In Search of Projectively Equivariant Neural Networks

Georg Bökman
Chalmers University of Technology
{bokman, fredrik.kahl}@chalmers.se

Axel Flinth
Umeå University
axel.flinth@umu.se

Abstract

Equivariance of linear neural network layers is well studied. In this work, we relax the equivariance condition to only be true in a projective sense. In particular, we study the relation of projective and ordinary equivariance and show that for important examples, the problems are in fact equivalent.

The rotation group in 3D acts projectively on the projective plane. We experimentally study the practical importance of rotation equivariance when designing networks for filtering 2D-2D correspondences. Fully equivariant models perform poorly, and while a simple addition of invariant features to a strong baseline yields improvements, this seems to not be due to improved equivariance.

1 Introduction

Deep neural networks [21] have been successfully applied across a large number of areas, including but not limited to computer vision [20], natural language processing [9], game play [36], biology [18]. In many of these areas the data contains geometric properties or symmetries that can exploited when designing neural networks. For instance, AlphaFold [18] models proteins as graphs on which a network respecting the graph structure operates. Much of the work on neural networks respecting geometry and symmetry of the data can be boiled down to formulating the symmetries in terms of group equivariance. Group equivariance of neural networks is a currently very active area of research starting with [42] and brought in to the deep networks era by [8], recent surveys include [4, 12].

The typical way of formulating an equivariance condition is by considering the neural network as a mapping from a vector space $V$ to another vector space $W$ and requiring application of the network to commute with group actions on $V$ and $W$. As $V, W$ are vector spaces the setting is nicely framed in terms of representation theory (we will provide a brief introduction in Section 2). What happens however when $V$ and/or $W$ are not vector spaces? This work covers the case when $V$ and $W$ are projective spaces, i.e. vector spaces modulo multiplication by scalars. One might intuitively say that cases where $V$ and $W$ are projective spaces could be quite obscure and not relevant to practical machine learning applications, but the utility of homogeneous coordinates in computer vision shows that projective spaces can be highly relevant. In Example 2.2 we describe the motivating example of rotation equivariance of pinhole projections in detail.

Our main theoretical result Theorem 2.16 relates the projective equivariance problem with a linear equivariance problem for the commutator subgroup of the relevant group. We then use this to show that in many cases, the projectively equivariant neural network layers are exactly the same as the equivariant ones, at least when the projective representation is a projection of an ordinary one. The equivalence is slightly surprising, since projective equivariance is a weaker condition and hence should allow more expressive architectures. In Appendix B, we go one slight step further and consider general projective representations in the setting of Lie groups. In this case, the equivariance problem can be formulated with the help of a representation of the Lie algebra of the group, and we prove a result similar to Theorem 2.16.

This work also has an experimental part. First, Section 3.1 presents a proof-of-concept toy experiment. We find that an equivariant model in some extreme cases is better at detecting lines in noisy 2D point clouds than non-equivariant models are (even when the latter are trained on augmented data).

*Equal contribution.

Preprint. Preliminary work.
In Section 3.2 we turn to more practically relevant experiments on filtering outliers from a set of 2D-2D correspondences of keypoints in two images of the same scene. We build on T-Net [46] to create networks that are equivariant to rotations of the scene. It turns out that these purely equivariant models compare very badly to T-Net. We therefore also consider a T-Net with added rotation invariant features based on nearest neighbours of the keypoints. This does increase performance, but not rotation equivariance.

In summary, the two main takeaways of this work are in essence negative:

1. Theorem 2.16 implies that neural network layers that are projectively equivariant are in many cases no more expressive than linearly equivariant ones.
2. Our experiments indicate that incorporating 3D rotation equivariance into practical problems involving pinhole projections might not be the key to improving performance of SOTA models.

However, our experiments do lead to improvements on T-Net and it is our hope that the theoretical contributions will be found interesting by the community.

2 Projective equivariance

In this work, we are concerned with networks that are projectively equivariant. Towards giving a formal definition, let us first discuss how equivariance can be formulated using the notion of a representation of a group.

A representation of a group $G$ on a vector space $V$ is a group homomorphism $\hat{\rho} : G \to GL(V)$. That is, a map $\hat{\rho}$ associating each group element $g$ with an isomorphism $\hat{\rho}(g)$ in a manner that respects the structure of the group, i.e.,

$$\hat{\rho}(g) \hat{\rho}(h) = \hat{\rho}(gh) \quad \text{for all } g, h \in G. \quad (1)$$

As we will also consider projective representations in this paper, we will refer to ‘ordinary’ representations (1) as linear representations. Given linear representations $\hat{\rho}_0$ and $\hat{\rho}_1$ on spaces $V$ and $W$, respectively, we say that a map $\Phi : V \to W$ is equivariant if

$$\hat{\rho}_1(g) \circ \Phi = \Phi \circ \hat{\rho}_0(g) \quad \text{for all } g \in G. \quad (2)$$

If in (2), $\hat{\rho}_1(g)$ is the identity transformation for every $g \in G$, then $\Phi$ is called invariant.

