THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Equivariant deep learning with applications in computer vision

Georg Bökman

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2024

Equivariant deep learning with applications in computer vision

Georg Bökman
ISBN 978-91-8103-102-7

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5560
ISSN 0346-718X

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000
www.chalmers.se

Cover image: A windmill in Fiskebäckskil photographed by the author. *The blades of the windmill have rotational symmetry. Should a neural network have the same internal representation for the position highlighted in blue and the corresponding position in orange?*

Back page image: The author photographed through the kaleidoscope at the Museum of Illusions in New York.

Printed by Chalmers digitaltryck
Gothenburg, Sweden, September 2024

Equivariant deep learning with applications in computer vision

Georg Bökman
Department of Electrical Engineering
Chalmers University of Technology

Abstract

We study equivariant deep learning for image data. A neural network is said to be equivariant to a transformation of its input if there is a corresponding transformation of the network output. In the first part of this thesis we cover key definitions and background theory in the hope of providing a useful reference for newcomers to the field.

Papers A and B study equivariance in the context of keypoint description. We introduce the concept of steerers, which are transformations of keypoint descriptions that correspond to specific transformations of the input images. We argue why rotation steerers appear naturally when training keypoint descriptor neural networks. Further, we propose affine steerers for arbitrary differentiable image transformations.

Paper C studies to what extent each layer of a convolutional neural network becomes close to reflection equivariant when trained on natural images. We find that the closer a convolutional neural network is to being equivariant, the closer it is to satisfying an equivariance constraint in each layer.

Convolutional neural networks are equivariant to translations of the input. In paper D, we show that image translations do not correspond to rigid motions of the camera taking the images. Instead, we propose methods to make neural networks more equivariant to rotational homographies, the only image transformations corresponding to rigid camera motions.

In paper E, we demonstrate that a state-of-the-art (semi-)dense image-matching neural network can be made close to rotation invariant without a drop in performance on upright images. This is done by replacing the backbone feature extractor neural network with one layerwise constrained to be equivariant.

Keywords: Computer vision, deep learning, equivariance, image matching.

List of papers

This thesis is based on the following publications:

[A] **Georg Bökman**, Johan Edstedt, Michael Felsberg, Fredrik Kahl, "Steerers: A framework for rotation equivariant keypoint descriptors". CVPR 2024.

[B] **Georg Bökman**, Johan Edstedt, Michael Felsberg, Fredrik Kahl, "Affine steerers for structured keypoint description". ECCV 2024.

[C] **Georg Bökman**, Fredrik Kahl, "Investigating how ReLU-networks encode symmetries". NeurIPS 2023.

[D] Lucas Brynte*, **Georg Bökman***, Axel Flinth, Fredrik Kahl , "Rigidity preserving image transformations and equivariance in perspective". SCIA 2023.

[E] **Georg Bökman**, Fredrik Kahl, "A case for using rotation invariant features in state of the art feature matchers". CVPRW 2022.

Other publications by the author, not included in this thesis, are:

[F] J. Edstedt, **G. Bökman**, Z. Zhao, "DeDoDe v2: Analyzing and Improving the DeDoDe Keypoint Detector". CVPRW 2024.

[G] J. Edstedt, Q. Sun, **G. Bökman**, M. Wadenbäck, M. Felsberg, "RoMa: Revisiting Robust Losses for Dense Feature Matching". CVPR 2024.

[H] J. Edstedt, **G. Bökman**, M. Wadenbäck, M. Felsberg, "DeDoDe: Detect, Don't Describe—Describe, Don't Detect for Local Feature Matching". 3DV 2024.

[I] **G. Bökman***, A. Flinth*, F. Kahl, "In search of projectively equivariant networks". TMLR 2024.

[J] **G. Bökman**, F. Kahl, A. Flinth, "ZZ-Net: A universal rotation equivariant architecture for 2D point clouds". CVPR 2022.

[K] C. Toft, **G. Bökman**, F. Kahl, "Azimuthal rotational equivariance in spherical convolutional neural networks". ICPR 2022.

*Equal contribution

Dedicated to my parents

Acknowledgments

One of my favourite things to do is to learn new things. I have learned a lot in the past five years, and this is in large part thanks to my supervisor, Fredrik Kahl. I thank Fredrik for his invaluable advice and support throughout.

I thank all my collaborators for the many inspiring and fruitful research discussions and their hard work. In addition to Fredrik, I will mention here my closest collaborators at Chalmers, including Axel Flinth, Lucas Brynte, and Carl Toft, and my collaborators at the Computer Vision Laboratory in Linköping, including Johan Edstedt, Michael Felsberg, and Mårten Wadenbäck. The close collaboration with Linköping started after Johan spotted my `reproduce_loftr` job script (part of Paper E) in the job queue of the national compute cluster Berzelius.

I was fortunate to be part of the WASP graduate school and the founding group of the Geometric Deep Learning cluster, which I led for its first years of existence. I thank all the members for their enthusiasm in jointly decrypting difficult papers and the successful study trips to the Netherlands and Switzerland. Particular thanks go to the most active members, including Pavlo Melnyk, Karl Bengtsson Bernander, Jimmy Aronsson, Oscar Carlsson, Yaroslava Lochman, Lucas Brynte, Fabian Sinzinger, Axel Berg, Yifei Jin, Johan Edstedt and Yifan Ding. I wish the new guard, including Elias Nyholm and Philipp Misof, all the best for the future.

I thank the past and present members of the computer vision group at Chalmers for making the past five years fun, through research and non-research discussions, social activities, support and advice. Thanks to Lucas, Yara, Rasmus, Kunal, Xixi, José, Carl, Jennifer, Dorian, Alex, Josef, Roman, Victor, Sofie, David, Christopher, Carl, Ida, Erik, Ji, Richard, Axel, Måns, Huu, Eskil, James, Torsten and Fredrik.

Finally, I thank Elina and my family and friends for their support and for giving me other things to think about than work.

This work was supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation.

Contents

Part I

Overview

CHAPTER 1

Introduction

Computer vision can be summarised briefly as the study of visual information using computers. Much of computer vision concerns algorithms with visual data as input and more structured data as output. For instance, computer vision covers algorithms that output semantic information, such as the type of bird seen in an image. Other examples are algorithms that reconstruct 3D information from images, *e.g.* producing 3D models of cities from images only.

Over the past decade, computer vision has gone from primarily algorithms hand-crafted by researchers and engineers to algorithms based on *machine learning* being the best for many tasks. The idea in machine learning is to start with a flexible algorithm with free parameters that can be changed to modify the algorithm's behaviour. This flexible algorithm is called a *model*. The model's parameters are tuned based on *training data*, *i.e.* large collections of pairs of input and output for the task at hand, so that the model with tuned parameters is as consistent with the training data as possible. The tuning of the model parameters on training data is also called *training* the model. After training, the model will approximate a function for which we know the correct output on our training data, but the aim is that the model will generalise to new data as well. The specific branch of machine learning currently domi-

nating computer vision research is *deep learning*, machine learning with deep neural networks. A neural networks is a specific form of computer algorithm that is structured in layers and contains many tuneable parameters.

In this thesis, we study both how geometric symmetries are learnt by neural networks when they are trained with image data and how we can use prior knowledge of what symmetries are present in the data when we construct neural networks. *Equivariant deep learning* is the sub-field of deep learning that considers approximating functions that respect some symmetry. Specifically, an algorithm is equivariant if applying the symmetry to the input before using the algorithm gives the same result as applying the symmetry to the output of the algorithm. For instance, an algorithm that detects forests in aerial imagery should be equivariant to rotations.

