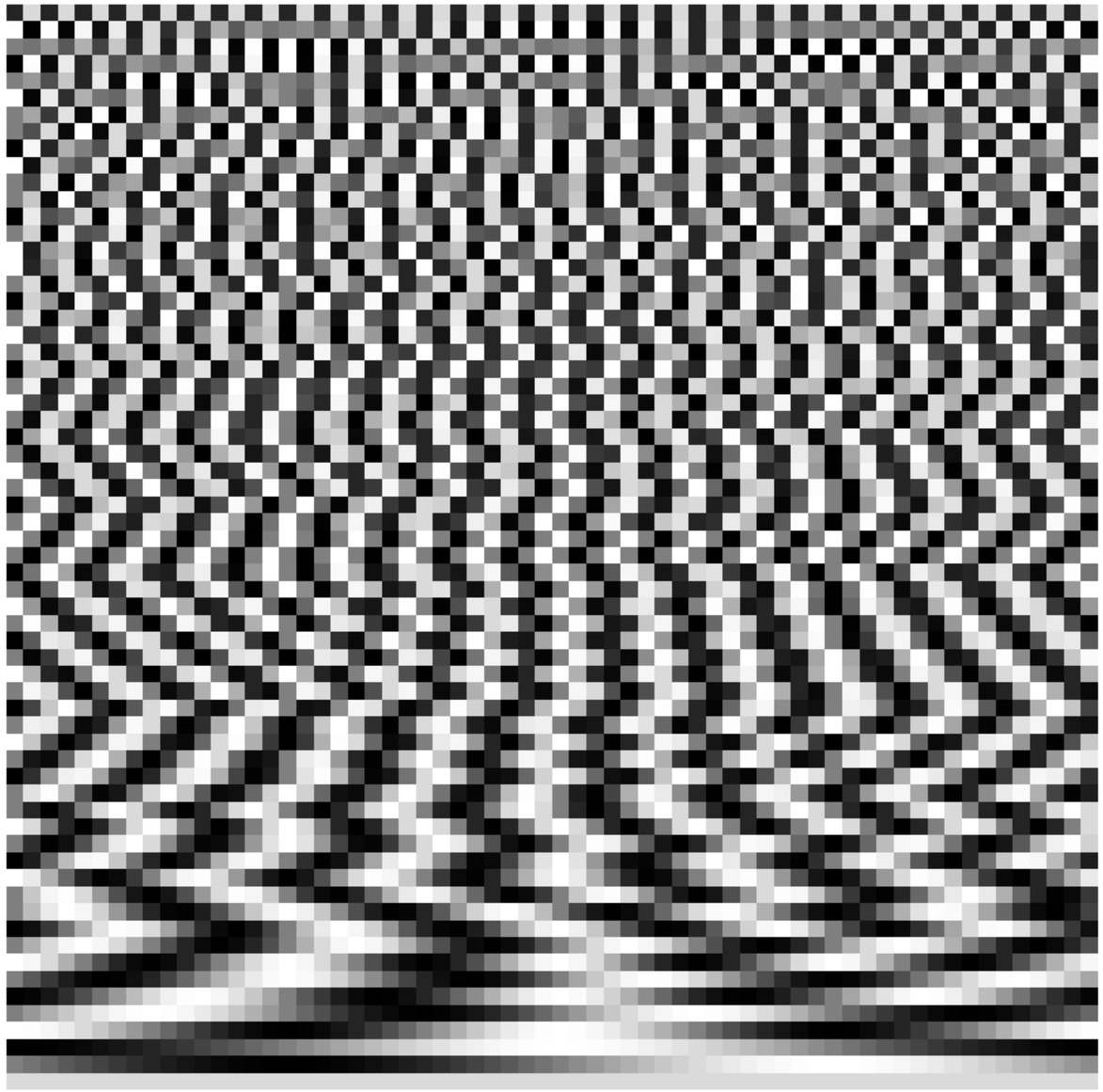
n = 128

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

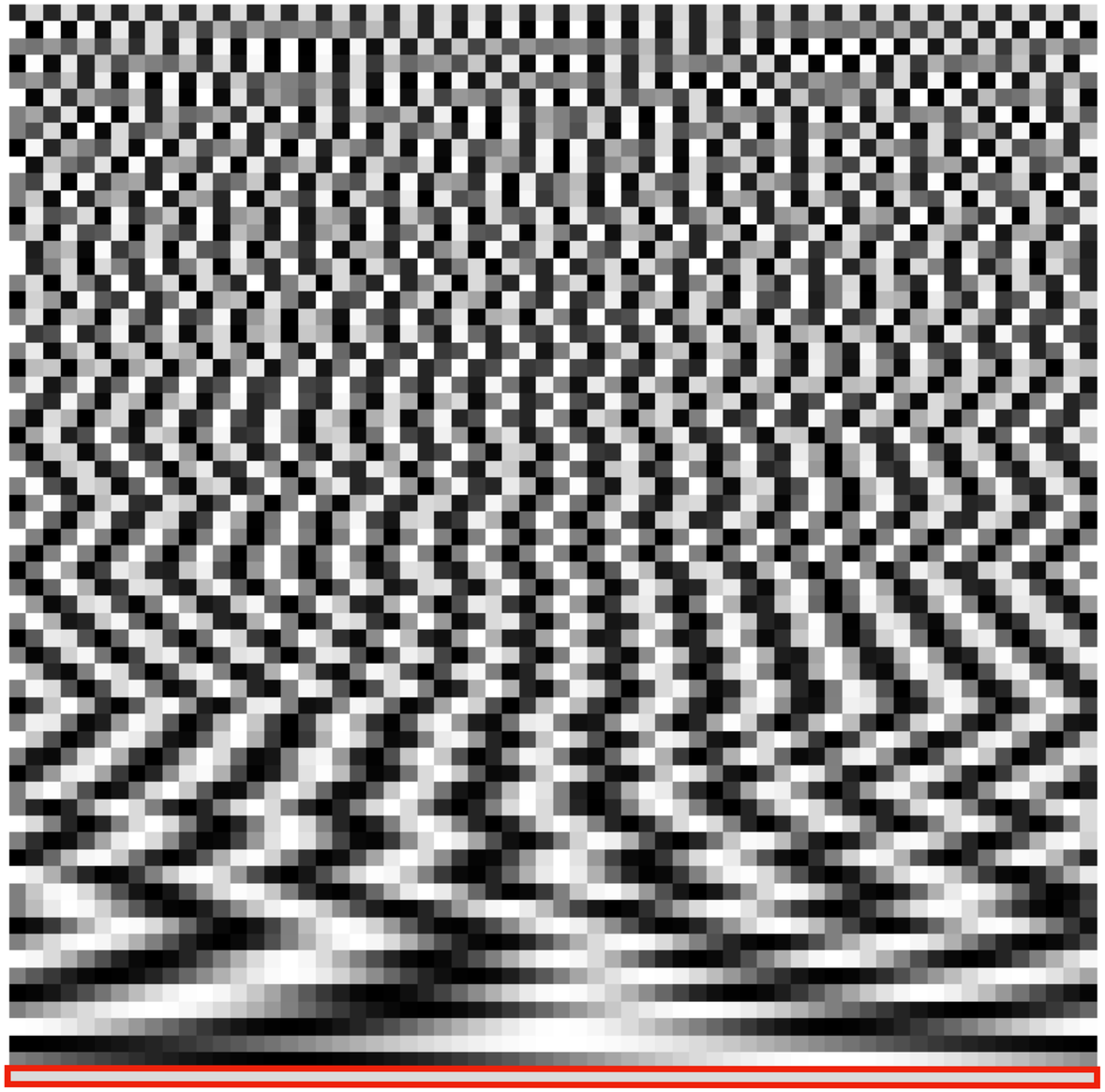
L = np.diag(main_diag) + np.diag(off_diag, 1) + np.diag(off_diag, -1)
L[0, -1] = 1
L[-1, 0] = 1

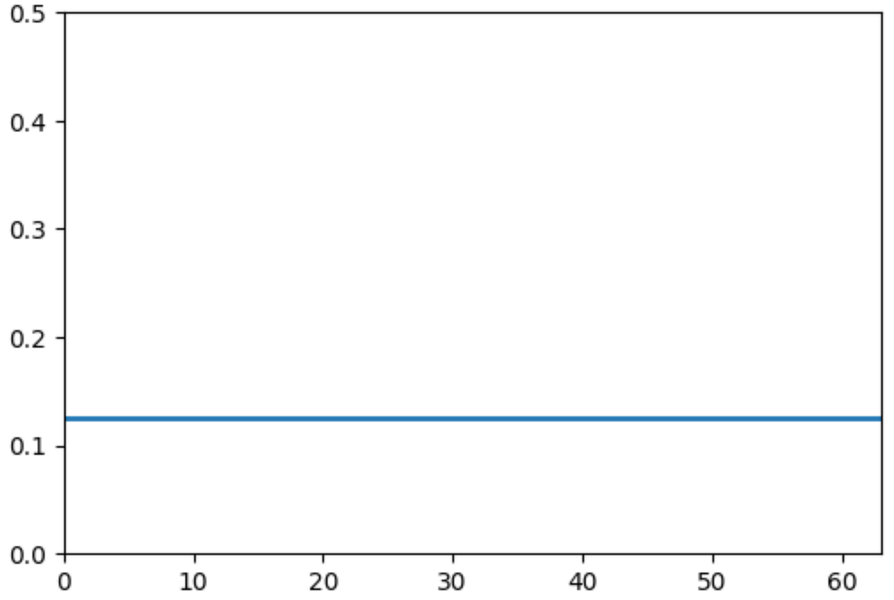
# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eigh(L)
```

Eigenfunctions

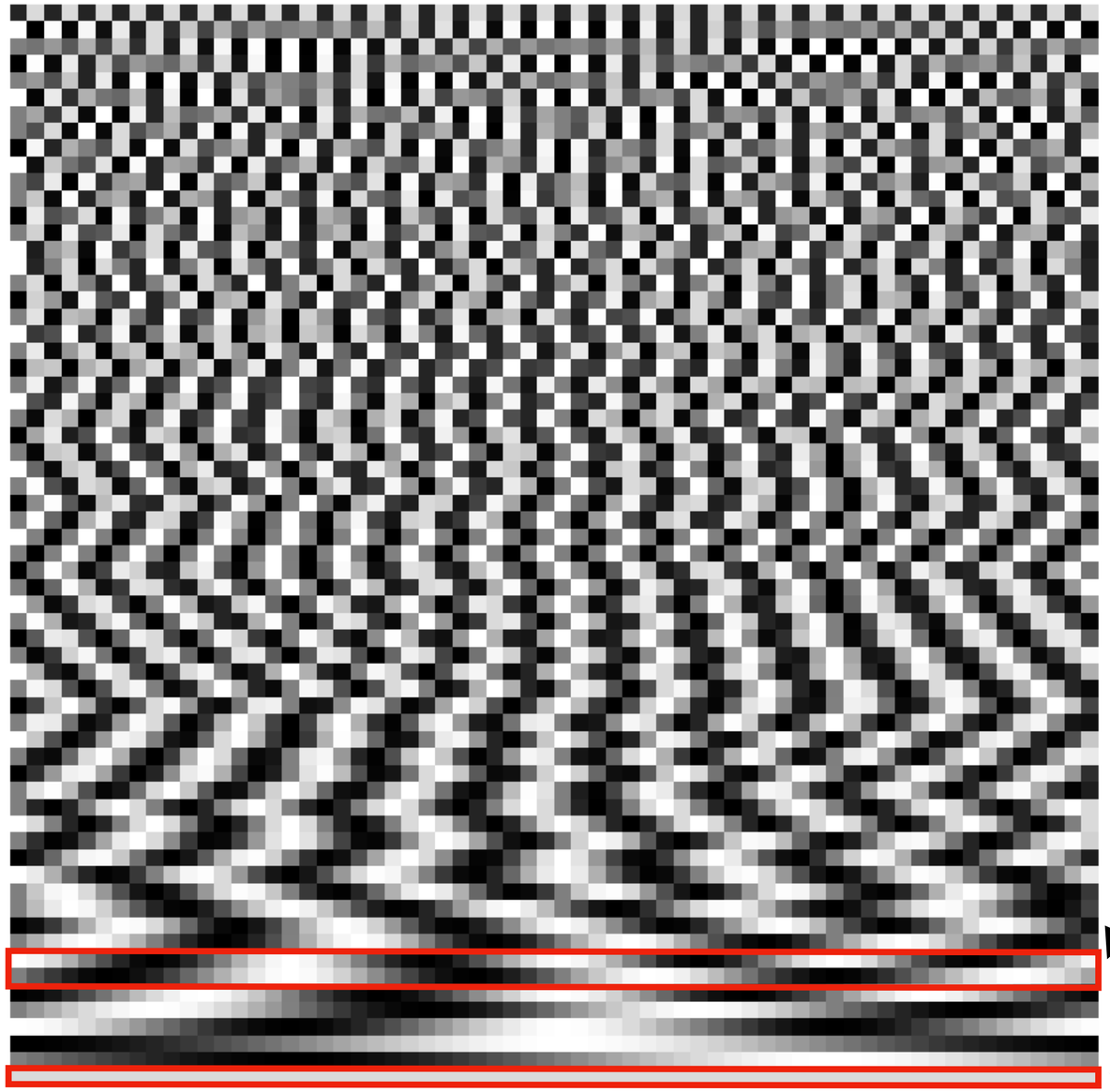


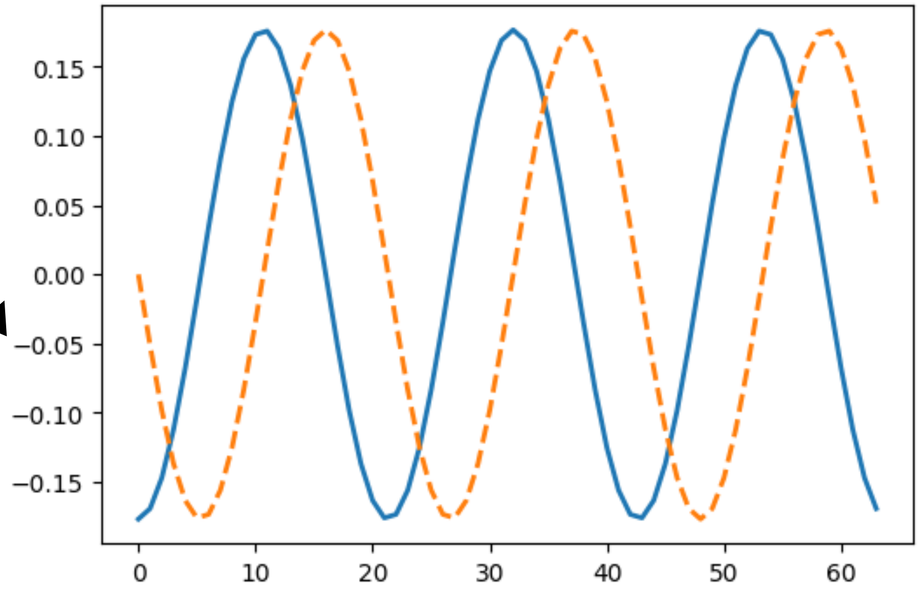
Eigenfunctions

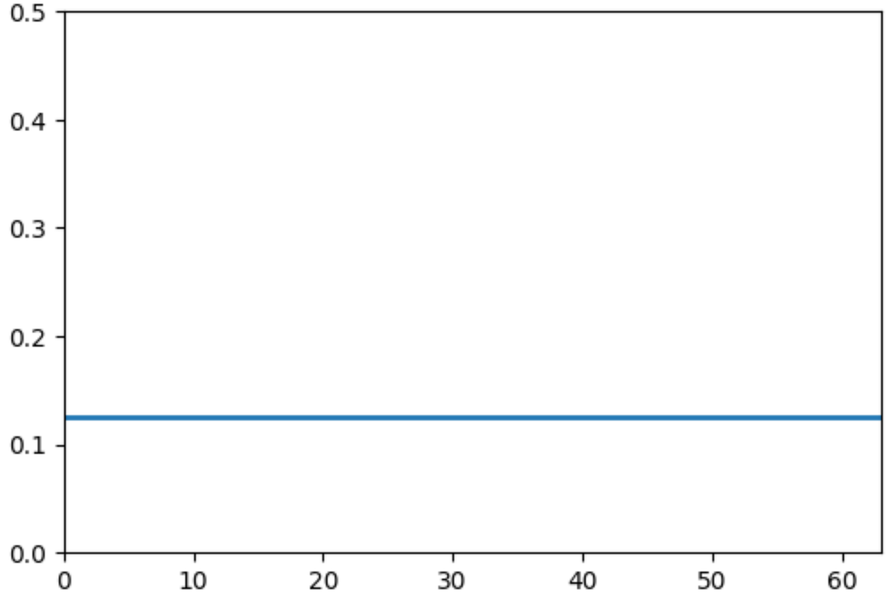




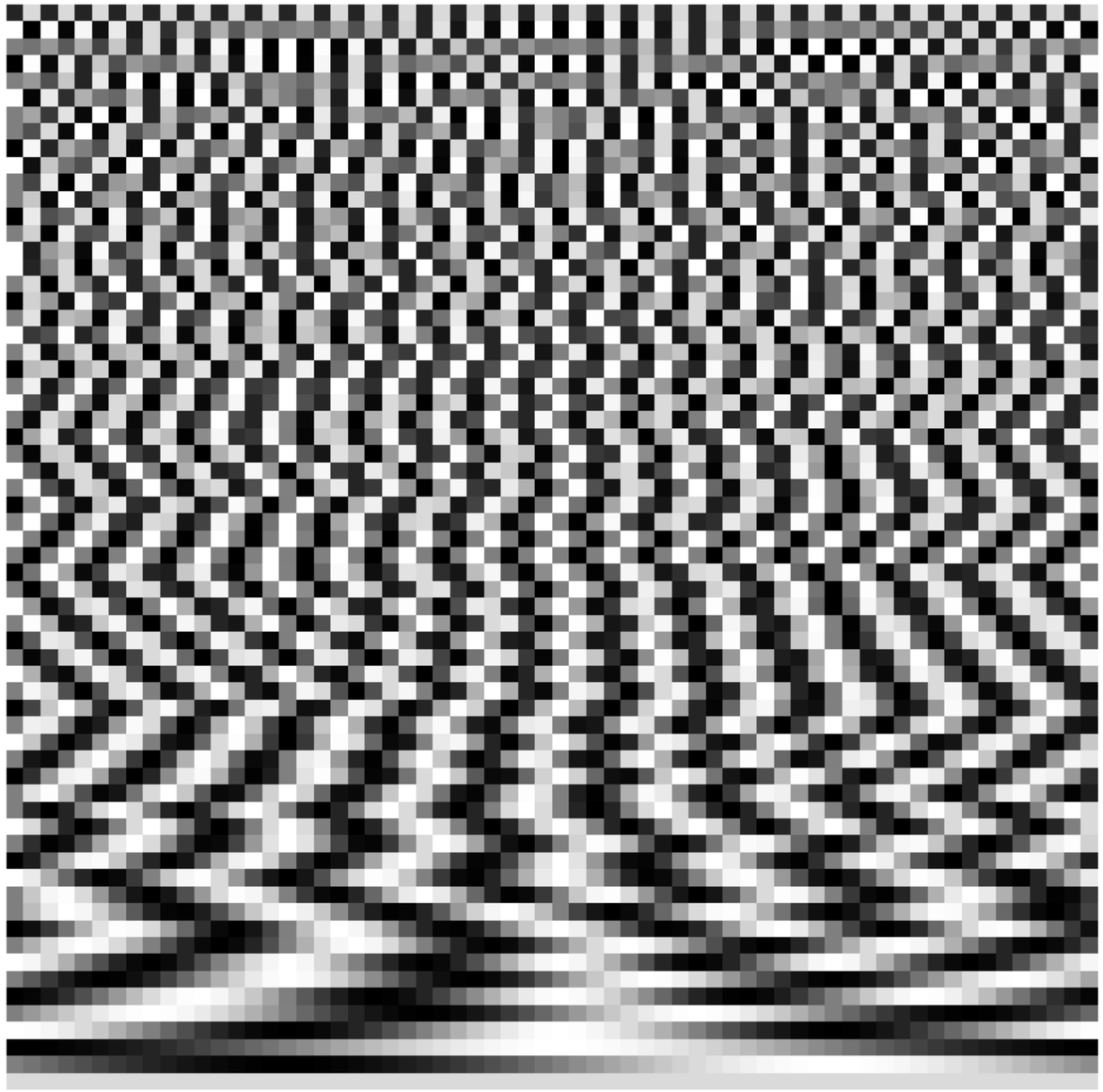
Eigenfunctions





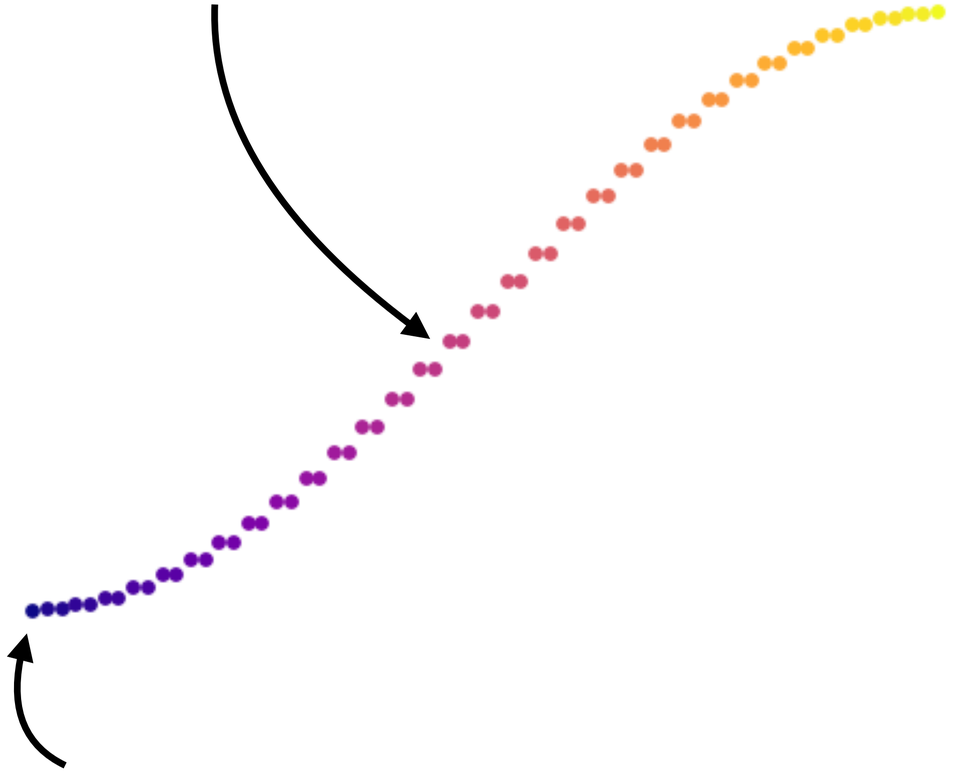


Eigenfunctions



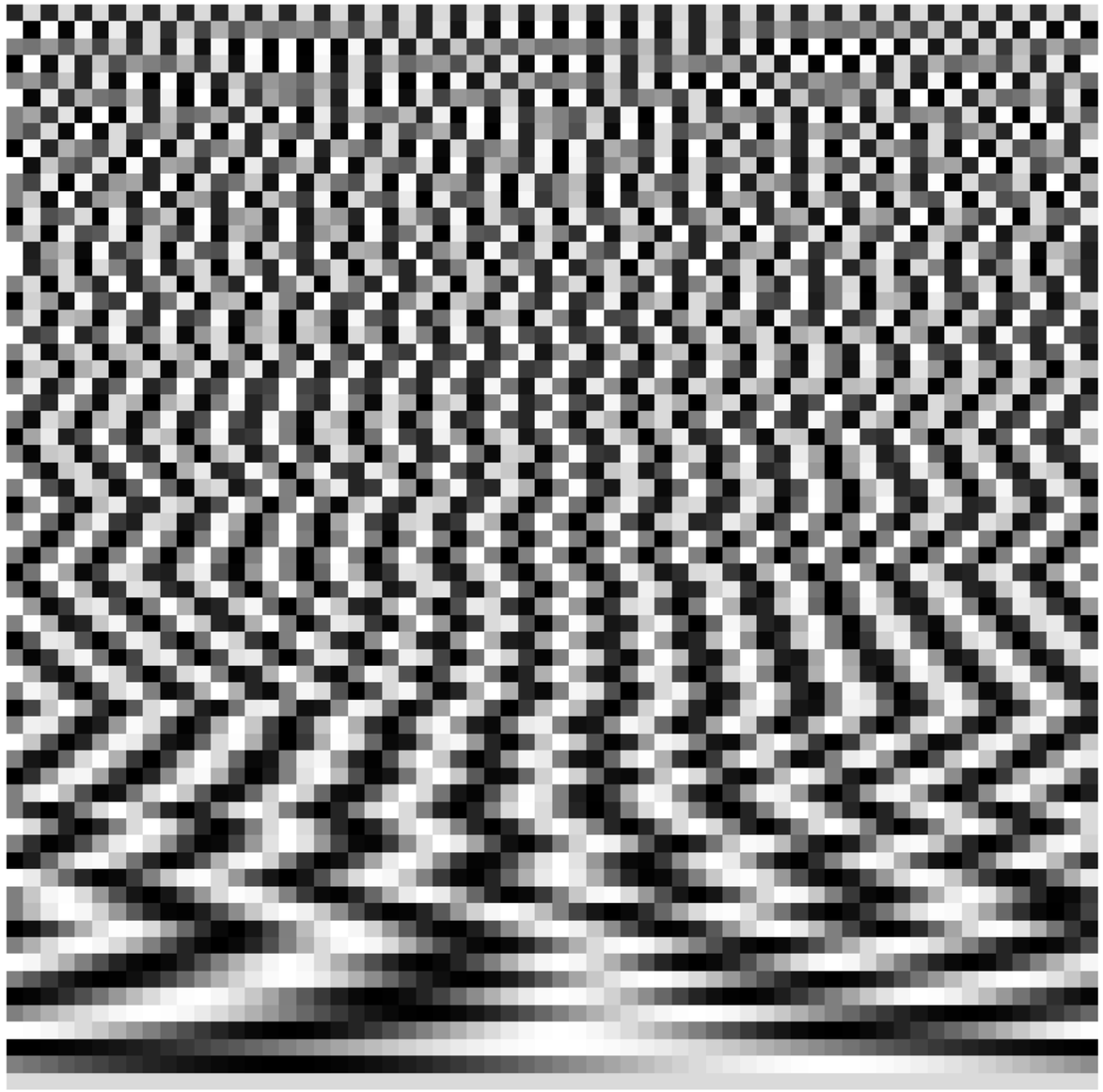
Eigenvalues

Then, pairs of equal eigenvalues



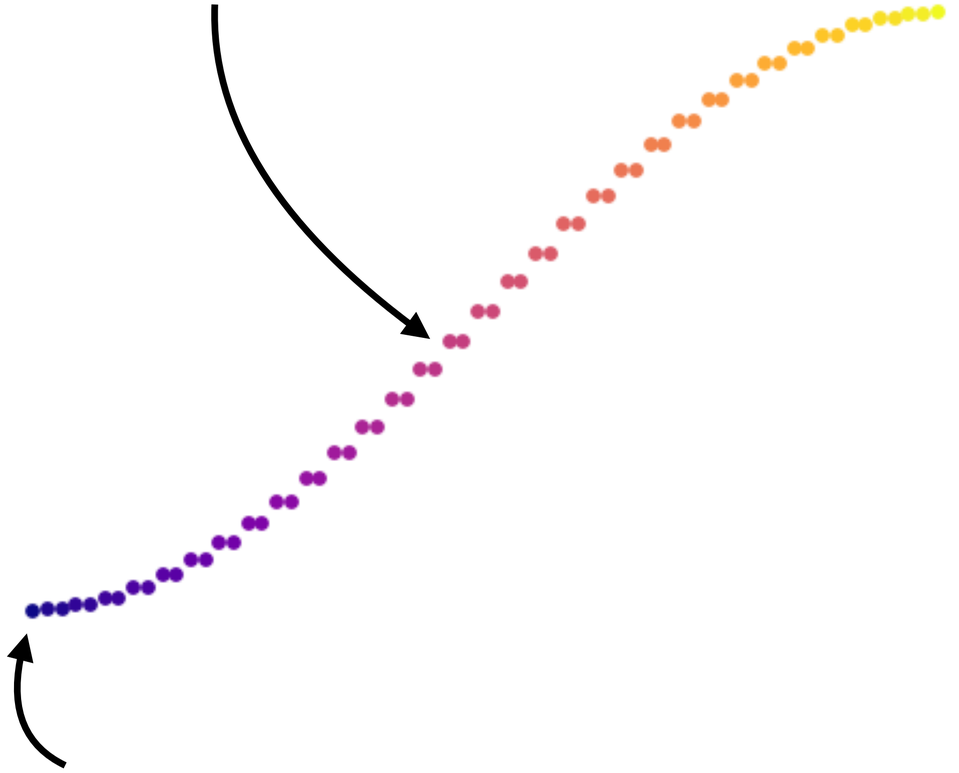
One “zero” eigenvalue

Eigenfunctions



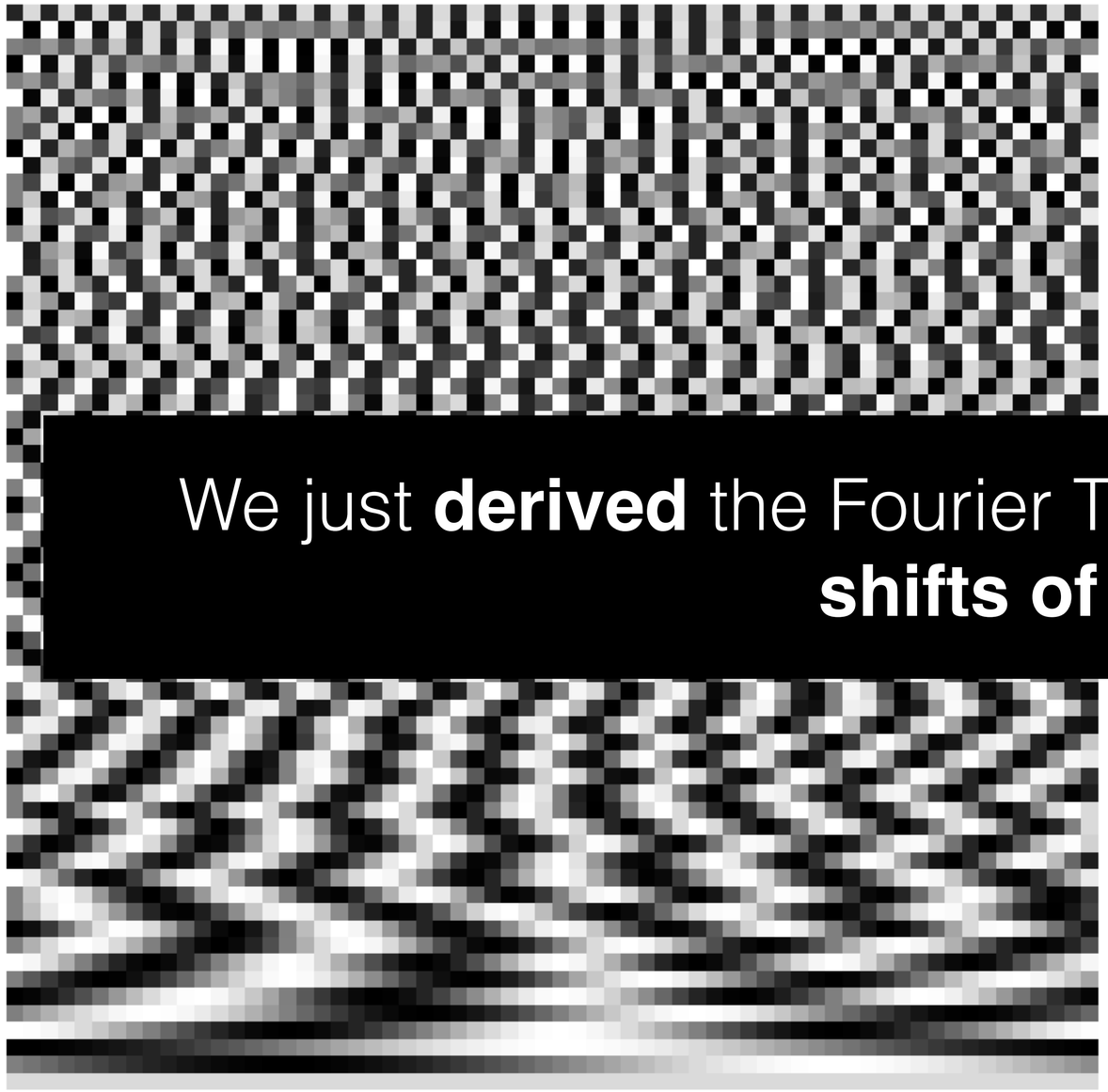
Eigenvalues

Then, pairs of equal eigenvalues



One “zero” eigenvalue

Eigenfunctions



Eigenvalues

Then, pairs of equal eigenvalues

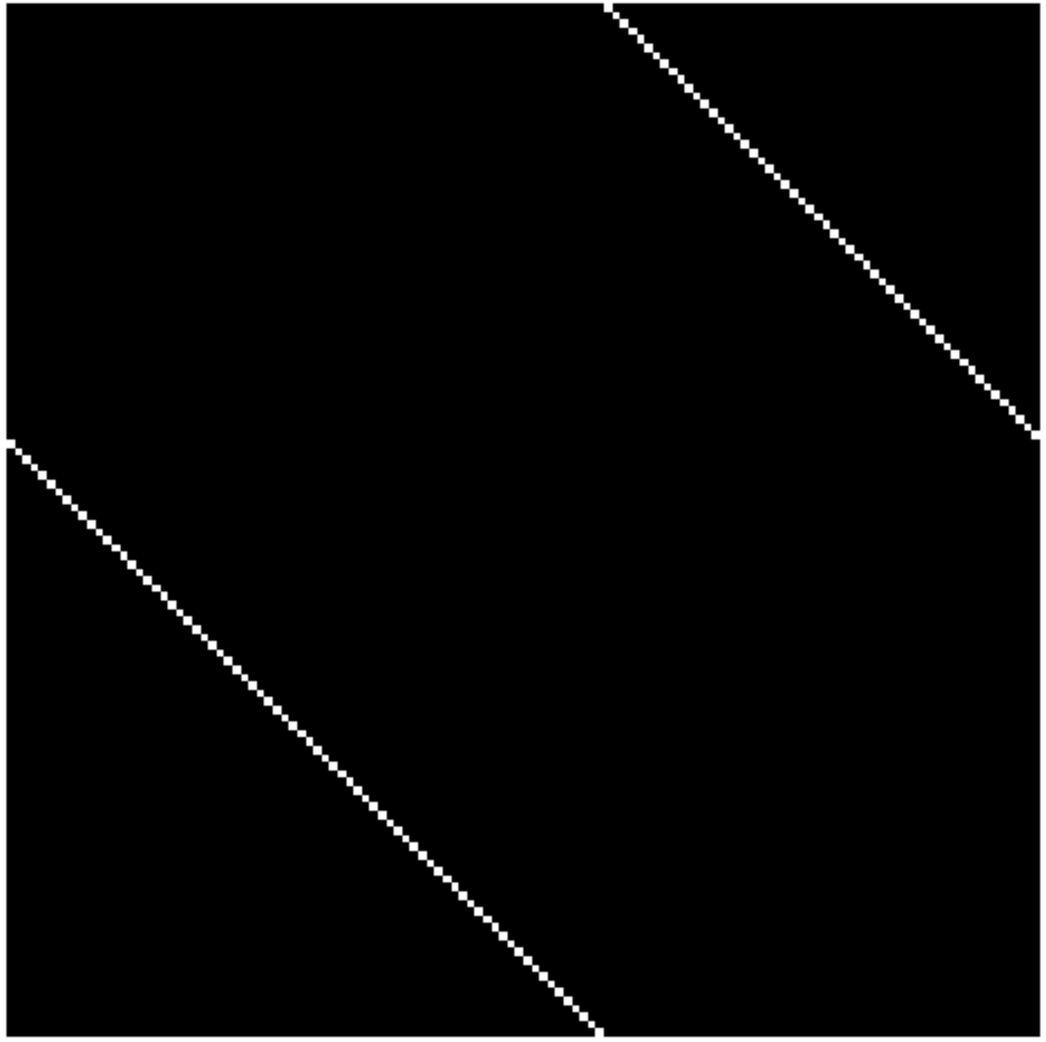


One “zero” eigenvalue

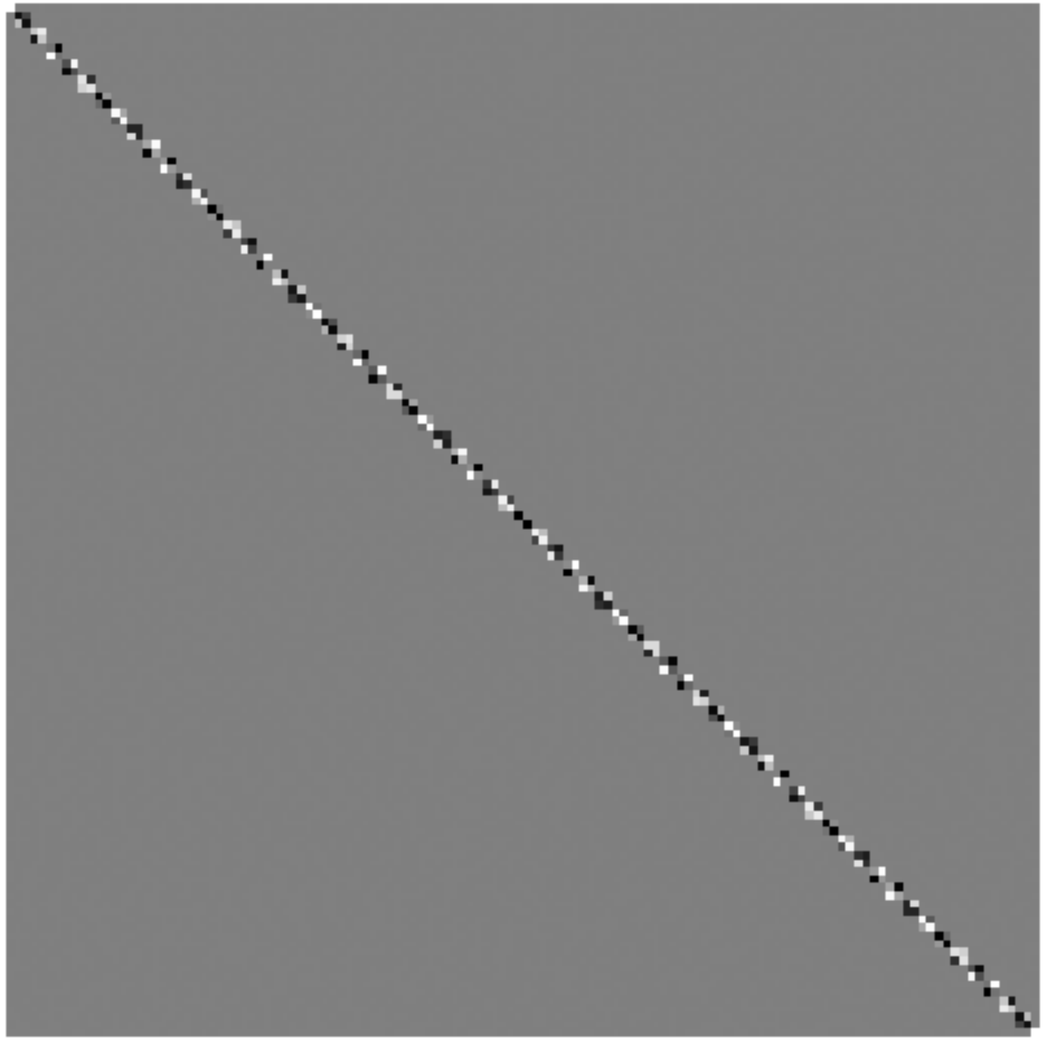
We just **derived** the Fourier Transform only from considering **shifts of functions**!