Example 2.1. Consider a set classification task: A classifier $\Phi$ maps a set of $m$ one-dimensional elements to a probability distribution over $c$ classes. So here $V = \mathbb{R}^m$ and $W = [0, 1]^c$. The classification of an element $v \in V$ should however not depend on the order of the points, since that does not change the set of scalars in $v$. Hence, $\Phi$ should be invariant to permutations. Letting $\rho_0(\pi)$ being the permutation matrix given by $\pi$ and $\rho_1(\pi) = \text{id}$ for all $\pi$ (i.e., $\rho_0$ be the canonical representation of $S_m$ on $V$, and $\rho_1$ the trivial representation) Then ‘a set should be classified in the same way regardless of order of the input points’ means exactly the same as (2). The question of how to design deep neural networks satisfying this symmetry has been extensively studied, starting with DeepSets [44] and PointNet [32].

Many further symmetry conditions can be phrased in terms of (2), and ways to design neural networks $\Phi$ satisfying (2) have been extensively studied for e.g. rigid motions in 2D/3D/nD [2, 6, 8, 38, 41], permutations of graph nodes [27] and for more general groups $G$ [1, 7, 10, 19, 42]. The most famous example is probably the translation equivariance of CNNs [11, 22].

In this work, we want to study networks that are projectively in- and equivariant. To explain this notion, let us first introduce the notation $P(V)$ the projective space associated to a vector space $V$ over a field $\mathbb{F}$. That is, $P(V)$ is the space of equivalence classes of $V$ under the equivalence relation $v \sim w \iff \exists \lambda \in \mathbb{F} \setminus \{0\}$ s.t. $v = \lambda w$. Furthermore, we will write $\Pi_V$ for the projection that maps $v \in V$ to its equivalence class $\Pi_V(v) \in P(V)$.

Example 2.2. The pinhole camera model is well-known and popular in camera vision [15]. In essence, the idea is to identify a point in an image with the line in $\mathbb{R}^3$ that this point can be an image of. Mathematically, this amounts to embedding the 2D-point $y \in \mathbb{R}^2$ to the 3D-point $x = [y, 1] \in \mathbb{R}^3$, and then consider it as a point in $P(\mathbb{R}^3)$. See also Figure 1. A 2D-point-cloud can in the same way
We will use the short hand PGL(we do not consider this model here. Could however still be more expressive than a bona-fide invariant one, whence the question of Φ with a scalar written in homogeneous coordinates as a vector in the direction of the ray passing through the point and the camera centre. Rotations act projectively on homogeneous coordinates. Right A 2D line is in homogeneous coordinates defined by a plane ⟨w, x⟩ = 0. Applying a homography to the line corresponds to rotating the plane (and therefore w) in R^3.

be identified with a set of points in homogeneous coordinates, say x_i = [y_i, 1] \in R^3, i \in [m]. In this manner, we can think of them elements X of V = (R^3)^m. Of course, multiplying all points in X with a scalar λ does not change the image of the points, so that we can consider X as only being defined modulo R, i.e., an element of P(V).

Let us now consider the task of estimating a line that is present in the cloud, as in Figure 1. The line can be described through an equation ⟨w(X), x⟩ = 0 for a w(X) ∈ W = R^3. Note that w(X) also is only defined up to a multiplicative scalar, and hence can be regarded as an element of P(R^3). The map w : P(V) → P(R^3) is still invariant to permutations, and also equivariant to rotations of the cloud X, i.e. w(RX) = Rw(X). This equivariance however only needs to hold in a projective sense. To make the last point more explicit, consider a ‘representative map’ ˜w : V → R^3 of w, meaning that ˜w(X) is a member of the equivalence class w(Π_V(X)) for all X ∈ V. The equivariance of w does not imply that ˜w(ρ_0(g))X = ρ_1(g)w(X) for all g – that equation rather only needs to hold up to a multiplicative constant. That multiplicative constant can furthermore depend on both π and X. Enforcing projective equivariance is hence a priori less restrictive than equivariance in the ordinary sense, and could therefore potentially open up for a more expressive architecture.

Remark 2.3. In a classification task, with W = [0, 1]^e, it is not wise to model the map Φ as a map from P(V) → P(W), since the interpretation of Φ(V) as a class of probabilities is lost when going over to P(W). A network Ψ : P(V) → W built by normalizing a projectively equivariant backbone Φ could however still be more expressive than a bona-fide invariant one, whence the question of building equivariant networks is interesting also here.

Remark 2.4. One could also consider the point clouds of members of (P(R^3))^m, rather than P((R^3)^m). This amounts to considering each point to be known only up to a scalar. While this certainly would be interesting, there are very few linear maps (R^3)^m → (R^3)^m that are well defined as maps P(R^3)^m to P(R^3)^m (the interested reader is referred to Appendix A.6 for details). Therefore, we do not consider this model here.

Motivated by Example 2.2, we are interested in maps that fulfil the equivariance relation (2) up to a multiplicative scalar. To formalise this notion, let us begin by introducing some notation.

We will use the short hand PGL(V) = P(GL(V)) which is called the projective general linear group. Note that an element M ∈ PGL(V) defines a projective-linear map P(V) → P(V).

Definition 2.5. A projective representation of a group G on a projective space P(V) is a group homomorphism ρ : G → PGL(V).

Put more concretely, a projective representation is a map associating each g ∈ G to an equivalence class ρ(g) of invertible linear maps. ρ respects the group structure in the sense of (1) – however, only in the sense of elements in PGL(V), i.e., up to a multiplicative scalar.