The computer vision task covered most in this thesis is *image matching*, which entails finding corresponding points in two different images of the same scene or object. This is a precursor task to many computer vision applications, such as 3D reconstruction. We will see that image matching is a good example of a task that is equivariant to geometric transformations of the input images.

The first part of the thesis covers background theory, while the second part contains the five research papers that form the basis of the thesis. Basic knowledge of deep learning will be assumed of the reader.

CHAPTER 2

Image representations and group representations

The word *representation* is overloaded with several meanings in the computer vision and machine learning literature. Two of these meanings are central to this work and will be covered in this chapter.

First, we have *image representations*, meaning the encodings of images obtained by feeding them through an algorithm, typically a neural network. Neural networks are composed of layers, and the output of each layer is said to be a representation of the image.

Second, we have the mathematical construct of *group representations*. A group representation is the concretisation of an abstract mathematical group (such as the group of rotations) as a group of linear maps. The branch of mathematics studying group representations is called *representation theory*.

Notably, group representations can act on image representations. This will be the topic of the next chapter.

2.1 Image representations

The most general definition of *image representations* is that they are the output of algorithms taking images as input. For instance, object bounding

boxes, segmentation masks and keypoint locations are all examples of image representations. More abstract image representations are, *e.g.* the output of compression algorithms or the output of intermediate layers of a neural network. Intermediate outputs of neural networks are also often called feature maps. We show a feature map from a neural network used in this thesis in Figure 2.1.

Digital images are arrays of pixels, where we associate each pixel with the red, green and blue colour intensities of its location in the array. Hence, an image can be considered a $3 \times H \times W$ array, where H, W are the image height and width. Image representations are often themselves arrays of shape $c \times h \times w$, where the first dimension is the *channel* or *feature* dimension, here with c channels, and h, w are the height and width. h, w are not necessarily the same as H, W since the algorithm might downsample or rescale the image's spatial dimensions. However, the idea is that if we rescale the image representation to size H, W, we can associate each pixel in the image with a c-dimensional *feature vector*. For example, a segmentation mask associates a vector to each pixel, where each dimension encodes the probability that the pixel corresponds to a specific segmentation class. Similarly, the output of a keypoint detector can be considered a $1 \times H \times W$ array, where each pixel is given a probability of being a keypoint. $h = w = 1$ is a special case where the image representation is a single c-dimensional vector associated with the entire image.

It is generally impossible to give a concrete interpretation of the c dimensions encoded in the intermediate outputs of neural networks. Instead, we consider the image representation an abstract quantity useful for producing the network's final output. In this context, the image representation is often called a *latent representation*.

Rather than thinking of an image as a finite array of pixels, it is often beneficial to consider it a function on an infinite pixel grid $I : \mathbb{Z}^2 \to \mathbb{R}^3$, or even take a continuous viewpoint and think of an image as a function $I : \mathbb{R}^2 \to \mathbb{R}^3$, assigning an RGB tuple to every point in the plane. The digital array of pixels is then a discrete approximation of this function. Similarly, an image representation can be thought of as a function $F : \mathbb{R}^2 \to \mathbb{R}^c$, assigning a c-dimensional feature vector to each point in the plane. This framing is extra useful when the resolution of the image input to an algorithm is different to the resolution of the output image representation.

Figure 2.1: Example of an input image and 3 dimensions of an intermediate feature map of a neural network from Paper A.

2.2 Representation learning

Representation learning is a subfield of machine learning where the aim is to use training data to tune a neural network[1] to output representations that are more useful for downstream tasks than the raw data is. What is a useful representation of course depends on the task, however, a guiding principle for representation learning is often *invariance*. That is, we would like to have image representations that are invariant to such perturbations of the input that are not relevant to the task at hand. For instance, in many image processing tasks, the illumination of the image should not influence the output of our algorithm. If we have image representations that are invariant to illumination changes and feed them as input to an algorithm, we guarantee that the algorithm output is not influenced by illumination.

One important aspect of representation learning is that it does not require labelled data. For instance, in contrastive learning, one specifies a set of transformations that should not influence the output representation and trains the neural network to output the same representation for images related by one of the specified transformations, but different representations for images that are not related by one of the specified transformations.

In the next chapter, we will see that invariance can often be formalized in terms of mathematical group theory, for which we need the formalism of group representations introduced in the next section. Further, we will contrast invariance with equivariance. For instance, a neural network being equivariant to illumination changes would mean that the output of the network changes when the input changes illumination, but the change of the output is given by a specific function. Knowing the function in output space that corresponds to certain transformations in input space can make the learned representations even more useful for downstream tasks than invariant representations. It allows us to modify the network's output as if the input were modified, but without actually rerunning the network. We will see this in Papers A and B.

[1] Other models than neural networks can also be used, but will not be considered in this thesis.

2.3 Group representations

This section assumes basic knowledge of groups and vector spaces. We will only consider vector spaces over the real numbers $\mathbb{R}$.

Definition 1: Given a group G, a (linear) *representation* of G consists of a vector space V and a group homomorphism $\rho : G \to \mathrm{GL}(V)$.

Here $\mathrm{GL}(V)$ is the general linear group of V, *i.e.*, the group of all invertible linear maps from V to itself. Often, we will refer to ρ as the representation, although strictly the representation is the pair (V, ρ). Occasionally, we will also refer to V as the representation. When we have a representation ρ of G, we will say that G acts on V by ρ.

If V has finite dimension D, then $\mathrm{GL}(V)$ can be viewed as the group of invertible $D \times D$ matrices. Hence, a group representation ρ is a map that encodes group elements as matrices and the group operation as matrix multiplication.

The following examples are illustrated in Figure 2.2.

Example 1: Let $G = \mathrm{SO}(2)$ be the group of planar rotations. $\mathrm{SO}(2)$ can be defined as the interval $[0, 2\pi)$ with the group operation being addition modulo 2π. The most common representation of $\mathrm{SO}(2)$ is on the vector space $\mathbb{R}^2$ by the ordinary rotation matrices

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{2.1}$$

For illustration, we sketch how to check that R is a representation. First, $R(\theta)$ is linear by definition and invertible since it has the easily checked inverse $R(\theta)^\mathsf{T}$. Finally, R is a group homomorphism since $R(\theta + \omega) = R(\theta)R(\omega)$ by the addition formulae of sine and cosine.