Recall: Circular Shift in Fourier Basis

$\mathbf{T}$

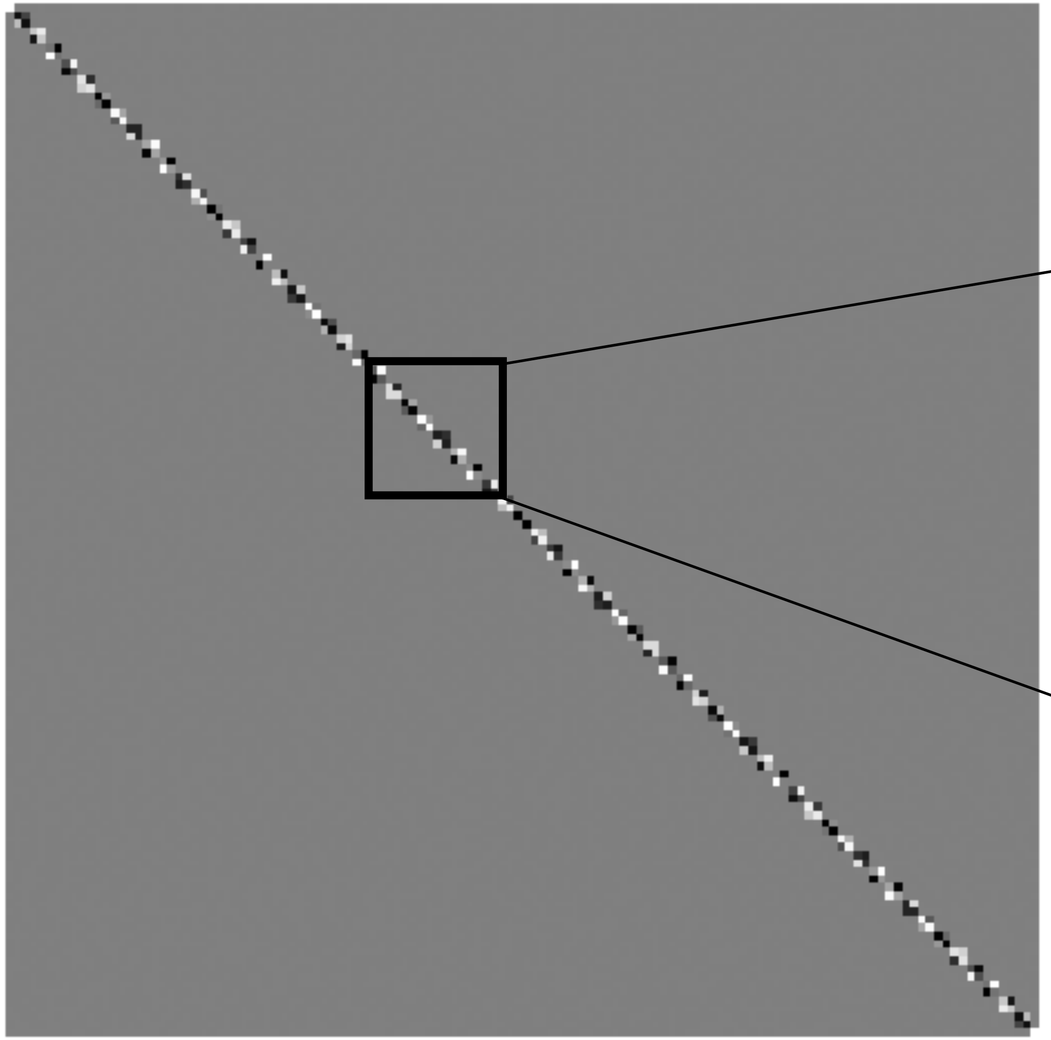


$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$



Recall: Circular Shifts in Fourier Basis

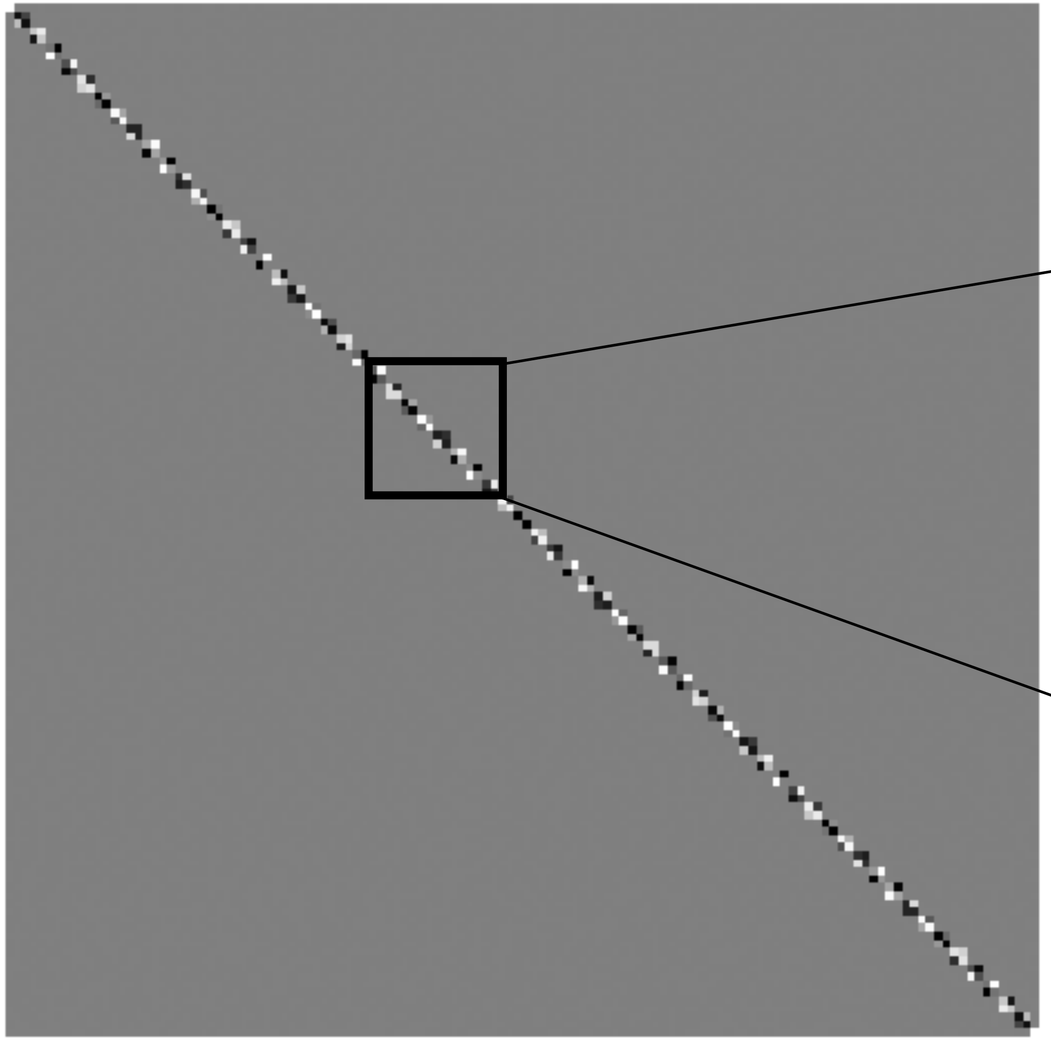
$$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$$



$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos\theta_3 & -\sin\theta_3 \\ 0 & 0 & 0 & 0 & \sin\theta_3 & \cos\theta_3 \end{bmatrix}$$

Recall: Circular Shifts in Fourier Basis

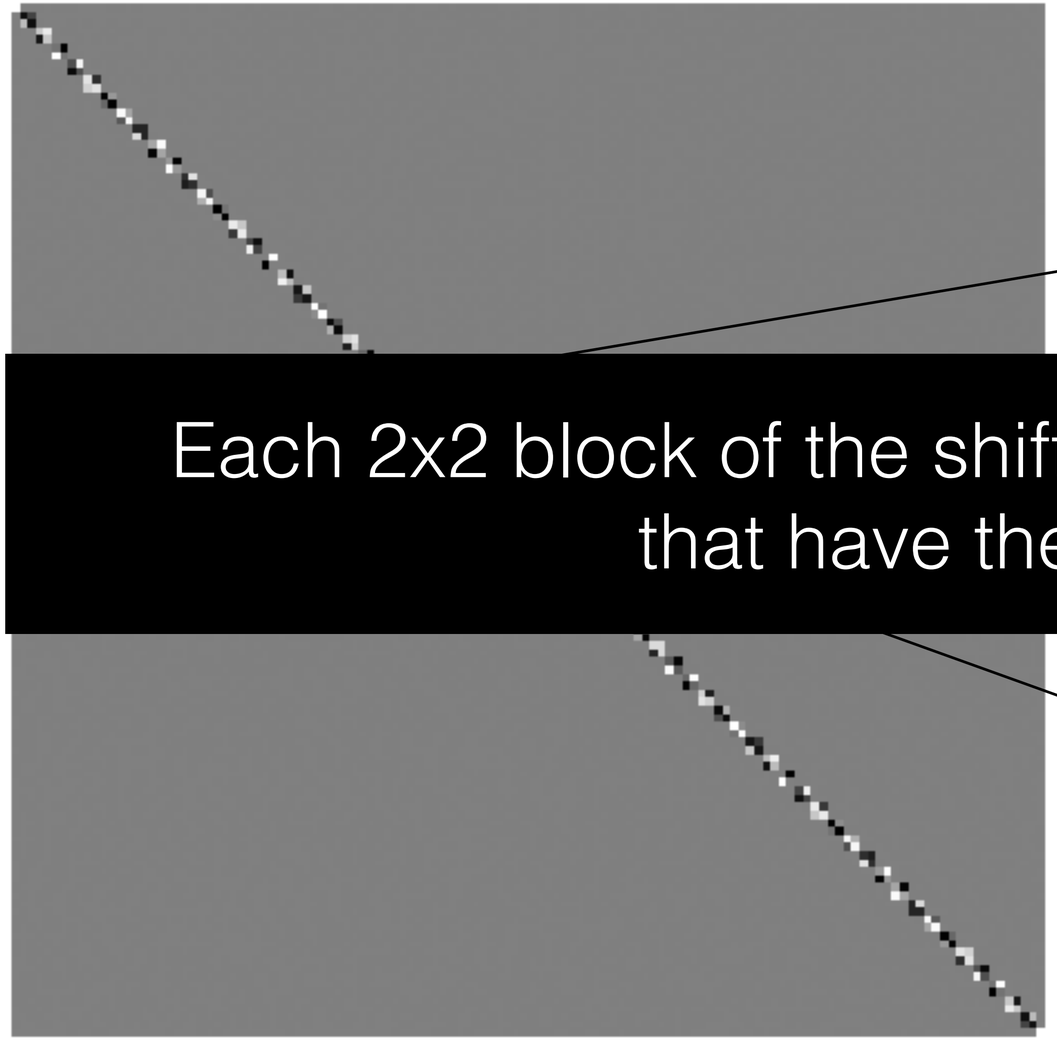
$$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$$



$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos\theta_3 & -\sin\theta_3 \\ 0 & 0 & 0 & 0 & \sin\theta_3 & \cos\theta_3 \end{bmatrix}$$

Recall: Circular Shifts in Fourier Basis

$$\mathcal{F}\mathbf{T}\mathcal{F}^{-1}$$

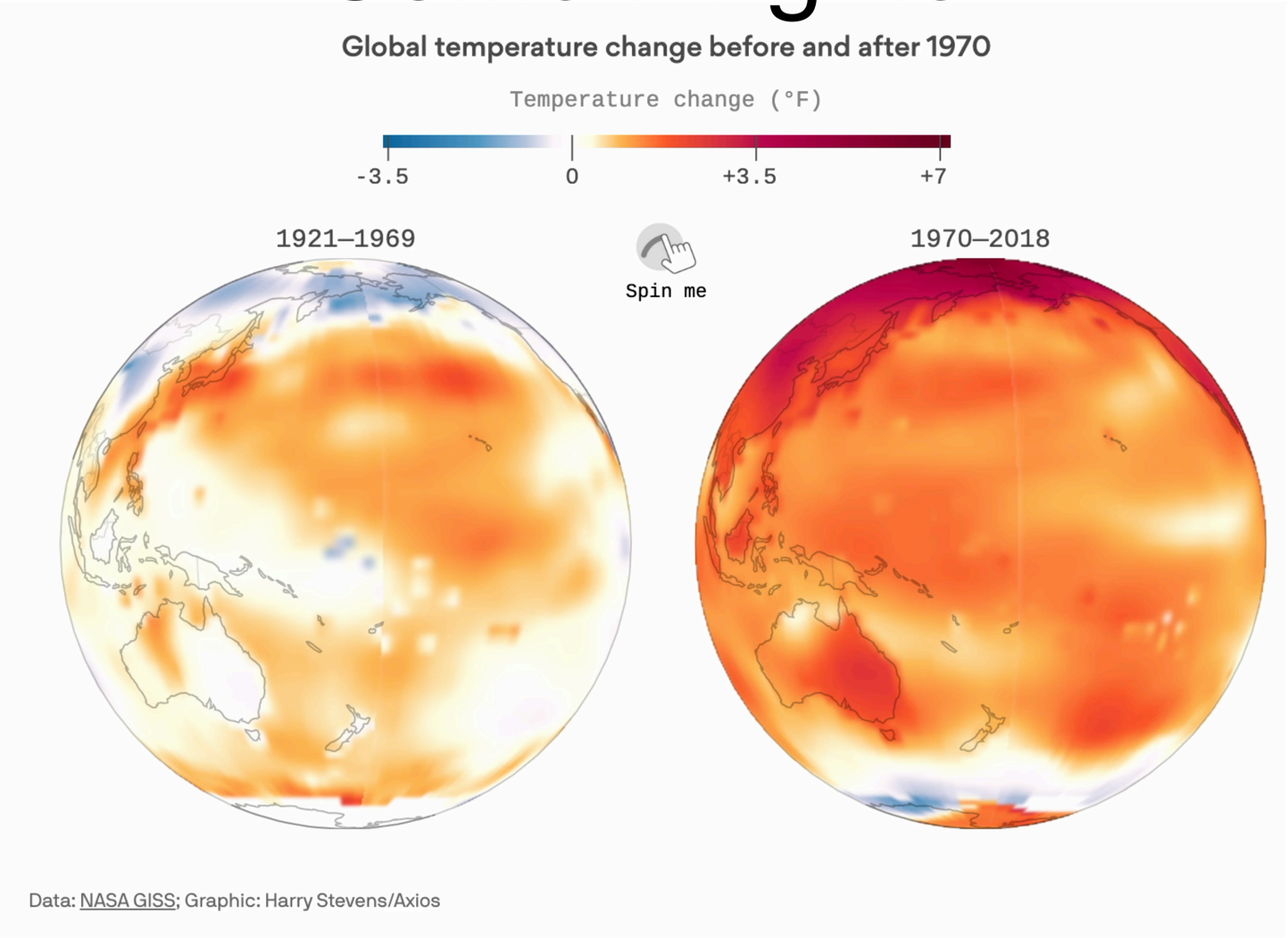


$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 & 0 & 0 \\ & & & & & \\ & & & & & \\ & & & & & \\ 0 & 0 & 0 & 0 & \sin\theta_3 & \cos\theta_3 \end{bmatrix}$$

Each 2x2 block of the shift operator only mixes coefficients that have the same eigenvalue!

Spherical harmonics

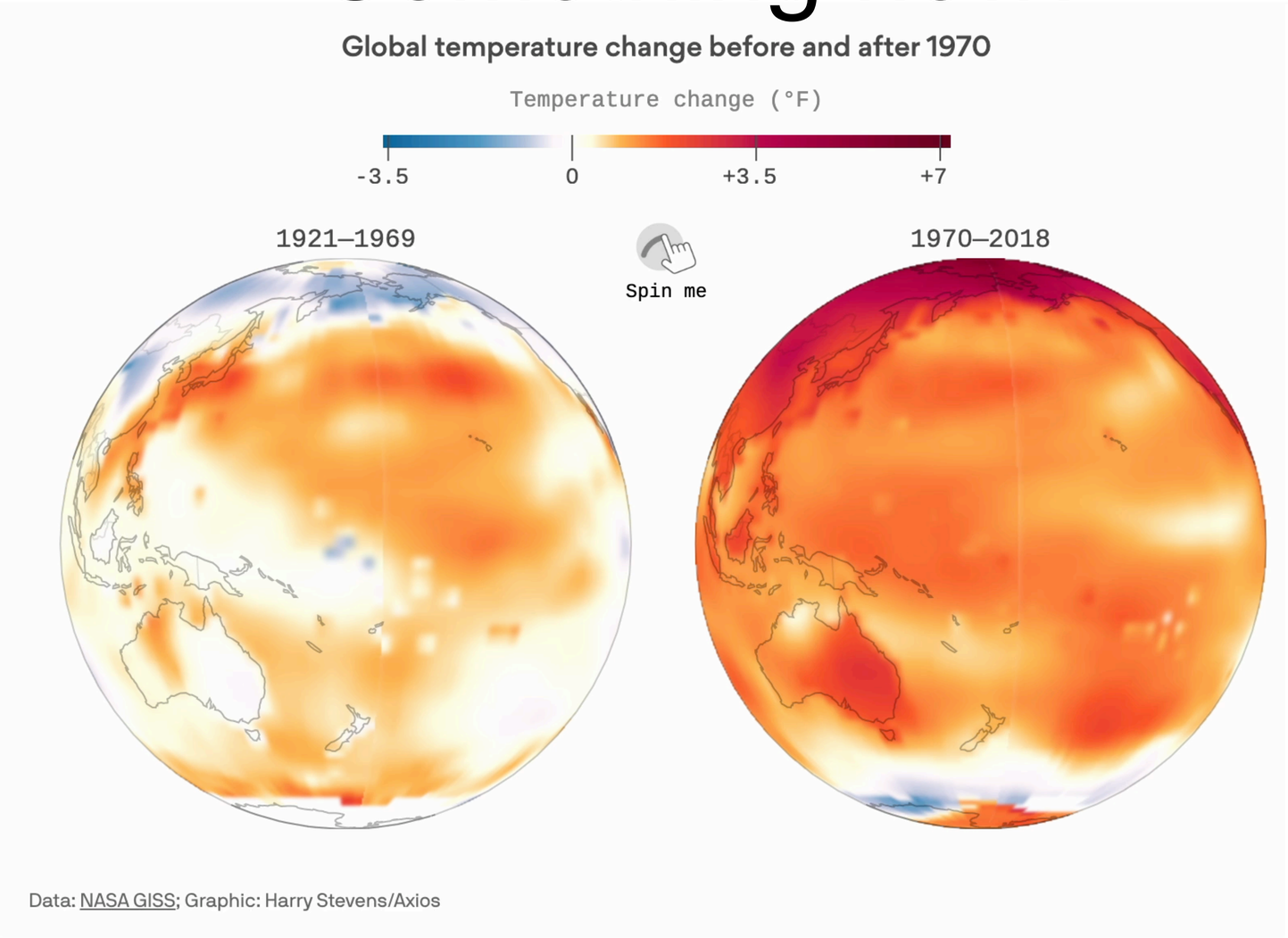
Something new!



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

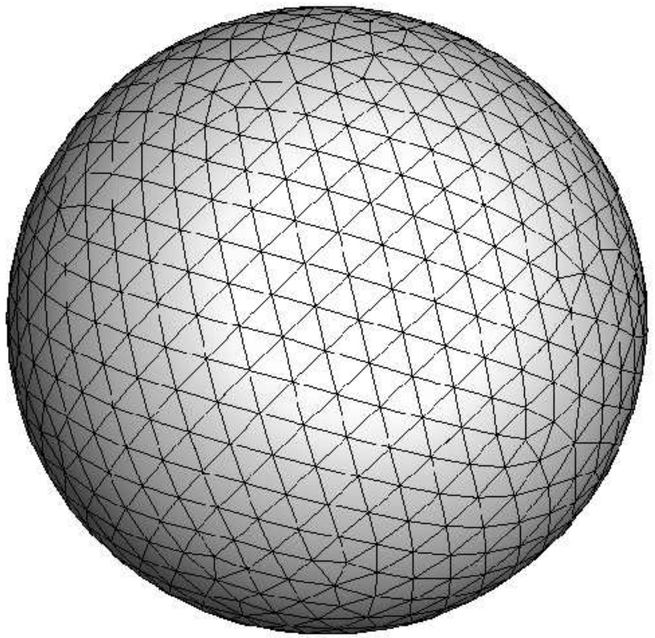
Is there a *rotation-steerable basis* for these functions?

Recipe for finding a steerable basis

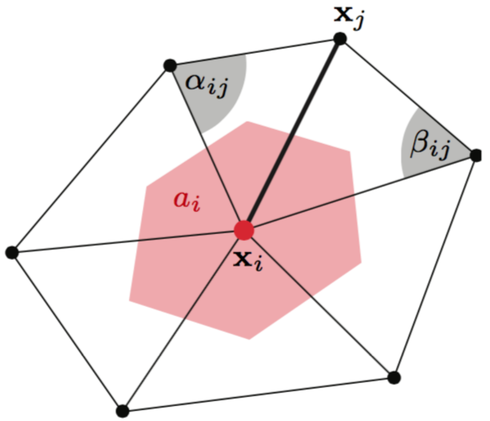
1. Find a *linear operator* such that…
    1. …it *commutes* with the group action. This means that its eigenspace is unchanged under the group action.
    2. …its eigenvectors are guaranteed to be a complete basis for the function space (it is self-adjoint).
2. Compute its Eigendecomposition.
3. The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

Cutting to the Chase: The Laplace-Beltrami Operator

Equivalent of Laplacian on surfaces



$$\Delta f := \operatorname{div}(\nabla f),$$



On meshes:

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

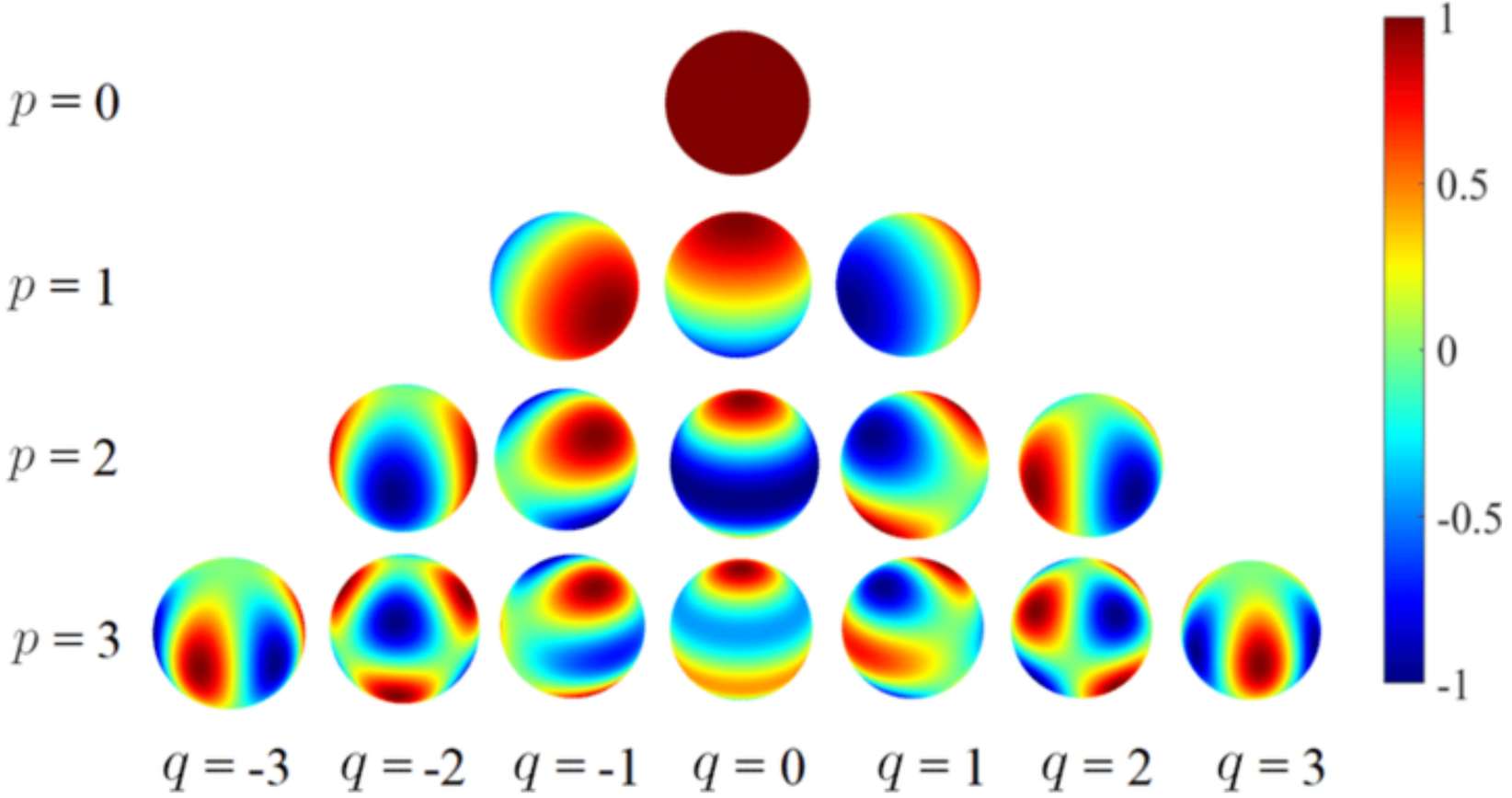
$$L = MC$$

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

Eigenbasis of Laplace Beltrami on sphere ?

- Spherical harmonics !

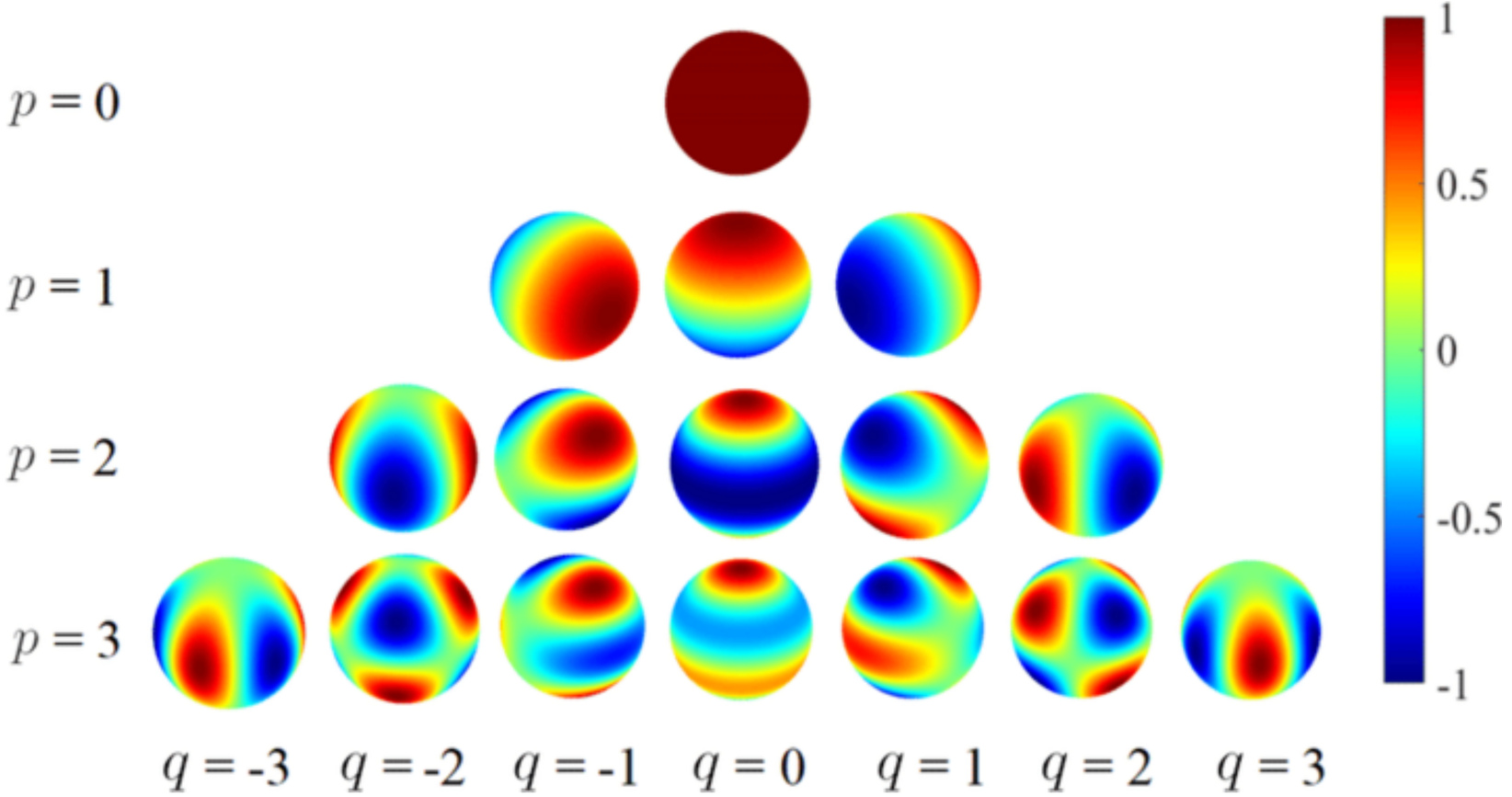




Merci Laplace

The Spherical Harmonics

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



Formulas

- Combination of complex exponentials and Legendre polynomials

$$Y_0^0(\theta,\phi)=\frac{1}{2}\,\frac{1}{\sqrt{\pi}}$$

$$Y_1^{-1}(\theta,\phi)=\frac{1}{2}\sqrt{\frac{3}{2\pi}}\,\sin\theta\,e^{-i\phi}$$

$$Y_1^0(\theta,\phi)=\frac{1}{2}\sqrt{\frac{3}{\pi}}\,\cos\theta$$

$$Y_1^1(\theta,\phi)=-\frac{1}{2}\sqrt{\frac{3}{2\pi}}\,\sin\theta\,e^{i\phi}$$

$$Y_2^{-2}(\theta,\phi)=\frac{1}{4}\sqrt{\frac{15}{2\pi}}\,\sin^2\theta\,e^{-2i\phi}$$

$$Y_2^{-1}(\theta,\phi)=\frac{1}{2}\sqrt{\frac{15}{2\pi}}\,\sin\theta\cos\theta\,e^{-i\phi}$$

$$Y_2^0(\theta,\phi)=\frac{1}{4}\sqrt{\frac{5}{\pi}}\,\left(3\cos^2\theta-1\right)$$

$$Y_2^1(\theta,\phi)=-\frac{1}{2}\sqrt{\frac{15}{2\pi}}\,\sin\theta\cos\theta\,e^{i\phi}$$

$$Y_2^2(\theta,\phi)=\frac{1}{4}\sqrt{\frac{15}{2\pi}}\,\sin^2\theta\,e^{2i\phi}$$

$$Y_3^{-3}(\theta,\phi)=\frac{1}{8}\sqrt{\frac{35}{\pi}}\,\sin^3\theta\,e^{-3i\phi}$$

$$Y_3^{-2}(\theta,\phi)=\frac{1}{4}\sqrt{\frac{105}{2\pi}}\,\sin^2\theta\cos\theta\,e^{-2i\phi}$$

$$Y_3^{-1}(\theta,\phi)=\frac{1}{8}\sqrt{\frac{21}{\pi}}\,\sin\theta\left(5\cos^2\theta-1\right)e^{-i\phi}$$

$$Y_3^0(\theta,\phi)=\frac{1}{4}\sqrt{\frac{7}{\pi}}\,\left(5\cos^3\theta-3\cos\theta\right)$$

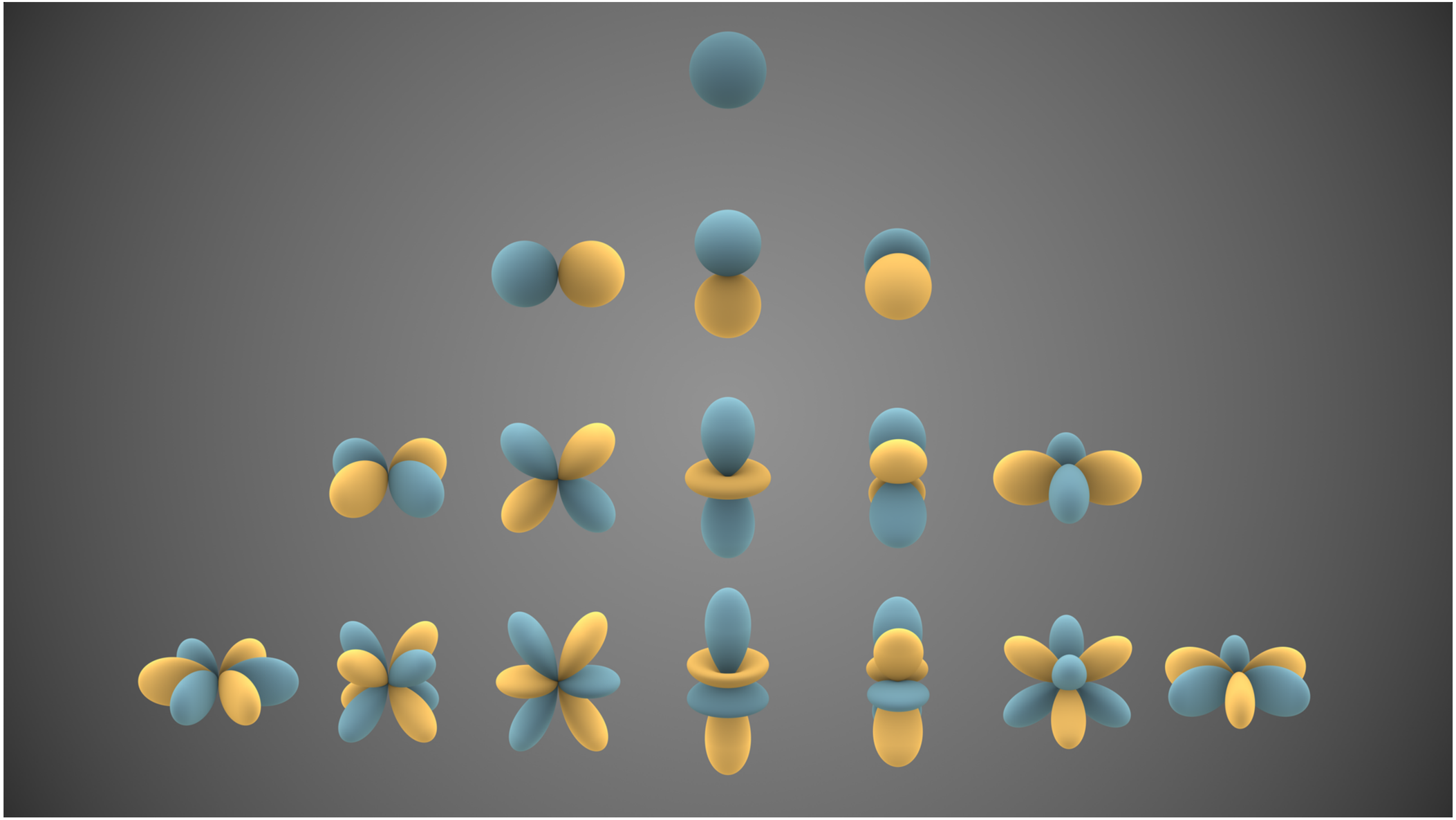
$$Y_3^1(\theta,\phi)=-\frac{1}{8}\sqrt{\frac{21}{\pi}}\,\sin\theta\left(5\cos^2\theta-1\right)e^{i\phi}$$

$$Y_3^2(\theta,\phi)=\frac{1}{4}\sqrt{\frac{105}{2\pi}}\,\sin^2\theta\cos\theta\,e^{2i\phi}$$

$$Y_3^3(\theta,\phi)=-\frac{1}{8}\sqrt{\frac{35}{\pi}}\,\sin^3\theta\,e^{3i\phi}.$$

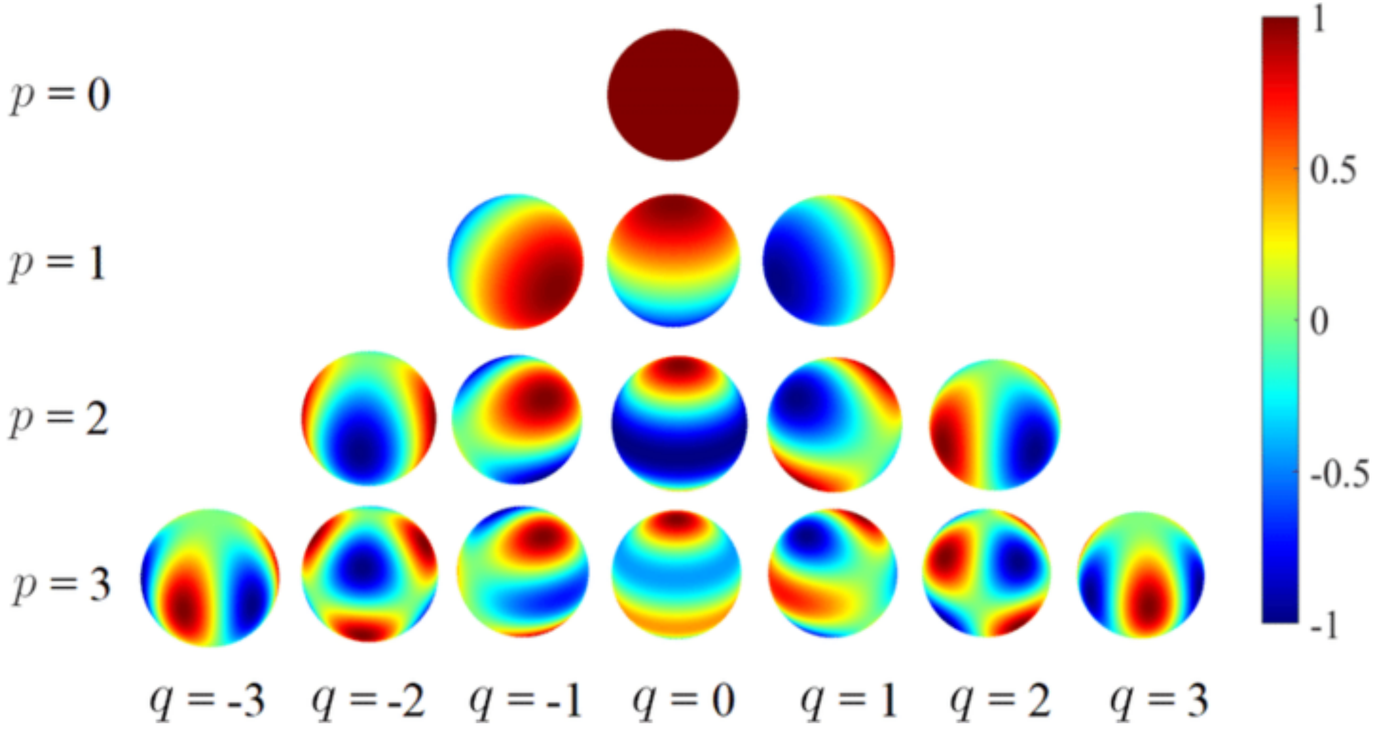
https://mathworld.wolfram.com/SphericalHarmonic.html

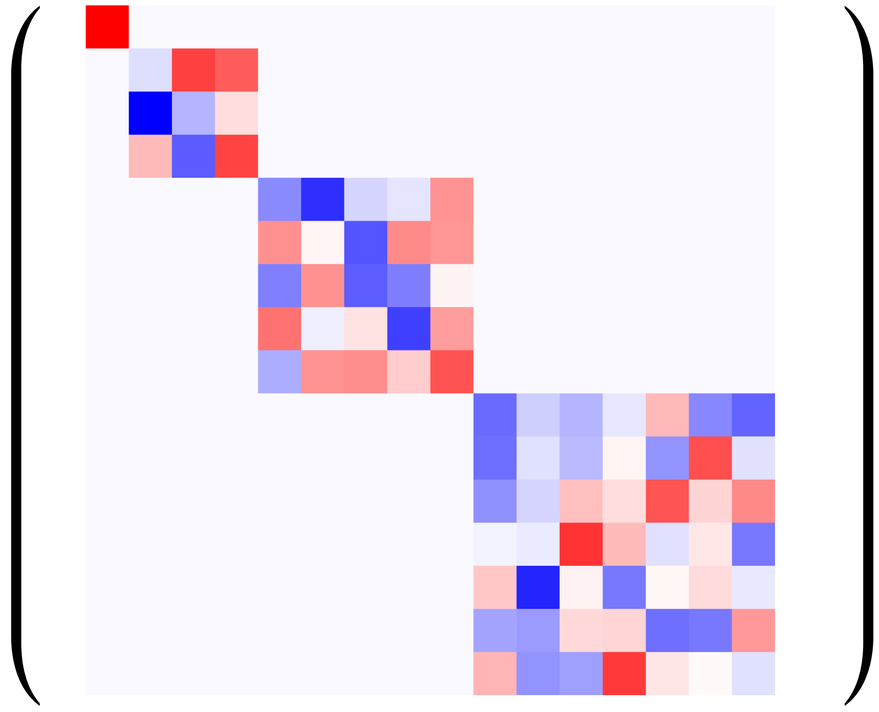
Other visualization



Steering the spherical harmonics basis via the “Wigner-D Matrices”

$$f(\mathbf{Rn}) = [\mathcal{F}_{S^2}^{-1} \quad \mathbf{D}(\mathbf{R}) \quad \mathcal{F}_{S^2} \quad f]](\mathbf{n})$$

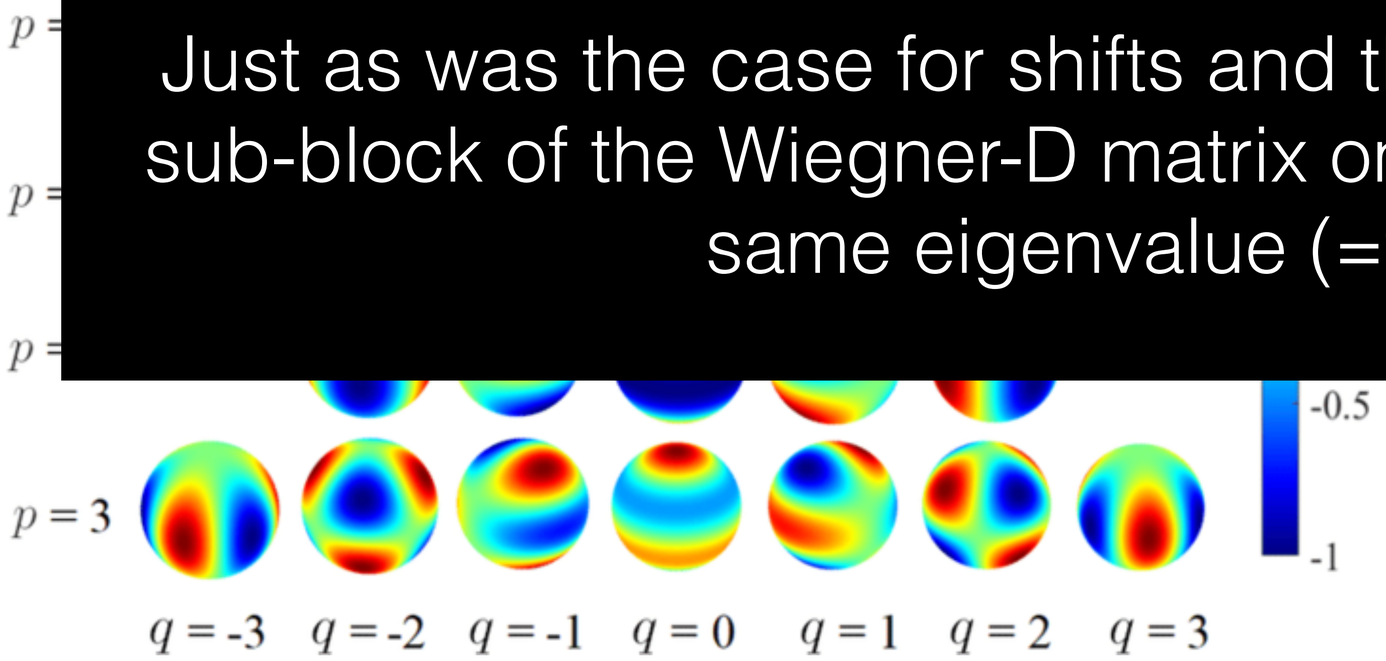


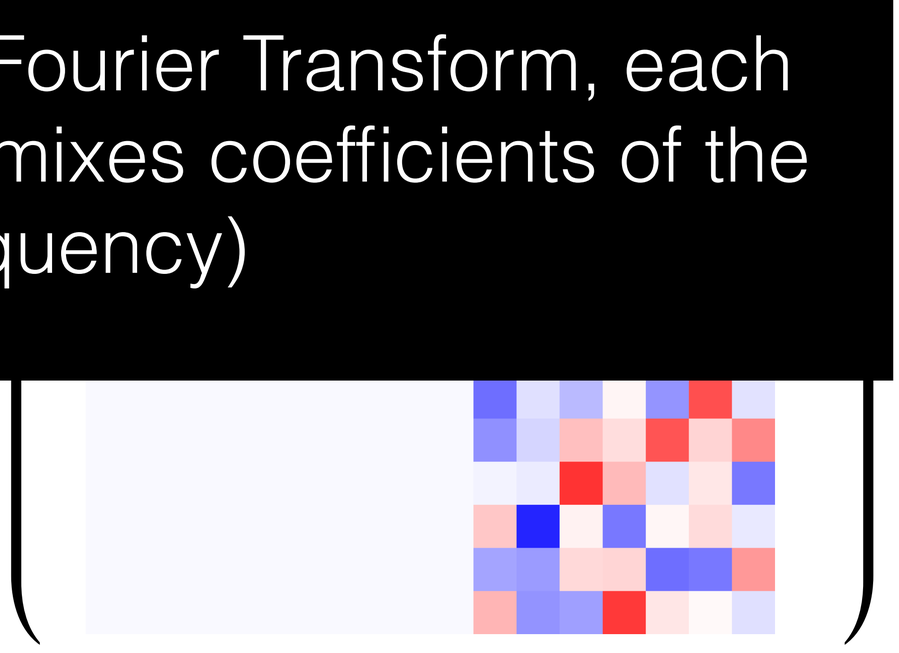


Steering the spherical harmonics basis via the “Wigner-D Matrices”

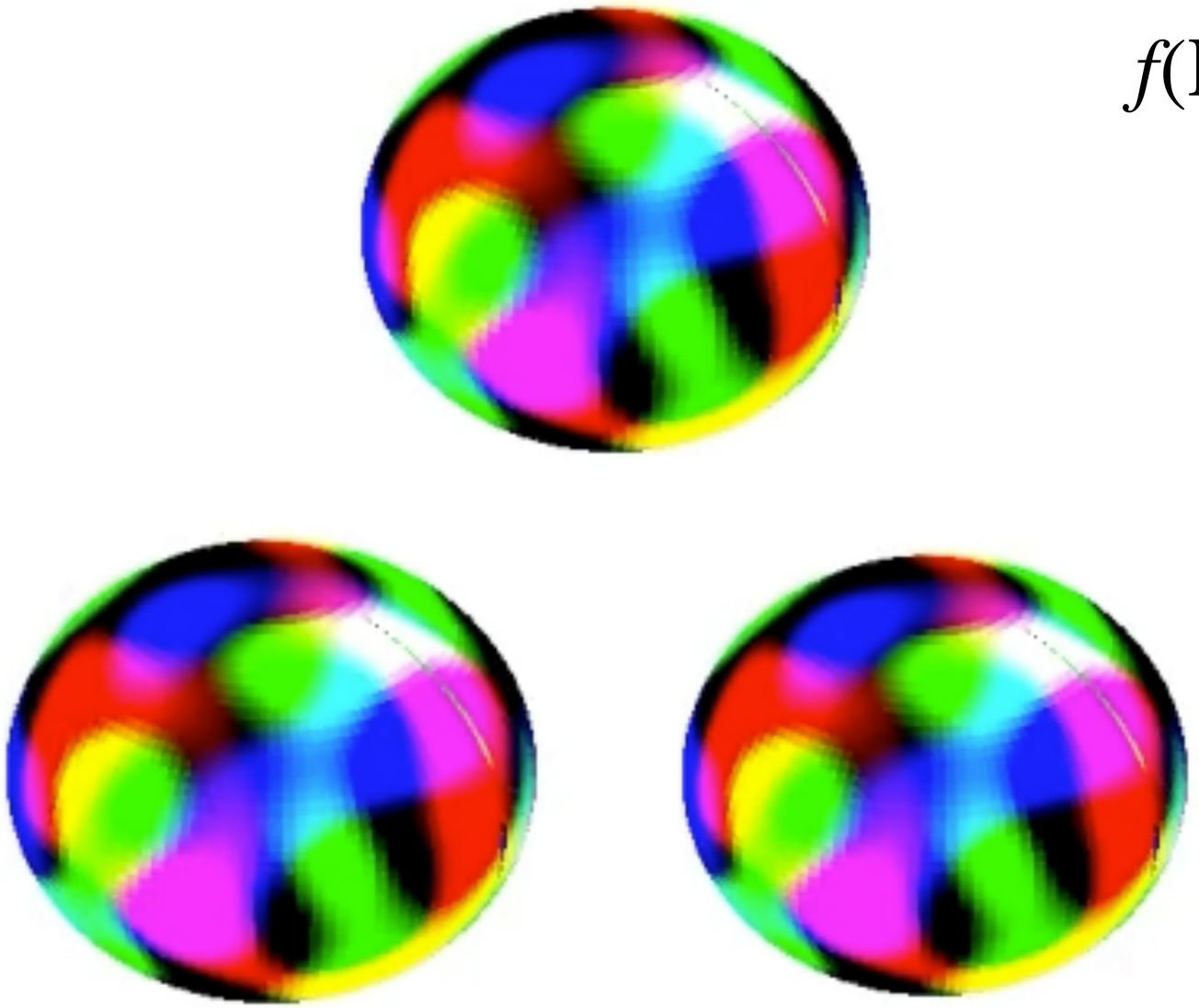
$$f(\mathbf{Rn}) = [\mathcal{F}_{S^2}^{-1} \quad \mathbf{D}(\mathbf{R}) \quad \mathcal{F}_{S^2} \quad f]](\mathbf{n})$$

Just as was the case for shifts and the Fourier Transform, each sub-block of the Wiegner-D matrix only mixes coefficients of the same eigenvalue (=frequency)

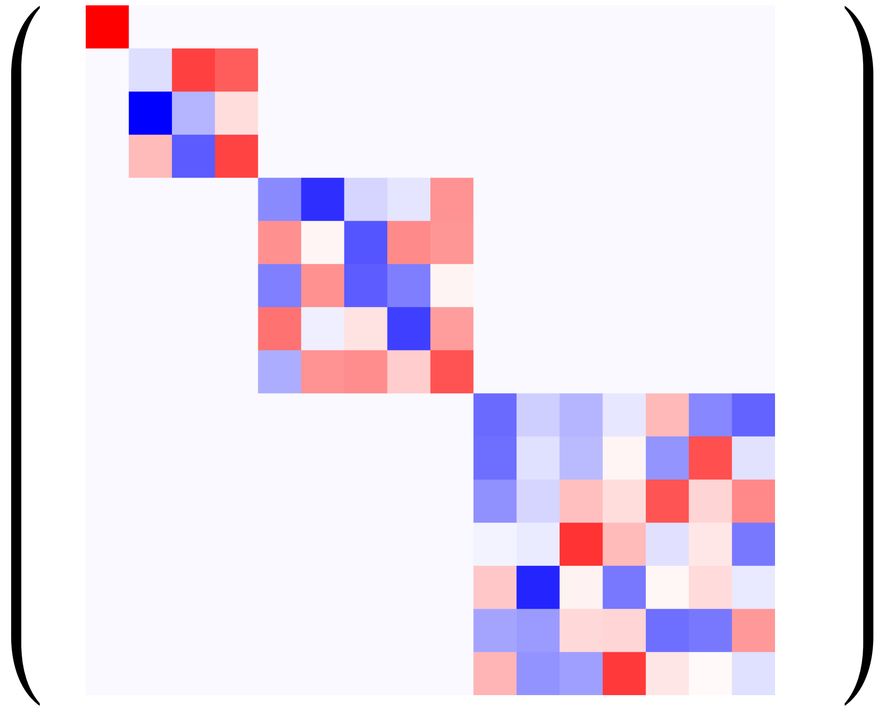




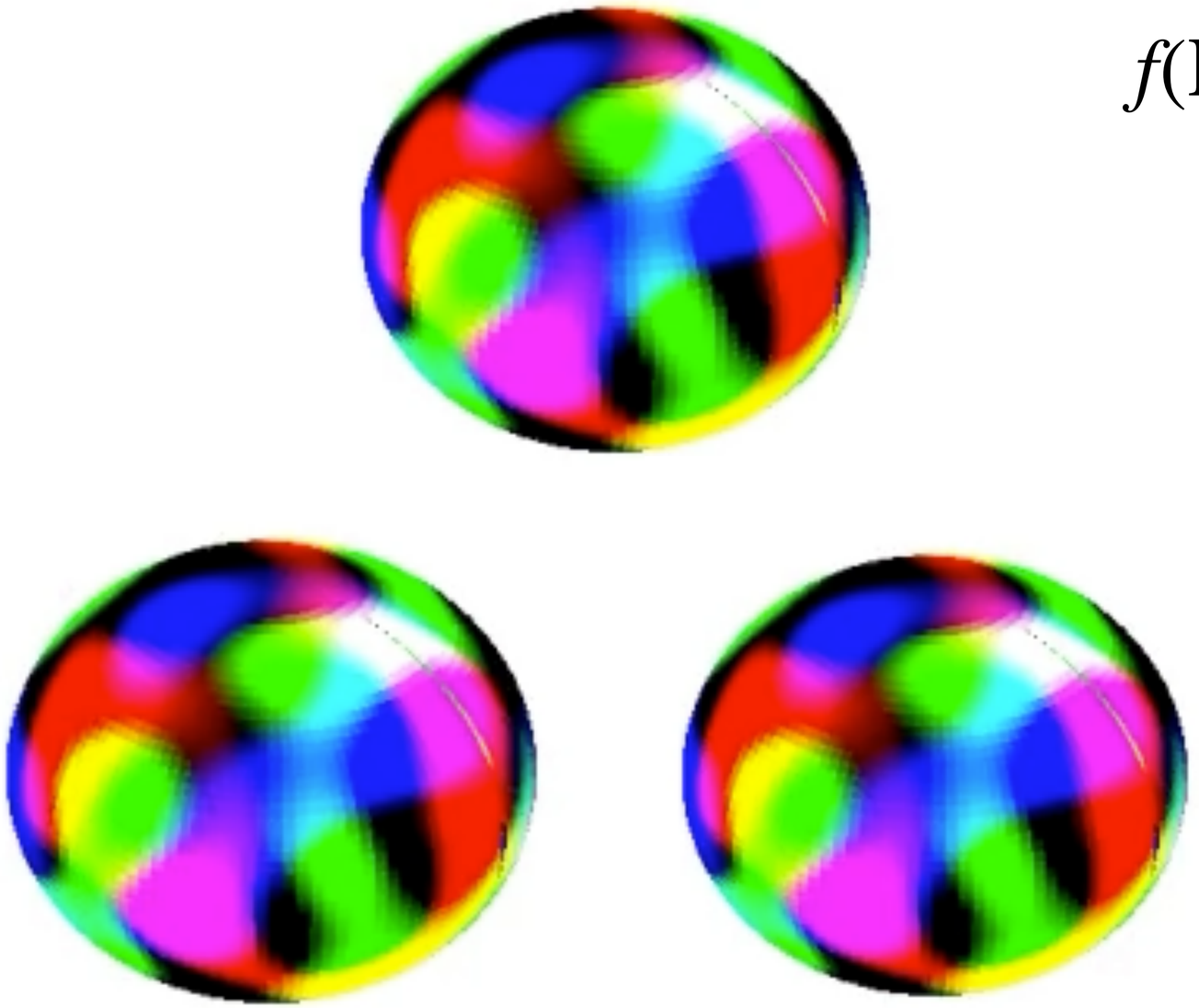
Steering the spherical harmonics basis via the “Wigner-D Matrices”



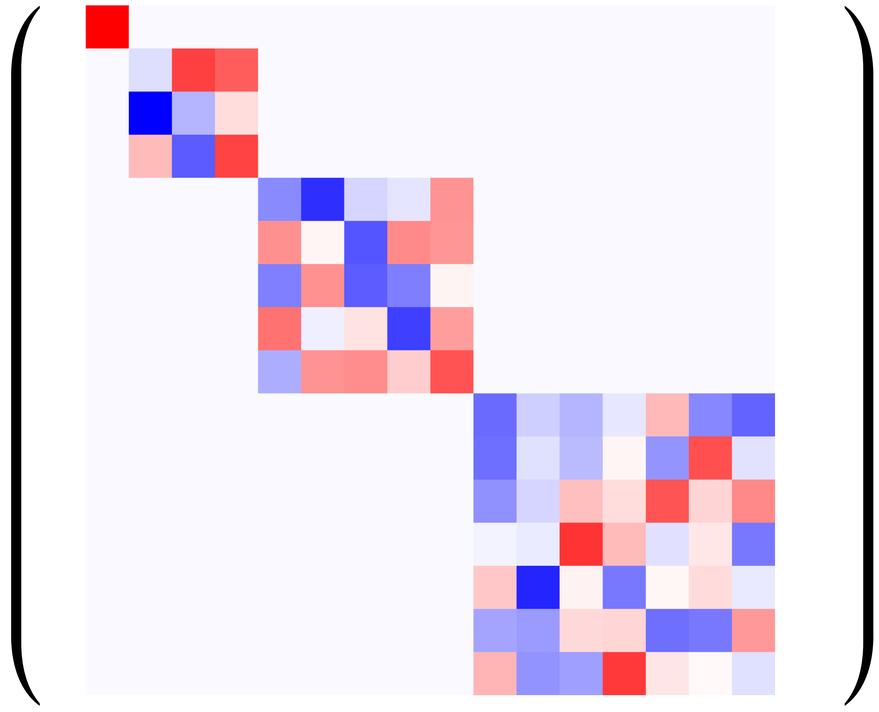
$$f(\mathbf{Rn}) = [\mathcal{F}_{S^2}^{-1} \quad \mathbf{D}(\mathbf{R}) \quad \mathcal{F}_{S^2} \quad f]](\mathbf{n})$$



Steering the spherical harmonics basis via the “Wigner-D Matrices”



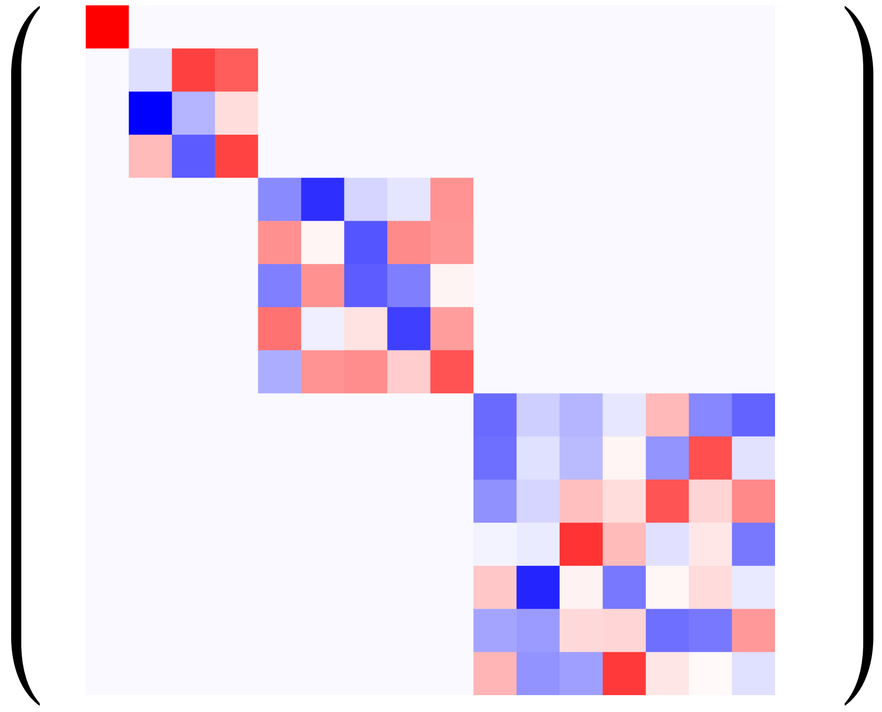
$$f(\mathbf{Rn}) = [\mathcal{F}_{S^2}^{-1} \quad \mathbf{D}(\mathbf{R}) \quad \mathcal{F}_{S^2} \quad f]](\mathbf{n})$$