Example 2.6. The simplest form of a projective representation is a projected linear one. That is, given a linear representation ˜ρ : G → GL(V), we immediately obtain a projective representation through ρ = Π_{GL(V)} o ˜ρ. In this work, we will heavily focus on this type of projective representations.

Example 2.7. [14] An interesting example is related to spin within quantum mechanics. For every n, there is a linear representation ˆρ of the group SU(2) on C^n. Using the fact that SU(2) is a global double cover of SO(3), this can be used to define a map ρ on SO(3) to GL(C^n) up to a scalar, i.e., a
projective representation. For odd $n$, this is a projection of a linear representation of $SO(3)$, but for even $n$, it is not. In quantum mechanics, odd an even $n$ correspond to integer and half-integer spin.

Given projective representations $\rho_0, \rho_1$ on $V$ and $W$, respectively, we can now state equivariance of a map $\Phi : P(V) \to P(W)$ exactly as before in (2):

$$\rho_1(g) \circ \Phi = \Phi \circ \rho_0(g) \text{ for all } g \in G.$$  \hfill (3)

Note that we only demand that the equality holds in $P(W)$ and not $W$ which was the case in (2). We refer to $\Phi$ satisfying (3) as being projectively equivariant.

2.1 Projectively equivariant linear maps

The canonical way to construct equivariant neural networks is to restrict both the activation functions as well as the linear layers to be equivariant. This method easily generalizes to projective equivariance, motivating the following question: Which linear maps $A : V \to W$ define projectively equivariant transformations $P(V) \to P(W)$? If the projective representations $\rho_0$ and $\rho_1$ are projections of linear representations $\hat{\rho}_0$ and $\hat{\rho}_1$, respectively, we immediately see that every equivariant $A \in \text{Hom}(V, W)$ defines a projectively equivariant transformation. The purpose of this section is to show that for many important groups we cannot find any other projectively equivariant transformations.

Let us first reformulate the equivariance problem slightly as is typically done [10, 41]. Given projective representations $\rho_0$ and $\rho_1$ on $V$ and $W$, we may define a projective representation on the space $\text{Hom}(V, W)$ of linear maps between $V$ and $W$, through

$$\rho : G \to \text{PGL}(\text{Hom}(V, W)), \quad \rho(g)A = \rho_1(g)A\rho_0^{-1}(g).$$

Just as in the linear case, we can reformulate projective equivariance of an $A \in \text{Hom}(V, W)$ as an invariance equation under $\rho$.

**Lemma 2.8.** A linear map $A : V \to W$ is projectively equivariant if and only if $A$ is invariant under $\rho$, that is

$$\rho(g)A = A \text{ for all } g \in G.$$

The above equation is understood in $P(\text{Hom}(V, W))$.

The proof, which is slightly less trivial than in the linear case, can be found in the appendix. The lemma implies that we in the following can concentrate on analyzing invariance equations instead

$$\rho(g)v = v \text{ for all } g \in G. \quad \text{(Proj)}$$

Note that these equations are understood in $P(V)$. Our next goal will be to find corresponding equations in $V$. In the case of $\rho$ being a projection of a linear representation $\hat{\rho}$, as described in Example 2.6, the obvious equations to compare to are

$$\hat{\rho}(g)v = v \text{ for all } g \in G. \quad \text{(Lin)}$$

Instead of investigating the differences and similarities between (Proj) and (Lin) directly, we will treat a more general question that also pertains to projective representations $\rho$ that are not projections of linear ones. To generalize, we will 'lift' $\rho$ to a linear representation of a so-called covering group of $G$. We need the following two definitions.

**Definition 2.9** ([5, Def. 5.36]). Let $G$ be a topological group. A group $H$ is called a covering of $G$ if there exists a group covering $\varphi : H \to G$, i.e. a surjective continuous group homomorphism which maps some neighbourhood of the unit element $e_H \in H$ to a neighbourhood of the unit element $e_G \in G$ homeomorphically.

**Definition 2.10.** Let $\rho : G \to \text{PGL}(V)$ be a projective representation, and $H$ a covering group of $G$, with group covering $\varphi$. A lift of $\rho$ is a linear representation $\hat{\rho} : H \to \text{GL}(V)$ with

$$\rho \circ \varphi = \Pi_{\text{GL}(V)} \circ \hat{\rho}.$$  

**Example 2.11.** If $\rho : G \to \text{PGL}(V)$ is a projection of a linear representation $\hat{\rho} : G \to \text{GL}(V)$ as in Example 2.6, then $\hat{\rho}$ is a lift of $\rho$ (the covering group is simply $G$ itself).

**Example 2.12.** The linear representations related to spin discussed in Example 2.7 are defined on $\text{SU}(2)$, which is a covering group of $SO(3)$. The linear representations are lifts of the projective representations of $SO(3)$ discussed there.
Given a lift $\hat{\rho}$ of $\rho$, we would like to understand the relation between the projective invariance equations $(\text{Proj}_G)$ of $\rho$ and the linear invariance equations of $\hat{\rho}$,

$$\hat{\rho}(h)v = v \text{ for all } h \in H. \quad (\text{Lin}_H)$$

We remind the reader that the equations in $(\text{Lin}_H)$ are understood in $V$, whereas the ones in $(\text{Proj}_G)$ are only understood in a projective sense. $(\text{Lin}_H)$ is hence a priori more restrictive than $(\text{Proj}_G)$.