Example 2: The following gives another representation of $\mathrm{SO}(2)$. Consider the vector space $\mathbb{R}^{2\times 2}$ of 2×2 matrices and let for $M \in \mathbb{R}^{2\times 2}, \theta \in \mathrm{SO}(2)$

$$R_2(\theta)[M] = R(\theta) M R(\theta)^\mathsf{T}, \tag{2.2}$$

with R as in (2.1). R_2 is, in fact, the tensor product of R by itself and can be interpreted as rotating linear maps M by θ. By "flattening" $\mathbb{R}^{2\times 2}$ we obtain the isomorphic vector space $\mathbb{R}^4$, and $R_2(\theta)$ is equivalent to the Kronecker

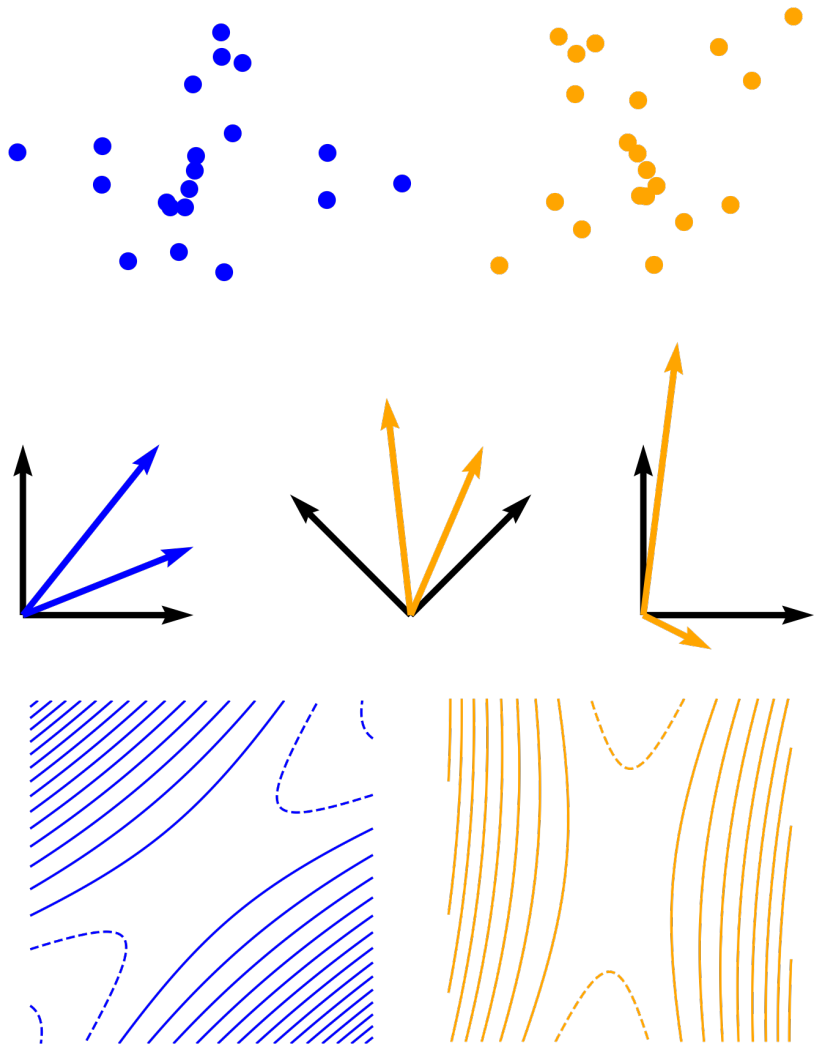

Figure 2.2: The planar rotation group acts differently on different objects. **Top:** The standard representation $R(\pi/4)$ in Example 1 rotates points in the plane, *e.g.* the blue points to the orange points. **Center:** The representation in Example 2 rotates linear maps. Left: A linear map $M : \mathbb{R}^2 \to \mathbb{R}^2$ is defined by where it maps the standard basis vectors, *i.e.* the mapping from the black to the blue vectors. Center: The rotated version of the mapping does onto the rotated standard basis as the original mapping does onto to the standard basis. Right: This means that on the standard basis, the rotated version of the mapping is $R(\pi/4)MR(\pi/4)^\mathsf{T}$. **Bottom:** Contour lines of the polynomial $p(x) = 5x_1^2 - 16x_1x_2 + 8x_2^2$ in blue, and of $p(R(\pi/4)^\mathsf{T}x)$ in orange, corresponding to the representation in Example 3.

product $R(\theta) \otimes R(\theta)$ acting on $\mathbb{R}^4$. By slight abuse of notation we can write

$$R_2(\theta) = R(\theta) \otimes R(\theta) = \frac{1}{2} \begin{bmatrix} \cos 2\theta + 1 & -\sin 2\theta & -\sin 2\theta & 1 - \cos 2\theta \\ \sin 2\theta & \cos 2\theta + 1 & \cos 2\theta - 1 & -\sin 2\theta \\ \sin 2\theta & \cos 2\theta - 1 & \cos 2\theta + 1 & -\sin 2\theta \\ 1 - \cos 2\theta & \sin 2\theta & \sin 2\theta & \cos 2\theta + 1 \end{bmatrix} \tag{2.3}$$

Example 3: SO(2) can also act on functions $\mathbb{R}^2 \to \mathbb{R}$, *e.g.* we can let V be the three-dimensional vector space of homogeneous polynomial functions $p : \mathbb{R}^2 \to \mathbb{R}$ of degree 2 and define

$$\tilde{R}(\theta)[p](x) = p\left(R(\theta)^\mathsf{T} x\right). \tag{2.4}$$

If we set $p(x) = a \cdot x_1^2 + b \cdot x_1 x_2 + c \cdot x_2^2$, we obtain after some algebraic manipulation, that

$$p\left(R(\theta)^\mathsf{T} x\right) = \frac{1}{2} \left(\begin{bmatrix} 1 + \cos 2\theta & \sin 2\theta & 1 - \cos 2\theta \\ -2 \sin 2\theta & 2 \cos 2\theta & 2 \sin 2\theta \\ 1 - \cos 2\theta & -\sin 2\theta & 1 + \cos 2\theta \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \right) \cdot \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}, \tag{2.5}$$

i.e. the coefficient vector $(a, b, c)^\mathsf{T}$ (which defines the polynomial) transforms under rotation by the matrix

$$\tilde{R}(\theta) = \frac{1}{2} \begin{bmatrix} 1 + \cos 2\theta & \sin 2\theta & 1 - \cos 2\theta \\ -2 \sin 2\theta & 2 \cos 2\theta & 2 \sin 2\theta \\ 1 - \cos 2\theta & -\sin 2\theta & 1 + \cos 2\theta \end{bmatrix} \tag{2.6}$$

where we use the notation $\tilde{R}$ again for simplicity.

Group representations can often be decomposed into smaller parts, called sub-representations.

Definition 2: Given a representation (ρ, V) of a group G, a representation (σ, W) of G is called a *sub-representation* of ρ, if $W \subset V$ and for all $g \in G$ and $w \in W$

$$\sigma(g)w = \rho(g)w. \tag{2.7}$$

Definition 3: A *proper* sub-representation of (ρ, V) is a sub-representation (σ, W), where W is neither 0-dimensional nor equal to V.

Example 4: Consider again the representation R_2 of SO(2) on 2×2-matrices from Example 2. Clearly, for any scaling of the identity matrix $M = \lambda I$, we have $R_2(\theta)[\lambda I] = \lambda I$. Furthermore, since planar rotations commute, we have for scalings of the 90° rotation matrix

$$M = \lambda R(\pi/2) = \lambda \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \tag{2.8}$$

that $R_2(\theta)[\lambda R(\pi/2)] = \lambda R(\pi/2)$. Thus, on the subspaces W_0 and W_1 of $\mathbb{R}^{2\times 2}$ spanned by

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{2.9}$$

respectively, we have sub-representations σ_0 and σ_1 of R_2 defined by

$$\sigma_0(\theta)[M] = M \text{ and } \sigma_1(\theta)[M] = M. \tag{2.10}$$

σ_0 and σ_1 are both *trivial* representations, so called because they are constant.

Example 5: The representation $\tilde{R}$ in Example 3 also has a trivial sub-representation. This is easily seen from the fact that any multiple of the polynomial $x_1^2+x_2^2$ is invariant to rotations. *I.e.*, the coefficient vector $(1, 0, 1)^\mathsf{T}$ spans a subspace W corresponding to a trivial representation.

Definition 4: An *irreducible representation* (short *irrep*) is a representation which has no proper sub-representations.