Steering the spherical harmonics basis via the “Wigner-D Matrices”

$$(\mathbf{Rn}) = [\mathcal{F}_{S^2}^{-1} \quad \mathbf{D}(\mathbf{R}) \quad \mathcal{F}_{S^2} \quad f]](\mathbf{n})$$

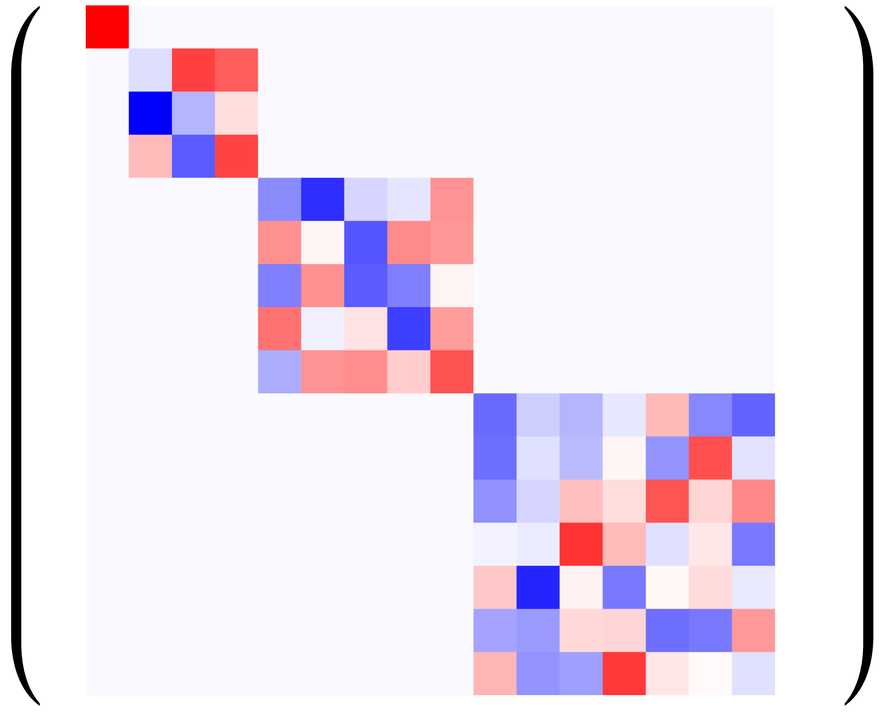




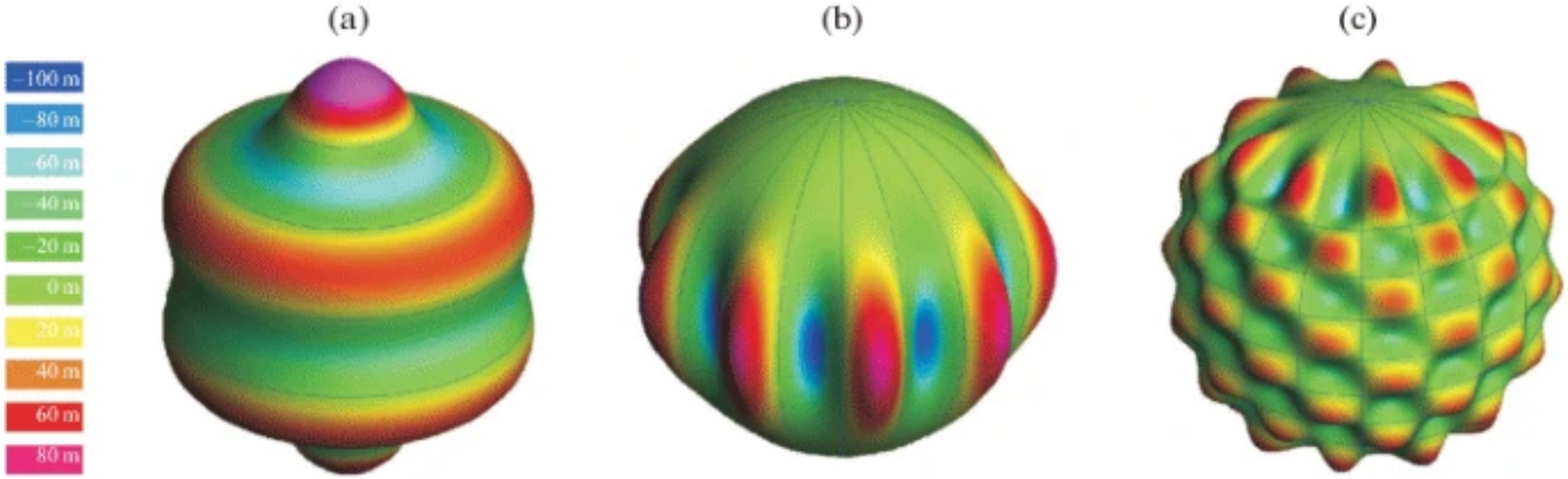
Steering the spherical harmonics basis via the “Wigner-D Matrices”

$$(\mathbf{Rn}) = [\mathcal{F}_{S^2}^{-1} \quad \mathbf{D}(\mathbf{R}) \quad \mathcal{F}_{S^2} \quad f]](\mathbf{n})$$

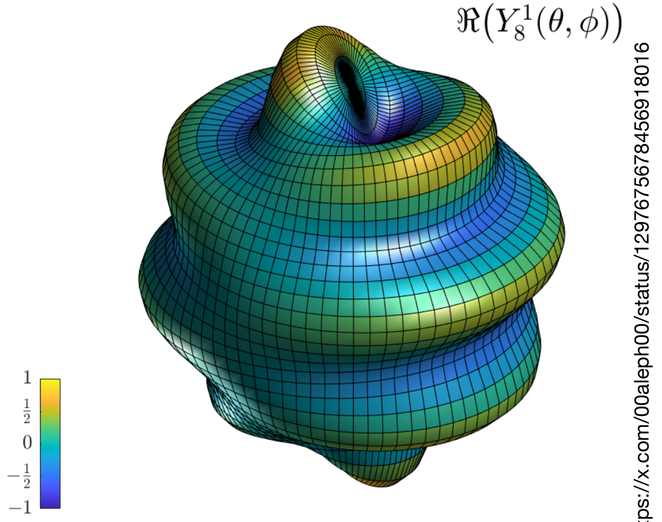




Also Spherical Harmonics are just very pretty :)



Mikhailov et al. High-Degree Models of the Earth's Gravity Field: History of Development, Assessment of Prospects and Resolution. Seism. Instr. 57, 446–461 (2021).



https://x.com/00aleph00/status/1297675678456918016

SH application: directional representation

PlenOctrees for Real-time Rendering of Neural Radiance Fields

Alex Yu
UC Berkeley

Ruilong Li
UC Berkeley
& USC ICT

Matt Tancik
UC Berkeley

Hao Li
Pinscreen
& UC Berkeley

Ren Ng
UC Berkeley

Angjoo Kanazawa
UC Berkeley



Project website & online demo: alexyu.net/plenoctrees

PlenOctrees for Real-time Rendering of Neural Radiance Fields

Alex Yu
UC Berkeley

Ruilong Li
UC Berkeley
& USC ICT

Matt Tancik
UC Berkeley

Hao Li
Pinscreen
& UC Berkeley

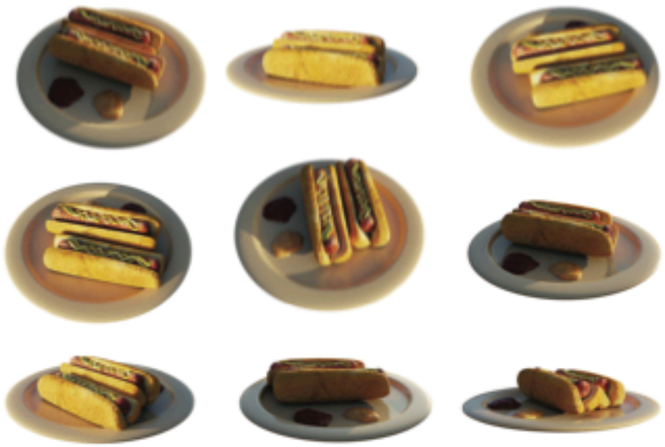
Ren Ng
UC Berkeley

Angjoo Kanazawa
UC Berkeley

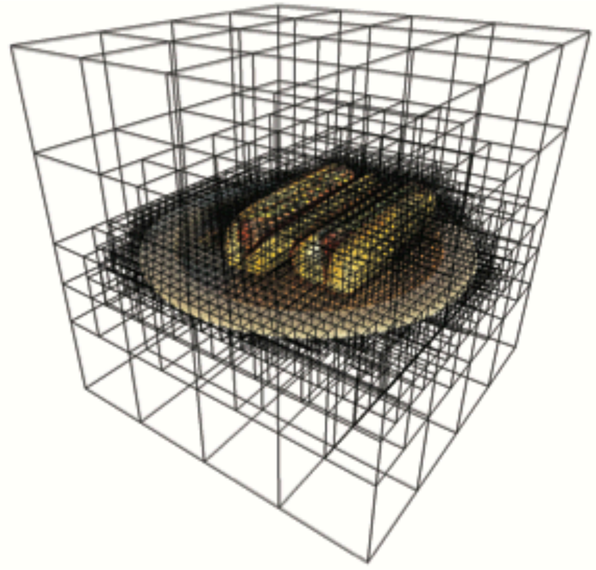


Project website & online demo: alexyu.net/plenoctrees

Offline Model Generation







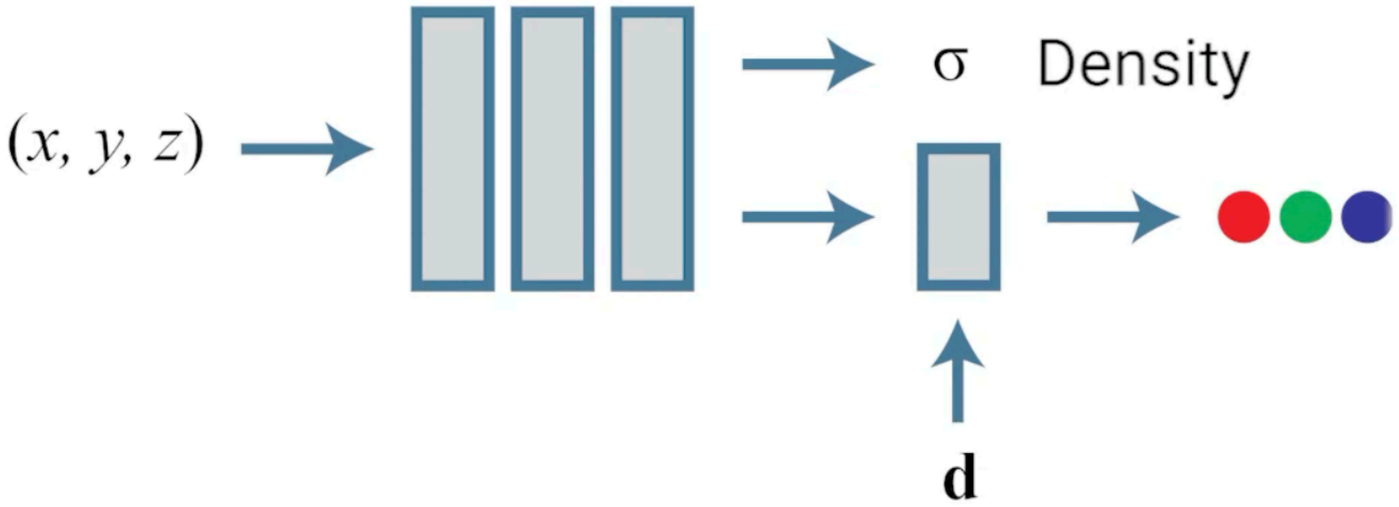
Input Images

NeRF-SH

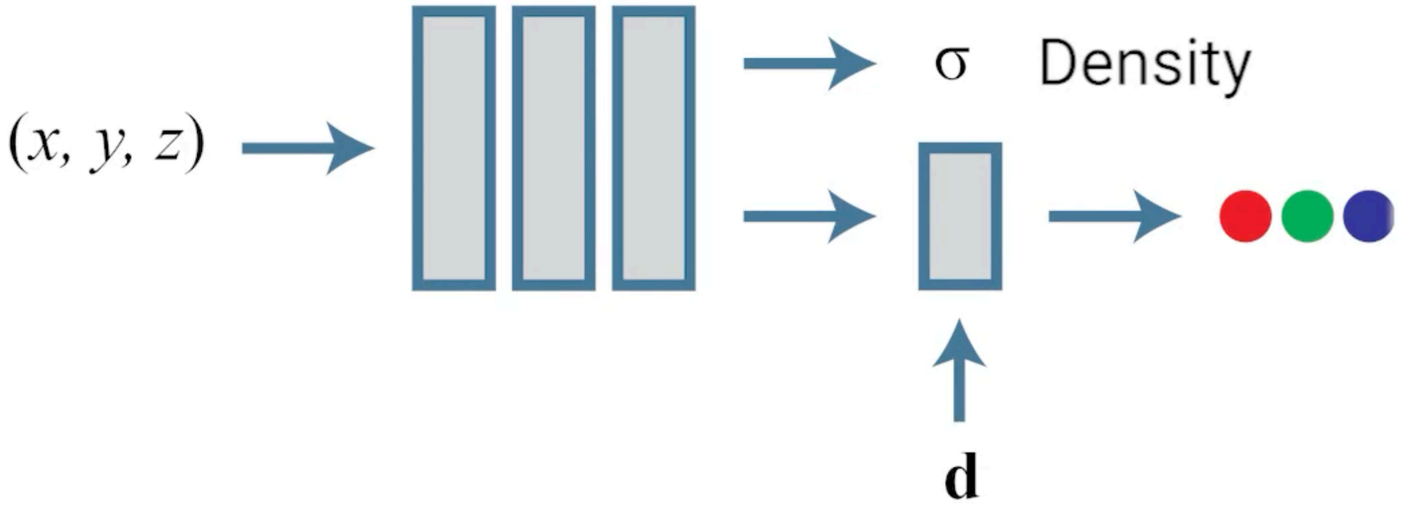
PlenOctree

Real-time Rendering

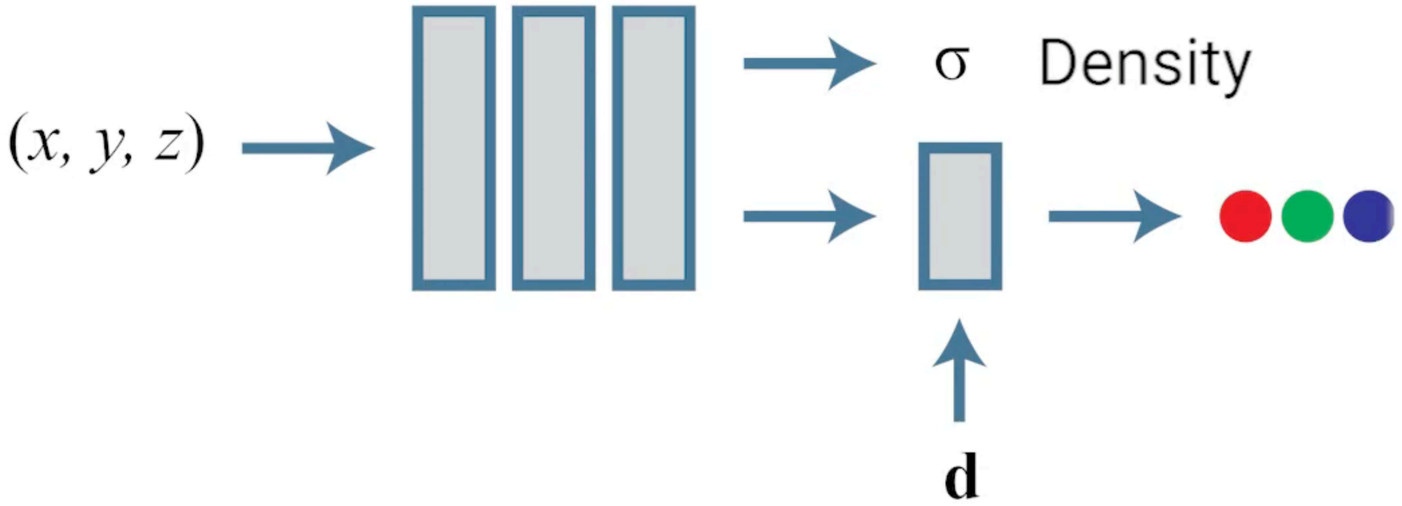
Original NERF: no output directionality



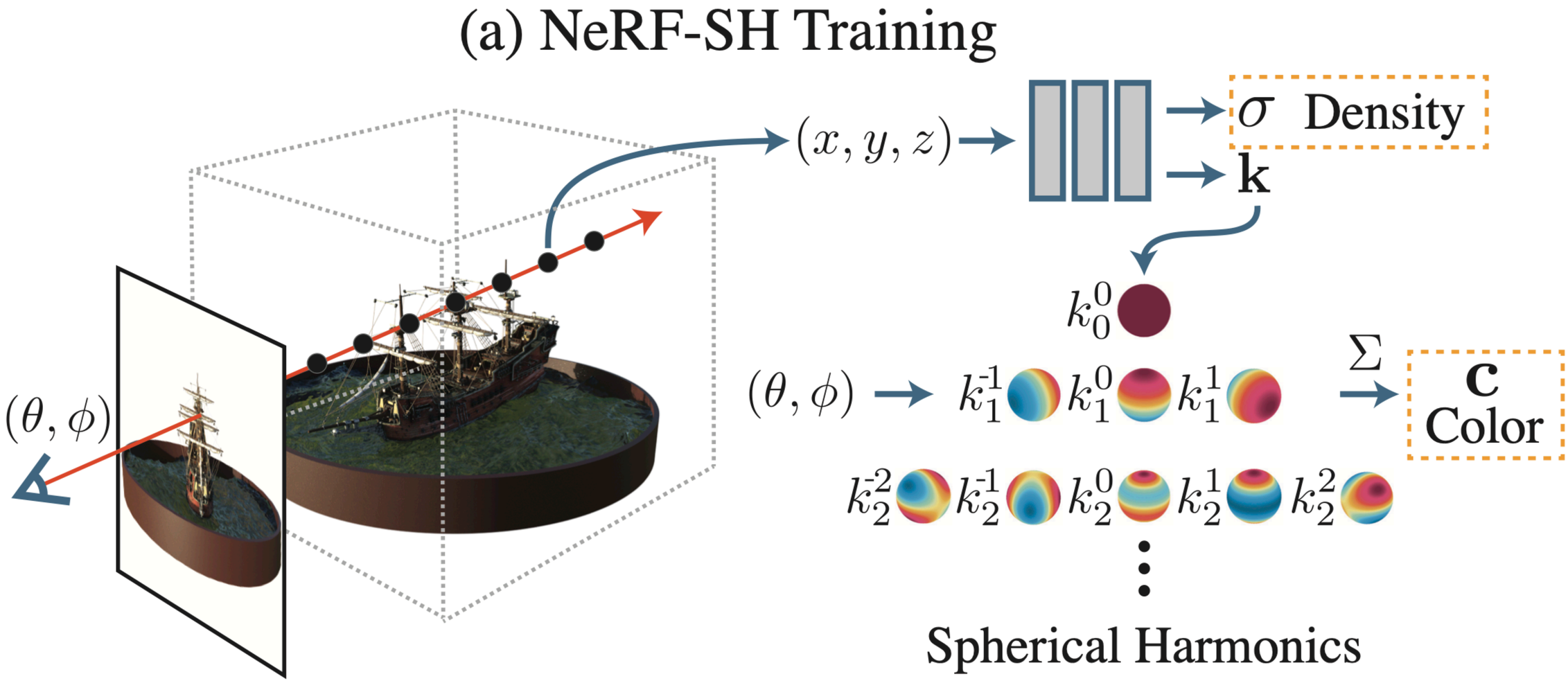
NeRF



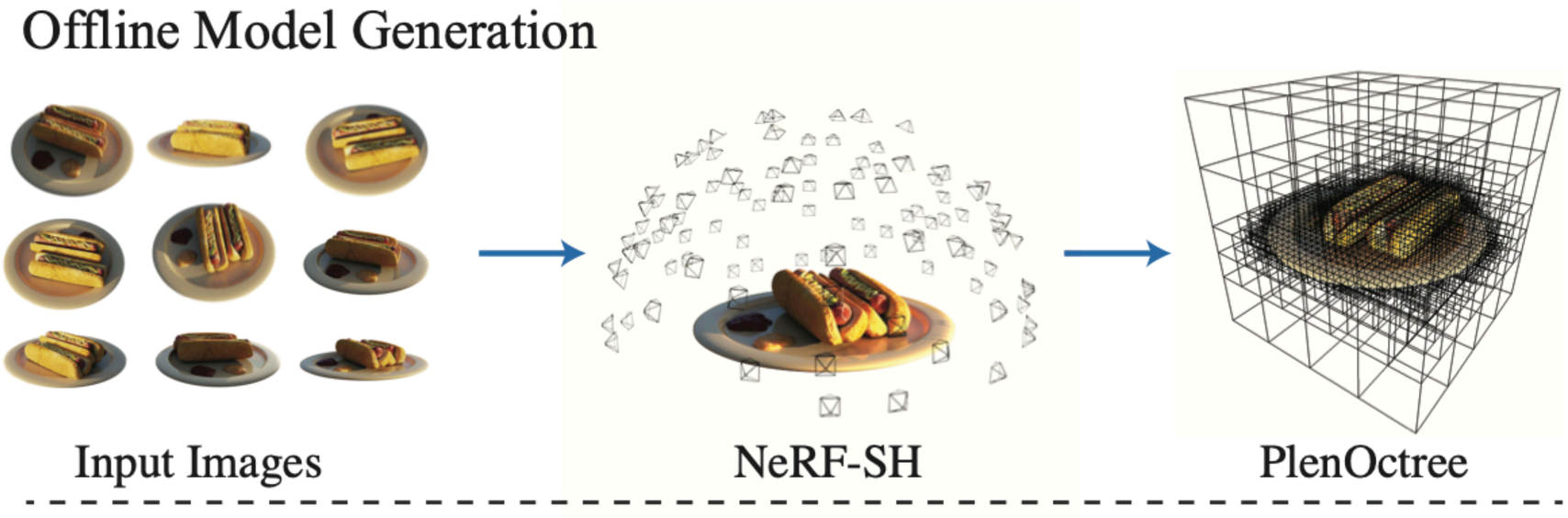
NeRF



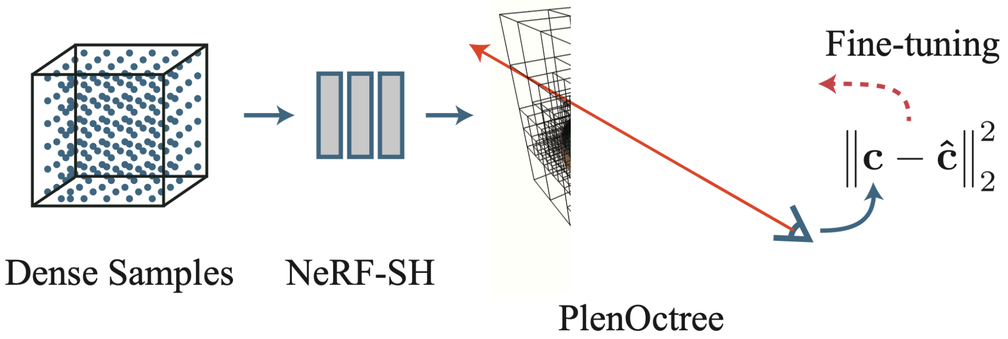
Recap: NERF SH



They also distill NERF into a non-neural representation based on octrees



(b) Conversion to PlenOctree



PlenOctree Viewer

https://alexyu.net/plenoctrees/demo/?load=../data/materials.npz

Search

Other Bookmarks

PlenOctree Viewer Options



PlenOctree Viewer

https://alexyu.net/plenoctrees/demo/?load=../data/materials.npz

Search

Other Bookmarks

PlenOctree Viewer Options



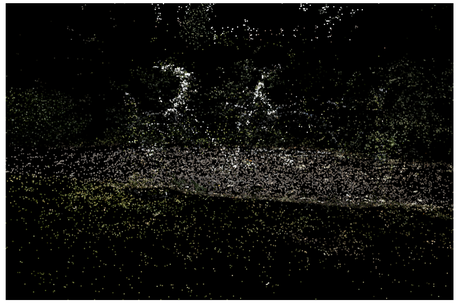
Interesting lessons

- Spherical harmonics are a nice representation of directional info
  - Reconstruct naturally into smooth functions
- They don’t directly reconstruct their octree representation
  - The neural NERF intermediate step appears to help
  - Neural networks appear to have good regularization

More recently: Gaussian splatting

- Also rely on spherical harmonics for directional information

2024: Gaussian Splatting

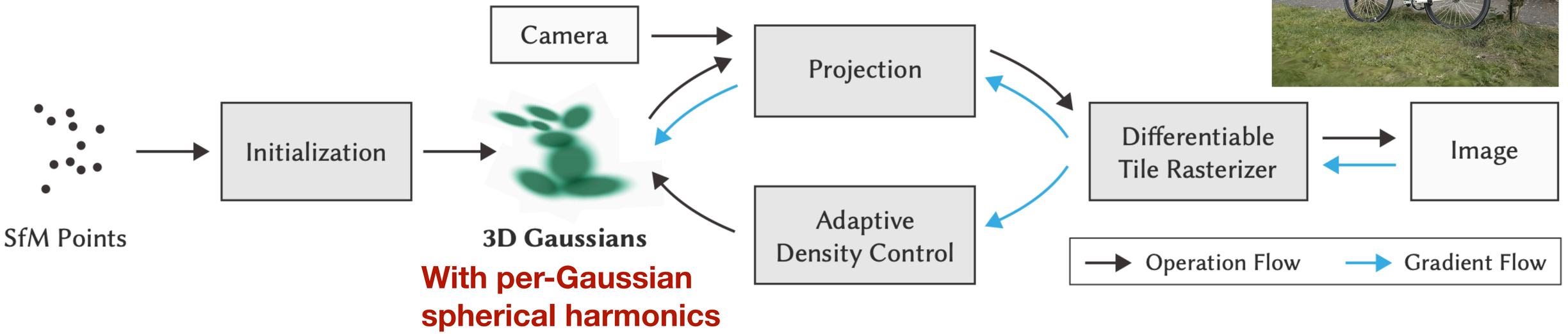






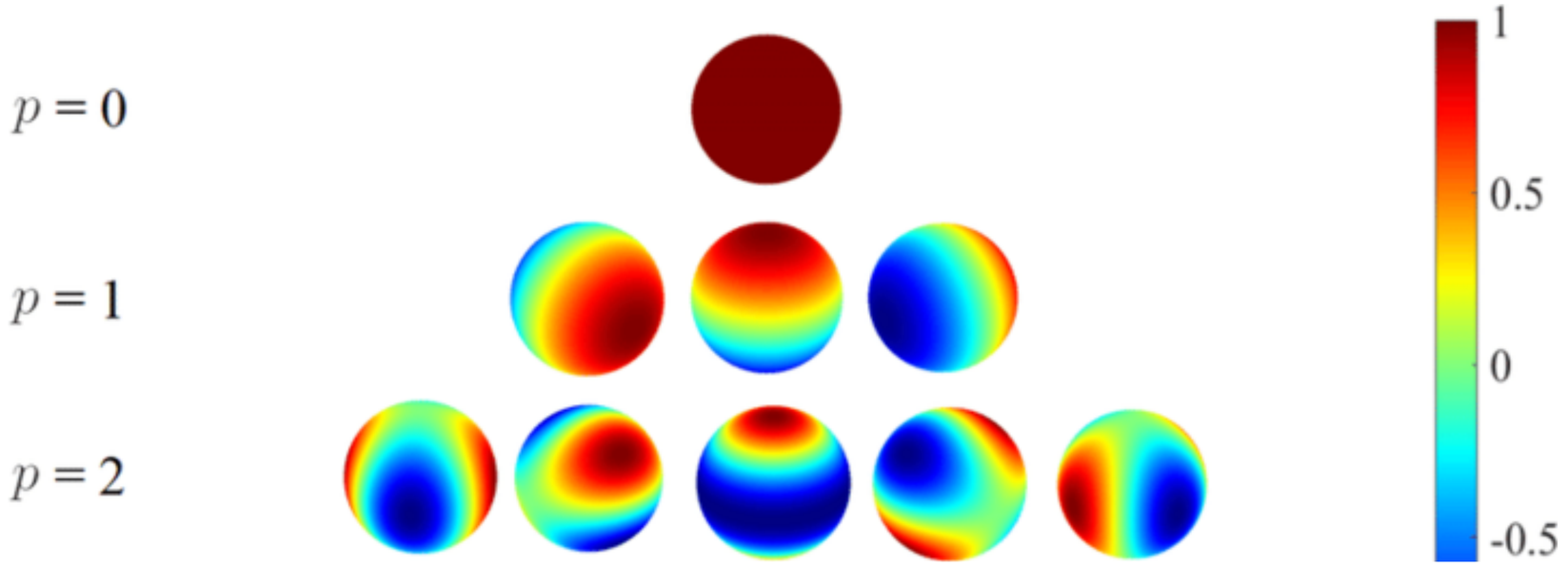
- Represent the scene as a collection of points with size: 3D Gaussians (mean & covariance) + color & opacity
- Each 3D Gaussian stores a set of spherical harmonic coefficients for directional color

Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*



With per-Gaussian spherical harmonics

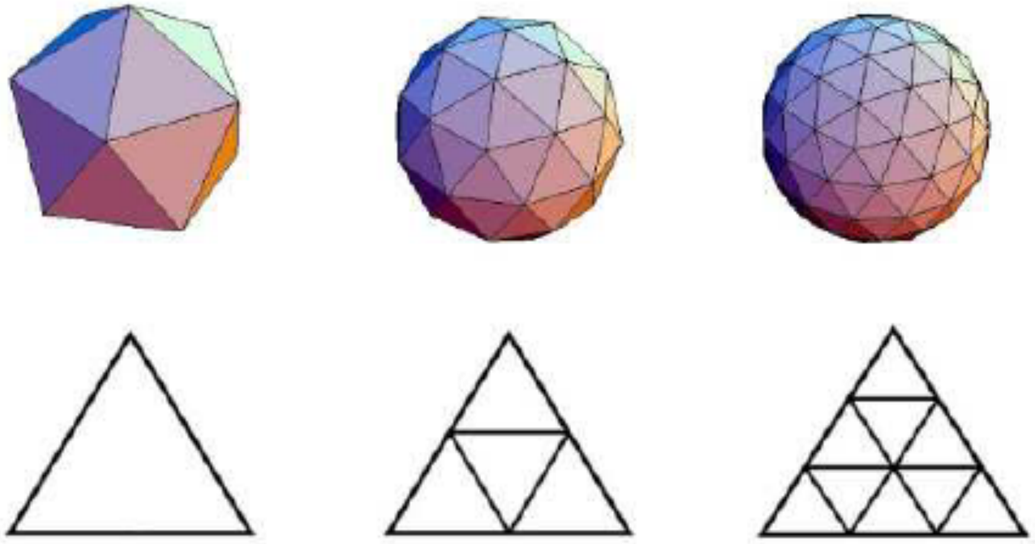
They go to degree 2



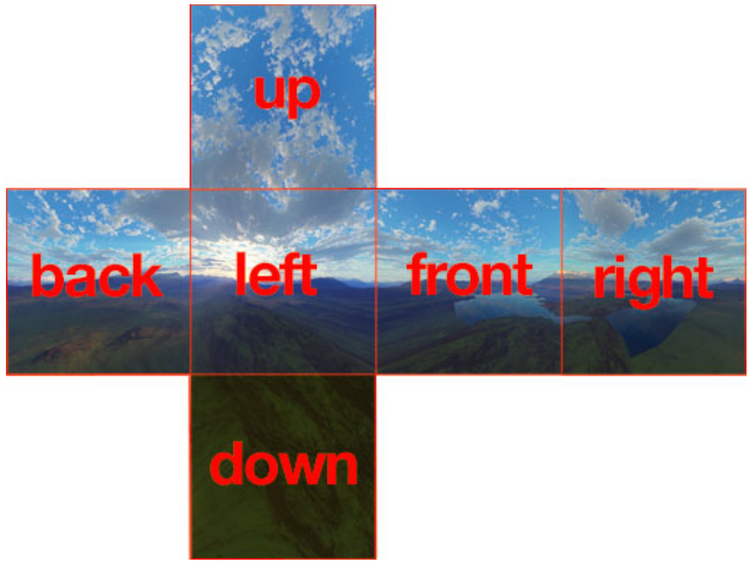

Menu
Views
Capture
3D Gaussians
Camera Point view
Metrics
57.56 (17.37 ms)
VSync On


Menu
Views
Capture
3D Gaussians
Camera Point view
Metrics
57.56 (17.37 ms)
VSync On

Alternatives: more local / canonical bases

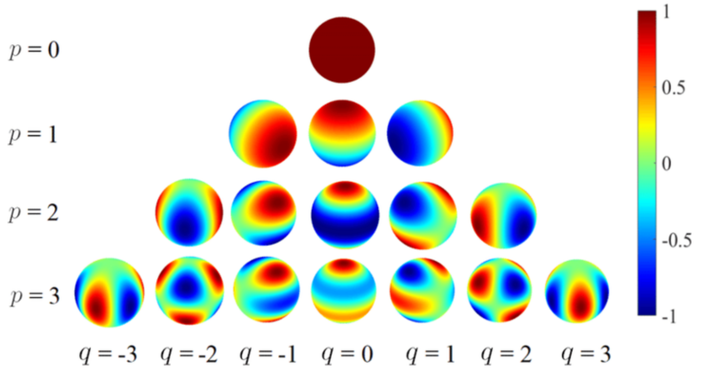


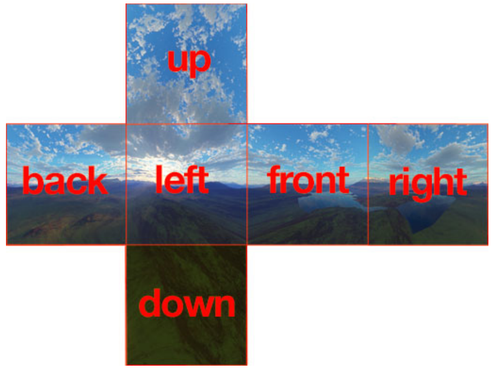
- Geodesic sphere / subdivision of icosahedron
- Cube maps
  - Very popular in graphics
  - Easier to comprehend,
  - But harder to get smooth reconstruction especially from low number of samples
  - But cheap reconstruction (SH require you to evaluate and sum all basis functions to reconstruct for one direction)
- Traditional mapping to rectangle, spherical coordinates
  - Very non-uniform and messy topology at poles
- Less uniform, not steerable
  - But to be fair, people don't steer SH that often



Discussion

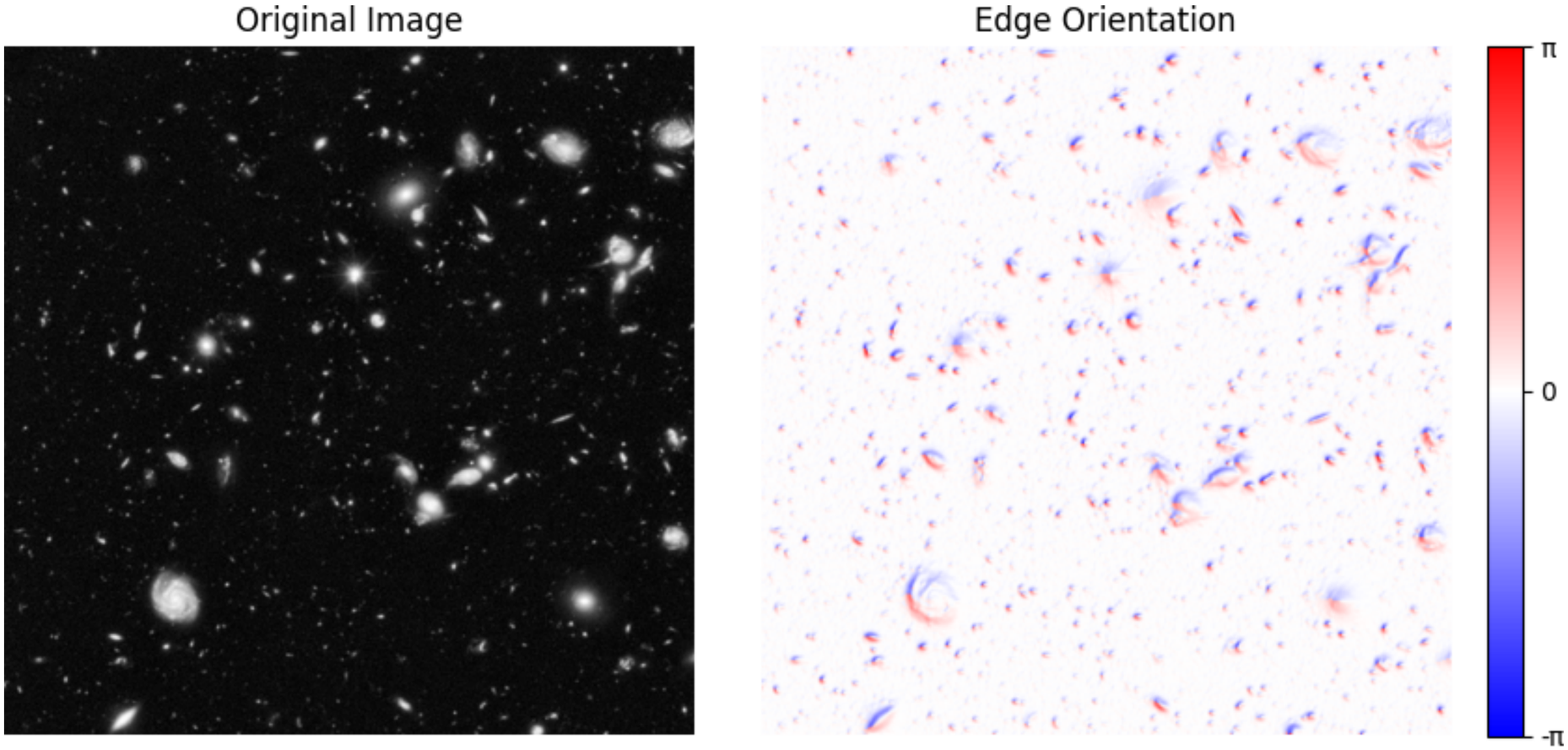
- Perhaps the one case where the spectral basis is used the most compared to the canonical one
  - Make its easier to get nice smooth reconstruction
  - (The equivalent of Lanczos/bicubic is not easy)
- In general, the canonical representation is messy because the sphere doesn't have a nice lattice like the square grid
- In practice, people usually use cube maps for high resolution and spherical harmonics for low resolution (because they reconstruct so smoothly)





PSet 2 is out!

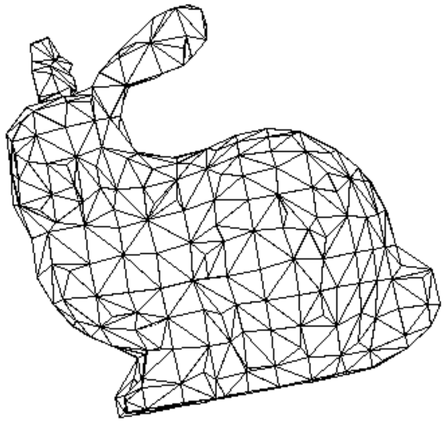
https://github.com/6-8300/pset-2/



Fourier Transform, steerable bases, steerable filters!

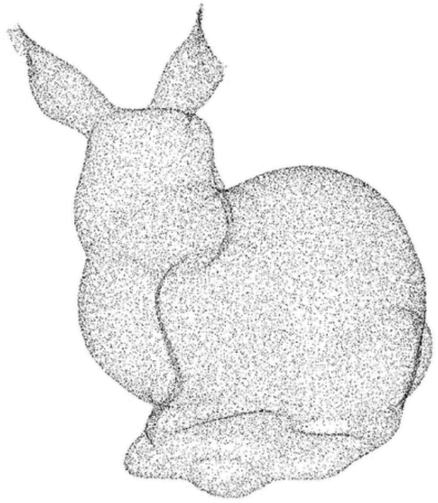
Geometric signal processing

Challenges of geometry processing



The domain is also the range!

- With images, there is a clear separation between (x,y) 2D domain and RGB/intensity range. In images it's all in (x,y,z)
- Order of points is irrelevant/distraction

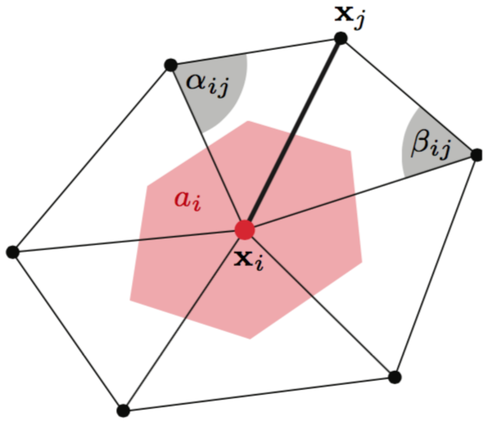


Extending signal processing to meshes

Start from Fourier's original perspective (heat transfer / Laplacian)

Extend Laplacian to surfaces: Laplace-Beltrami