**Lemma 2.13.** If $x \in V$ solves $(\text{Lin}_H)$, its equivalence class in $P(V)$ solves $(\text{Proj}_G)$.

The (simple) proof is presented in Appendix A. The converse direction, i.e., that $(\text{Proj}_G)$ imply $(\text{Lin}_H)$, is sometimes true, but not always.

**Example 2.14.** Consider the canonical linear representation of $\mathbb{Z}_n$ on $\mathbb{C}^n$, $\hat{\rho}(p)(x) = x_{m-p}$, $m,p \in \mathbb{Z}_n, x \in \mathbb{C}^n$. It is clear that all elements fixed under this action are multiples of the all-one vector, $x^0 = 1, m \in \mathbb{Z}_n$. However, if we project $\hat{\rho}$ to a projective representation $\rho$, all of the vectors $x^k = \omega^{km}, \omega_n = \exp(2\pi i/n)$ are invariant, since

$$(\rho(p)x^k)_m = x^{p(m-p)} = \omega^{k(m-p)} = \omega^{-kp} = \omega^{-km}x^m,$$

i.e $\rho(p)x^k = x^k$ in a projective sense.

**Example 2.15.** Now consider the canonical representation of $\text{SO}(3)$ on $\mathbb{R}^3$, $\hat{\gamma}(R)x = Rx$. It is clear that there are no $x$ with $Rx = x$ for all $R \in \text{SO}(3)$. This is however also true in a projective sense - given an $x \in \mathbb{R}^3$, we can always find an $R \in \text{SO}(3)$ so that $x$ and $Rx$ are not collinear, so that $x \neq \pi(R)x$ in $P(\mathbb{R}^3)$. That is, the solution sets of both problems are empty, and hence equal.

Note that Example 2.14 shows that the set of solutions of $(\text{Proj}_G)$ is in general not even a linear space, but rather a union of rays. It would be difficult to optimize over such a set, for instance in order to find an optimal neural network layer. In contrast, the solutions of $(\text{Lin}_H)$ are always a linear space – in fact, for any matrix group, we can in fact find the solutions by solving a finite dimensional linear equation system [10]. This further motivates us finding cases in which the two sets of problems are equivalent.

A fundamental difference between $\mathbb{Z}_n$ and $\text{SO}(3)$ is that $\mathbb{Z}_n$, for $n \geq 3$, is commutative, while $\text{SO}(3)$ is not. It turns out that the commutator $\{H,H\}$ of the group $H$, i.e. the group generated by the set of commutators $\{h,k\} = hh^{-1}kk^{-1}, h,k \in H$, in general plays a crucial role in the relation between $(\text{Proj}_G)$ and $(\text{Lin}_H)$. The commutator of a group can be seen as a measure of how commutative the group is – the smaller the commutator the more commuting elements the group contains. We can now formulate the crucial result of this section.

**Theorem 2.16.** Let $G$ be a group and $H$ a covering group of $G$ with covering map $\varphi$. Further, let $\rho: G \rightarrow \text{PGL}(V)$ be a projective representation of $G$ and $\hat{\rho}: H \rightarrow \text{GL}(V)$ a lift of $\rho$. Assume $v \in P(V)$ to be a solution of $(\text{Proj}_G)$. Then, every representing element $x$ of the equivalence class $v$ solves the following restricted linear invariance problem

$$\hat{\rho}(h)x = x \text{ for all } k \in \{H,H\}. \quad (\text{Lin}_{(H,H)})$$

**Proof.** If $v \in P(V)$ is a solution of $\rho(g)v = v$, every pair of representing elements $M$ of $\rho(g)$ and $x$ of $v$ must satisfy $Mx = \lambda(M,x,g)x$ for some $\lambda \in \mathbb{P}$. It is not hard to see that $\lambda$ only depends on $g$ and $M$. Now, for $h \in H$ arbitrary, since $\hat{\rho}$ is a lift of $\rho$, $\hat{\rho}(h)$ is a representing element of $\rho(\varphi(h))$. Hence, $(\text{Proj}_G)$ implies

$$\hat{\rho}(h)x = \lambda(\varphi(h),\hat{\rho}(h)M)x \text{ for all } h \in H$$

for some $\lambda(\varphi(h)) \in \mathbb{P}$. For simplicity, let us write $\lambda(\varphi(h),\hat{\rho}(h)M) = \lambda_h$. Now, $\hat{\rho}$ being a linear representation implies that

$$x = \hat{\rho}(h^{-1}h)x = \hat{\rho}(h^{-1})\hat{\rho}(h)x = \lambda_{h^{-1}}\lambda_hx,$$

so that $\lambda_{h^{-1}} = \lambda_h^{-1}$. For elements $\{H,H\} \ni k = hnh^{-1}m^{-1}$, we may now deduce

$$\hat{\rho}(k)x = \hat{\rho}(h)\hat{\rho}(m)\hat{\rho}(h^{-1})\hat{\rho}(m^{-1})x = \lambda_h\lambda_m\lambda_{h^{-1}}\lambda_{m^{-1}}x = x.$$

Since $\{H,H\}$ by definition is generated by the commutators, this proves the claim. \qed

This theorem looks simple, but it turns out to be enough to deduce strong statements for two very interesting examples: the group of rotations $\text{SO}(3)$ and the group of permutations $S_n$. 