An immediate consequence of the definition is that any one-dimensional representation is irreducible. Irreps are a form of basic building block for representations. For commonly encountered groups, all irreps are known, and for finite and compact groups, it is the case that all finite-dimensional representations are direct sums of irreps. *I.e.* any representation ρ can be written as

$$\rho(g) = Q \left(\bigoplus_{j=1}^{n} \sigma_j(g) \right) Q^{-1}, \tag{2.11}$$

for some irreps σ_j and an invertible linear map Q. In terms of matrices, $\oplus$ means assembling the matrices into a block diagonal. To illustrate (2.11), we will decompose the representations in Examples 1, 2 and 3 into irreps.

Example 6: For the ordinary 2×2 rotation matrices in Example 1, it is clear that there is no proper sub-representation, since there is no proper subspace of $\mathbb{R}^2$ that is closed under $R(\theta)$. Thus, R is irreducible.

Example 7: Let's now consider $R_2(\theta)$ from Examples 2 and 4. The trivial representations σ_0 and σ_1 in Example 4 are one-dimensional and hence irreducible. The orthogonal complement of the corresponding subspaces is the space spanned by

$$A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{2.12}$$

Plugging each of these into (2.2), we obtain

$$R(\theta) A R(\theta)^{\mathsf{T}} = \begin{bmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{bmatrix} = \cos(2\theta) A + \sin(2\theta) B \tag{2.13}$$

and

$$R(\theta) B R(\theta)^{\mathsf{T}} = \begin{bmatrix} -\sin 2\theta & \cos 2\theta \\ \cos 2\theta & \sin 2\theta \end{bmatrix} = -\sin(2\theta) A + \cos(2\theta) B. \tag{2.14}$$

Hence, in the flattened vector space $\mathbb{R}^4$, A and B span a two-dimensional subspace where SO(2) acts by the frequency two rotation

$$\begin{bmatrix} \cos 2\theta & -\sin 2\theta \\ \sin 2\theta & \cos 2\theta \end{bmatrix}, \tag{2.15}$$

which is irreducible by the same argument as in Example 6. Indeed, we can decompose (2.3) as

$$R(\theta) \otimes R(\theta) = Q \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos 2\theta & -\sin 2\theta \\ 0 & 0 & \sin 2\theta & \cos 2\theta \end{bmatrix} Q^{-1}, \tag{2.16}$$

where

$$Q = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \end{bmatrix}. \tag{2.17}$$

Example 8: Next, we consider the representation $\tilde{R}$ from Examples 3 and 5, which rotates polynomials $ax_1^2 + bx_1x_2 + cx_2^2$. We identify the function space of polynomials with the space of coefficients $(a, b, c)^\mathsf{T}$. We saw already in Example 5 that one trivial irrep corresponds to the subspace spanned by $(1, 0, 1)^\mathsf{T}$. The orthogonal subspace is spanned by $u = (-1, 0, 1)^\mathsf{T}$ and $v = (0, 2, 0)^\mathsf{T}$, which (after some calculation) are found to be acted on by the irrep

$$\begin{bmatrix} \cos 2\theta & -\sin 2\theta \\ \sin 2\theta & \cos 2\theta \end{bmatrix}. \tag{2.18}$$

Similarly to in the previous example, we can write

$$\tilde{R}(\theta) = \tilde{Q} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 2\theta & -\sin 2\theta \\ 0 & \sin 2\theta & \cos 2\theta \end{bmatrix} \tilde{Q}^{-1}, \tag{2.19}$$

for

$$\tilde{Q} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 2 \\ 1 & 1 & 0 \end{bmatrix}. \tag{2.20}$$

We conclude this series of examples by mentioning that one complete set of irreps for SO(2) consists of precisely the rotation matrices $R(k\theta)$ with frequencies $k \in \mathbb{Z}_{\geq 1}$ together with the one-dimensional trivial representation. This means that any finite-dimensional representation of SO(2) corresponds to a direct sum of the mentioned irreps.

To decompose a general representation into irreps, one can use so-called character theory, we refer to the literature for details. Hopefully, the examples in this chapter have given the reader a flavour of key concepts in elementary representation theory. For more in-depth treatments, which go beyond what will be required knowledge for this thesis, we recommend the books by Serre and Hall [1], [2].

CHAPTER 3

Equivariance

As mentioned in the previous chapter, neural network feature maps are typically not interpretable. However, if a group representation acts on a feature map, then we know how the feature map changes under the group. A neural network outputting feature maps changing under a specific group representation is called *equivariant*. In this chapter we define equivariance and discuss equivariance for linear functions, neural networks and neural networks acting on images specifically.

3.1 Equivariance

Given a function f, taking images as input, we are interested in how the output $f(I)$ changes as the image I undergoes some transformation, for example, rotation. If there exists a representation σ of $\mathrm{SO}(2)$ such that $f(\mathtt{rot}_\theta(I)) = \sigma(\theta)f(I)$, where $\mathtt{rot}_\theta$ signifies image rotation by θ radians, then we call f rotation equivariant. Given σ, we know precisely how f behaves under rotations of the input, which can be practically of great use. In this section, we will give a general description of equivariant linear maps using the so-called *Schur's lemma*.

Definition 5: Given a group G and two representations (V, ρ) and (W, σ) of G, a function $f : V \to W$ is called *equivariant* (with respect to ρ and σ) if for all $g \in G$ and $v \in V$

$$f(\rho(g)v) = \sigma(g)f(v). \tag{3.1}$$

A special case of equivariance is *invariance*, where the representation σ in (3.1) is taken to be constant equal to the identity transformation on W.

Example 9: Let's again consider the representations R_2 and $\tilde{R}$ of SO(2) acting on matrices (Example 2) and homogeneous quadratic polynomials in two variables (Example 3). We will try to find the general form of equivariant linear maps $L : \mathbb{R}^3 \to \mathbb{R}^4$. *I.e.* L such that

$$L\tilde{R}(\theta) = R_2(\theta)L. \tag{3.2}$$

By multiplying both sides by $\tilde{R}(-\theta)$ from the right and using the decompositions found in Examples 7 and 8 we can write

$$\begin{aligned} L &= R_2(\theta)L\tilde{R}(-\theta) \\ &= Q \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos 2\theta & -\sin 2\theta \\ 0 & 0 & \sin 2\theta & \cos 2\theta \end{bmatrix} Q^{-1} L \tilde{Q} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 2\theta & \sin 2\theta \\ 0 & -\sin 2\theta & \cos 2\theta \end{bmatrix} \tilde{Q}^{-1} \end{aligned} \tag{3.3}$$

which after the change of variables $\tilde{L} = Q^{-1}L\tilde{Q}$ becomes

$$\tilde{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos 2\theta & -\sin 2\theta \\ 0 & 0 & \sin 2\theta & \cos 2\theta \end{bmatrix} \tilde{L} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 2\theta & \sin 2\theta \\ 0 & -\sin 2\theta & \cos 2\theta \end{bmatrix}. \tag{3.4}$$

Since the left side is independent of θ, the θ's on the right side must cancel out. After staring at the equation for a while, one concludes that the only way to make this happen is if $\tilde{L}$ has the form

$$\tilde{L} = \begin{bmatrix} a & 0 & 0 \\ b & 0 & 0 \\ 0 & c & -d \\ 0 & d & c \end{bmatrix}, \tag{3.5}$$

where the structure of the lower right part comes from the fact that it has to commute with rotation matrices. Knowing that L is equivariant thus implies that it is parameterized by 4 values, rather than the $3 \cdot 4 = 12$ an arbitrary linear map requires.