Define discrete equivalent for meshes: cotangent Laplacian



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

Use to define equivalents of convolution/smoothing

Use for PDEs

Compute eigenvectors for fun & profit

See e.g. https://ddg.cs.columbia.edu/SGP2014/LaplaceBeltrami.pdf

Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow

Mathieu Desbrun Mark Meyer Peter Schröder Alan H. Barr

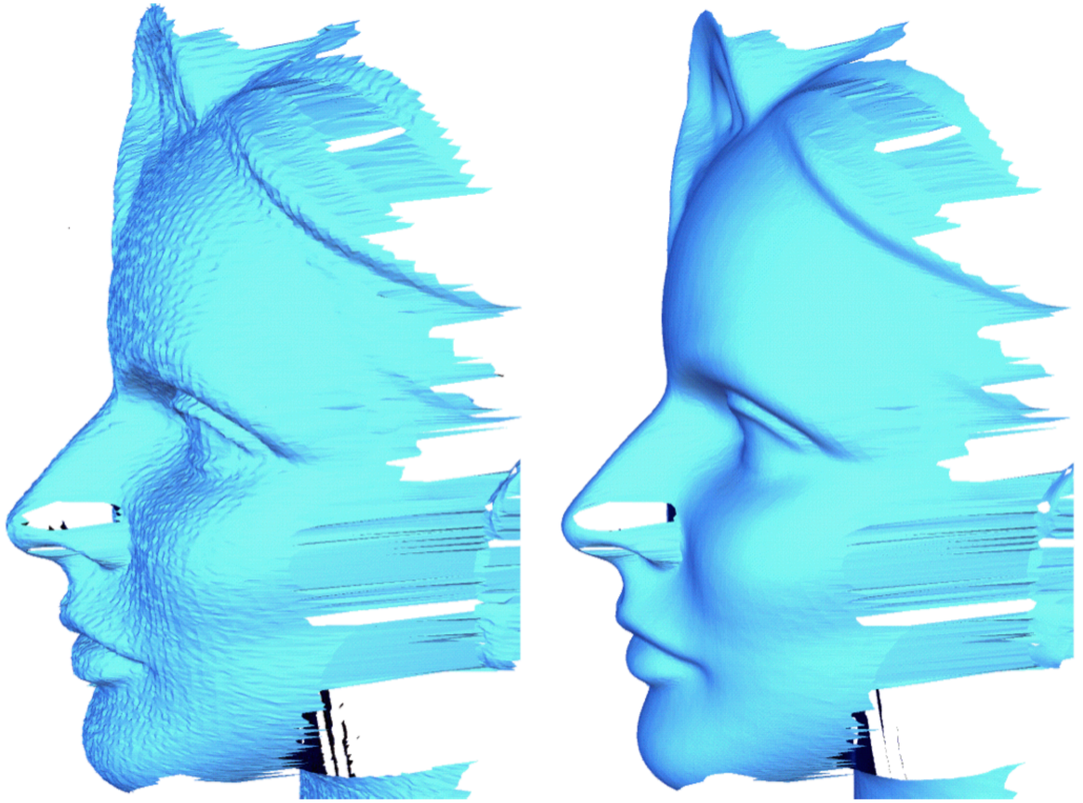
Caltech*

Abstract

In this paper, we develop methods to rapidly remove rough features from irregularly triangulated data intended to portray a smooth surface. The main task is to remove undesirable noise and uneven edges while retaining desirable geometric features. The problem arises mainly when creating high-fidelity computer graphics objects using imperfectly-measured data from the real world.

Our approach contains three novel features: an *implicit integration* method to achieve efficiency, stability, and large time-steps; a scale-dependent Laplacian operator to improve the diffusion process; and finally, a robust curvature flow operator that achieves a smoothing of the shape itself, distinct from any parameterization. Additional features of the algorithm include automatic exact volume preservation, and hard and soft constraints on the positions of the points in the mesh.

We compare our method to previous operators and related algorithms, and prove that our curvature and Laplacian operators have several mathematically-desirable qualities that improve the appearance of the resulting surface. In consequence, the user can easily select the appropriate operator according to the desired type of fairing. Finally, we provide a series of examples to graphically and numerically demonstrate the quality of our results.



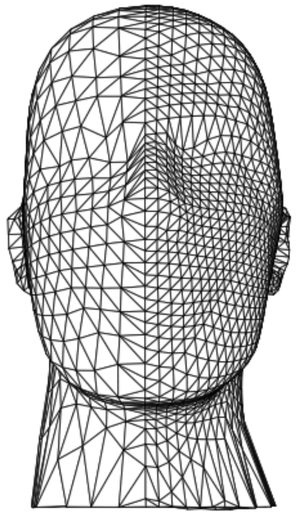
(a) (b)

Figure 1: *(a): Original 3D photography mesh (41,000 vertices). (b): Smoothed version with the scale-dependent operator in two integration step with* $\lambda dt = 5 \cdot 10^{-5}$*, the iterative linear solver (PBCG) converges in 10 iterations. All the images in this paper are flat-shaded to enhance the faceting effect.*

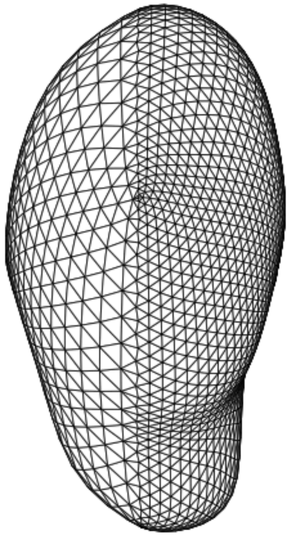
Benefits

Strong math underpinning

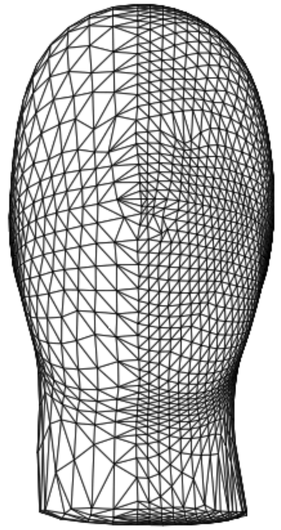
Invariance to discretization & parameterization



Mesh with irregular discretization



With “bad” smoothing, the amount of smoothing depends on discretization

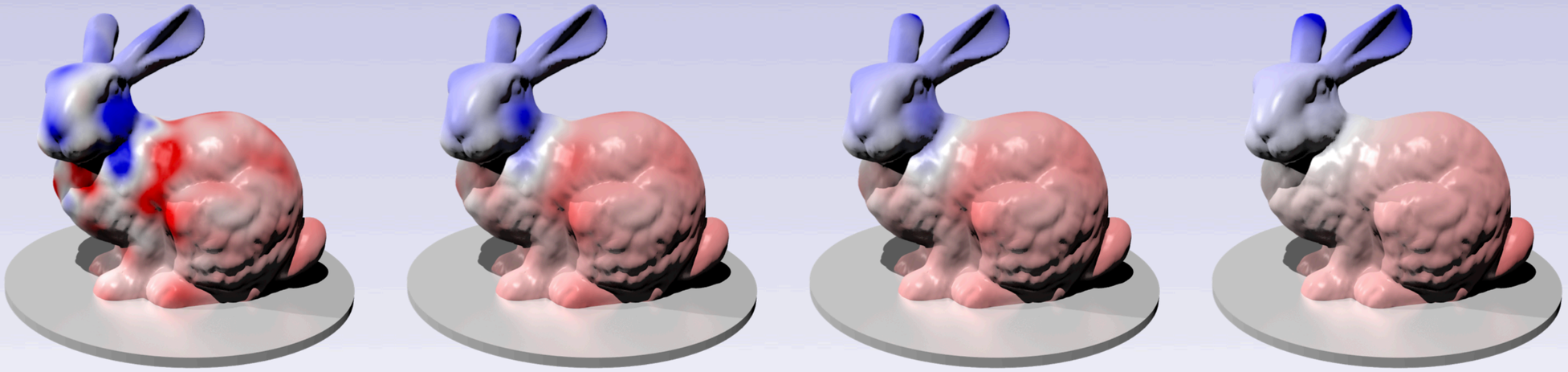


Better smoothing

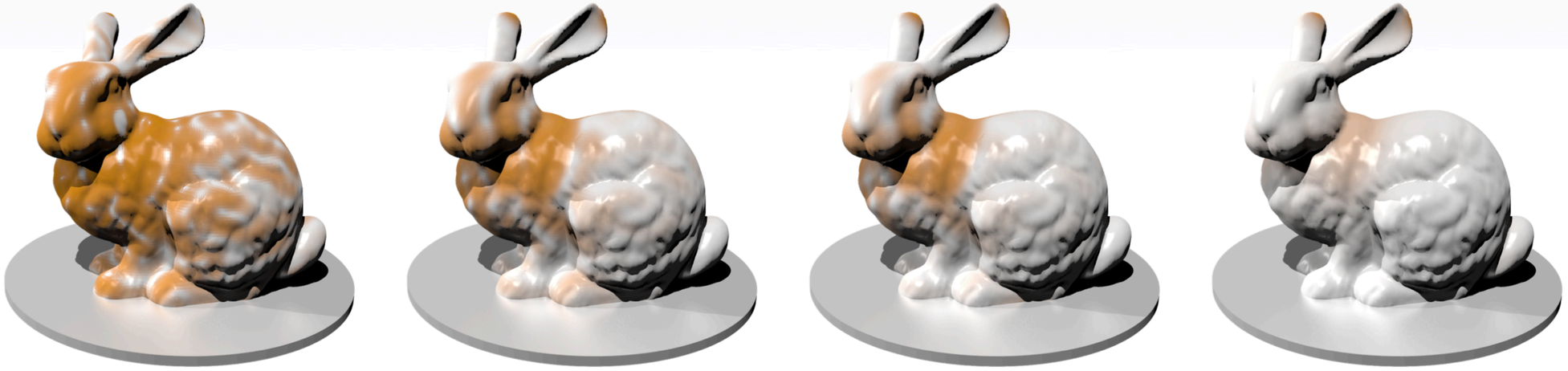
- *Laplace-Beltrami operator* ("Laplacian") provides a basis for a diverse variety of geometry processing tasks.
- Remarkably common pipeline:
  1. simple pre-processing (build f)
  2. solve a PDE involving the Laplacian (e.g., $\Delta u = f$)
  3. simple post-processing (do something with u)
- Expressing tasks in terms of Laplacian/smooth PDEs makes life easier at code/implementation level.
- Lots of existing theory to help understand/interpret algorithms, provide analysis/guarantees.
- Also makes it easy to work with a broad range of geometric data structures (meshes, point clouds, etc.)

https://ddg.cs.columbia.edu/SGP2014/LaplaceBeltrami.pdf

Heat Equation Illustrated



time



Discrete Laplace operators: No free lunch

Max Wardetzky[1] Saurabh Mathur[2] Felix Kälberer[1] Eitan Grinspun[2] †

[1]Freie Universität Berlin, Germany [2]Columbia University, USA

Abstract

Discrete Laplace operators are ubiquitous in applications spanning geometric modeling to simulation. For robustness and efficiency, many applications require discrete operators that retain key structural properties inherent to the continuous setting. Building on the smooth setting, we present a set of natural properties for discrete Laplace operators for triangular surface meshes. We prove an important theoretical limitation: discrete Laplacians cannot satisfy all natural properties; retroactively, this explains the diversity of existing discrete Laplace operators. Finally, we present a family of operators that includes and extends well-known and widely-used operators.

You can’t get everything you want from a discrete Laplacian

https://www.cs.columbia.edu/cg/pdfs/1180993110-laplacian.pdf

Functional maps

Similar approach works for correspondences on meshes - called “Functional Maps”

Eigenvectors as shape characterization

Regular Euclidean Fourier eigenbases are independent of the signal

But with geometry, domain and signal are conflated

Eigenvectors of Laplace Beltrami depend on the shape

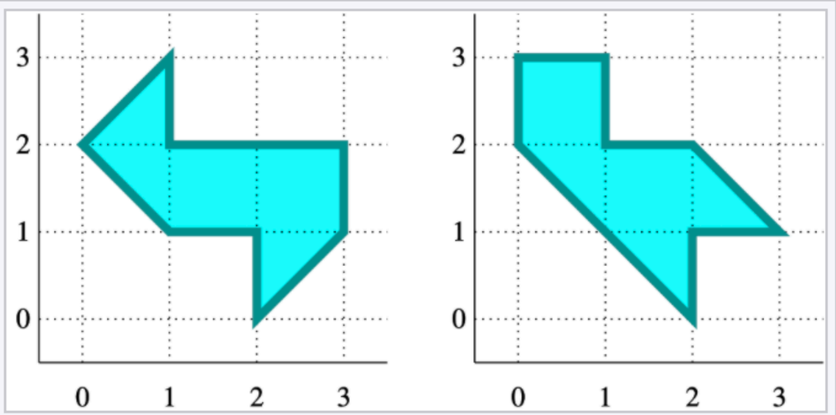
Related to shape of a drum

Mark Kac, 1966, Lipman Bers, Schuster 1882

The frequencies at which a drumhead can vibrate depend on its shape.

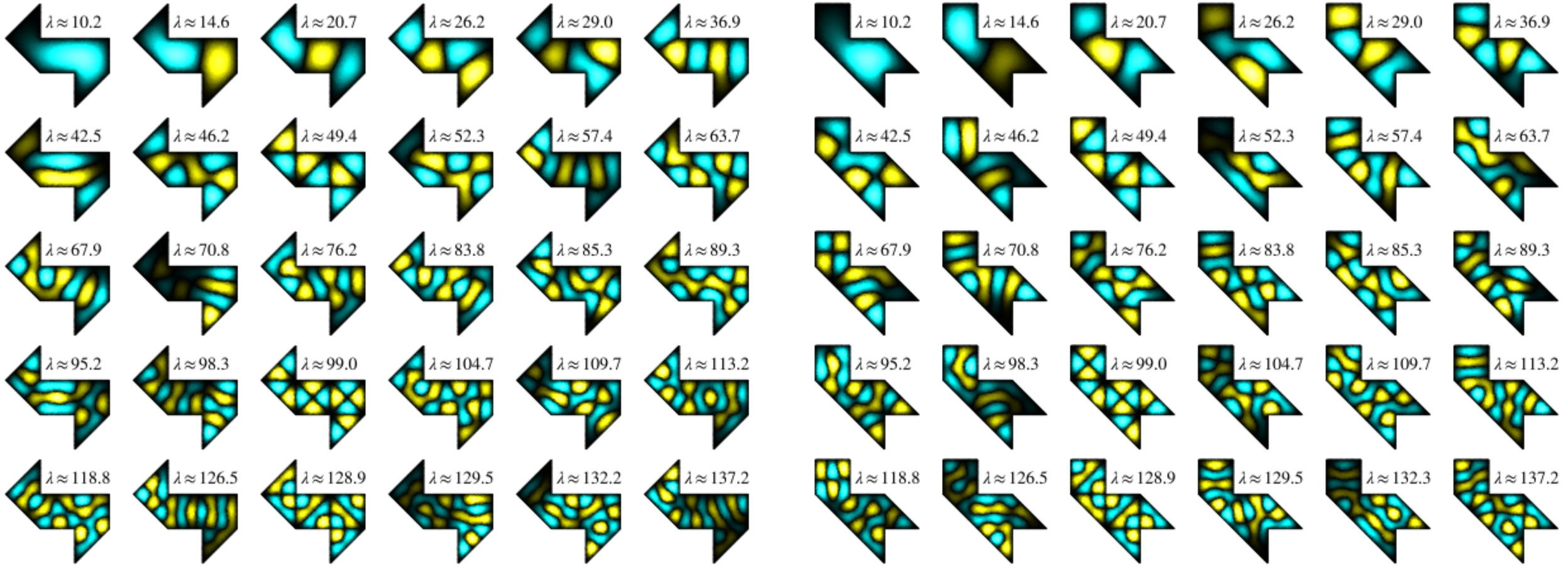
Inverse question: can the shape be predicted given the frequencies?

Answered in 1990s negatively by Carolyn S. Gordon, David Webb and Scott A. Wolpert.



Mathematically ideal drums with membranes of these two different shapes (but otherwise identical) would sound the same, because the eigenfrequencies are all equal, so the timbral spectra would contain the same overtones. This example was constructed by S. J. Chapman. Notice that both polygons have the same area and perimeter.