---

In Search of Projectively Equivariant Neural Networks
2.2 The rotation group

The rotation group SO(3) is a perfect group, meaning that \( \{ \text{SO}(3), \text{SO}(3) \} = \text{SO}(3) \). For convenience, we provide a proof of this in Appendix A. SO(3) being perfect allows us to draw the following corollary directly from Lemma 2.13 and Theorem 2.16.

**Corollary 2.17.** Let \( V \) be a vector space and \( \rho : \text{SO}(3) \rightarrow \text{PGL}(V) \) be a projection of a representation \( \hat{\rho} : \text{SO}(3) \rightarrow \text{GL}(V) \). Then, a vector \( x \in V \) is a solution of \((\text{Lin}_{\text{SO}(3)})\) if and only if its equivalence class in \( \text{P}(V) \) is a solution of \((\text{Proj}_{\text{SO}(3)})\).

In the appendix, we prove a theorem which also also covers the case when the projective representation is not the projection of a linear one. The gist of the result is that we in this case can find the set of invariant vectors by solving a set of linear (and not projective, as \((\text{Proj}_{\mathcal{C}})\)) equations involving the Lie algebra of \( G \), as in [10]. Since even formulating these equations needs quite a lot of notation we have not used up until know, we postpone it in its entirety to Appendix B.

2.3 The permutation group

The permutation groups acts on vectors in \( \mathbb{P}^n \) by permuting indices, \( (\hat{\rho}(\pi)x)_i = x_{\pi^{-1}(i)} \). This action induces an action on the tensor product of \( \mathbb{P}^n \) with itself \( k \) times, \( (\mathbb{P}^n)^{\otimes k} \), through \( \hat{\rho}(\pi)(x_0 \otimes \cdots \otimes x_{k-1}) = \hat{\rho}(\pi)x_0 \otimes \cdots \otimes \hat{\rho}(\pi)x_{k-1} \) and linear continuation. For these representations, the solutions of \( (\text{Lin}_{S_n}) \) were characterised in [27]. In the follow up work [28], it was shown that for \( n \geq k + 2 \), the solutions are precisely the same as the ones for the group \( A_n \), the alternating group. This has an important consequence: Theorem 2.16 states that all solutions of \((\text{Proj}_{A_n})\) are solutions of \((\text{Lin}_{S_n})\). However, since \( \{S_n, A_n\} = A_n \), this means that they are solutions of \((\text{Lin}_{A_n})\), and therefore also of \((\text{Lin}_{S_n})\). The following corollary, which we prove in detail in Appendix A, holds.

**Corollary 2.18.** If \( n \geq k + 2 \), a tensor \( T \in (\mathbb{P}^n)^{\otimes k} \) is invariant under the linear action of \( S_n \) exactly when its equivalence class in \( \text{P}((\mathbb{P}^n)^{\otimes k}) \) is invariant under the projective action of \( S_n \). The bound \( n \geq k + 2 \) is tight.

3 Experiments: projective equivariance and scene rotations

Our goal from the beginning has been to develop a neural network tailored for pinhole projected keypoints. The relevant group action here is the action of SO(3) on 2D point clouds, as in Example 2.2. We call such rotations scene rotations. The straightforward approach would be to look at linear layers \( (\mathbb{R}^3)^m \rightarrow V \), where \( V \) is some latent vector space, such that the resulting projective map \( \text{P}((\mathbb{R}^3)^m) \rightarrow \text{P}(V) \) is equivariant. Corollary 2.17 shows that in this case, projectively equivariant layers will not be more expressive than ordinary linear ones. Consequently, we use models in our experiments that are equivariant in the ordinary sense.

For our equivariant models, we will use EGNNs [35]. In short, they are message passing networks that take as input a graph, where each node has both coordinates in \( \mathbb{R}^D \) and abstract features \( h_i \in \mathbb{R}^{C_a} \). These are processed, in a message passing fashion, to produce new equivariant coordinates and invariant features. In our experiments, we can feed point clouds into the EGNN with either 2D coordinates – for equivariance to SO(2) – or 3D coordinates (projections to the unit sphere) – for equivariance to SO(3). For more details, we refer to Section C in the appendix.

3.1 Robust line extraction

Let us first consider the line fitting problem described in Example 2.2. To make the task non-trivial, we consider the case of point-clouds \( X \in \text{P}(\mathbb{R}^3) \) containing outliers, i.e., points that are not on the line. We perform an experiment to compare equivariant to nonequivariant models, trained with augmentation by scene rotations of different magnitudes. The equivariant models are EGNNs, and the nonequivariant are ordinary message passing networks. Details on the models are presented in Appendix D.

**Data.** Data is generated on the fly, similar to [37]. Each data example consists of a 2D point cloud in \([-1, 1]^2\) with a noisy line, as in Figure 1. The regression target is the line vector \( w \) normalized to unit length. Details on the data generation are presented in Appendix D. If a point \( x_i \) in an input cloud is given by \([y_i, 1] \in \mathbb{R}^2 \) in homogeneous coordinates, the input to the equivariant models is given
Figure 2: Linefitting results for an equivariant model and non-equivariant models with varying amounts of rotation augmentation. Scores are averages over 10 training runs, lower is better. The $x$-axis displays the amount of test time rotations in degrees.

by $\frac{x_i}{\|x_i\|}$, and the input to the non-equivariant models is $y_i$. The clouds are converted to $k$-nearest neighbour graphs, where $k = 24$ in our experiments.