A general specification of equivariant linear maps comes from *Schur's lemma*. Schur's lemma is typically stated for representations over the complex numbers, but for applications we are more interested in representations over the real numbers. We therefore give a version of Schur's lemma for real representations here, following [3], [4]. The statement is also included in a deep learning context in [5].

Lemma 1 (Schur's lemma for real irreps)**:**

1. *Let (V, ρ) and (W, σ) be (real) irreps of a group G and let $f : V \to W$ be an equivariant linear map. Then either f is identically 0 or f is invertible.*

2. *Let E be the vector space of equivariant linear maps from the finite-dimensional real irrep (V, ρ) of G to (V, ρ) itself. Then E is either one-, two-, or four-dimensional.*

Proof. Sketch:

1. One shows that the image $f(V)$ of f must be a sub-representation of W, *i.e.* since W is an irrep, either $f(V) = W$ or $f(V) = \{0\}$. Similarly, one shows that the nullspace $f^{-1}(0)$ of f is a sub-representation of V, so that either $f^{-1}(0) = V$ or $f^{-1}(0) = \{0\}$. The statement follows.

2. By 1, E consists of invertible linear maps and the 0-map. Thus E forms a finite-dimensional associative division algebra over the real numbers, the bilinear product of the algebra being composition of maps. By the so-called *Frobenius theorem*, E must be isomorphic (as a division algebra over the reals) to either the real numbers, the complex numbers or the quaternions—having dimensions 1, 2 and 4 respectively.

□

If $f : V \to W$ is an invertible equivariant linear map, we call it an isomorphism of representations and we call (V, ρ) and (W, σ) isomorphic whenever such f exists. (V, ρ) is clearly isomorphic to itself. The space E of all equivariant linear maps $V \to V$ is called the endomorphism space of (V, ρ). For

Lie groups, the dimensionality (one, two or four) of E for a given finite-dimensional irrep (V, ρ) can be determined using character theory [3].

In Example 9, Schur's lemma manifested itself through the fact that $\tilde{L}$ has zeros for all entries that correspond to mappings between non-isomorphic irreps in the decompositions of $\tilde{R}$ and R_2. Further, the endomorphism space of the frequency-two irrep was the two-dimensional space of matrices $\left[\begin{smallmatrix} c & -d \\ d & c \end{smallmatrix}\right]$, isomorphic to the complex numbers. All the two-dimensional irreps of SO(2) have the same two-dimensional endomorphism space.

Schur's lemma gives a recipe for equivariant linear maps between representations (V, ρ) and (W, σ), as a generalization of Example 9. First decompose ρ and σ into irreps, then solve the equivariance constraint (3.1) for each irrep. An alternative approach is to solve (3.1) numerically [6]. It is worth mentioning that the approach of decomposing a representation into irreps can fail for non-compact groups.

3.2 Equivariance in neural networks

Neural networks consist of simple layers that are stacked into a large function. For instance, so-called fully connected neural networks take the form

$$f(x) = L_K(\eta(\cdots L_2(\eta(L_1(x)))\cdots)), \tag{3.6}$$

where the L_i are affine maps and η is a non-linear function, typically applied to each component of its input vector independently. Each η and L_i is called a layer of the neural network, and the L_i's contain the trainable parameters of the network. One principled manner of making f equivariant to a group G is to specify L_i and η to be equivariant—this is sometimes referred to as the Geometric Deep Learning Blueprint [7]. Typically, the representation of G on the input and output is known (say, rotating an image and rotating an image segmentation). However, to make every L_i equivariant, we need to specify a representation of G on the output of every layer. This represents a hyperparameter choice, and optimally choosing the representations is an open research problem. Once we have chosen G-representations in every layer, we can solve for equivariant linear layers, for instance using Schur's lemma as described in the previous section. Similarly, the non-linearity η can be chosen to be equivariant.

Example 10: A common neural network type for image processing is the convolutional neural network (ConvNet) [8], [9]. In a ConvNet, the linear layers consist of correlations of the input with trainable filter kernels. If the input to a layer is $F : \mathbb{Z}^2 \to \mathbb{R}^C$, F is correlated with a matrix-valued filter $M : \mathbb{Z}^2 \to \mathbb{R}^{D\times C}$, where D is the output feature dimension.[1] The output of the correlation is a feature map $M \star F : \mathbb{Z}^2 \to \mathbb{R}^D$. Explicitly, we have

$$(M \star F)(x) = \sum_{x' \in \mathbb{Z}^2} M(x')F(x + x'). \tag{3.7}$$

If we translate the input image by t, then since the same filter M is applied at every spatial location, the output $M \star F$ will also be spatially translated by t. This means that the linear layers in ConvNets are translation equivariant.[2]

Example 11: Next, we want to make ConvNet-layers equivariant to, say, rotations by 90° [11], [12]. By choosing a representation of the group of 90° rotations, $(\mathbb{R}^C, \rho)$ on the input feature dimension and $(\mathbb{R}^D, \sigma)$ on the output, M defines a rotation equivariant correlation if rotating F followed by correlating with M yields the same result as rotating $M \star F$, *i.e.*

$$\sigma(\theta)(M \star F)\left(R(\theta)^\mathsf{T} x\right) = \sum_{x' \in \mathbb{Z}^2} M(x')\rho(\theta)F\left(R(\theta)^\mathsf{T}\left(x + x'\right)\right). \tag{3.8}$$

It is worth pointing out that if M satisfies (3.8), then the layer is equivariant to roto-translations since any roto-translation can be written as a translation followed by a rotation, and correlation is translation equivariant by the previous example. Changing variables $x \mapsto R(\theta)x$ and $x' \mapsto R(\theta)x'$ in (3.8) gives

$$\sigma(\theta)(M \star F)(x) = \sum_{x' \in \mathbb{Z}^2} M(R(\theta)x')\rho(\theta)F(x + x'). \tag{3.9}$$

And by equating $M \star F$ in (3.7) and (3.9) we find a condition on M in order for (3.9) to hold for all F:

$$\sigma(\theta)M(x') = M(R(\theta)x')\rho(\theta), \tag{3.10}$$

which can be solved numerically [12] or analytically [13]. The condition (3.10)

[1] Here we consider F and M to be 0 outside a small bounded region of the pixel grid $\mathbb{Z}^2$.

[2] In practice, translation equivariance can fail in ConvNets due to border effects and subsampling [10].

is known as the kernel constraint.[3] In Paper C, we will study how close ordinary ConvNets are to fulfilling the kernel constraint for the group of horizontal reflections when trained on natural images.