https://en.wikipedia.org/wiki/Hearing_the_shape_of_a_drum



Eigenmodes and corresponding eigenvalues of the Laplace operator on the GWW domains

https://en.wikipedia.org/wiki/Hearing_the_shape_of_a_drum

Functional Maps: A Flexible Representation of Maps Between Shapes

Maks Ovsjanikov† Mirela Ben-Chen‡ Justin Solomon‡ Adrian Butscher‡ Leonidas Guibas‡

† LIX, École Polytechnique ‡ Stanford University

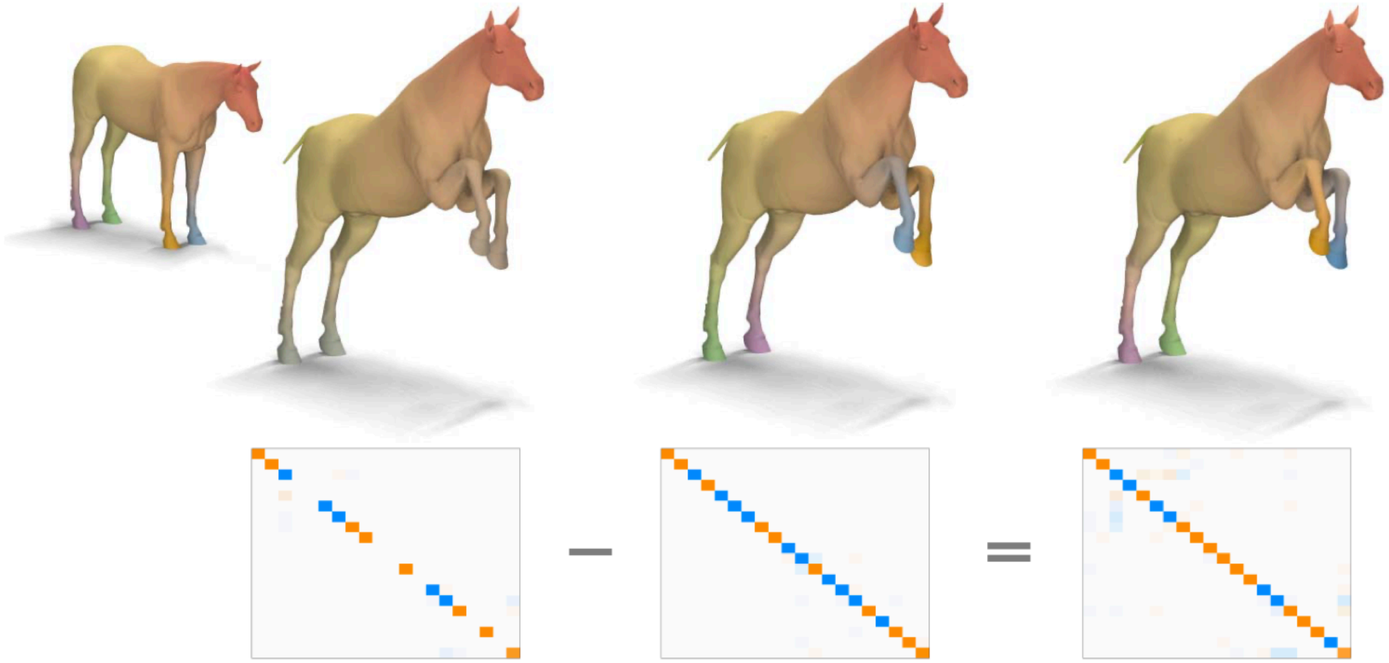
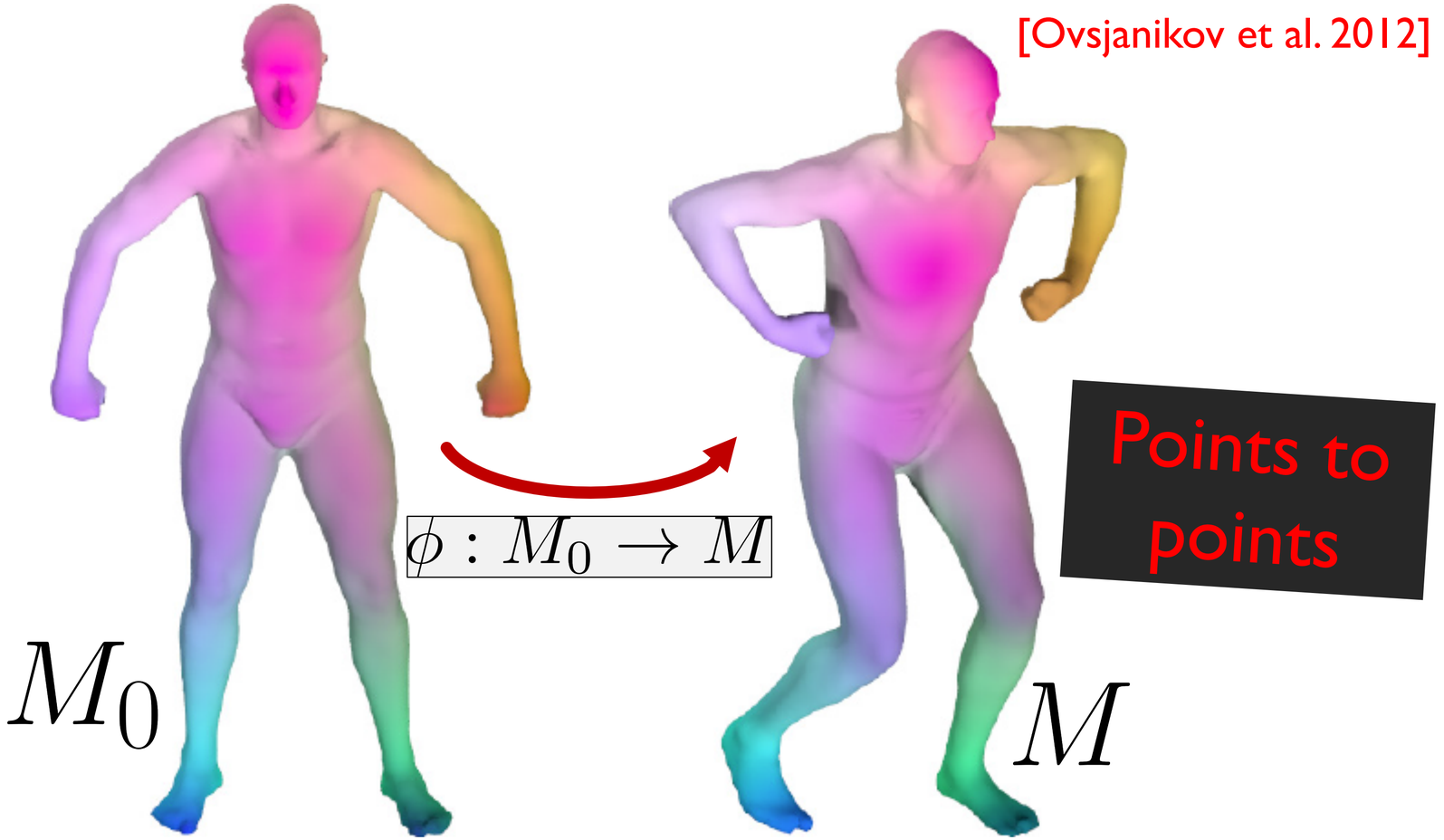


Figure 1: *Horse algebra: the functional representation and map inference algorithm allow us to go beyond point-to-point maps. The source shape (top left corner) was mapped to the target shape (left) by posing descriptor-based functional constraints which do not disambiguate symmetries (i.e. without landmark constraints). By further adding correspondence constraints, we obtain a near isometric map which reverses orientation, mapping left to right (center). The representation allows for algebraic operations on shape maps, so we can subtract this map from the ambivalent map, to retrieve the orientation preserving near-isometry (right). Each column shows the first 20x20 block of the functional map representation (bottom), and the action of the map by transferring colors from the source shape to the target shape (top).*

Functional Maps for Shape-to-Shape Correspondence

Slide Credit: Justin Solomon

[Ovsjanikov et al. 2012]

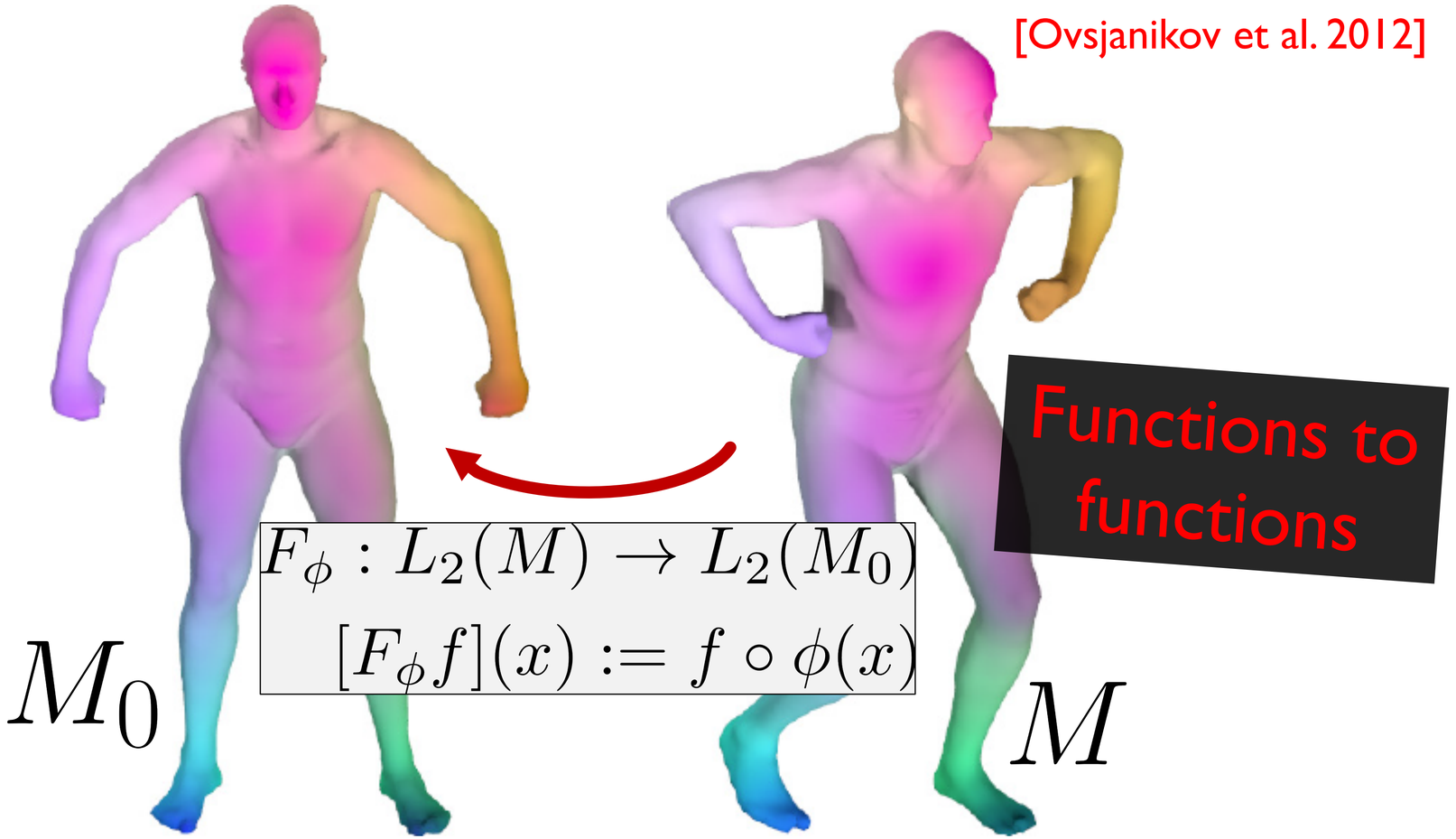


Points on M_0 to points on M

Functional Maps for Shape-to-Shape Correspondence

Slide Credit: Justin Solomon

[Ovsjanikov et al. 2012]

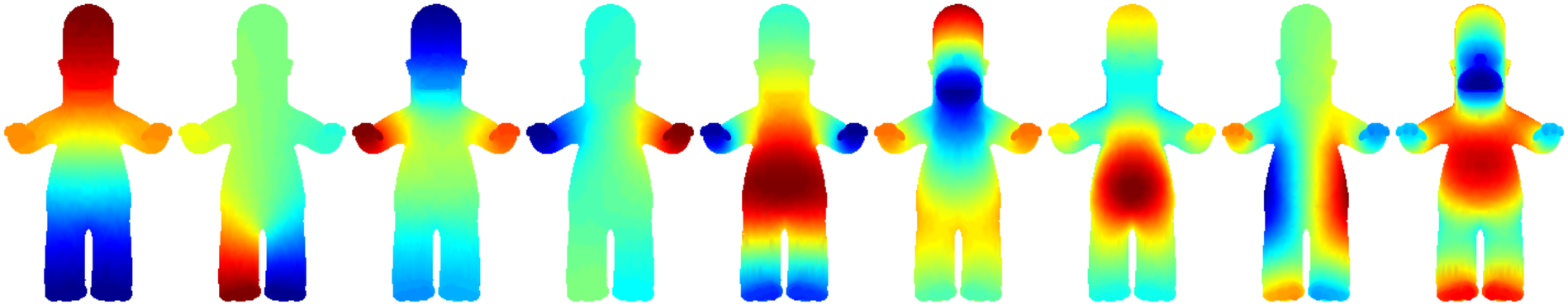


Functions on M to functions on M_0

Functional Maps for Shape-to-Shape Correspondence

Slide Credit: Justin Solomon

[Ovsjanikov et al. 2012]



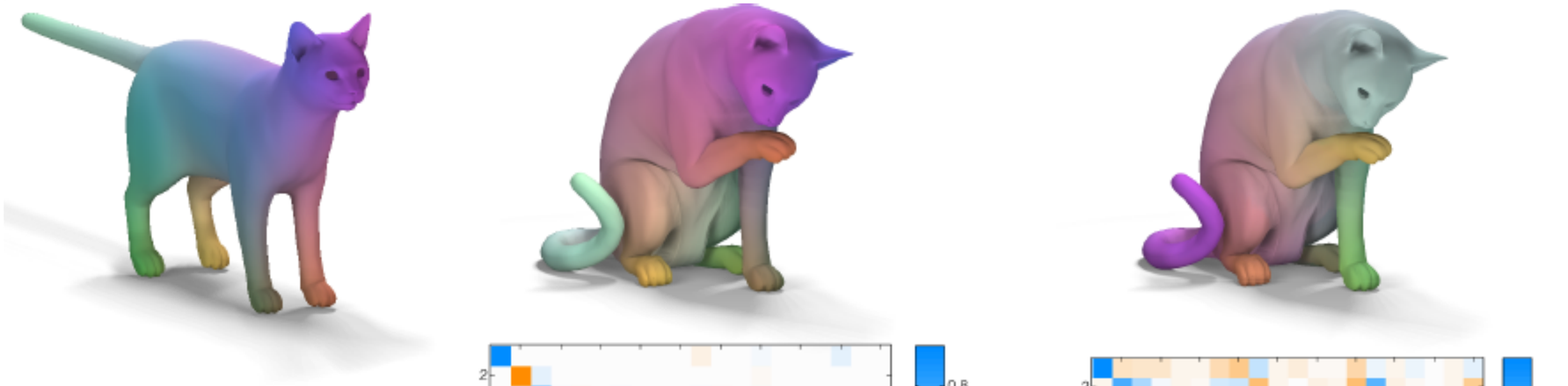
$$f(x) = \sum_i a_i \psi_i(x)$$

Functional map:

Matrix taking Laplace-Beltrami (Fourier) coefficients on M to coefficients on M_0

Example Maps

Slide Credit: Justin Solomon



Nearly identity:
Not a mistake!

(c) *left to right map*

(d) *head to tail map*

Adapted from slides by Q. Huang, V. Kim

Functional Maps for Shape-to-Shape Correspondence

Slide Credit: Justin Solomon

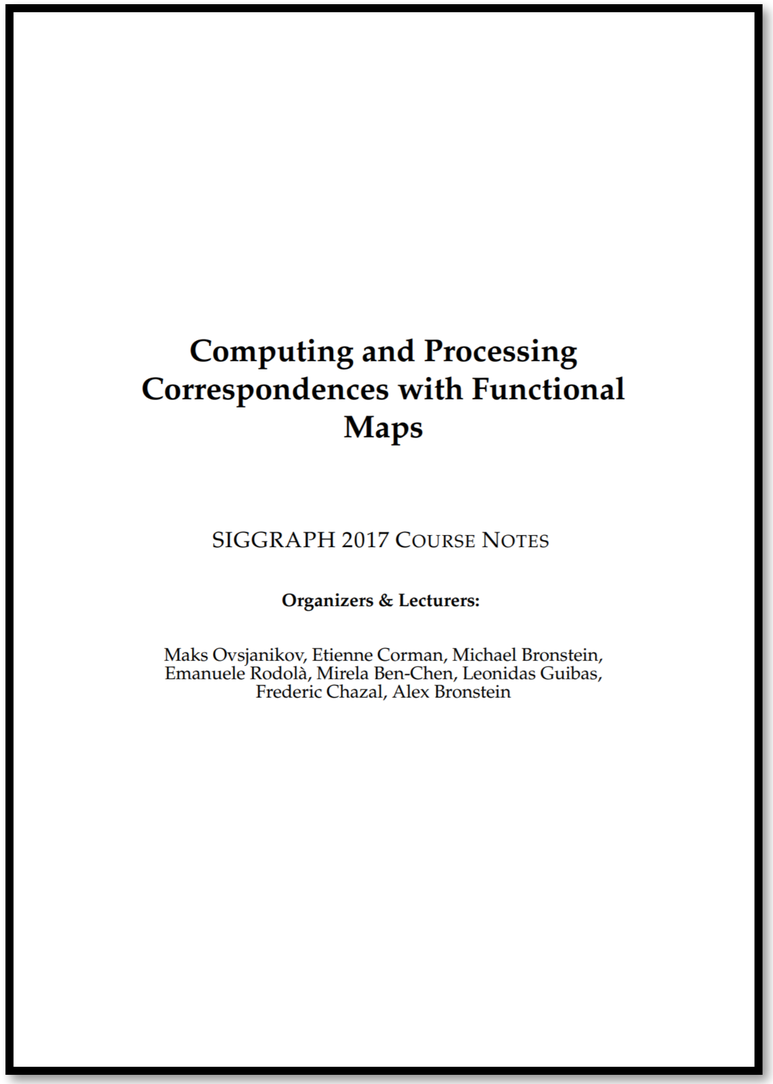
- Simple Algorithm
  - Compute some geometric functions to be preserved: A, B
  - Solve in least-squares sense for C: B = C A
- Additional Considerations
  - Favor commutativity
  - Favor orthonormality (if shapes are isometric)
  - Efficiently getting point-to-point correspondences

Adapted from slides by Q. Huang, V. Kim

Ovsjanikov'12

Useful Survey

Slide Credit: Justin Solomon



Computing and Processing Correspondences with Functional Maps

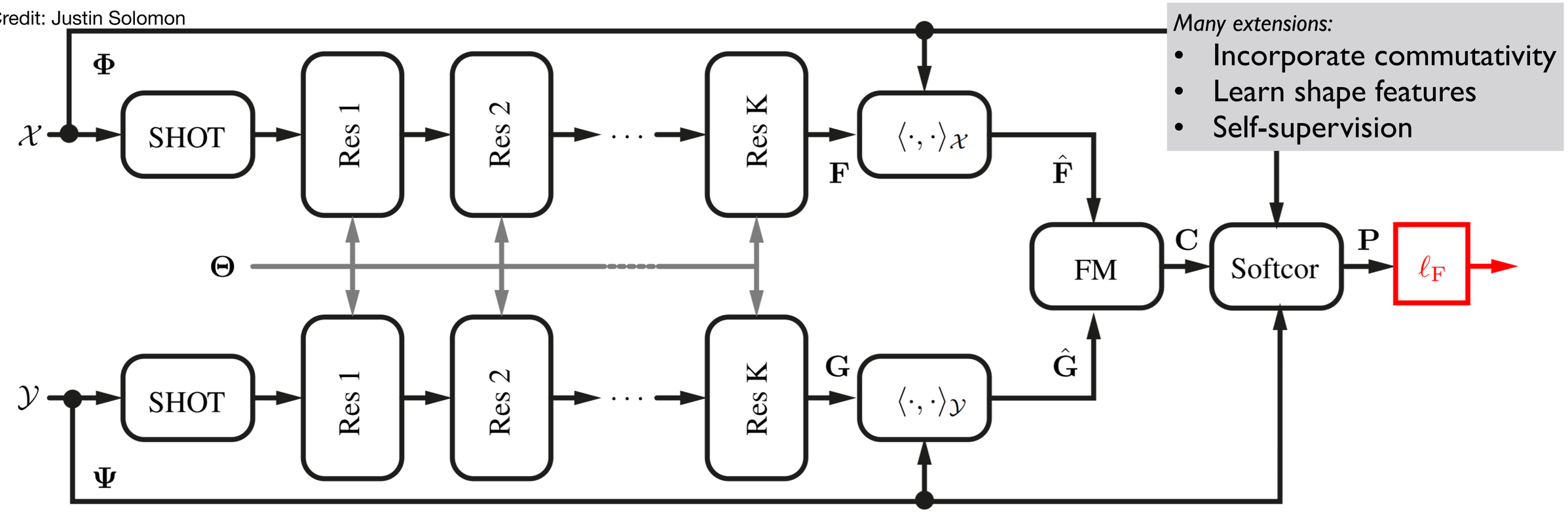
SIGGRAPH 2017 COURSE NOTES

Organizers & Lecturers:

Maks Ovsjanikov, Etienne Corman, Michael Bronstein, Emanuele Rodolà, Mirela Ben-Chen, Leonidas Guibas, Frederic Chazal, Alex Bronstein

Deep Functional Maps

Slide Credit: Justin Solomon



Many extensions:

- Incorporate commutativity
- Learn shape features
- Self-supervision

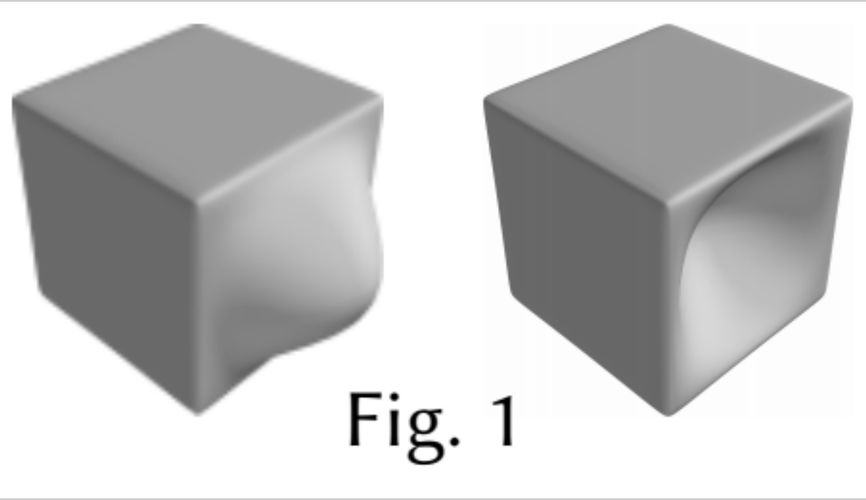
Figure 3. **FMNet architecture.** Input point-wise descriptors (SHOT [38] in this paper) from a pair of shapes are passed through an identical sequence of operations (with shared weights), resulting in refined descriptors $\mathbf{F}, \mathbf{G}$. These, in turn, are projected onto the Laplacian eigenbases $\mathbf{\Phi}, \mathbf{\Psi}$ to produce the spectral representations $\hat{\mathbf{F}}, \hat{\mathbf{G}}$. The functional map (FM) and soft correspondence (Softcor) layers, implementing Equations (3) and (6) respectively, are not parametric and are used to set up the geometrically structured loss ℓ_{F} (5).

"Deep functional maps: Structured prediction for dense shape correspondence" (Litany et al. 2017)

Eigenvectors for extrinsic geometry

Intrinsic vs. Extrinsic geometry

Same intrinsic geometry.
Different extrinsic



- Laplace Beltrami focuses on intrinsic geometry (observable by measurements purely on the 2D surface)
- Extrinsic geometry refers to the 3D ambient space
- Sometimes we car about extrinsic geometry

Steklov Spectral Geometry for Extrinsic Shape Analysis

YU WANG, Massachusetts Institute of Technology, USA
MIRELA BEN-CHEN, Technion – Israel Institute of Technology, Israel
IOSIF POLTEROVICH, Université de Montréal, Canada
JUSTIN SOLOMON, Massachusetts Institute of Technology, USA

Replace heat diffusion on surface to heat diffusion onside volume

Bring back to surface calculations (boundary elements)

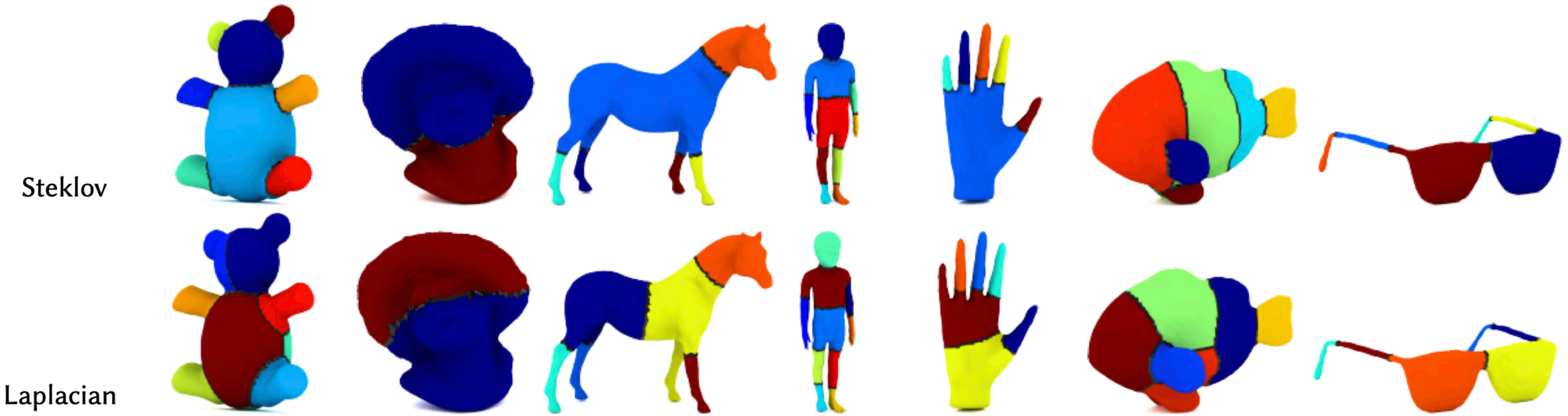


Fig. 22. Results of segmentation by the Steklov and Laplacian embeddings.

Geometric Deep Learning

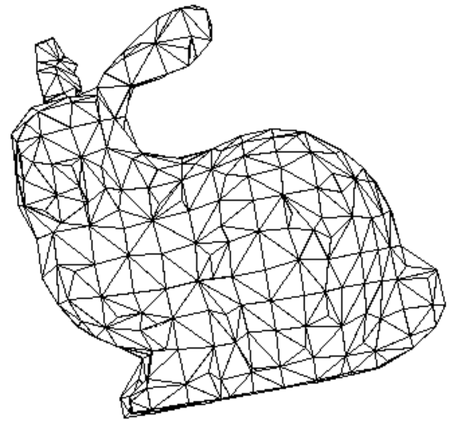
Challenges to extend CNNs to meshes / Points

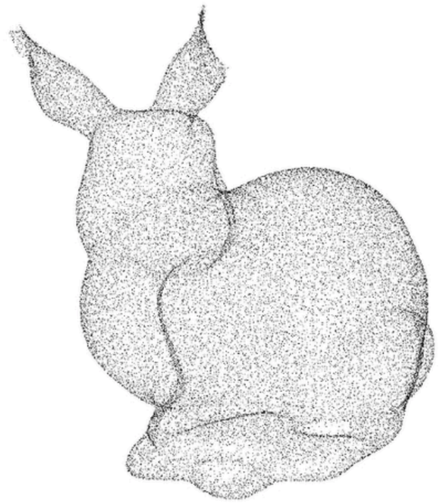
No nice regular lattice domain

Domain and range are conflated

Convolution not well defined

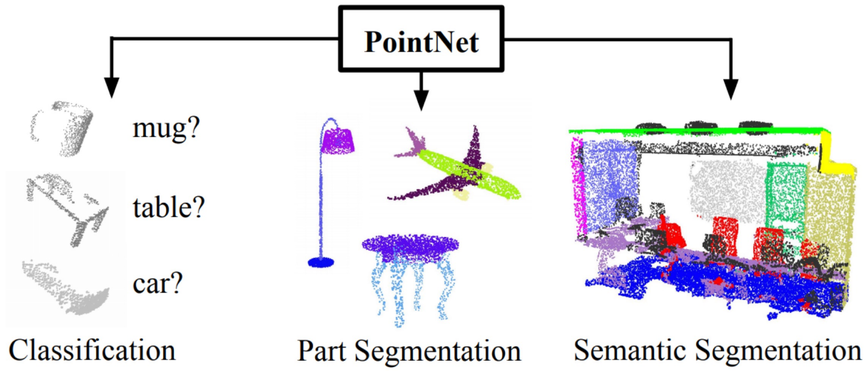
Point sets and triangle mesh encoded up to permutation

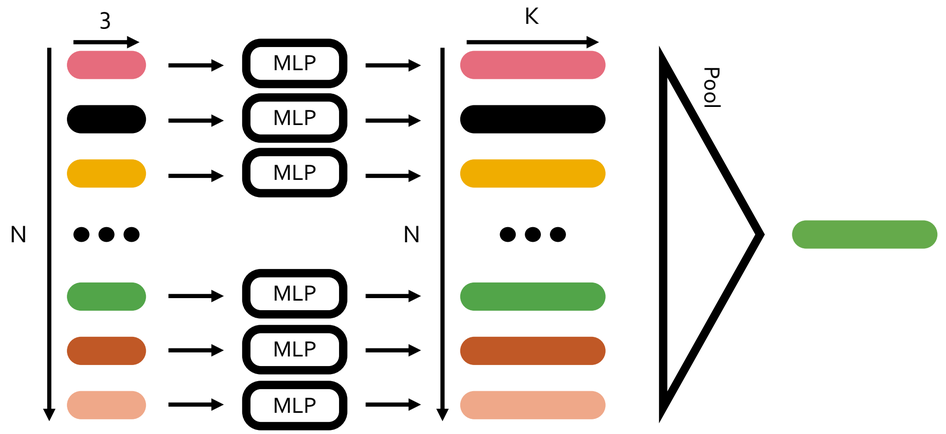




Models for Point Cloud Learning

PointNet pioneered deep learning on point clouds.





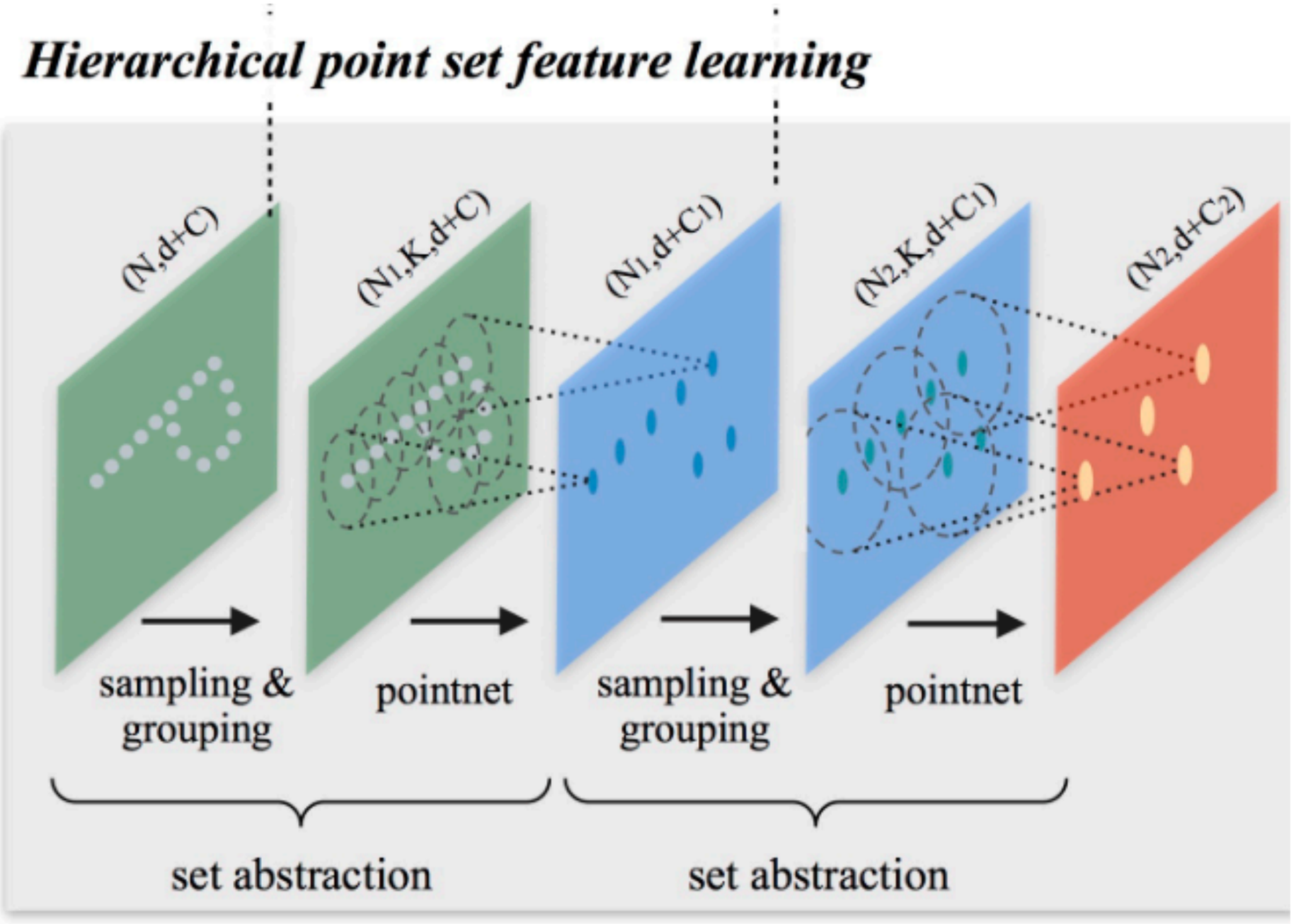
PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. Charles R. Qi, Hao Su, Kaichun Mo, Leonidas J. Guibas

But each point is handled independently!

Plus global max pooling

PointNet++

- Sample subset of points
- Build neighborhood (kNN or ball)
- Compute permutation-invariant MLP on neighbors (max pooling)
- Build hierarchy



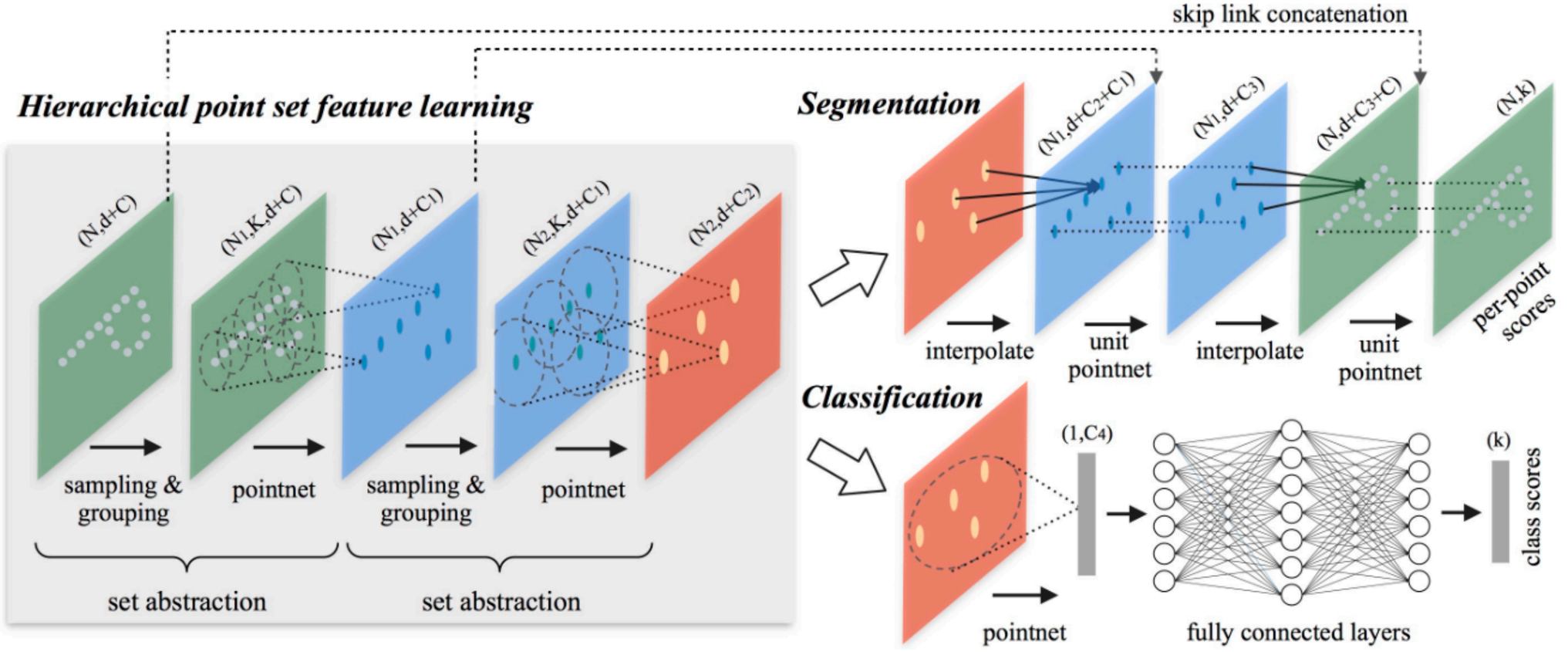


Figure 1. **PointNet++ Architecture for Point Set Segmentation and Classification.** We introduce a type of novel neural network, named as PointNet++, to process a set of points sampled in a metric space in a hierarchical fashion (2D points in Euclidean space are used for this illustration). The general idea of PointNet++ is simple. We first partition the set of points into overlapping local regions by the distance metric of the underlying space. Similar to CNNs, we extract local features capturing fine geometric structures from small neighborhoods; such local features are further grouped into larger units and processed to produce higher level features. This process is repeated until we obtain the features of the whole point set.