**Data augmentation.** We use the same augmentation scheme for training as well as generating more difficult test data by applying random scene rotations. The size of the out-of-plane part of these is chosen uniformly in $[0, \theta_{\text{max}}]$, where we vary $\theta_{\text{max}}$. We also renormalize the clouds we feed into the non-equivariant architectures to avoid clouds with points too far away from the origin. For more details, see Appendix D.

**Metric.** After every epoch of 2500 training examples, we test the models by averaging the projective $\ell^2$-distance, $\min_{\epsilon \in \{+1, -1\}} \|\epsilon w - w^0\|$, between the estimate and ground truth over 1000 examples augmented by rotations with $\theta_{\text{max}}$ set to 0°, 15°, 30° and 45°, respectively. Due to the stochastic nature of this performance measure, we give a model a score equal to its 90th percentile over all epochs. We train for 100 epochs.

**Results.** We consider three model sizes: 3, 4 and 5 message passing layers and repeat each experiment 10 times. The results are presented in Figure 2 and in more detail in Appendix D.1. The $\theta_{\text{max}} = 45^\circ$ model failed completely – that level of augmentation turned out to be too aggressive. This is not surprising, since this augmentation indeed produces some degenerate data points, in particular with a few outliers far away from the origin in 2D.

Excluding the broken 45°-model, the non-equivariant models do excel on the less diverse dataset, with heavier augmentation leading to better performance. They however get worse on the more diverse ones, and degrade so much that the equivariant model performs best for the 45°-test, despite having only seen non-augmented examples during training. This shows that for certain extreme datasets, the equivariant model can achieve generalization performance that we cannot obtain through (simple) training data augmentation of a simple model.

### 3.2 2D-2D correspondence filtering

In this section we will look at the problem of outlier filtering a set of given keypoint correspondences between two images of the same scene. A correspondence $(x, y)$ of one point in each image is an inlier if $x$ and $y$ correspond to projections of the same 3D point. See Figure 5 in the appendix for an illustration of correspondence data. Assuming that the given keypoints are calibrated, the inliers fulfil the *essential matrix equation* \( y^T Ex = 0 \). From a set of inliers, the essential matrix $E \in \mathbb{R}^{3 \times 3}$ can be estimated e.g. by finding the least squares solution to this equation. From $E$, the relative position of the cameras taking the two images can then be computed (the translation only up to scale), see [15] for details. We will follow the line of work starting with [43] and continuing with [37, 45, 46] and use a weighted least squares formulation:

$$ E = \arg \min_{\|E\| = 1} \sum_i w_i \left( y_i^T \tilde{E} x_i \right)^2, \quad (4) $$

where $w_i \in [0, 1]$ are inlier weights predicted by the neural network and $i$ ranges over all input correspondences. $w_i = 0 / w_i = 1$ means that $i$ is an outlier/inlier, respectively. Our aim will be to create a network estimating the $w_i$ that is invariant to scene rotations. We use SIFT [26] keypoint detection/description and nearest neighbour matching of the descriptors to generate the correspondences in our experiments, following the implementation of [45].
Related work. PointCN [43] uses a permutation equivariant MLP together with so-called context normalization. The latter refers to normalizing the features of each cloud in each layer to have zero mean and unit variance. In aCNE [37], this normalization is done in a weighted sense, where the weights are adaptively calculated for each layer. Another direction of improvement was taken by OANet [45] which improved on PointCN by introducing order-aware pooling and unpooling layers, allowing the network to cluster correspondences and better learn local context. OANet also (as did the concurrent [33]) introduced the technique of calculating an estimate of $E$ in each layer and feeding the resulting residuals together with estimated inlier weights iteratively into new layers. T-Net [46] improved on OANet by introducing squeeze-and-excitation blocks [17] into the architecture and by concatenating all the output features of each iteration (correspondence wise) into large features in a final step, processing these large features in a network dubbed the “|”-structure to obtain final inlier weights and a final $E$ matrix. The backbone OANet (with added squeeze-and-excitation) is instead referred to as “–”-structure.

We will build on T-Net in our experiments. The “|”-structure will remain the same as in T-Net, but we will modify the “–”-structure. First, we will try replacing each T-Block with an EGNN. Second, we will add invariant information to the ‘vanilla’ T-Net with the hope that the network can learn how to best use this information.

GraT-Net – Using equivariant graph networks. We build a graph on the input correspondences by adding edges for each correspondence to the $K$ nearest neighbours in both images. Since we want to have inlier weights that are invariant w.r.t. rotations of both cameras/images, we proceed as follows, similar to [3]: We first feed each cloud individually through an EGNN network to produce invariant features $h_{x}$ and $h_{y}$, respectively. Then for each correspondence $(x_i, y_i)$, we max-pool the features to obtain a feature for the correspondence: $h_i(c) = \max(h_{x_i}(c), h_{y_i}(c))$, where $c$ ranges over the dimensions of the abstract feature $h$. The abstract output feature of the last EGNN layer is taken as inlier weight $w_i$. The construction is detailed in Appendix E.2.