Example 12: Due to the presence of $R(\theta)$ on the right-hand-side of the kernel constraint (3.10), it is not at first sight an equivariance constraint on a linear map. In fact, in the literature (3.10) is typically solved directly without appealing to Schur's lemma [12], [13]. However, intuitively, Schur's lemma should apply here since correlations (3.7) consist of applying a linear function at each position of the incoming feature map. This local linear function defined by M takes feature maps $F : \mathbb{Z}^2 \to \mathbb{R}^C$ to feature vectors in $\mathbb{R}^D$:

$$F \mapsto \sum_{x' \in \mathbb{Z}^2} M(x')F\left(x'\right) \tag{3.11}$$

We can obtain this linear map by evaluating $M \star F$ at 0. In order to look at the equivariance properties of M, we let $\mathcal{R}$ be the action rotating feature maps spatially around 0 and get from (3.8)

$$\sigma(\theta)(M\star F)(0) = \sum_{x' \in \mathbb{Z}^2} M(x')\rho(\theta)F\left(R(\theta)^\mathsf{T}x'\right) = \sum_{x' \in \mathbb{Z}^2} M(x')\rho(\theta)(\mathcal{R}(\theta)[F])(x'), \tag{3.12}$$

which together with the definition (3.7) of $(M \star F)(0)$ means that the local linear map (3.11) defined by M must be equivariant with respect to the combined action of ρ and $\mathcal{R}$ on the input and σ on the output. We can use translation equivariance of the correlation to see that the layer then becomes equivariant to rotations for $x \neq 0$ as well. Since M in practice will have finite support, say some finite square $S \subset \mathbb{Z}^2$, we can consider (3.11) as a linear function from cropped feature maps $F : S \to \mathbb{R}^C$ to feature vectors in $\mathbb{R}^D$. $\mathcal{R}$ is then a finite-dimensional permutation representation and we can apply Schur's lemma to solve for valid kernels M.

Above, we discussed how to enforce equivariance in neural networks by hard-coding the constraint into the linear maps. An alternative approach to making a neural network equivariant is to add a loss during training that encourages equivariance. This has the benefit of not requiring the hyperparameter choice of a G-representation per layer, and instead relies on the

[3] The kernel constraint can also be derived for continuous groups in much the same way as above [13], [14]. Kernels fulfilling the kernel constraint can be found by decomposing ρ and σ into irreps and solving the kernel constraint for each irrep-pair.

flexibility of the neural network to approximate arbitrary functions (including equivariant functions). Furthermore, more complicated neural network architectures than the stacking of linear layers (3.6), such as the currently very popular Transformer [15], require more effort to specify each layer to be equivariant.

Some tasks have training losses that implicitly encourage equivariance, meaning that no extra loss has to be added to the training, and ordinary training over a data distribution that is symmetric under a group G[4] implicitly encourages equivariance. In the next chapter, we will consider keypoint description as an example of a task that is naturally equivariant to planar rotations.

One interesting question in this setting is whether each layer in (3.6) has to be equivariant for the entire network to be equivariant. In Paper C, we show experimentally that this is commonly the case in practice, but we also find theoretical counterexamples.

3.3 Steerable image representations

We will now specialize the discussion to image representations. As discussed in Section 2.1 we will take the view that images are functions $I : \mathbb{R}^2 \to \mathbb{R}^3$ and image representations are functions $F : \mathbb{R}^2 \to \mathbb{R}^c$, where c is the feature/channel dimension of the image representation.

Images and image representations can be warped by functions $\phi : \mathbb{R}^2 \to \mathbb{R}^2$ describing the movement of each position in the plane to a new position in the plane. We introduce a warping operator W_ϕ for the transformation of images and image representations by ϕ:

$$W_\phi[F](x) = F(\phi^{-1}(x)). \tag{3.13}$$

In particular, we will consider warps by differentiable functions, *i.e.* ϕ satisfying

$$\phi(x + \Delta x) = \phi(x) + J_\phi(x)\Delta x + o(\|\Delta x\|). \tag{3.14}$$

[4]By symmetric data distribution, we here mean that the probability of a particular training sample is equal to the probability of the same sample transformed by any $g \in G$. Data distributions that are symmetric under some transformation are commonly obtained through data augmentation.

In (3.14), $J_\phi : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$ is the Jacobian of ϕ. Note that J can also be warped by (3.13).

Example 13: If ϕ is a roto-translation, we can write $\phi(x) = t + R(\theta)x$ for some rotation angle θ and translation vector t. Thus,

$$\phi(x + \Delta x) = t + R(\theta)(x + \Delta x) = \phi(x) + R(\theta)\Delta x, \tag{3.15}$$

so the Jacobian $J_\phi(x)$ is constant in x and given by the rotation matrix $R(\theta)$.

If we have a set Φ of continuously differentible functions ϕ, with invertible Jacobians, such that Φ forms a group under function composition, then by the chain rule

$$J_{\phi_2 \circ \phi_1} = J_{\phi_2} J_{\phi_1} \tag{3.16}$$

and by the inverse function theorem,

$$J_{\phi^{-1}} = J_\phi^{-1} \tag{3.17}$$

so the Jacobians of the functions in Φ generate a subgroup of GL(2), which we will call G_Φ. In the case of Φ being roto-translations as in Example 13, G_Φ is the planar rotation group SO(2).

If there exists a group representation $(\mathbb{R}^c, \varsigma)$ of G_Φ such that

$$f(W_\phi[I]) = W_\phi[\varsigma(J_\phi) f(I)], \tag{3.18}$$

then we call ς a G_Φ-*steerer* for f.[5] We also say that ς steers $f(I)$ and that $f(I)$ is steerable. Equation (3.18) is a specific equivariance condition on f that means that the features at each location of $f(I)$ are warped spatially, but also modified by the steerer ς in the channel dimension.

Steerable image representations can be obtained by using neural networks built from equivariant layers as described in Section 3.2. However, it is also possible to train the networks to approximately satisfy the condition (3.18) by including it in the loss function. Specifically in the case of representation learning, where an image I and an augmented image $W_\phi[I]$ are input to a feature extractor f, trained to minimize the discrepancy between the features

[5] ς is the greek letter sigma in word-final position. Sigma stands for **s**teerer and we choose the word-final form since the steerer is applied to the output of the feature extractor f. The name steerer is an hommage to the steerable filters by Adelson and Freeman [16].

extracted from I and $W_\phi[I]$,

$$\left\| f(I) - W_{\phi^{-1}}[f(W_\phi[I])] \right\|, \tag{3.19}$$

it is very simple to include a steerer in the loss

$$\left\| \varsigma(J_\phi) f(I) - W_{\phi^{-1}}[f(W_\phi[I])] \right\|, \tag{3.20}$$

without introducing a new loss term. As a special case of representation learning, we will study steerers for keypoint description in Papers A and B.

CHAPTER 4

Keypoint description and matching

This chapter reviews the basics of keypoint description and matching, which is the topic of Papers A, B and E of this thesis. A keypoint is a position in an image that contains some distinctive content. For instance, given multiple images taken in the same scene, keypoints would ideally be projections of recognizable 3D points, so that the same 3D point will be detected as a keypoint in multiple images. Such keypoints are helpful for 3D reconstruction because with enough keypoint correspondences in multiple images, we can use *structure-from-motion* algorithms to calculate both the camera motion from each image capture to the next, as well as the 3D structure of the objects in the scene. To go from keypoints to keypoint correspondences, we will assign to each keypoint a description—a D-dimensional vector—and match keypoints in different images with similar descriptions. An example of matches is shown in Figure 4.1. To obtain descriptions, we will use neural networks, which will be called descriptors. Often in the literature, what we call keypoint descriptions are called keypoint descriptors. We will stick to the convention that a descriptor is a function that outputs descriptions.[1] We will also use the verb *describe* for the application of a keypoint descriptor.

[1] This convention was suggested to the author by Johan Edstedt.

Figure 4.1: Example of matching keypoints in two images of the same building.

4.1 Background on keypoint description

We will consider keypoint descriptors as functions f that take as input an image I and a keypoint x and output a description $v = f(I, x)$. This notation is less cumbersome than the more accurate $f(I)(x)$, which is what we would have if we view the image representation $f(I)$ as a function $\mathbb{R}^2 \to \mathbb{R}^D$, from which we extract descriptions. In practice, a descriptor will take multiple keypoints as input and describe them all in parallel, but we consider the single keypoint case for conceptual simplicity. Once we have descriptions for a set of keypoints in two images, we can match the descriptions in the first image with the ones in the second image. The matching is typically done through variants of mutual nearest neighbours.