Dynamic Graph CNN for Learning on Point Clouds

YUE WANG, Massachusetts Institute of Technology
YONGBIN SUN, Massachusetts Institute of Technology
ZIWEI LIU, UC Berkeley / ICSI
SANJAY E. SARMA, Massachusetts Institute of Technology
MICHAEL M. BRONSTEIN, Imperial College London / USI Lugano
JUSTIN M. SOLOMON, Massachusetts Institute of Technology

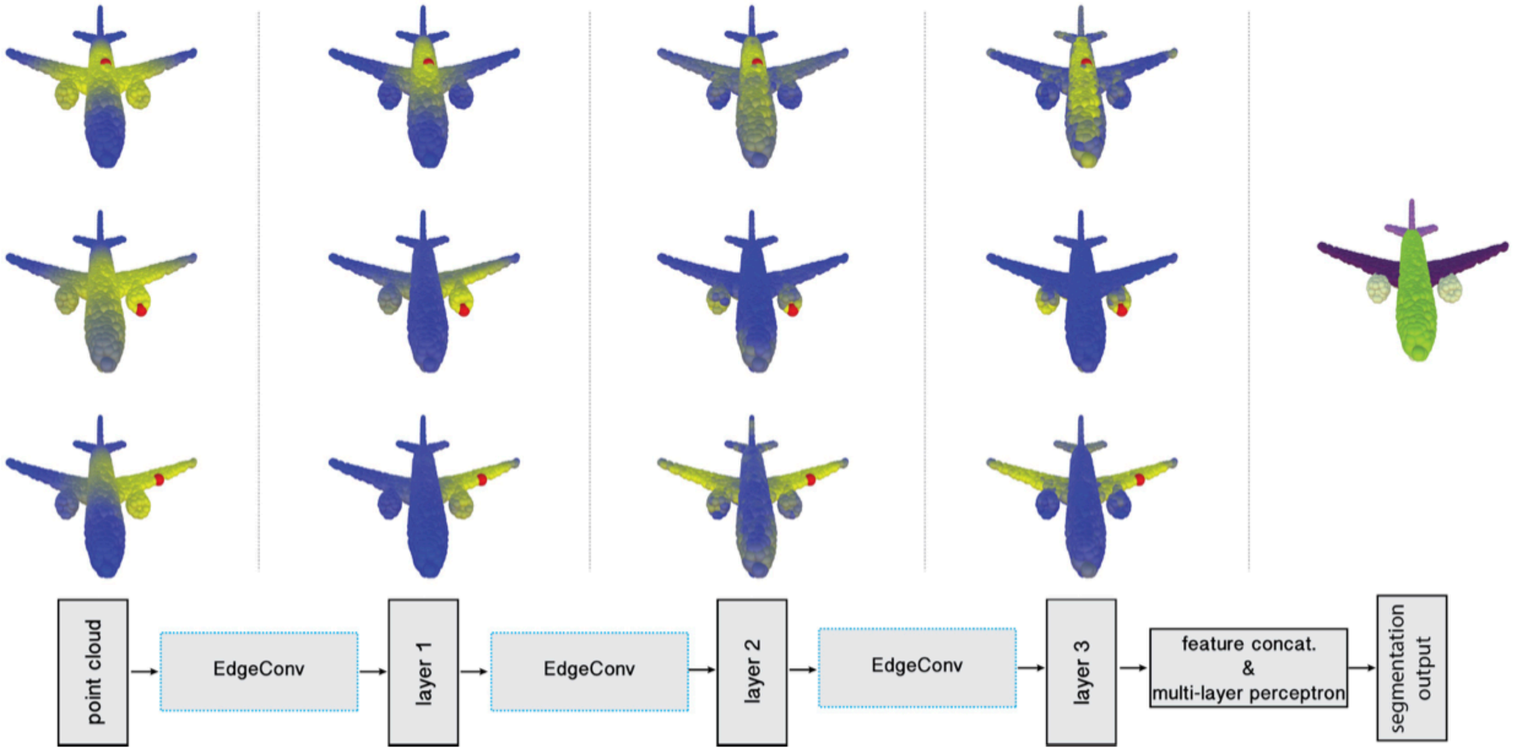
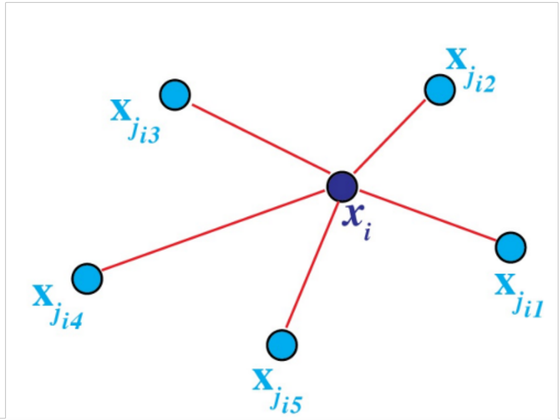


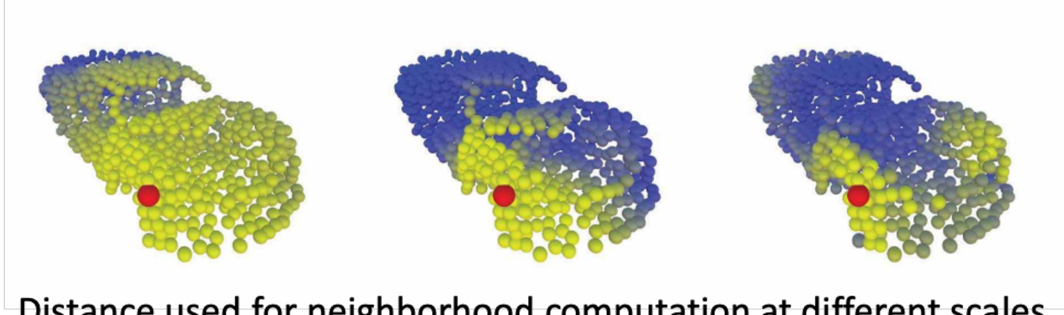
Fig. 1. **Point cloud segmentation using the proposed neural network.** Bottom: schematic neural network architecture. Top: Structure of the feature spaces produced at different layers of the network, visualized as the distance from the red point to all the rest of the points (shown left-to-right are the input and layers 1-3; rightmost figure shows the resulting segmentation). Observe how the feature space structure in deeper layers captures semantically similar structures such as wings, fuselage, or turbines, despite a large distance between them in the original input space.

Dynamic-Graph CNN

- Two main ideas
  - general function on neighbors (takes xi and xj into account, not just xj)
  - Recompute neighborhood (graph) at different scales based on previous features

    Predates but is related to attention



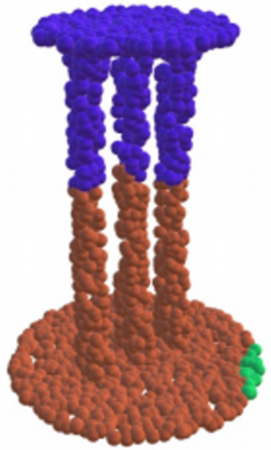


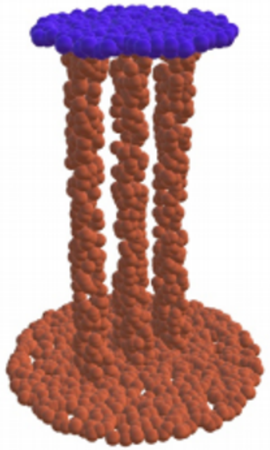
Distance used for neighborhood computation at different scales
Note how dynamic the neighborhood is and how semantically-similar parts like the headlights become despite being far in the original space

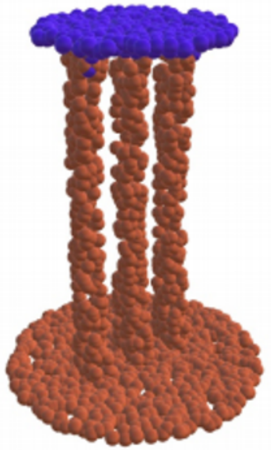
PointNet

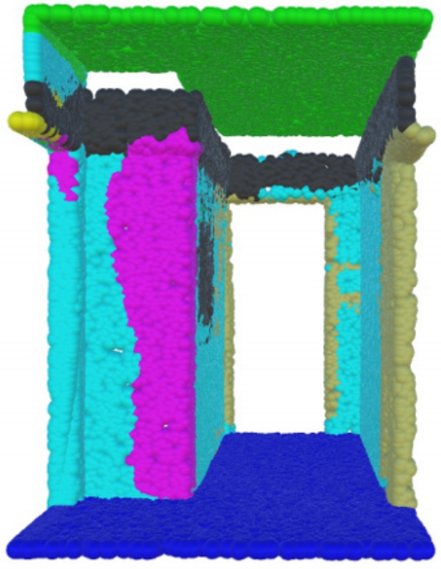
EdgeConv

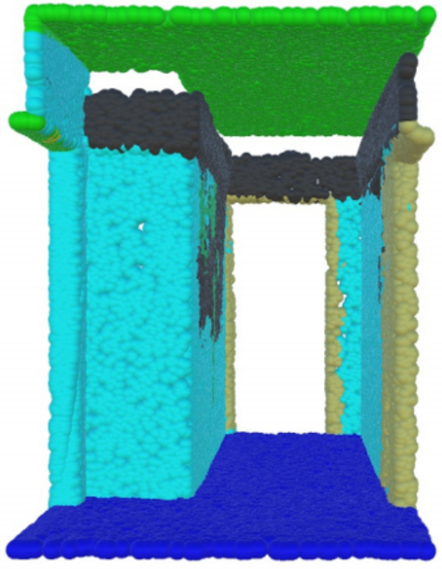
Ground Truth

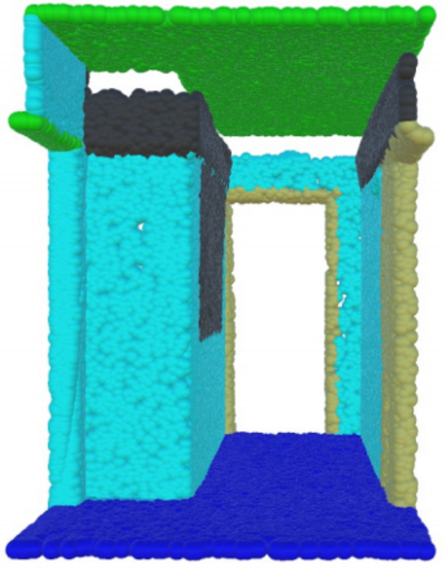












Part and Semantic segmentation results

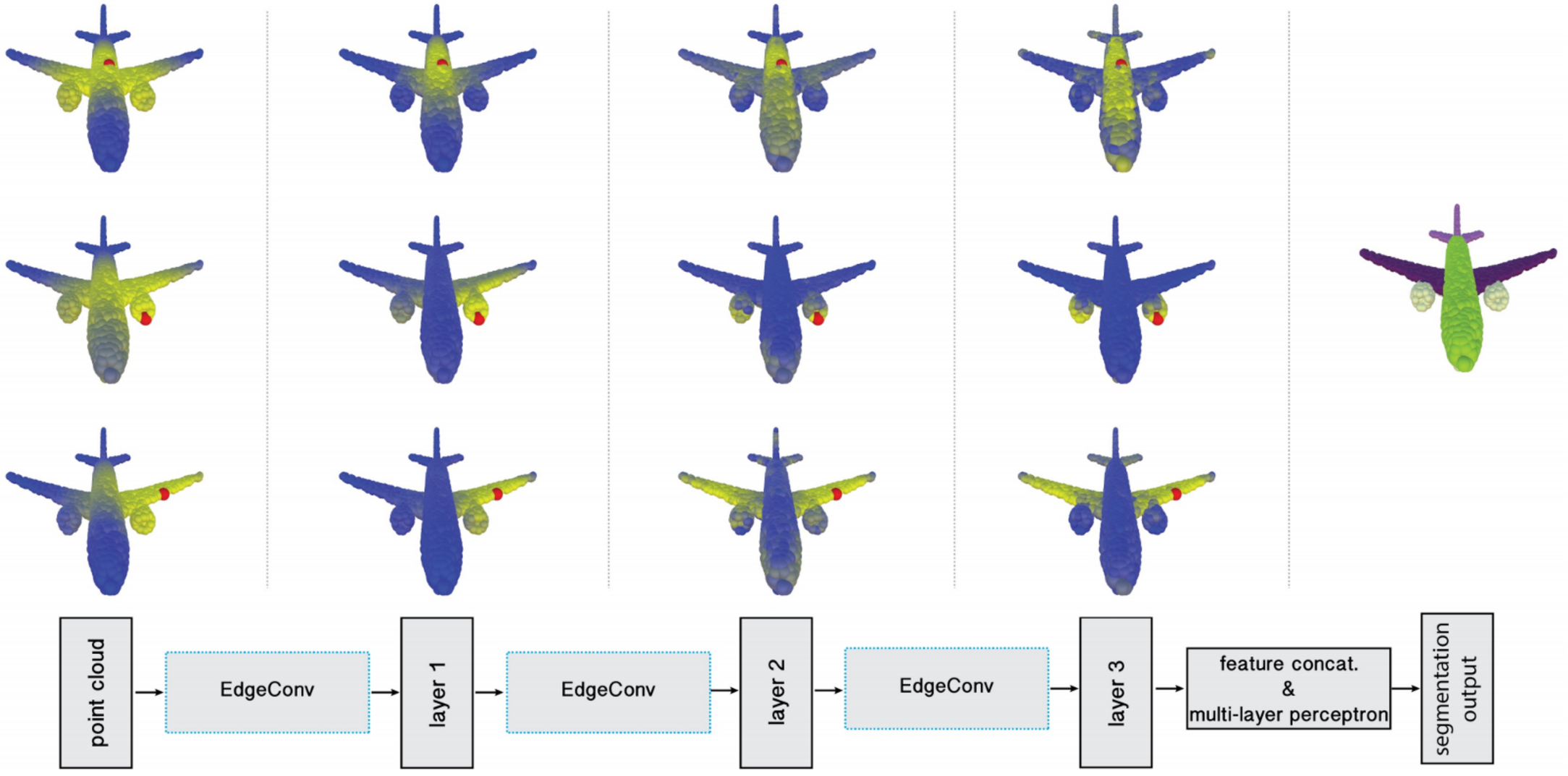


Fig. 1. **Point cloud segmentation using the proposed neural network.** Bottom: schematic neural network architecture. Top: Structure of the feature spaces produced at different layers of the network, visualized as the distance from the red point to all the rest of the points (shown left-to-right are the input and layers 1-3; rightmost figure shows the resulting segmentation). Observe how the feature space structure in deeper layers captures semantically simila structures such as wings, fuselage, or turbines, despite a large distance between them in the original input space.

Other approaches

Voxelize everything to get back to square lattice

- Possibly add sparsity for efficiency
- Possibly keep points inside voxel for fine details

Sparse voxel CNN, MinkovskiNet, VoxelNet

Hybrid point-voxels (PV-RCNN / PV-RCNN++ ; SPVCNN / SPVConv)

4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks

Christopher Choy chrischoy@stanford.edu
JunYoung Gwak jgwak@stanford.edu
Silvio Savarese ssilvio@stanford.edu

Abstract

In many robotics and VR/AR applications, 3D-videos are readily-available sources of input (a continuous sequence of depth images, or LIDAR scans). However, these 3D-videos are processed frame-by-frame either through 2D convnets or 3D perception algorithms in many cases. In this work, we propose 4-dimensional convolutional neural networks for



Figure 1: An example of 3D video: 3D scenes at different time steps. Best viewed on display.

VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection

Yin Zhou
Apple Inc
yzhou3@apple.com
Oncel Tuzel
Apple Inc
otuzel@apple.com

Abstract

Accurate detection of objects in 3D point clouds is a central problem in many applications, such as autonomous navigation, housekeeping robots, and augmented/virtual reality. To interface a highly sparse LiDAR point cloud with a region proposal network (RPN), most existing efforts have focused on hand-crafted feature representations, for example, a bird's eye view projection. In this work, we remove the need of manual feature engineering for 3D point clouds and propose VoxelNet, a generic 3D detection network that unifies feature extraction and bounding box prediction into a single stage, end-to-end trainable deep network. Specifically, VoxelNet divides a point cloud into equally spaced 3D voxels and transforms a group of points within each voxel into a unified feature representation through the newly introduced voxel feature encoding (VFE) layer. In this way,

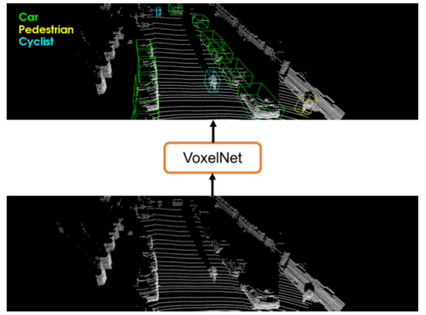


Figure 1. VoxelNet directly operates on the raw point cloud (no need for feature engineering) and produces the 3D detection results using a single end-to-end trainable network.

Equivariant deep learning

Goal

Build in symmetries to make learning more efficient

e.g. shift invariance, rotation invariance

Also related to physics-informed deep learning

Form of inductive bias

Vocabulary discussion

Invariant as constant

Invariant as commutative

Equivariant / steerable

Most Slides Adapted From:

UNIVERSITY OF AMSTERDAM

AMLAB Amsterdam Machine Learning Lab

Group Equivariant Deep Learning

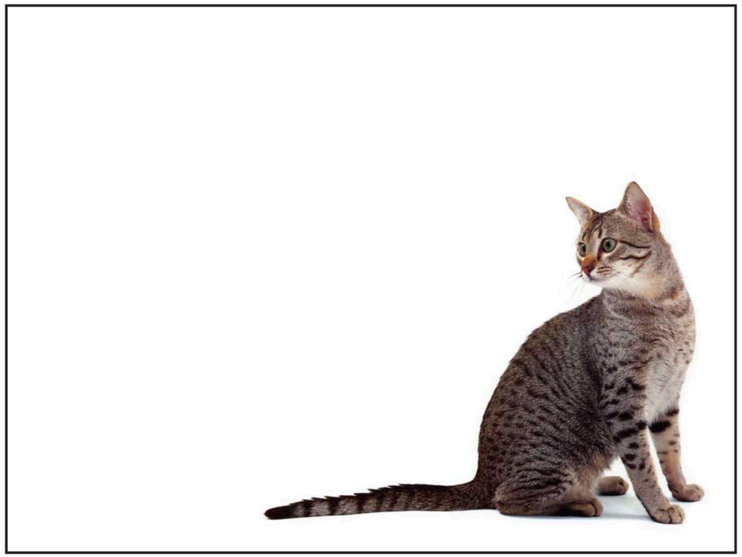


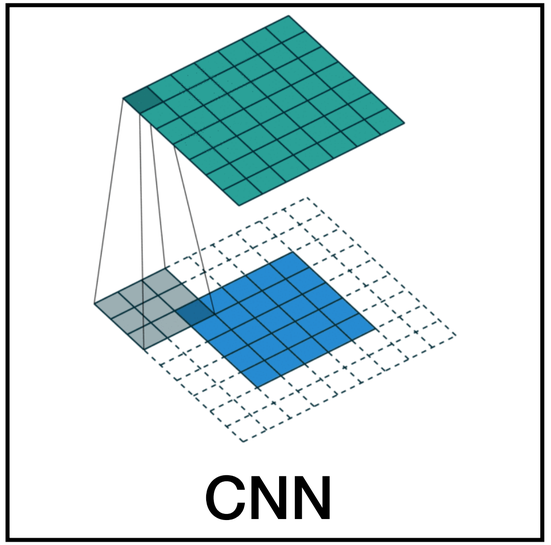
Erik Bekkers, Amsterdam Machine Learning Lab, University of Amsterdam
This mini-course serves as a module with the UvA Master AI course Deep Learning 2 https://uvadl2c.github.io/

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

Go there for more complete/serious treatment

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





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

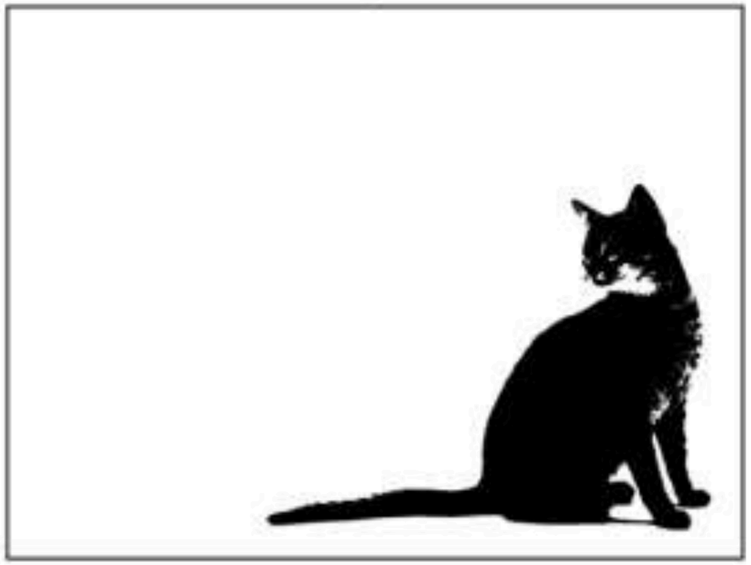
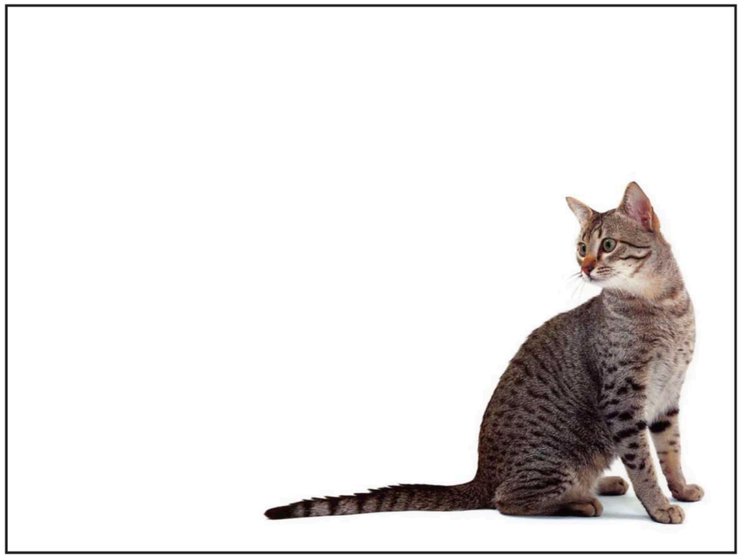
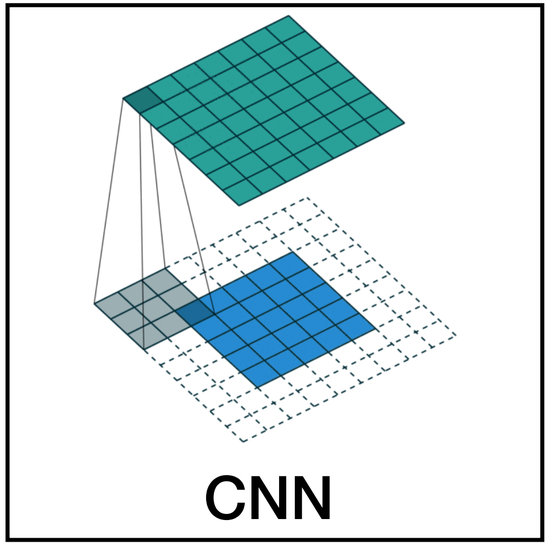


Image: Bernhard Kainz

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





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

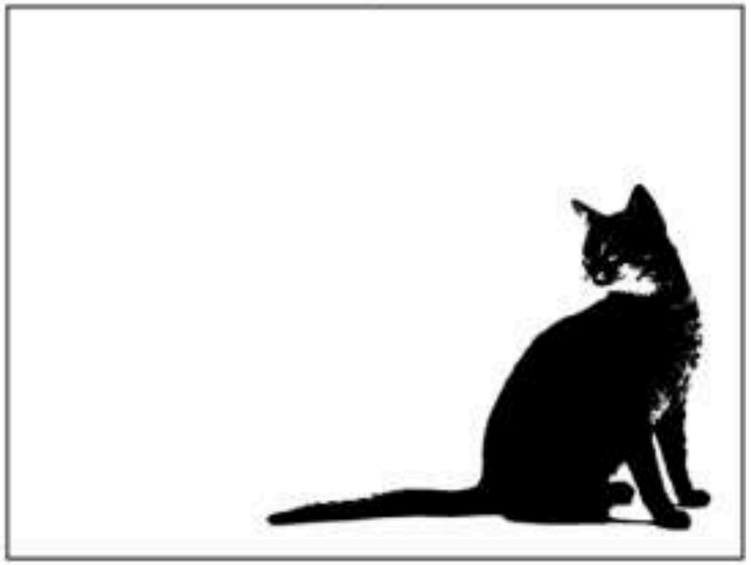


Image: Bernhard Kainz

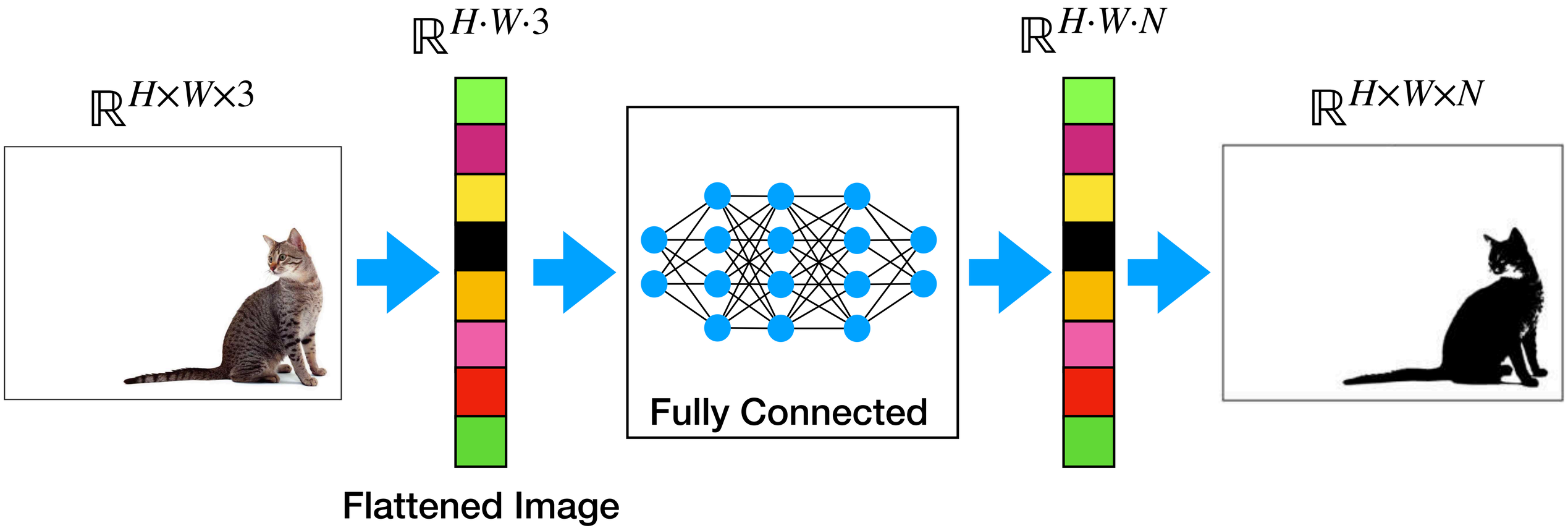


Image: Bernhard Kainz

Why is this a bad idea?

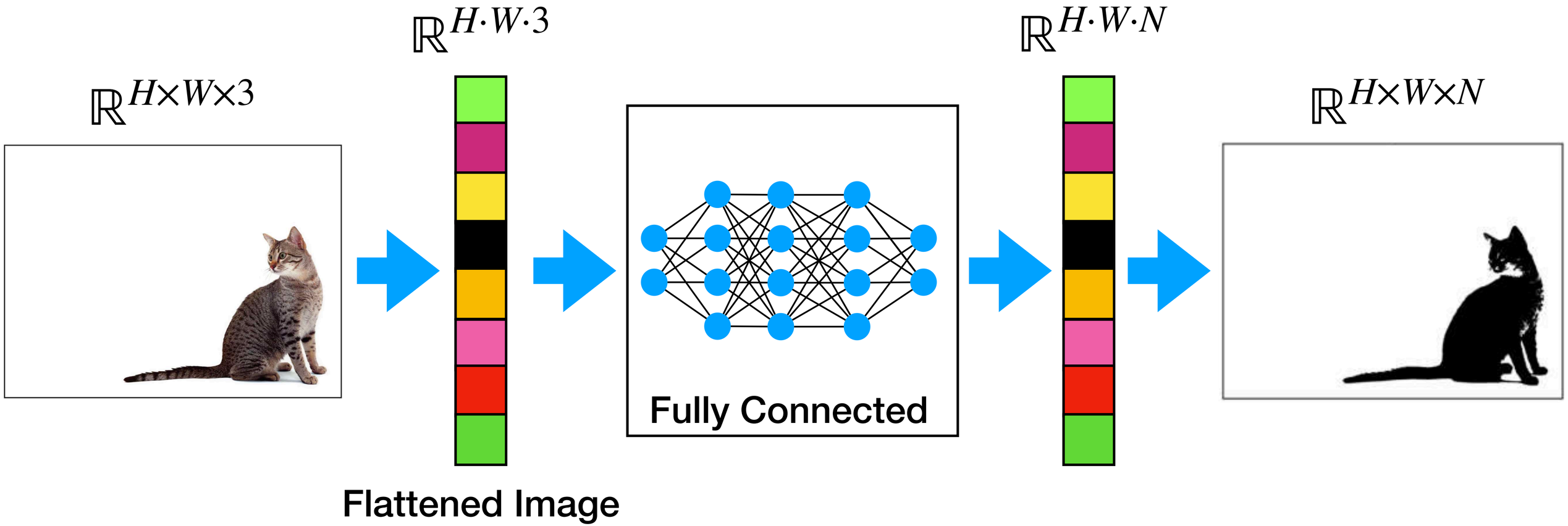


Image: Bernhard Kainz

Why is this a bad idea?

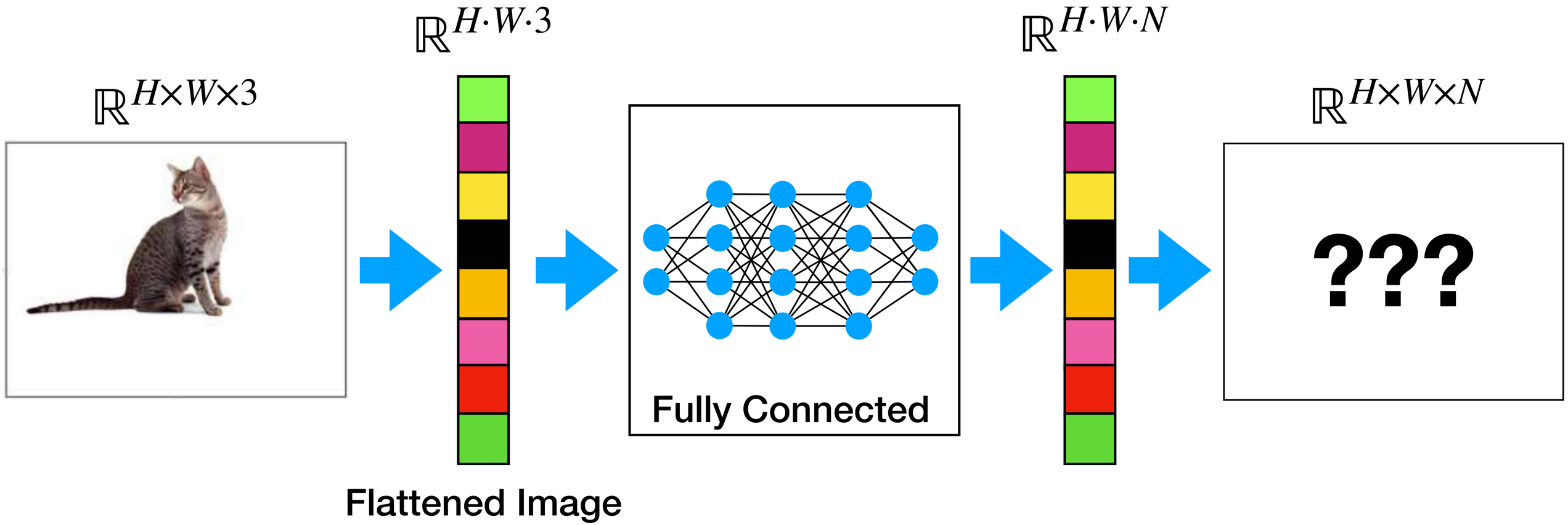
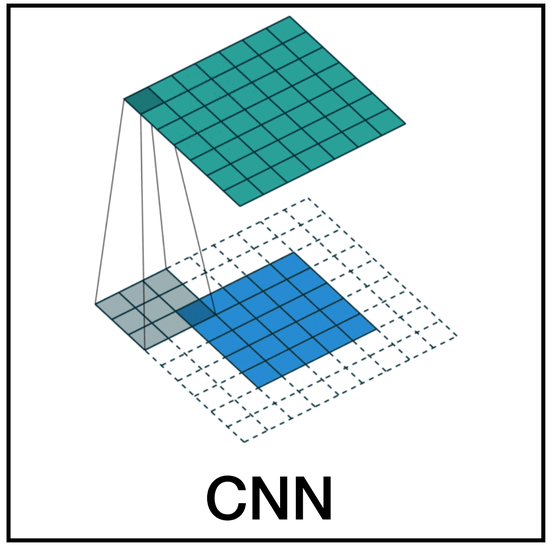


Image: Bernhard Kainz

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





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

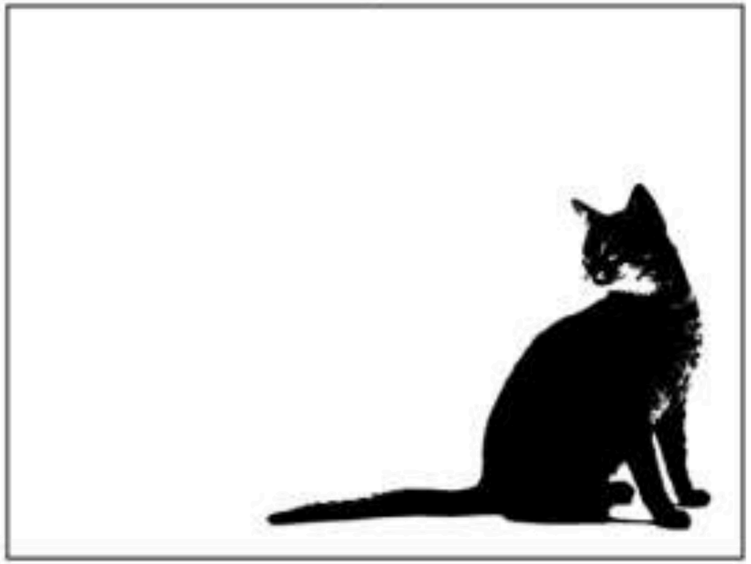
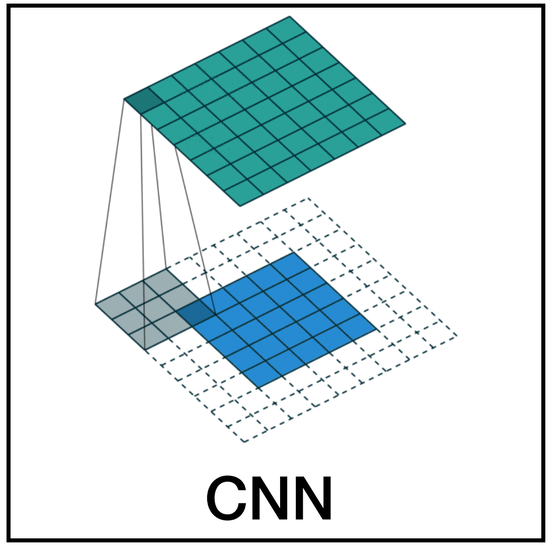


Image: Bernhard Kainz

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





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

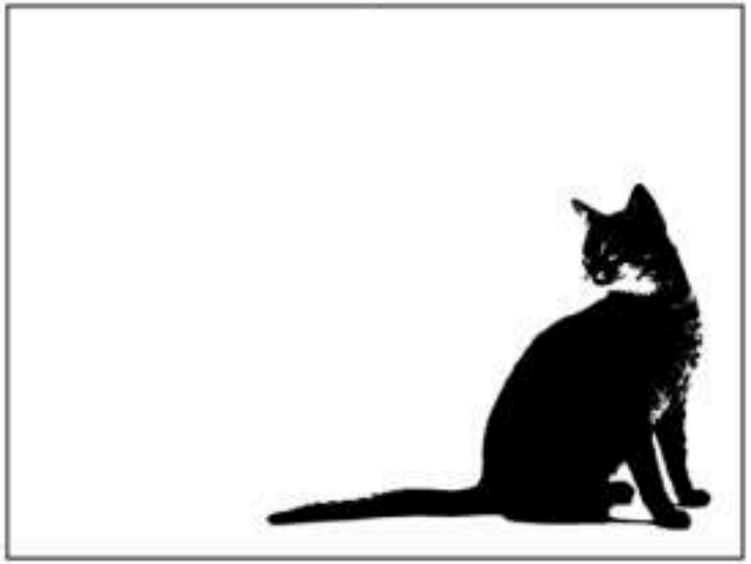
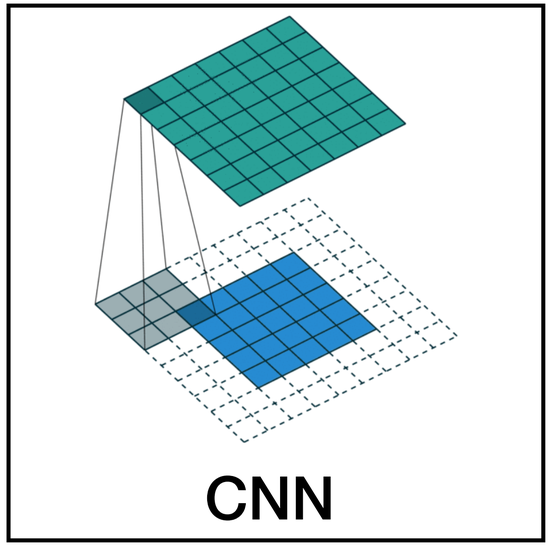


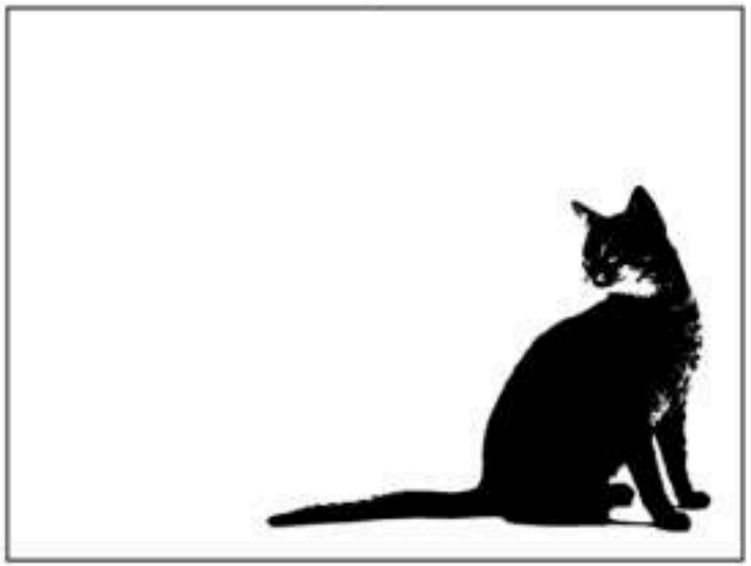
Image: Bernhard Kainz

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

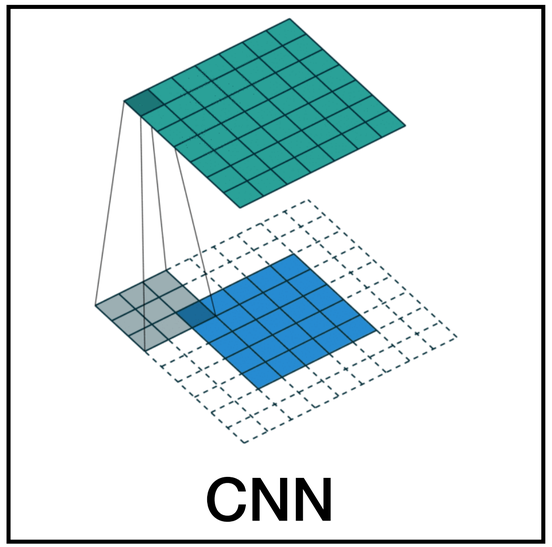




$\mathbb{R}^{H \times W \times N}$







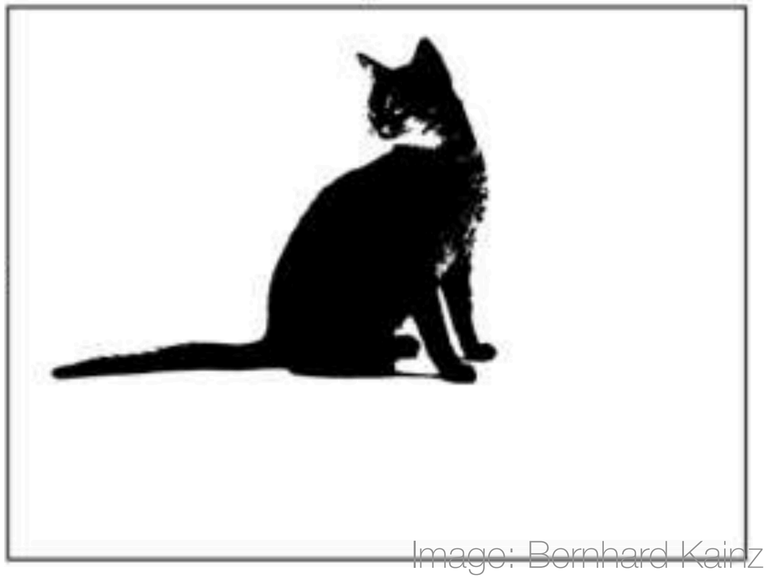
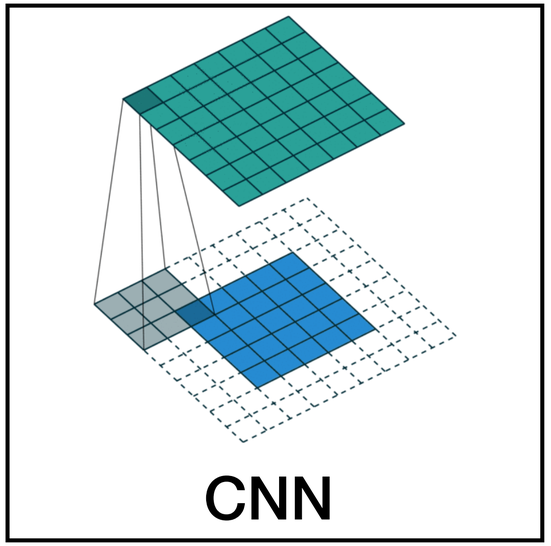


Image: Bernhard Kainz

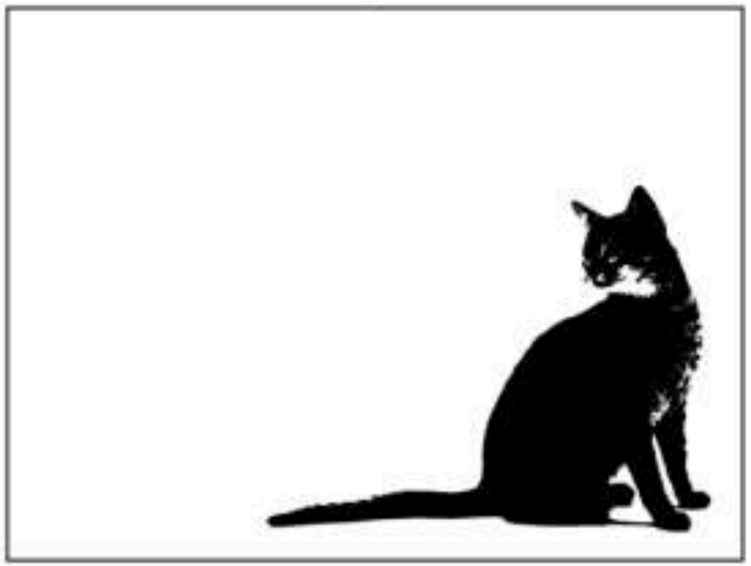
“Shift Equivariance”

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

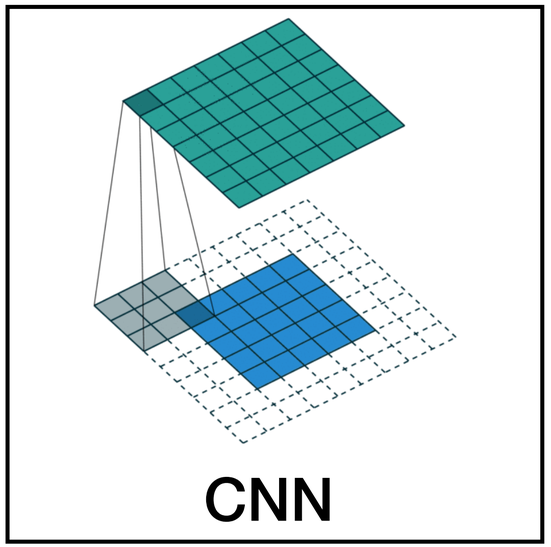




$\mathbb{R}^{H \times W \times N}$







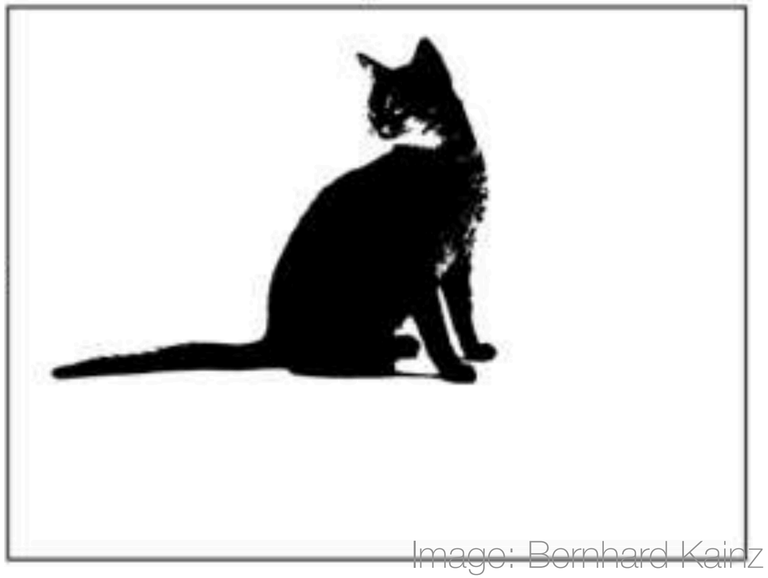


Image: Bernhard Kainz

To be fancy: Group Representations

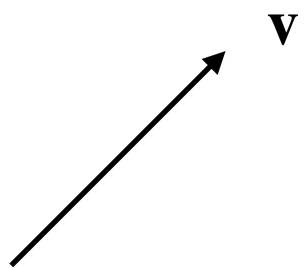
A representation $\rho : G \rightarrow GL(V)$ is a function from G to the general linear group $GL(V)$.