Our first implementation used constant input features in the first layer in order to guarantee perfect rotation equivariance\(^3\). This approach turned out to work extremely poorly, likely because the absolute coordinates of the correspondences carries quite a lot of information that is lost in the equivariant model. In a second attempt, we break the equivariance by using the four coordinates of each correspondence as abstract input features at each graph node. This worked better, but the performance is still far from T-Net – in part due to the fact that we used smaller networks in order to be able to fit the network on a single GPU. T-Net is more lightweight since it does not contain graph layers – it uses no edge information. We refer to our models as GraT-Net2 and GraT-Net3, for the 2D and 3D coordinate cases respectively (Gra for Graph). The fully equivariant graph networks without any coordinate information will be called GraT-Net2-- and GraT-Net3--. More details on how we construct and train the networks is deferred to Appendix E.2.

ET-Net – Adding invariant input features to T-Net. In a final experiment, we use minimally modified T-Net: We simply feed it, along the already used four coordinates per correspondence, with extra rotation invariant input features. Using the graph structure defined above, we define an initial feature in each node $i$ given as the set of distances, in the each image, to its neighbouring correspondences. are then fed through a two layer MLP to produce the rotation invariant features. We use this approach with both 2D and 3D features, in order to test whether theoretically more appealing 3D features have an actual advantage over the simpler 2D baseline. We refer to these new networks as ET-Net2 and ET-Net3 (E for Extra or Equivariant). Again, more details for the construction and training of the networks can be found in Appendix E.3. Using local geometric features in the correspondence filtering problem has been done in earlier work [25, 31], but our approach is conceptually simpler and the first to use 3D information.

Experiment details. For our experiments we use the YFCC100M dataset [39] which was processed by [16] into 72 sequences of images of tourist attractions and following [45] we reserve 4 of these for testing and use the rest for training. We use the public code of [45] to generate a dataset consisting of SIFT nearest-neighbour correspondences between image pairs and ground truth relative positions based on the structure-from-motion pipeline from [16]. We retrained T-Net to the best of our ability

\(^3\)A good alternative would likely be to use the SIFT-descriptors that were used to match the correspondences, but as we want to compare to T-Net, we stick to the ‘pure’ geometric data only. For a graph network that does use the SIFT-descriptors for both matching and filtering, but is much heavier computationally than T-Net, see SuperGlue [34].
using the official code, with slight modifications to make it run, and include the results from our retraining as well as the results reported in [46]. The authors do not provide pretrained weights. Upon publication we will make our code public, as well as the trained weights. We did three reruns of the ET-Net and T-Net training and report the median scores in Table 1. Due to the very unpromising performance of the GraT-Net models, we chose not to rerun them.

The performance is measured via comparing the relative position of the cameras calculated from the estimated $E$ to the ground truth. As in earlier work, we evaluate both directly estimating $E$ from the predicted inliers by a least squares approach, as well as instead using RANSAC with a minimal solver on the set of predicted inliers (i.e. correspondences with positive weights), which will filter out further remaining outliers and hence lead to better scores.

Testing equivariance. To measure the level of equivariance of the networks, we perform a second test, where the models are tested on data on which scene rotations with a random angle in $(0^\circ, 15^\circ)$, about a random axis are applied. This yields a more difficult test set, as the camera poses in the data set are typically fairly gravity aligned.

Results and conclusions from experiments. Results from the outlier filtering experiments can be found in Table 1. We use the same AUC@$\alpha$-metric as [43, 45, 46], which is a measure of how many predicted relative rotations and translations are within a certain angle $\alpha$ from the ground truth.

We see in Table 1 that ET-Net3 is the best performing model. In particular it outperforms ET-Net2, indicating that 3D-information is more useful than 2D-information for the task. However, when looking at the version of the test data with camera poses rotated up to $15^\circ$ about an arbitrary axis, we see that ET-Net3 and ET-Net2 are no more robust to these deviations from the training data, than T-Net. The improved performance hence does not seem to stem from improved rotation equivariance. The only models that are rotation equivariant are GraT-Net2-- and GraT-Net3-- which perform very poorly overall.

4 Conclusion

In this paper, we considered projective equivariance in neural networks. From a theoretical point of view, we studied the relation between projective and linear equivariance for neural network linear layers. Our main finding is that these problems in many important cases are equivalent. On the experimental side, we compared the performance of equivariant and non-equivariant networks, as well as non-equivariant ones explicitly working on invariant features. We did find a case where the equivariant model outperformed its non-equivariant competitor in the very diverse test data sets on robust line-fitting experiment. In the practically more relevant 2D-2D-correspondence experiment, we did manage to improve a competitive baseline, but did not find strong evidence of projective rotation equivariance being beneficial.