For keypoint descriptors to be useful, they must output descriptions that can disambiguate keypoints that belong to different 3D points. On the other hand, the descriptions must not lead to false negatives when the same 3D point is photographed in different lighting conditions or from different viewing directions. Of course, whether false negatives are obtained or not is not only dependent on the descriptions but also on the matching procedure, *i.e.*, a matching procedure that is more strict will produce more false negatives.

Before neural networks were introduced to the problem of keypoint description, there were many hand-crafted descriptors. The most famous one is probably the Scale Invariant Feature Transform (SIFT) [17], which is still heavily used in practice.

Example 14: SIFT is an algorithm for both keypoint detection and description, but we will focus on the description part here. Importantly, however, we must mention that SIFT keypoints consist not only of a position in an image but also of an orientation and a scale. The SIFT descriptor computes the description of a keypoint (in a grayscale image) through the following steps.

1. For each keypoint, a square grid of 16 image patches is centred at the keypoint. The scale of the keypoint determines the scale of the square, and the square is rotated according to the orientation of the keypoint.

2. Each of the 16 image patches obtained is further divided into 16 smaller sub-patches.

3. In each of the sub-patches, the image gradient is calculated and weighted by the amplitude of a Gaussian window centred at the keypoint.

4. For each of the 16 image patches, the scaled gradients of its sub-patches are summarized in a histogram with 8 bins, corresponding to 8 angle intervals.

5. The final description of the keypoint consists of all the histograms from the image patches. A SIFT description is, therefore, $16 \cdot 8 = 128$ dimensional.

Recent developments have seen neural network-based descriptors outperform classical methods such as SIFT by large margins. The drawback is that the descriptions are less interpretable.

Example 15: DeDoDe [18] is a recent neural network-based keypoint descriptor. Training data is extracted from large-scale 3D reconstructions (reconstructed using *e.g.* SIFT!). A diverse set of image pairs with varying amounts of overlap between them is needed for descriptor training. Furthermore, we need a way to obtain keypoints and determine whether keypoints in two images correspond to the same 3D point. DeDoDe uses a separately trained DeDoDe keypoint detector to obtain keypoints, but it is also common to train a detector and a descriptor jointly. To determine whether a keypoint correspondence is correct, we backproject the keypoints into 3D using depth maps and camera poses from the 3D reconstructions. Keypoint correspondences that correspond to close enough 3D points are determined to be correct matches. For DeDoDe, descriptions are always normalized to unit norm. The similarity between descriptions is measured in cosine similarity, *i.e.* by their scalar product. Given two images and n keypoints in each, we form the similarity matrix

$$S_{ij} = \left\langle f(I, x_i), f(I', x'_j) \right\rangle . \tag{4.1}$$

S is normalized by using softmax over the rows and columns, and the training objective is to maximize the entries of S corresponding to correct matches.[2] This normalization ensures that the similarity of correct matches is maximized and the similarity of incorrect matches minimized during training.

[2] In fact, the negative logarithms of the entries are minimized during training.

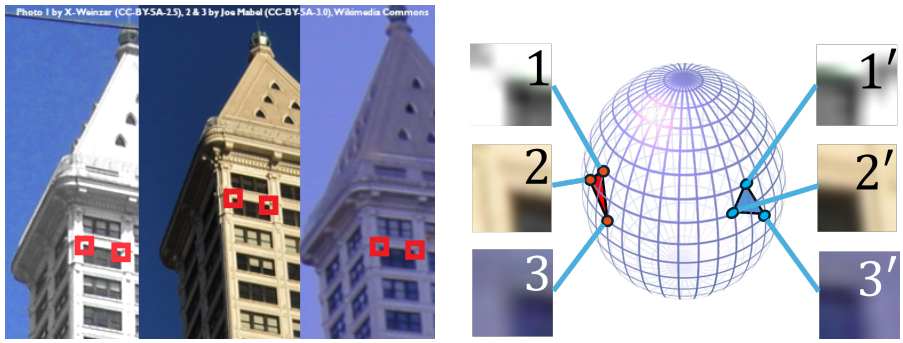

Figure 4.2: Illustration of the fact that keypoint patches are likely to appear in all orientations while training a keypoint descriptor on upright images. Left: Three upright images with corners in all 90° rotations. Right: abstract illustration of the latent space where descriptions of the keypoints lie on a $(D-1)$-dimensional hypersphere. Different orientations will be pushed to different locations in latent space. If we assume that the description of 1 is as close to the description of 2 as the description of $1'$ is to the description of $2'$, then the red and blue triangles must be isometric. An extension of this argument serves as the motivation for rotation equivariance given in Section 4.2.

4.2 Rotation equivariant keypoint description

In this section, we will explain why rotation equivariance is a natural property of keypoint descriptors, *i.e.* we will explain why keypoint descriptors have steerers. The argument stems from Paper A, Section 5, but we aim to make it more precise here, highlighting what parts of the argument are hand-wavy and what parts are rigorous. Figure 4.2 might help when following the argument.

Consider the setting of the DeDoDe descriptor in Example 15 again. We will argue that DeDoDe has a steerer for 90° rotations and make the following assumptions:

1. The descriptor network f only considers the local neighbourhood of a keypoint x when generating a description for it. Such local neighbourhoods will be called patches.

2. Each patch is as likely to appear in each 90° orientation.

Assumption 1 is likely to be approximately true in practice, especially when using ConvNets with small kernel sizes. Assumption 2 is likely to be approximately true when using datasets with man-made environments (*e.g.* corners of windows) but also in natural environments.

Using Assumption 1, we will suppress the dependence of f on an image and simply write $f(x)$ for the description of keypoint x, associated with some image patch. To rotate x from here on means to rotate the patch around x. Let's take two pairs of (not necessarily matching) keypoints x, x' and y, y', such that the patches around x and x' are $90°$ rotations of the patches around y and y'. Recall that the network is trained by maximizing cosine similarities between matching keypoint pairs while minimizing cosine similarities of non-matching pairs. Using Assumption 2, we make the case that the pair x, x' is treated equivalently during training as the pair y, y'. Thus, it is reasonable to assume that after training, the similarity of $f(x)$ and $f(x')$ should be approximately the same as the similarity of $f(y)$ and $f(y')$:

$$\langle f(x), f(x') \rangle \approx \langle f(y), f(y') \rangle . \tag{4.2}$$

From (4.2), we conclude that there must be some orthogonal transformation mapping the line segment $f(x), f(x')$ to the line segment $f(y), f(y')$ on the hypersphere of descriptions. Thus, a rotation on the input space is encoded as a high-dimensional orthogonal transformation of the description space. We call this orthogonal transformation of description space a steerer. The steerer is under-specified so far, but it will be uniquely defined by considering more keypoints.