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

To be fancy: Group Representations



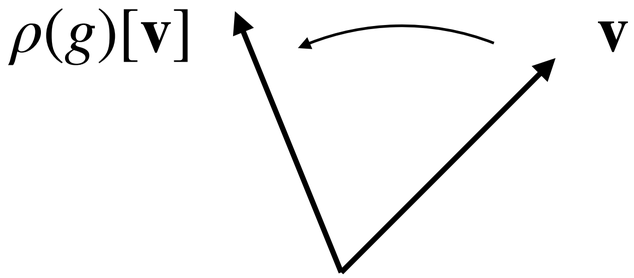
A representation $\rho : G \rightarrow GL(V)$ is a function from G to the general linear group $GL(V)$.

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

To be fancy: Group Representations



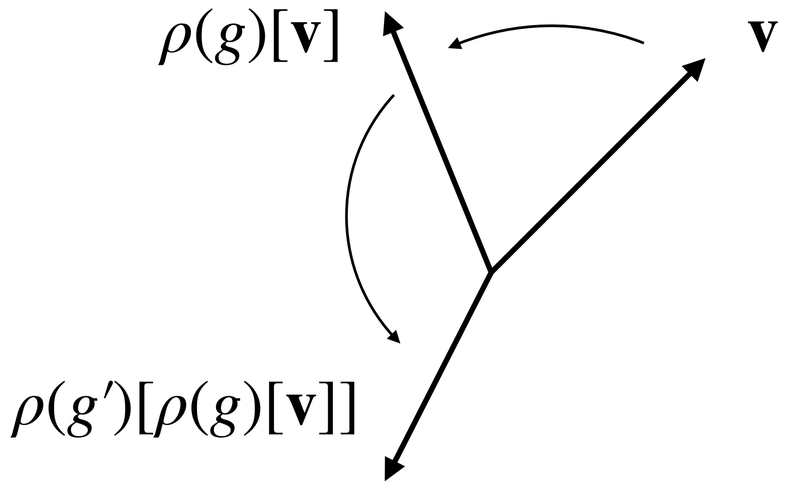
A representation $\rho : G \rightarrow GL(V)$ is a function from G to the general linear group $GL(V)$.

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

To be fancy: Group Representations



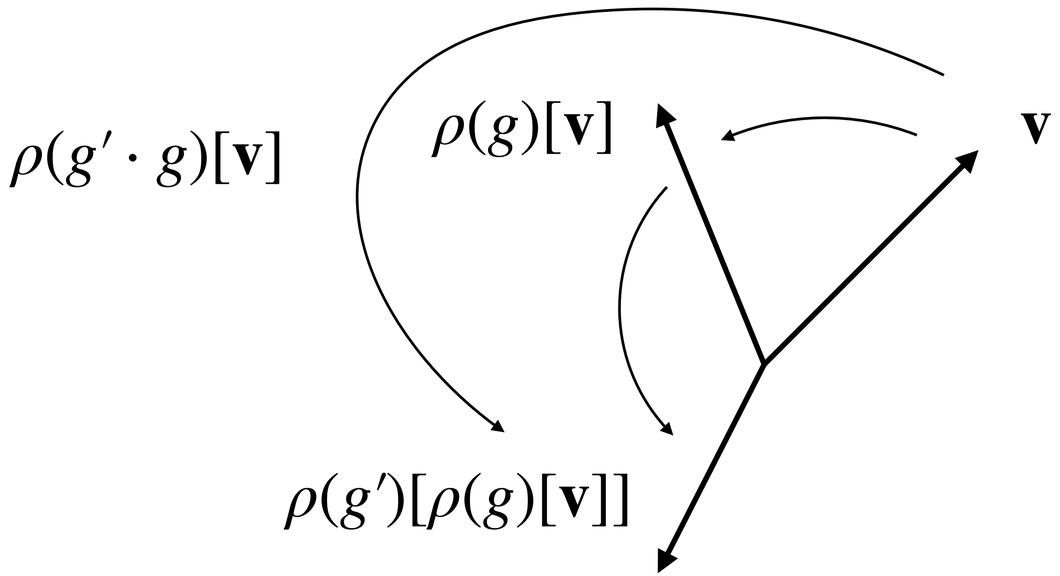
A representation $\rho : G \rightarrow GL(V)$ is a function from G to the general linear group $GL(V)$.

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

To be fancy: 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

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

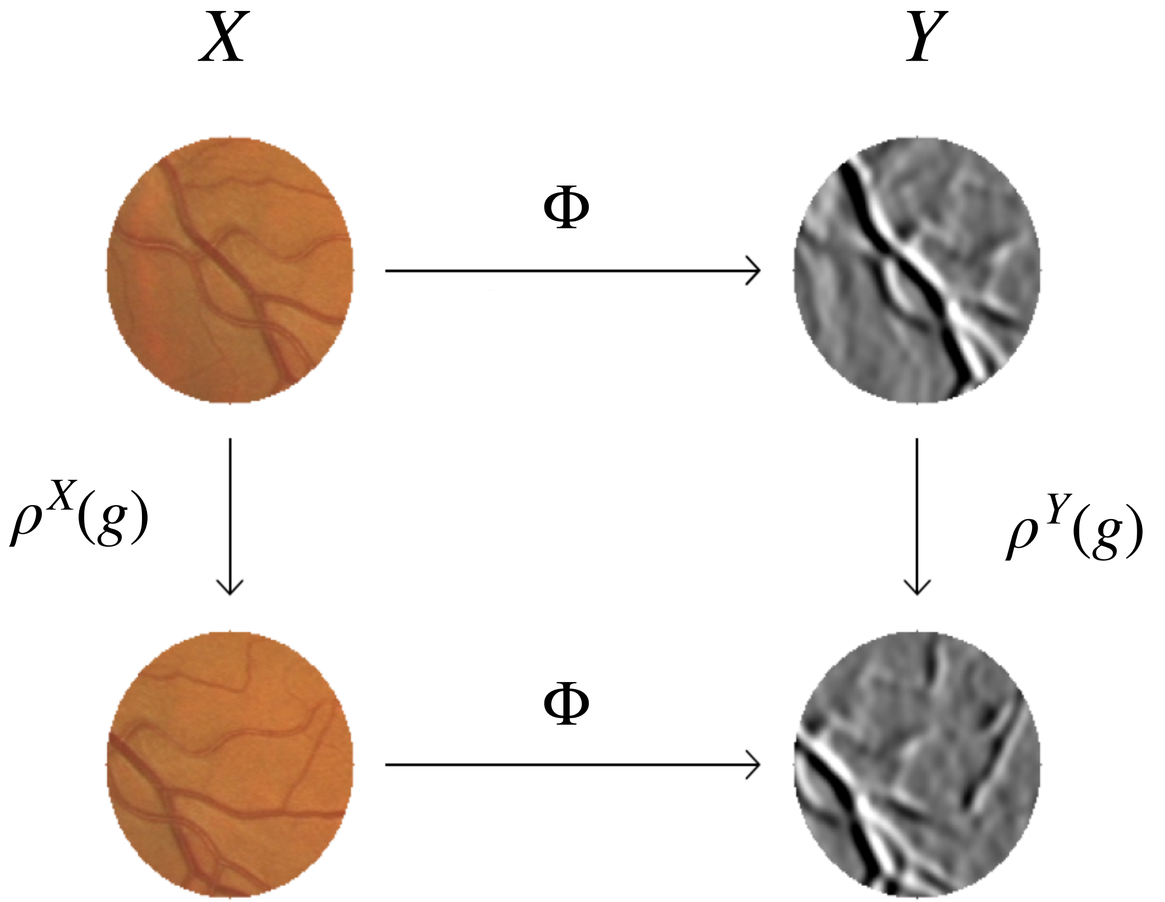
Slide credit: Erik Bekkers, University of Amsterdam

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

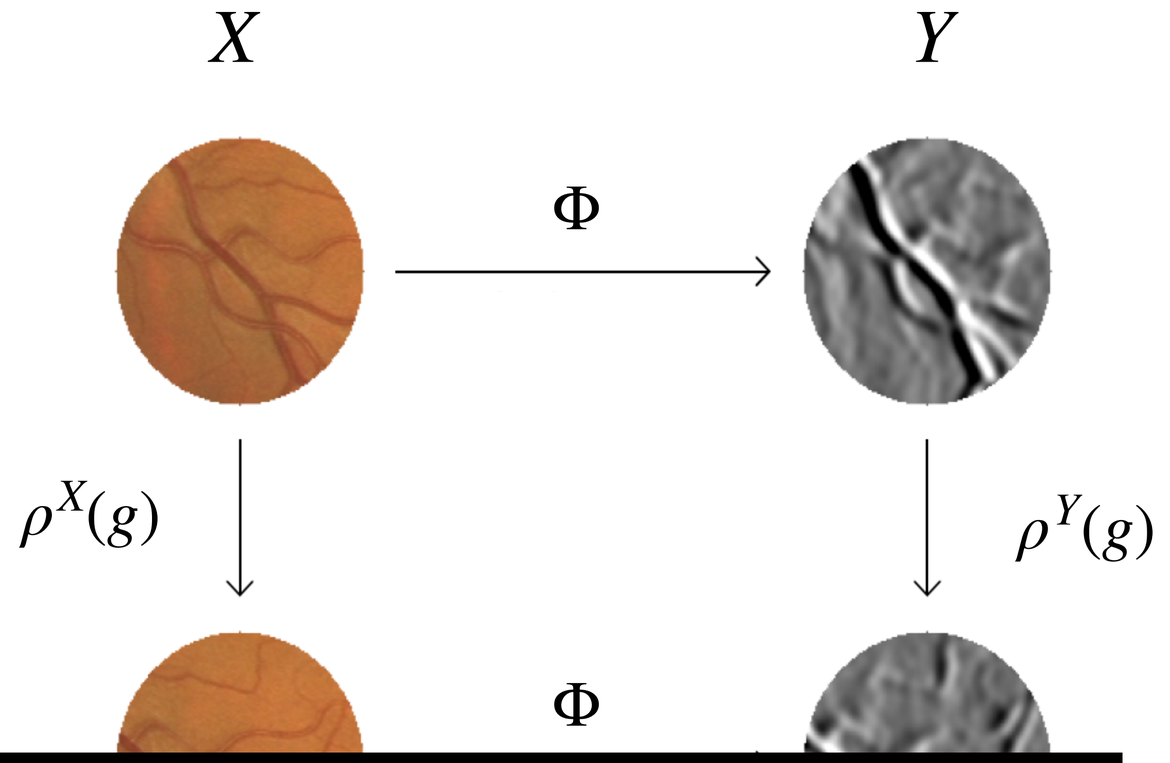


Slide credit: Erik Bekkers, University of Amsterdam

Equivariance

Equivariance is a property of a function $\Phi : X \rightarrow Y$ (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

Well at least that's one definition of invariance :-(depends on context. See next slide

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

Well at least that's one definition of invariance :-(depends on context. See next slide

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

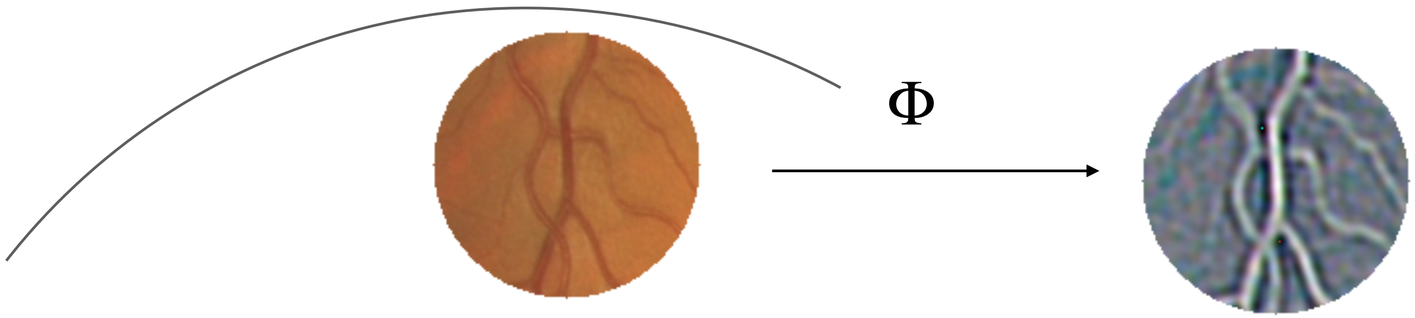
Slide credit: Erik Bekkers, University of Amsterdam

Invariance

Well at least that's one definition of invariance :-(depends on context. See next slide

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

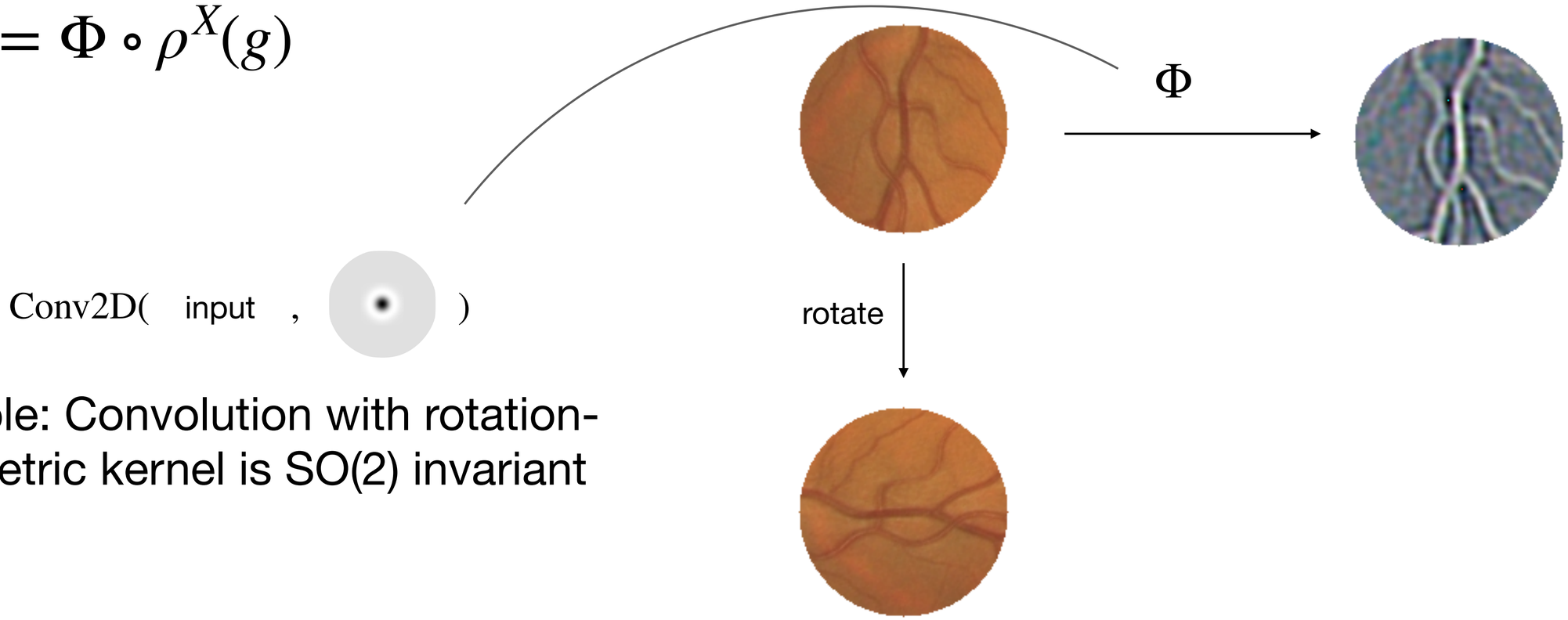
Slide credit: Erik Bekkers, University of Amsterdam

Invariance

Well at least that's one definition of invariance :-(depends on context. See next slide

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

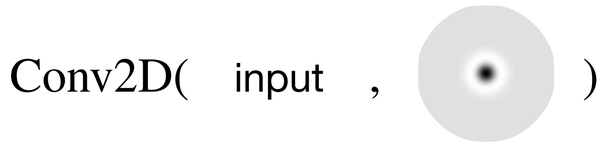
Slide credit: Erik Bekkers, University of Amsterdam

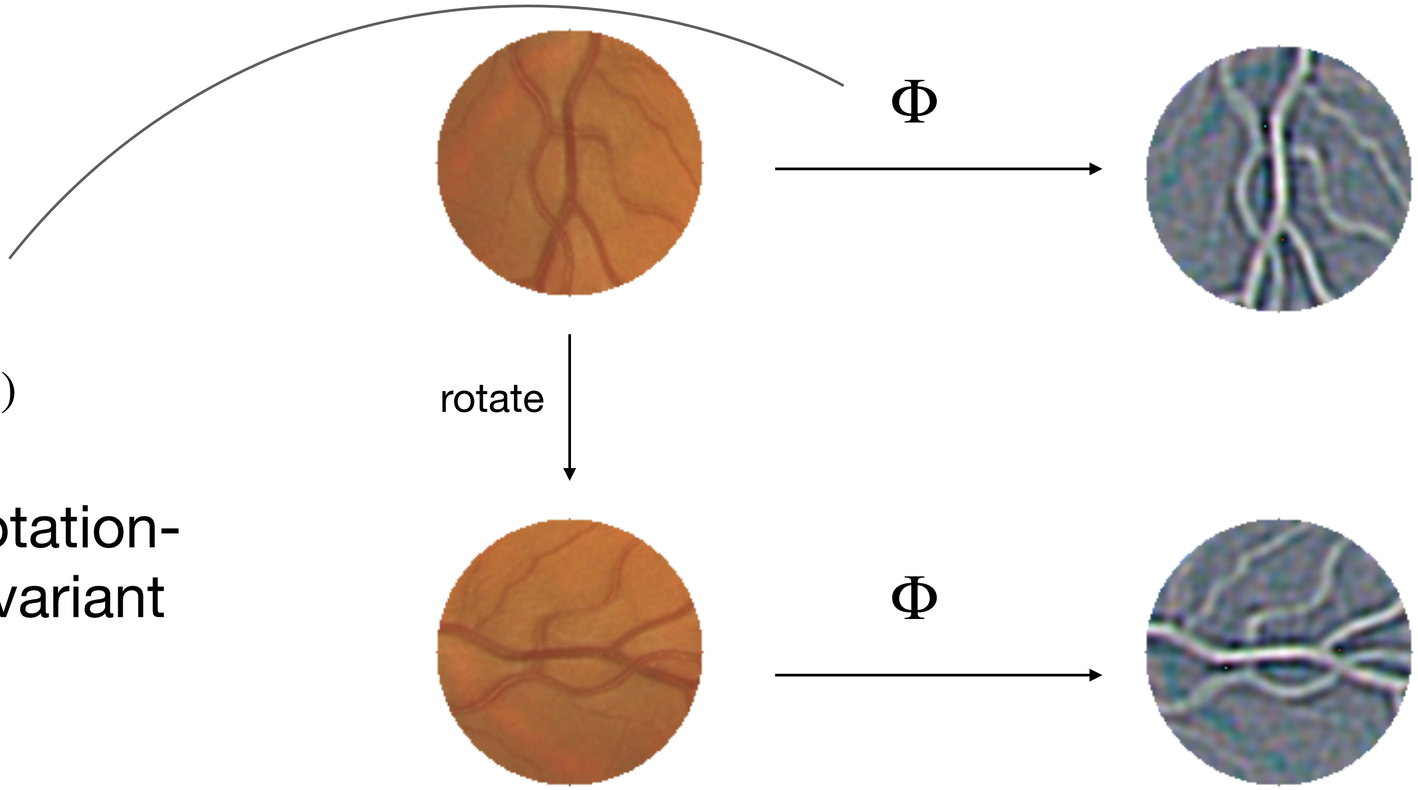
Invariance

Well at least that's one definition of invariance :-(depends on context. See next slide

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

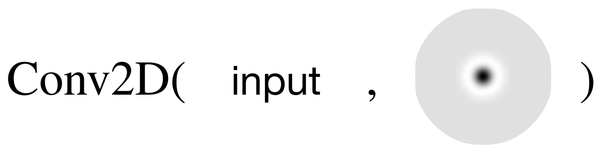
Slide credit: Erik Bekkers, University of Amsterdam

Invariance

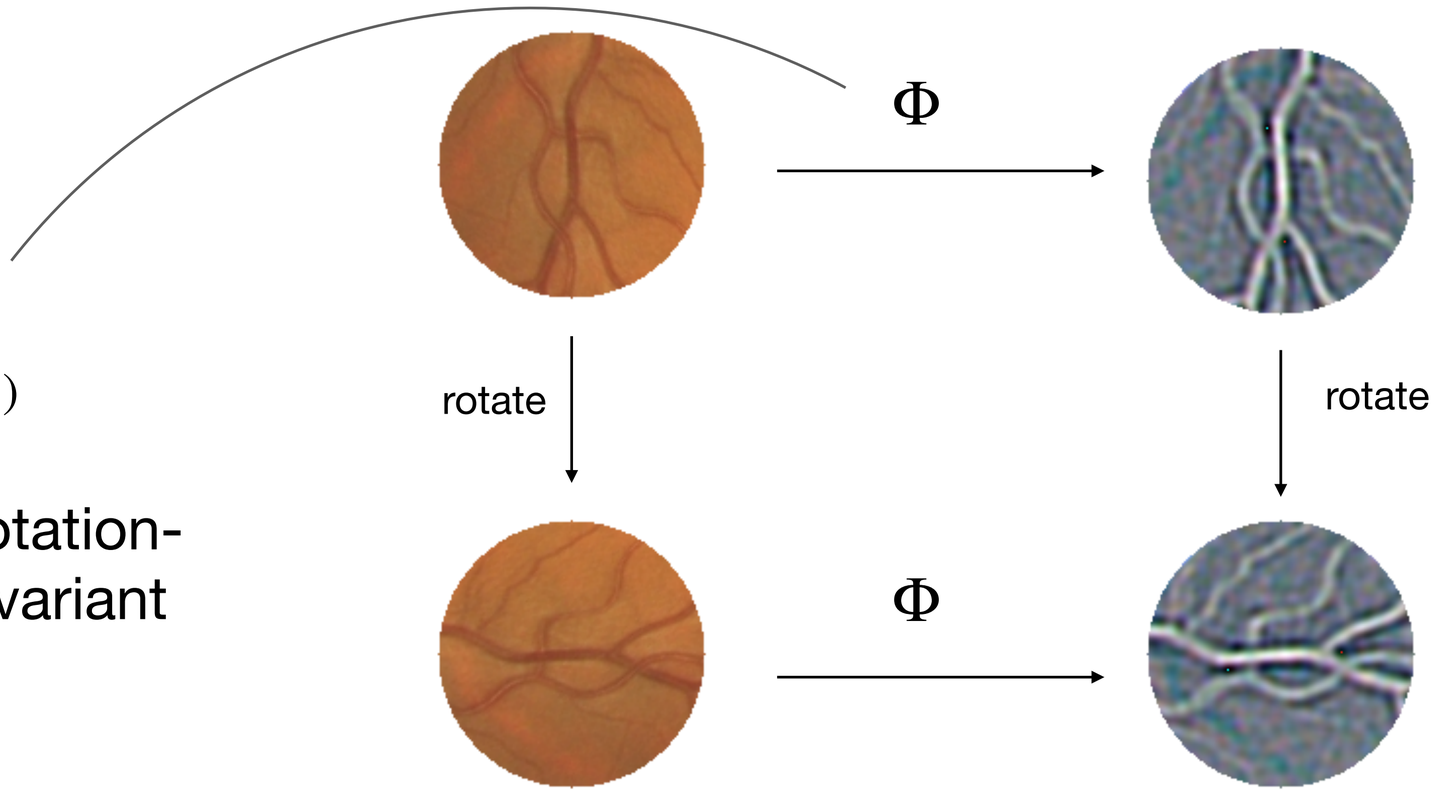
Well at least that's one definition of invariance :-(depends on context. See next slide

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

Vocabulary discussion

Invariant as constant

Invariant as commutative

Equivariant / steerable

Shift invariance in CNN?

Geometric guarantees (equivariance)

CNNs are translation equivariant

Mostly



Via convolutions



Geometric guarantees (equivariance)

CNNs are translation equivariant

Mostly



Via convolutions



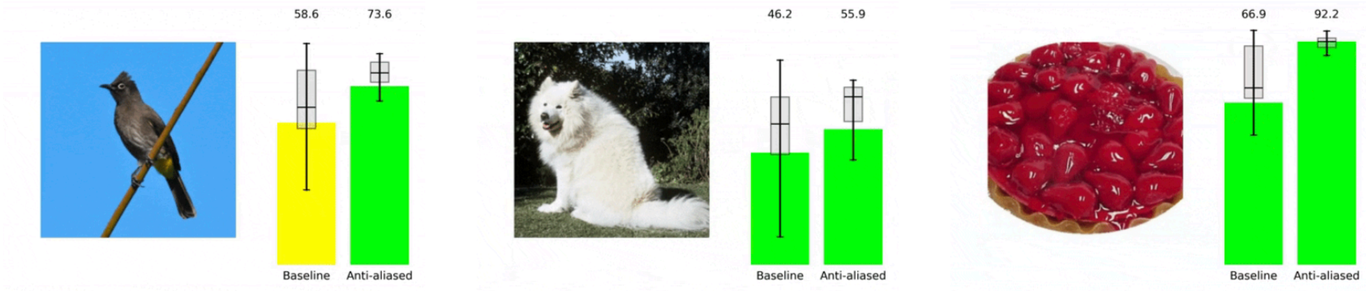
Actually not 100% shift in/equi-variant by construction

Making Convolutional Networks Shift-Invariant Again

Richard Zhang

Adobe Research

[Paper] [GitHub] [Talk] [Slides] [Poster]



We show that **classic antialiasing applied to modern deep networks** can stabilize outputs and improve accuracy. Try a pretrained antialiased network as a backbone for your application.

The C in CNNs (convolution) is shift in/equi-variant. But NOT the pooling

Their idea: add band limited (blur) before pooling.