Thus, we must explain why the steerer is the same for all keypoints. To argue this, let $x_1, x_2, \ldots, x_d$ be keypoints such that $\{f(x_i)\}_{i=1}^d$ form a linear basis for the span of the range of f. Let y_i be keypoints whose patches correspond to $90°$ rotations of the patches of the x_i. Then, by (4.2),

$$\langle f(x_i), f(x_j) \rangle \approx \langle f(y_i), f(y_j) \rangle \tag{4.3}$$

for all pairs i, j. The sequence of descriptions $(f(x_i))_{i=1}^d$ and the sequence $(f(y_i))_{i=1}^d$ are thus related by some orthogonal matrix Q:

$$f(y_i) \approx Q f(x_i). \tag{4.4}$$

Next, we show that Q works for any keypoint x, and its rotated version y. We assume (again from (4.2)) that

$$\langle f(x), f(x_i)\rangle \approx \langle f(y), f(y_i)\rangle \tag{4.5}$$

for all i, meaning that

$$\langle Q^\mathsf{T} f(y), f(x_i)\rangle \approx \langle Q^\mathsf{T} f(y), Q^\mathsf{T} f(y_i)\rangle = \langle f(y), f(y_i)\rangle \approx \langle f(x), f(x_i)\rangle. \tag{4.6}$$

But since the $f(x_i)$'s form a basis of the span of the range of f, this means that $Qf(x) \approx f(y)$.

Finally, the fact that the steerer Q generates a group representation follows immediately from applying the preceding argument to the rotated keypoint, *i.e.* from the fact that rotating a keypoint twice corresponds to multiplying the description by Q^2. A similar argument to the above can also be made for continuous rotations, see further Theorem 5.1 in Paper A.

Steerers enable rotation invariant matching in several ways. We outline the simplest version here. Refer to Paper A for more details. Given a 90°-steerer Q, we can match images up to a global rotation by matching descriptions $f(I, x)$ in one image with rotated versions of descriptions $Q^k f(I', x'), k = 0, 1, 2, 3$ in a second image and keeping the matches from the multiple k of 90° with the most matches. Using the steerer gives a clear computational benefit over having to apply the descriptor network f once for each rotation $k \cdot 90°$.

The argument given in this section did not preclude Q being the identity matrix. In short, Q becomes the identity when there is no benefit in f being able to differentiate between x and its rotated version y. However, when training a keypoint descriptor on upright images, the training loss encourages f to distinguish *e.g.* corners in different orientations (see for instance Figure 4.2). So, Q will not be the identity in this case. If we apply rotation augmentation to upright images naively during training, the loss will encourage f to be rotation invariant, *i.e.* Q to be the identity. Rotation invariance typically leads to worse performance on upright images.

CHAPTER 5

Summary of included papers

This chapter provides a summary of the included papers.

5.1 Paper A

Georg Bökman, Johan Edstedt, Michael Felsberg, Fredrik Kahl
Steerers: A framework for rotation equivariant keypoint descriptors
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),
pp. 4885-4895, 2024
`https://arxiv.org/abs/2312.02152` .

We demonstrate experimentally that neural network keypoint descriptors trained on upright images become equivariant to rotations. We call the group representation in the latent space steerer. We give a geometrical argument for why it is not entirely surprising that steerers exist for descriptors trained on upright images. Having argued for the naturality of rotation equivariance for the problem, we train descriptors and steerers jointly and also train descriptors for fixed steerers. Practically, we show state-of-the-art results.

5.2 Paper B

Georg Bökman, Johan Edstedt, Michael Felsberg, Fredrik Kahl
Affine steerers for structured keypoint description
To be presented at *European Conference on Computer Vision (ECCV)*, 2024
`https://arxiv.org/abs/2408.14186` .

We generalize steerers from the rotation group to the affine group GL(2). This allows us to treat equivariance to arbitrary differentiable image warps. We show experimentally that affine steerers work and that pretraining with affine steerers leads to SotA results on upright images.

5.3 Paper C

Georg Bökman, Fredrik Kahl
Investigating how ReLU-networks encode symmetries
Advances in Neural Information Processing Systems (NeurIPS), vol. 36, pp. 13720–13744, 2023
`https://arxiv.org/abs/2305.17017` .

We investigate how neural networks encode symmetries. In particular, we show a connection between the permutation conjecture by Entezari et al. and the layerwise equivariance of convolutional neural networks. Experimentally, we observe that the more invariant to horizontal flips a convolutional neural network is, the closer it is in structure to a group convolutional neural network.

5.4 Paper D

Lucas Brynte*, **Georg Bökman***, Axel Flinth, Fredrik Kahl
Rigidity preserving image transformations and equivariance in perspective
Scandinavian Conference on Image Analysis (SCIA),
pp. 59–76, 2023
`https://arxiv.org/abs/2201.13065` .

Convolutional neural networks are equivariant with respect to pixel translations. We show that pixel translations do not correspond to rigid motions of the camera. The only pixel transformations that correspond to any rigid motions of the camera are rotational homographies (corresponding to rotations of the camera). We propose two methods for increased equivariance to rotational homographies, namely data augmentation and warping the images from rectilinear to an azimuthal equidistant projection.

5.5 Paper E

Georg Bökman, Fredrik Kahl
A case for using rotation invariant features in state of the art feature matchers
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops,
pp. 5110-5119, 2022
`https://arxiv.org/abs/2204.10144` .

We modify the LoFTR semi-dense image matching model to make it more robust to inplane rotations. This is done by replacing all convolutional layers with layers equivariant under rotations. Experiments show that this leads to large improvements in performance on image pairs with large inplane rotations while not sacrificing much performance on upright image pairs.

*Equal contribution

References

[1] J.-P. Serre, *Linear representations of finite groups*. Springer, 1977.

[2] B. C. Hall, *Lie groups, Lie algebras, and representations*. Springer, 2013.

[3] J. M. Boardman, *Real and Complex Representations*, 2007.

[4] T. Bröcker and T. Tom Dieck, *Representations of compact Lie groups*. Springer, 2013.

[5] M. Weiler, P. Forré, E. Verlinde, and M. Welling, *Equivariant and Coordinate Independent Convolutional Networks*. 2023.

[6] M. Finzi, M. Welling, and A. G. G. Wilson, 'A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups,' in *Int. Conf. Machine Learning*, 2021.

[7] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, 'Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges,' *arXiv:2104.13478 [cs, stat]*, May 2021.

[8] K. Fukushima, 'Cognitron: A self-organizing multilayered neural network,' *Biological cybernetics*, vol. 20, no. 3, pp. 121–136, 1975.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, 'Gradient-based learning applied to document recognition,' *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[10] A. Azulay and Y. Weiss, 'Why do deep convolutional networks generalize so poorly to small image transformations?' *J. Mach. Lear. Res.*, vol. 20, no. 184, pp. 1–25, 2019.

[11] T. Cohen and M. Welling, 'Group equivariant convolutional networks,' in *Int. Conf. Machine Learning*, 2016.

[12] T. Cohen and M. Welling, 'Steerable CNNs,' in *Int. Conf. Learn. Represent.*, 2017.

[13] M. Weiler and G. Cesa, 'General $E(2)$-equivariant steerable CNNs,' in *Adv. Neural Inform. Process. Syst.*, 2019.

[14] M. Weiler, F. A. Hamprecht, and M. Storath, 'Learning steerable filters for rotation equivariant CNNs,' in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[15] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, 'Attention is All you Need,' in *Adv. Neural Inform. Process. Syst.*, 2017.

[16] W. T. Freeman, E. H. Adelson, *et al.*, 'The design and use of steerable filters,' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 9, pp. 891–906, 1991.

[17] D. G. Lowe, 'Distinctive Image Features from Scale-Invariant Keypoints,' *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[18] J. Edstedt, G. Bökman, M. Wadenbäck, and M. Felsberg, 'Dedode: Detect, don't describe—describe, don't detect for local feature matching,' in *Int. Conf. 3D Vis.*, 2024.