Helps when data is limited

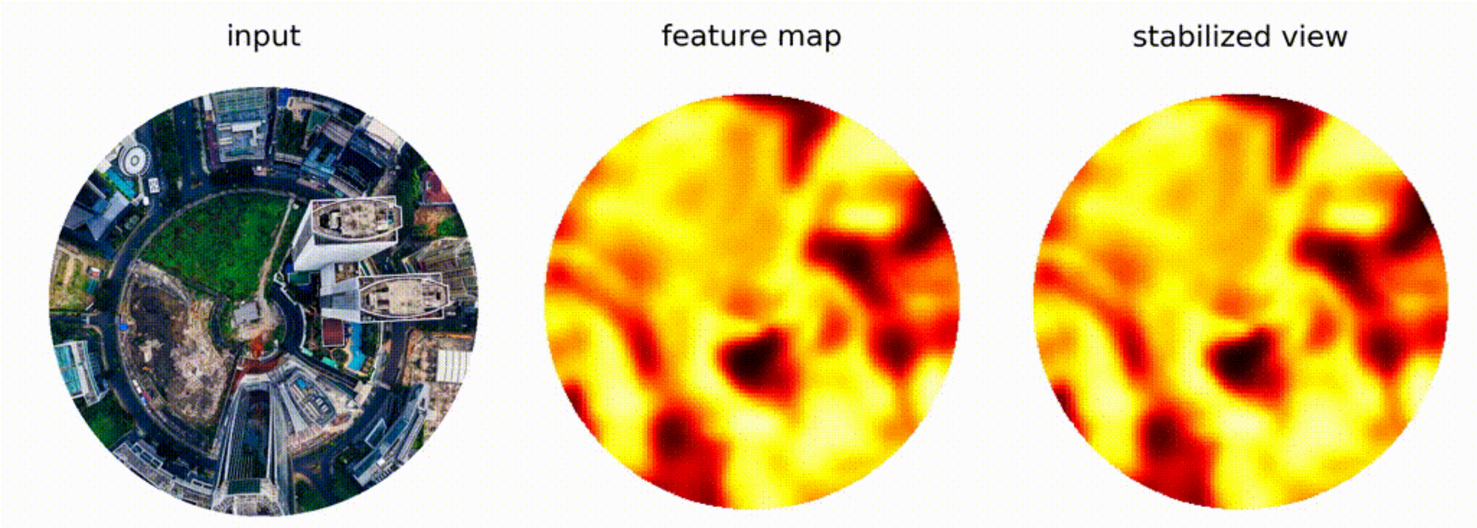
But with enough training data the trained network can be

https://richzhang.github.io/antialiased-cnns/

Rotation equivariance

Geometric guarantees (equivariance)

CNN

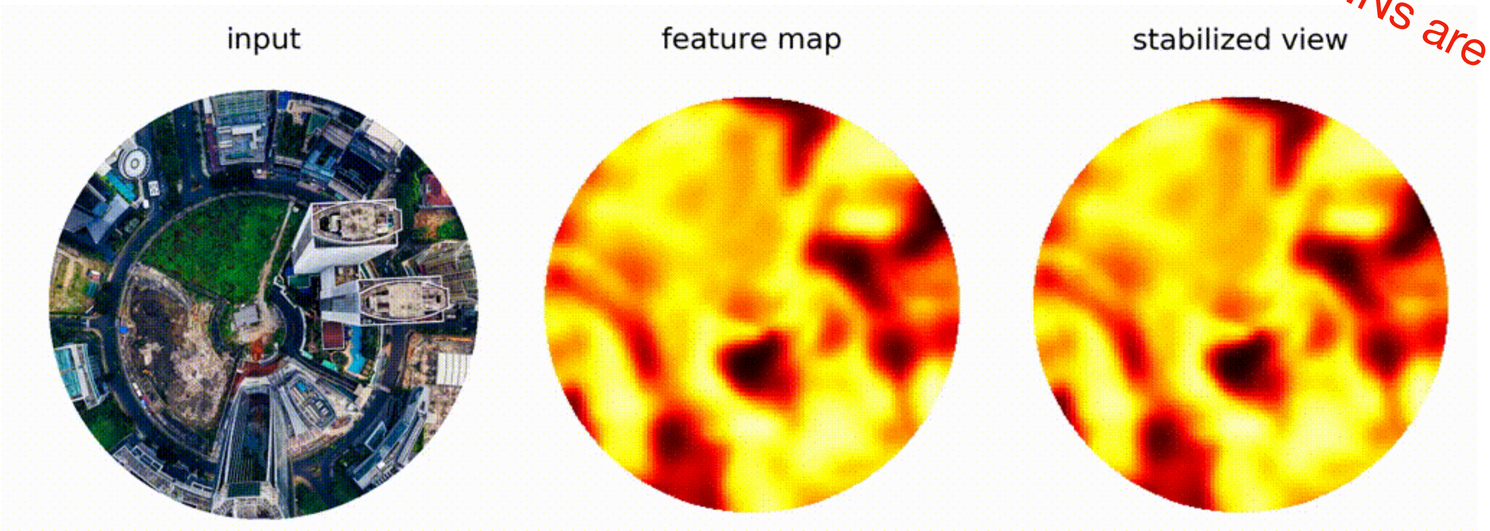


Figures source:
https://github.com/QUVA-Lab/e2cnn

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Geometric guarantees (equivariance)

CNN



CNNs are not rotation equivariant

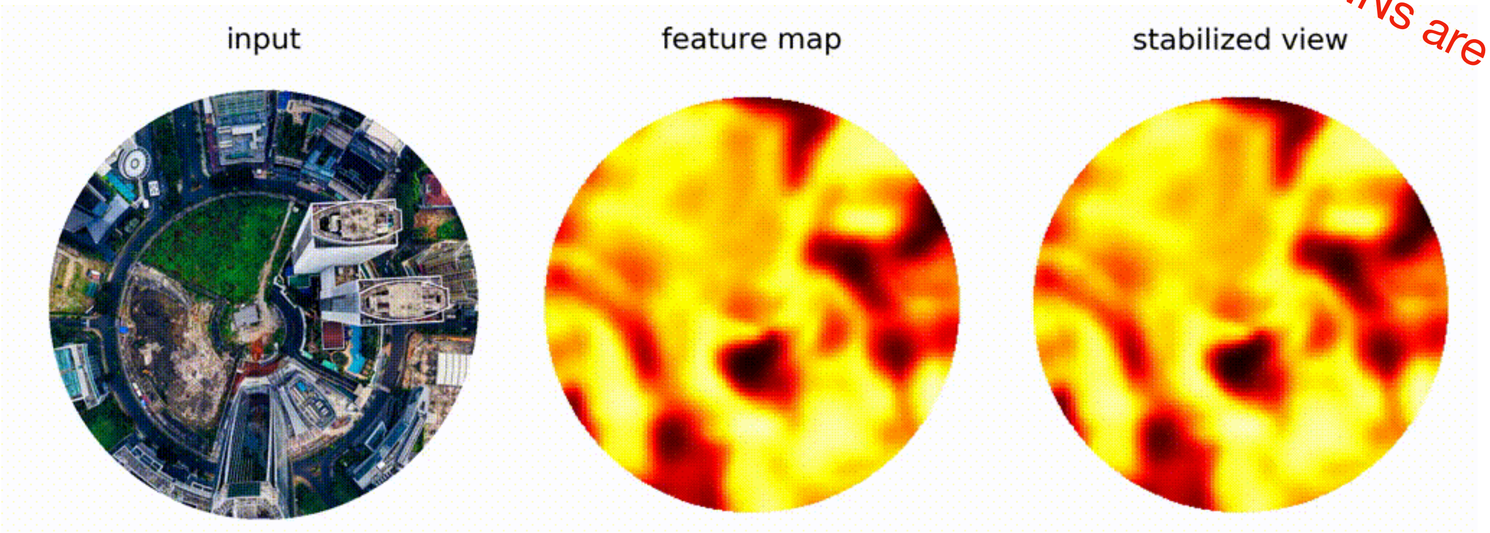


Figures source:
https://github.com/QUVA-Lab/e2cnn

Slide curtesy of Erik Bekkers from UVA Deep Learning II Course

Geometric guarantees (equivariance)

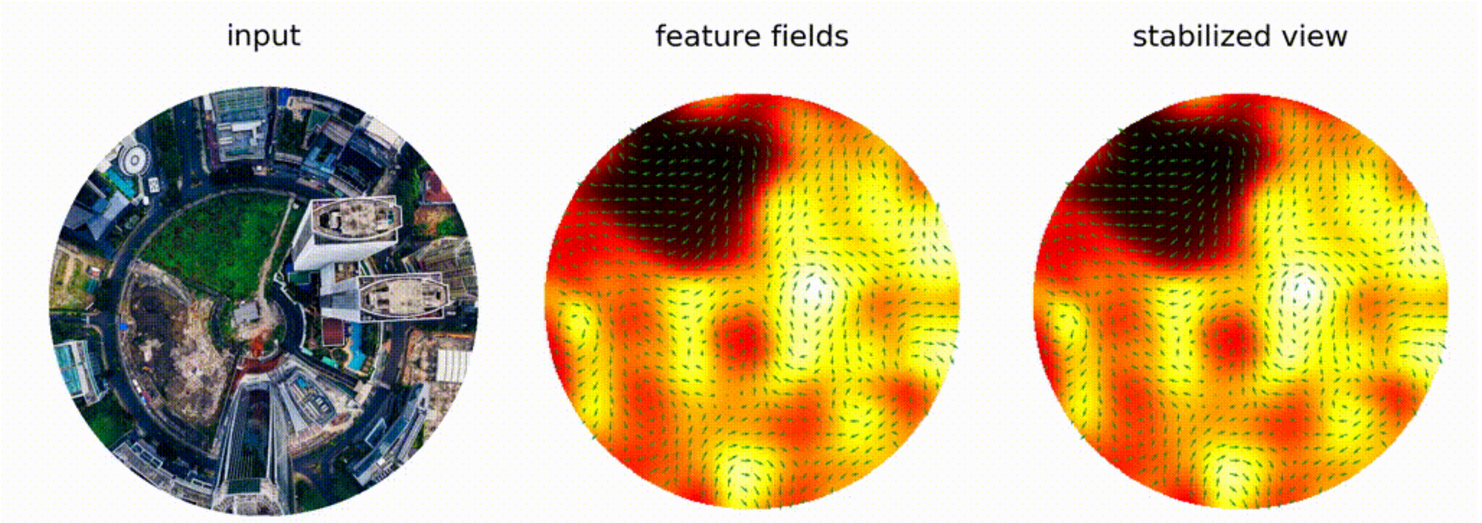
CNN



CNNs are not rotation equivariant



Equivariant NN

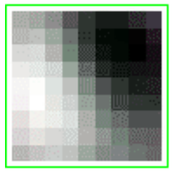




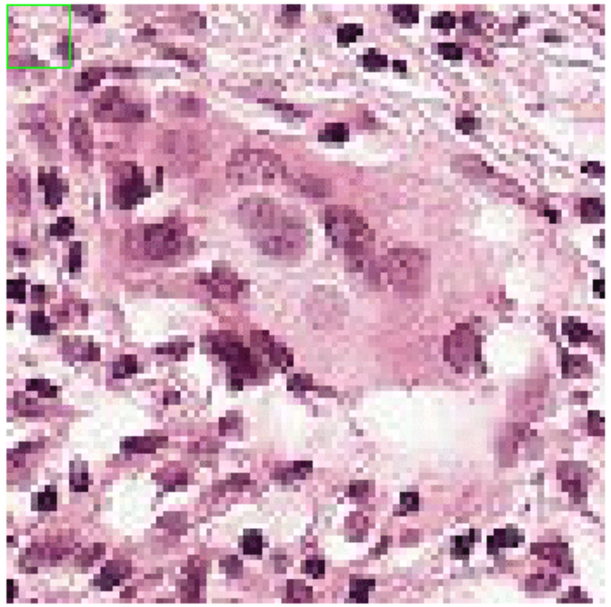
Figures source:
https://github.com/QUVA-Lab/e2cnn

Cross-correlations

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



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



$=$



k
2D convolution kernel

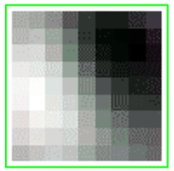
f^{in}
2D feature map

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

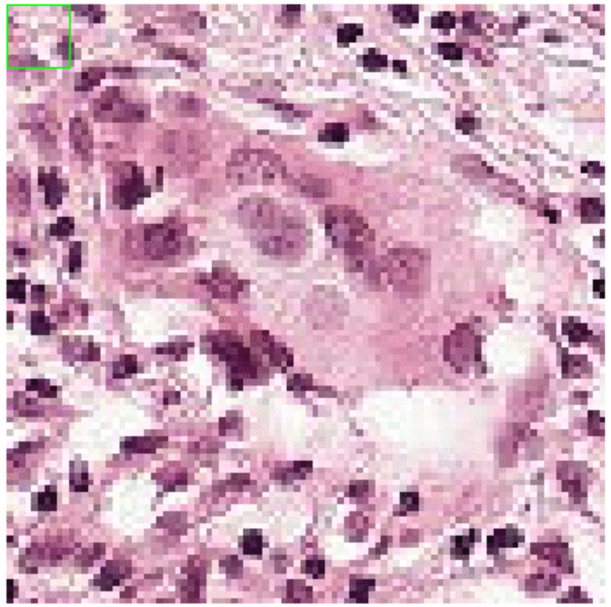
Slide credit: Erik Bekkers, University of Amsterdam

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

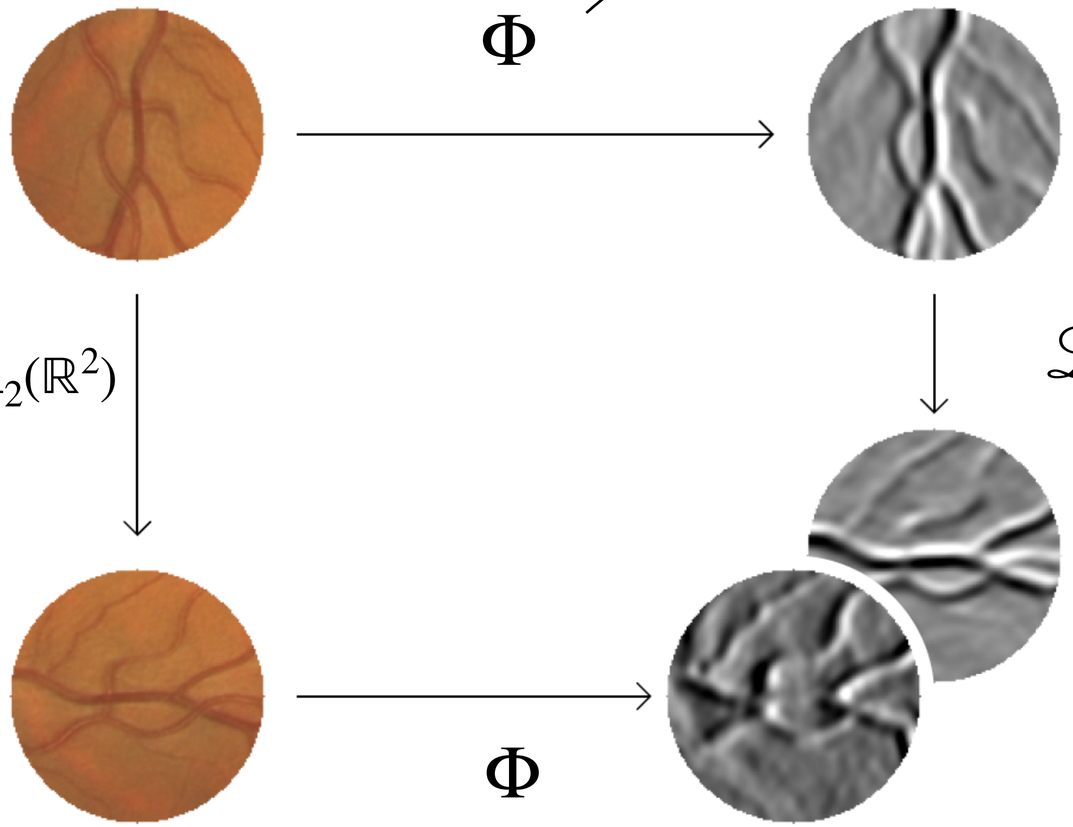
f^{out}
2D feature map (after ReLU)

Slide 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

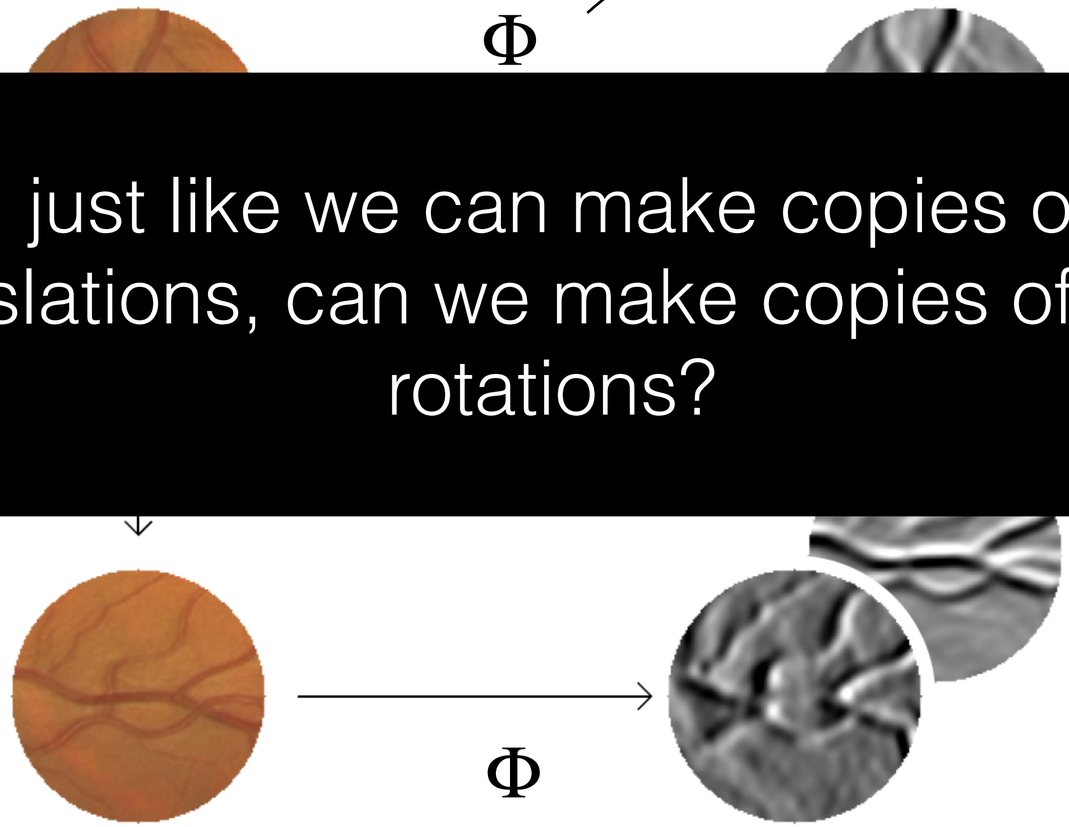
Slide credit: Erik Bekkers, University of Amsterdam

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

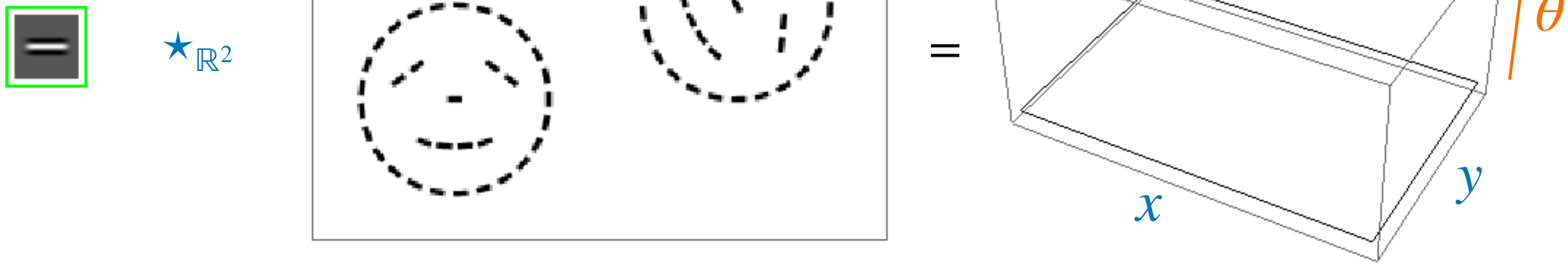
Slide credit: Erik Bekkers, University of Amsterdam

SE(2) equivariant cross-correlations

Representation of the roto-translation group!

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

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



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

f^{in}
2D feature map

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

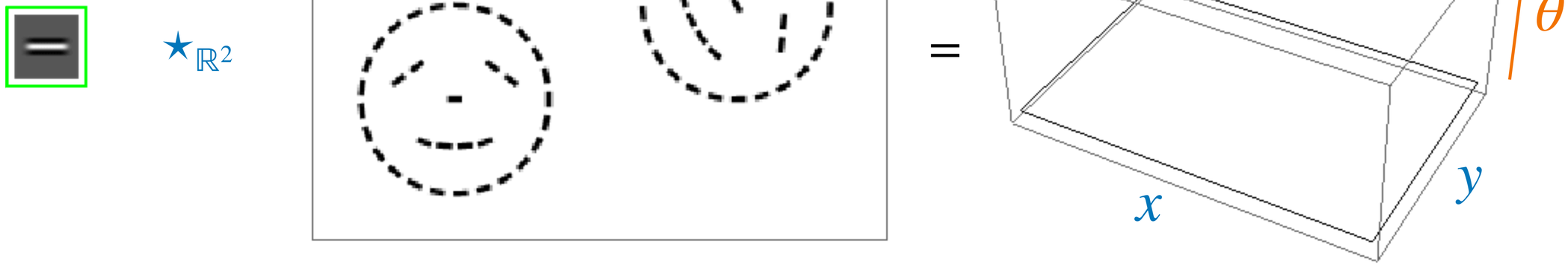
Slide credit: Erik Bekkers, University of Amsterdam

SE(2) equivariant cross-correlations

Representation of the roto-translation group!

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

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



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

Recap: Regular Group Convolutions

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

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

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

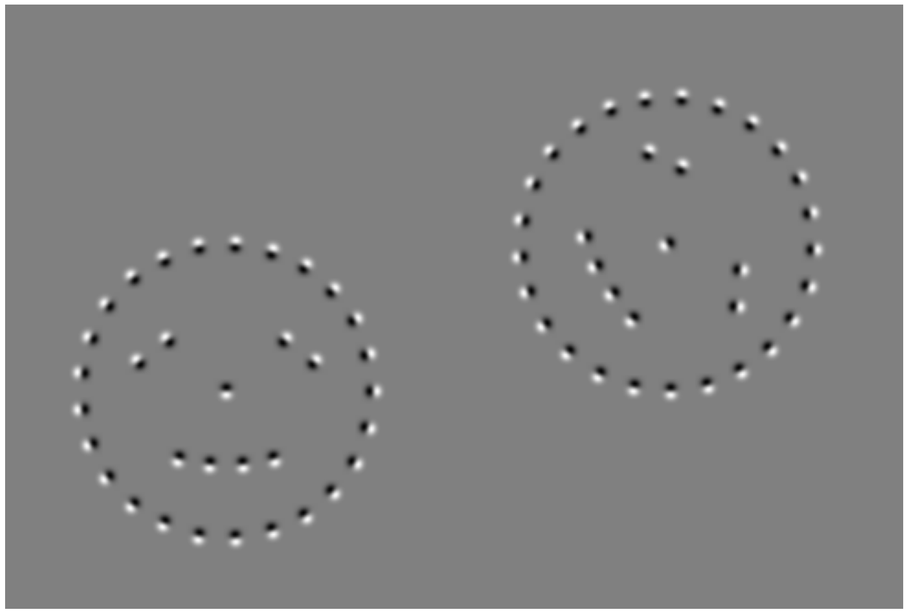
$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

---

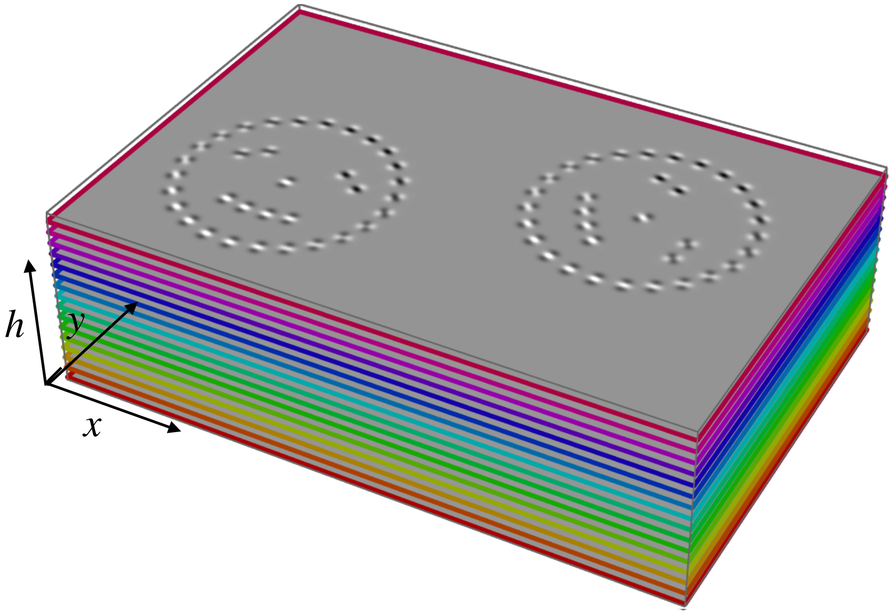
translation
"template matching"



$\tilde{\star}_{\mathbb{R}^2}$



$=$



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

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

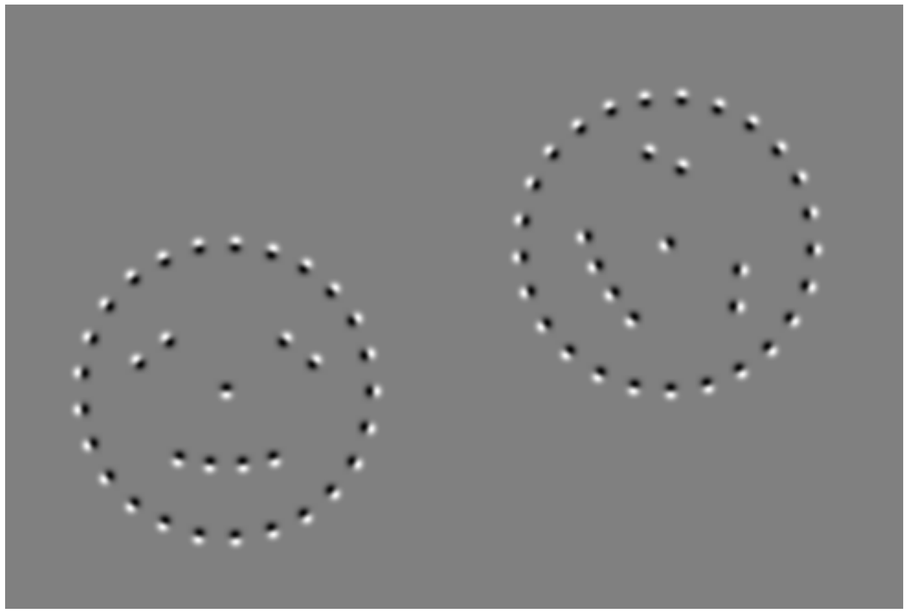
$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

---

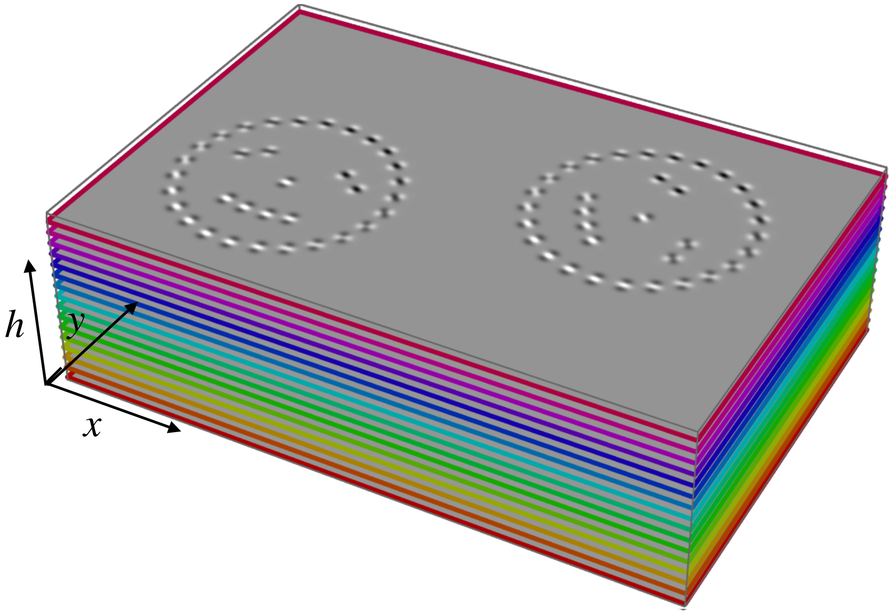
translation
"template matching"



$\tilde{\star}_{\mathbb{R}^2}$



$=$



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

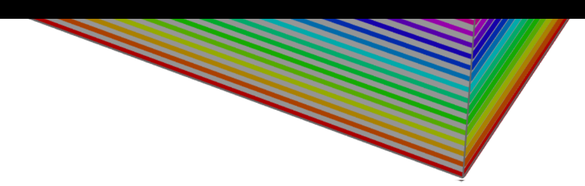
(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

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

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

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

Steerable basis to the rescue

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

Lifting convolution with steerable kernel

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

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

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

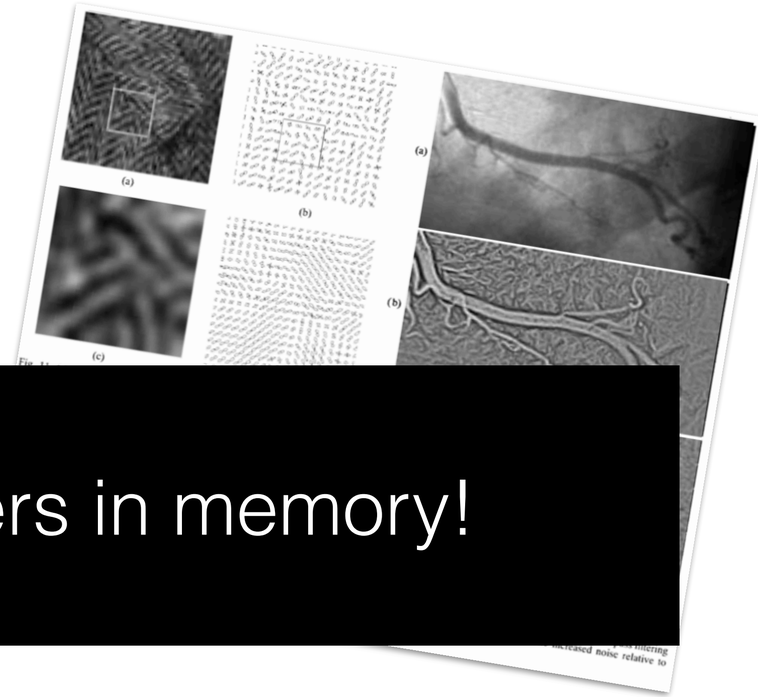
$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

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

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



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

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

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

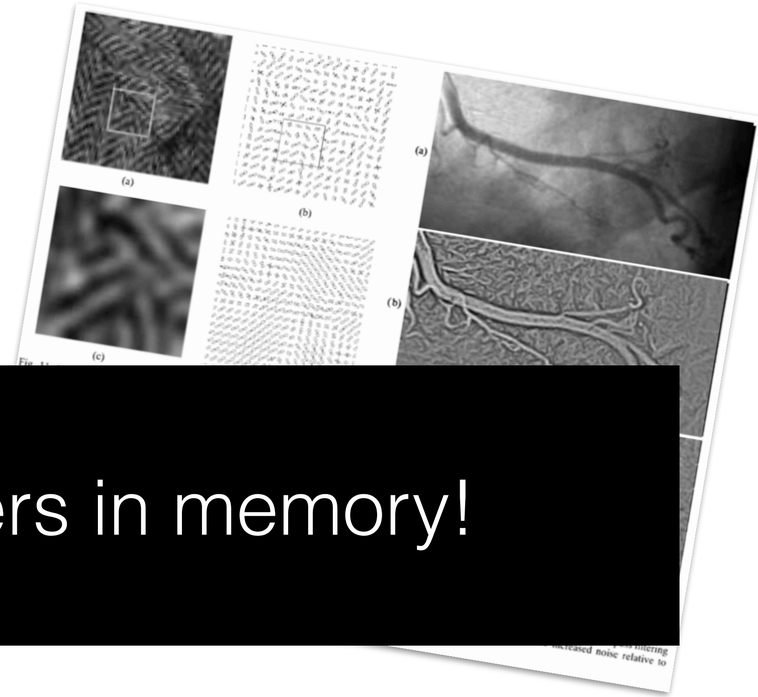
$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

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

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



! 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.



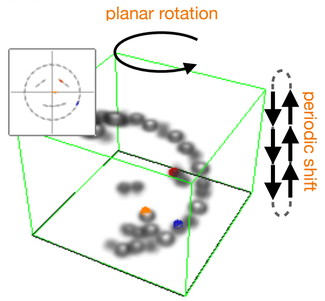
Only need to store responses of bases filters in memory!

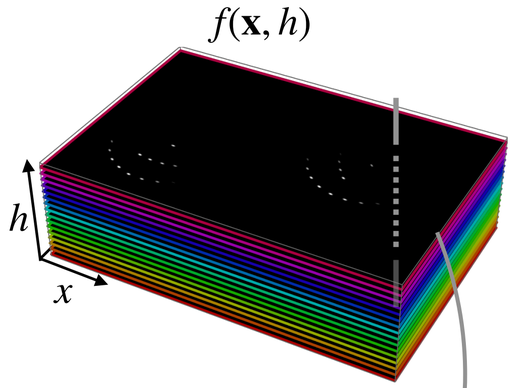
Slide credit: Erik Bekkers, University of Amsterdam

Feature field and induced representation

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

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

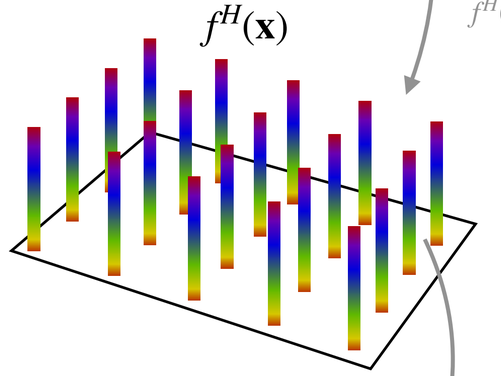




Regular H feature fields: Let $f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$ be the field of functions $f^H(\mathbf{x}) : H \to \mathbb{R}$ on the subgroup H, then the functions (**fibers**) transform via the regular representation $\mathscr{L}^H_h$

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

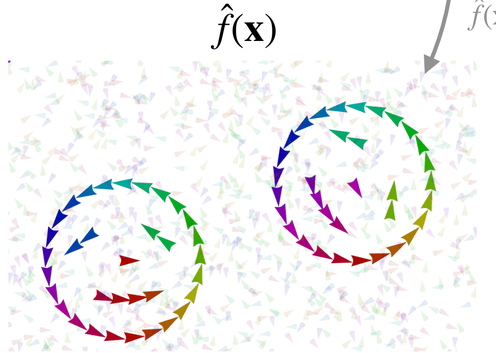
$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}^G_H[\ \mathscr{L}^H_h\ ](\mathbf{x}, h) f^H\right)(\mathbf{x}')$$



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

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 $\rho(h) = \oplus_l \rho_l(h)$

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



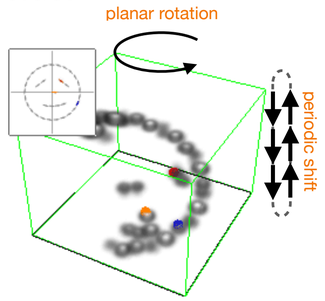
$\hat{f}(\mathbf{x}) = \mathscr{F}_H[f^H(\mathbf{x})]$

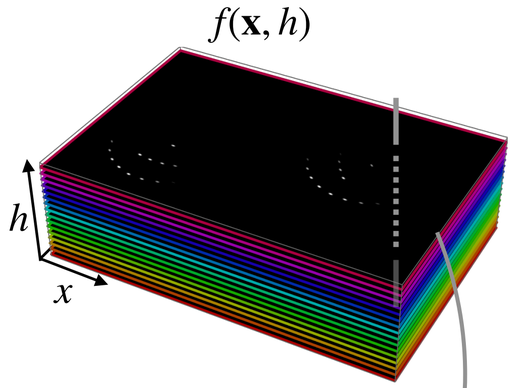
Slide credit: Erik Bekkers, University of Amsterdam

Feature field and induced representation

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

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

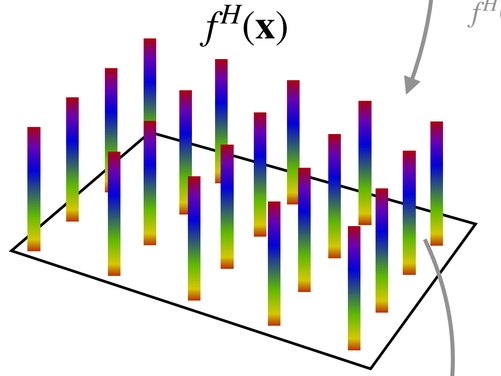




Regular H feature fields: Let $f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$ be the field of functions $f^H(\mathbf{x}) : H \to \mathbb{R}$ on the subgroup H, then the functions (**fibers**) transform via the regular representation $\mathscr{L}^H_h$

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

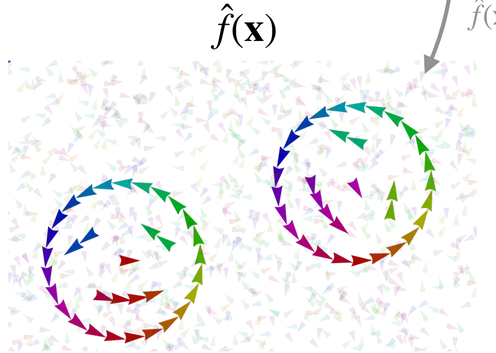
$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}^G_H[\ \mathscr{L}^H_h\ ](\mathbf{x}, h) f^H\right)(\mathbf{x}')$$



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

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 $\rho(h) = \oplus_l \rho_l(h)$

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

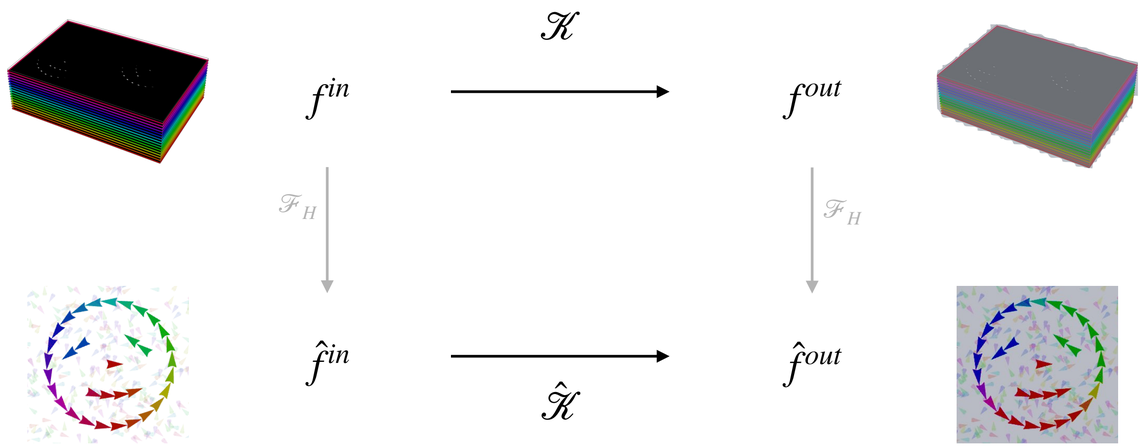


$\hat{f}(\mathbf{x}) = \mathscr{F}_H[f^H(\mathbf{x})]$

Slide credit: Erik Bekkers, University of Amsterdam

Steerable group convolutions

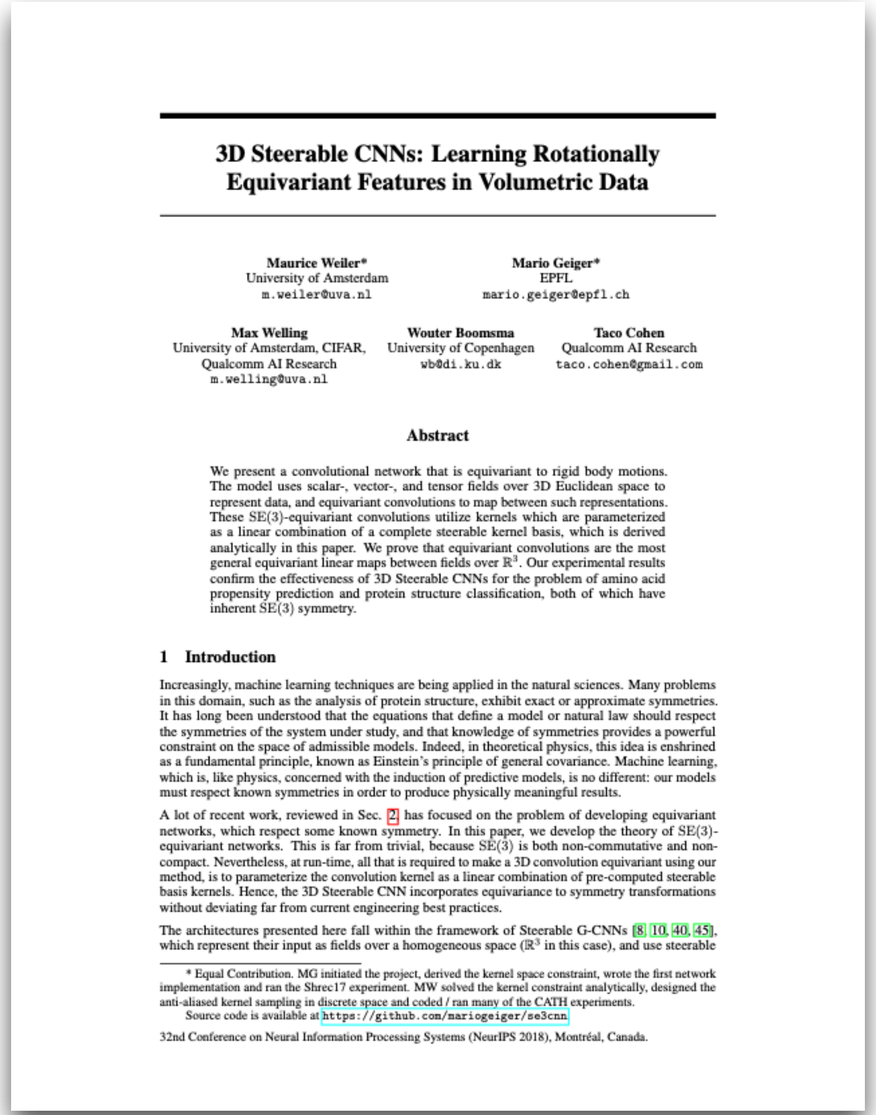
Group convolution $\mathscr{K}[f](g) = \int_G k(g^{-1}g')f(g)\mathrm{d}\mathbf{x}'\mathrm{d}g$



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

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.

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.

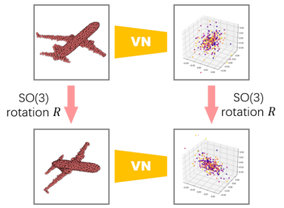


Figure 1: By lifting latent representations from vectors of scalar entries to vectors of 3D points (i.e., matrices) we facilitate the creation of a simple rotation equivariant toolbox allowing the implementation of fully equivariant pointcloud networks.

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-

arXiv:2204.02394v2 [cs.CV] 10 Feb 2023

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

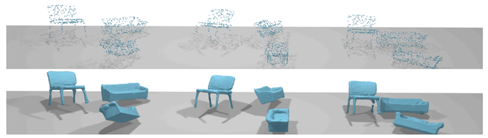


Figure 1: (Above): A scene-level point cloud produced by individual SE(3)-transformations of three sparse object point clouds.(Below): Our equivariant reconstruction. *The whole scene* is given as input to our network. The network, trained only on *single objects in canonical poses* and agnostic to the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

arXiv:2112.05124v1 [cs.RO] 9 Dec 2021

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.

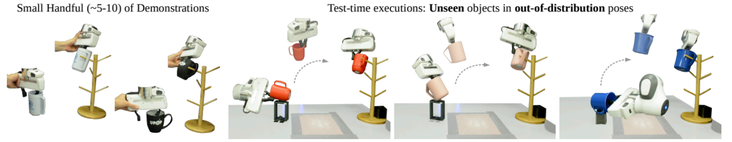


Fig. 1: Given a few (∼5-10) demonstrations of a manipulation task (left), Neural Descriptor Fields (NDFs) generalize the task to novel object instances in any 6-DoF configuration, *including those unobserved at training time*, such as mugs with arbitrary 3D translation and rotation (right). NDFs are 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.

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

arXiv:2410.23179v1 [cs.LG] 30 Oct 2024

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

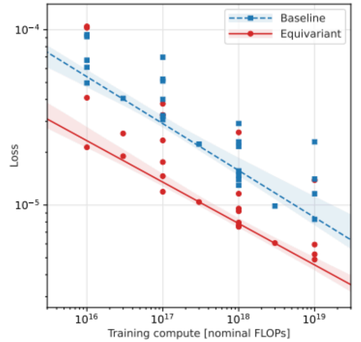


Figure 1: **Scaling with compute**. The dots show the training compute budget and test loss in our experiments, the lines indicate the compute-optimal performance according to the scaling laws we find. The test losses of both non-equivariant (– –) and equivariant (——) transformers scale as a power law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

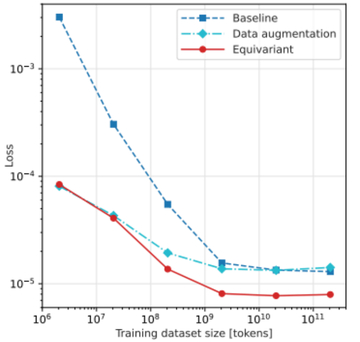


Figure 2: **Scaling with training data**. We show the performance of the non-equivariant transformer (– –), non-equivariant transformer trained with data augmentation (—·—), and equivariant transformer (——) as a function of the number of unique tokens in the training dataset. All experiments use the 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.

THE LIE DERIVATIVE FOR MEASURING LEARNED EQUIVARIANCE

Nate Gruver*, **Marc Finzi***, **Micah Goldblum**, **Andrew Gordon Wilson**
New York University

Equivariance out of distribution From our analysis above, large models appear to learn equivariances that rival architecture engineering in the classification setting. When learning equivariances through data augmentation, however, there is no guarantee that the equivariance will generalize to data that is far from the training distribution. Indeed, Engstrom et al. (2019) shows that carefully chosen translations or rotations can be as devastating to model performance as adversarial examples. We find that vision models do indeed have an *equivariance gap*: models are less equivariant on test data than train, and this gap grows for OOD inputs as shown in Figure 7. Notably, however, architectural biases do not have a strong effect on the equivariance gap, as both CNN and ViT models have comparable gaps for OOD inputs.

Also the world is full of biases

- The world is not that equivariant to rotations and even translation
- For example:
  - Gravity imposes directional structure
  - The Ground imposes translational structure

At the end of the day

- Shift invariance as captured by the C in CNNs is powerful to make deep learning scalable
  - But strict shift invariance (in particular of pooling) may not matter
- Sherical harmonics are powerful directional representations especially for low-complexity (they reconstruct nice and smooth)
- Extension of Fourier to meshes/geometry has been powerful
- But more fancy versions of equivariance and representation theory haven’t panned out as effectively in my experience
  - Except maybe special case of physics and data-scarce domains
  - Although data augmentation is powerful there.

The Bitter Lesson: Brute-Force tends to Win

General methods leveraging massive computation (like search and learning) consistently outperform human-crafted, task-specific systems, even though the latter feel clever initially. Examples: Chess, Go, Language Modeling.

Geometric deep learning in vision is an example of this. Trying to manually build models that capture the incredible complexity of the real world is possibly futile; we may need to *learn* these rules from data.

How? We don’t know.

But Effective Brute Force takes method and intelligence

- There are many versions of brute force
  and finding the one that works is not so easy
- Developing high performance solutions that allow us to scale up the force we deploy is not easy

- Some insights from clever solutions help refine brute force
  - e.g. practical geometry processing & physically based rendering have been deeply impacted by mathematical analysis
  - Casting CNNs and Deep learning in general in terms of linear algebra operators has greatly facilitated the development of high-performance implementations