---

\[\text{Table 1: Results for relative pose regression from outlier ridden sets of correspondences between images from the YFCC100M dataset. The numbers are percentages and we report AUC scores (w/o RANSAC)/(w/ RANSAC). Results for ET-Net and T-Net (retrained) are median values from three training runs, the rest (including T-Net (paper)) are single runs.}\]

| Method | AUC @ $\alpha$ | Test time rotation $0^\circ$ | Test time rotation $15^\circ$ |
|--------|----------------|-----------------|-----------------|
| RANSAC | 8.5/113.8/18.4 | 22.3/8.5/13.9 | 22.4/13.8/18.5 |
| GraT-Net2-- | 2.5/6.0/10.5/14.4 | 13.7/9.9/6.0/14.0 | 17.0/13.7/9.9/6.0 |
| GraT-Net3-- | 2.5/6.0/10.5/14.4 | 13.7/9.9/6.0/14.0 | 17.0/13.7/9.9/6.0 |
| GraT-Net2 | 25.9/51.0/61.0 | 74.5/58.9/69.2 | 71.6/55.8/67.3 |
| GraT-Net3 | 26.0/51.0/61.0 | 74.5/58.9/69.2 | 71.6/55.8/67.3 |
| ET-Net2 | 47.9/56.2/61.6 | 74.5/58.9/69.2 | 66.5/51.0/64.7 |
| ET-Net3 | 49.7/57.1/62.8 | 74.5/58.9/69.2 | 66.5/51.0/64.7 |
| T-Net (retrained) | 47.9/56.2/61.6 | 74.5/58.9/69.2 | 66.5/51.0/64.7 |
| T-Net (paper) | 52.3/56.2/61.6 | 74.5/58.9/69.2 | 66.5/51.0/64.7 |

4 We report the the maximum scores in Appendix E.1, Table 7. None of our three T-Net reruns provided as good performance as the single run reported by [46].
Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE) partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

References

[1] Jimmy Aronsson. “Homogeneous vector bundles and G-equivariant convolutional neural networks”. In: arXiv:2105.05400 [cs, math, stat] (May 2021). arXiv: 2105.05400. URL: http://arxiv.org/abs/2105.05400.

[2] Erik J. Bekkers et al. “Roto-Translation Covariant Convolutional Networks for Medical Image Analysis”. In: arXiv:1804.03393 [cs, math] (June 2018). arXiv: 1804.03393. URL: http://arxiv.org/abs/1804.03393.

[3] Georg Bökman, Fredrik Kahl, and Axel Flinth. “ZZ-net: A universal rotation equivariant architecture for 2d point clouds”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022, pp. 10976–10985.

[4] Michael M. Bronstein et al. “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges”. In: arXiv:2104.13478 [cs, stat] (May 2021). arXiv: 2104.13478. URL: http://arxiv.org/abs/2104.13478.

[5] Brian C. Hall. “Lie Groups, Lie Algebras, and Representations: An Elementary Introduction”. In: Cham: Springer International Publishing, 2015, pp. 49–76. ISBN: 978-3-319-13467-3. URL: https://arxiv.org/pdf/math-ph/0005032.pdf.

[6] Gabriele Cesa, Leon Lang, and Maurice Weiler. “A Program to Build E(N)-Equivariant Steerable CNNs”. en. In: Sept. 2021. URL: https://openreview.net/forum?id=WE4qe9xInQw&referrer=the%20profile%20of%20%20Gabriele%20Cesa}/profile?id=Gabriele_Cesa.2

[7] Taco Cohen, Mario Geiger, and Maurice Weiler. “A General Theory of Equivariant CNNs on Homogeneous Spaces”. In: arXiv:1811.02017 [cs, stat] (Jan. 2020). arXiv: 1811.02017. URL: http://arxiv.org/abs/1811.02017.

[8] Taco S. Cohen and Max Welling. “Group Equivariant Convolutional Networks”. In: arXiv:1602.07576 [cs, stat] (June 2016). arXiv: 1602.07576 [cs, stat].

[9] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[10] Marc Finzi, Max Welling, and Andrew Gordon Wilson. “A Practical Method for Constructing Equivariant Multilayer Perceptrons for Arbitrary Matrix Groups”. In: Proceedings of the 38th International Conference on Machine Learning. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 3318–3328. URL: https://proceedings.mlr.press/v139/finzi21a.html.

[11] Kunihiko Fukushima and Sei Miyake. “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition”. In: Competition and cooperation in neural nets. Springer, 1982, pp. 267–285.

[12] Jan E. Gerkens et al. “Geometric Deep Learning and Equivariant Neural Networks”. In: arXiv:2105.13926 [hep-th] (May 2021). arXiv: 2105.13926. URL: http://arxiv.org/abs/2105.13926.

[13] Justin Gilmer et al. “Neural message passing for quantum chemistry”. In: International conference on machine learning. PMLR, 2017, pp. 1263–1272.

[14] Brian C. Hall. Quantum Theory for Mathematicians. Graduate Texts in Mathematics. Springer, 2013.
[36] David Silver et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. In: Science 362.6419 (2018), pp. 1140–1144. 1

[37] Weiwei Sun et al. “Acne: Attentive context normalization for robust permutation-equivariant learning”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, pp. 11286–11295. 6–8, 21–23

[38] Nathaniel Thomas et al. “Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds”. In: arXiv:1802.08219 [cs] (May 2018). arXiv: 1802.08219. URL: http://arxiv.org/abs/1802.08219. 2

[39] Bart Thomée et al. “YFCC100M: the new data in multimedia research”. en. In: Communications of the ACM 59.2 (Jan. 2016), pp. 64–73. ISSN: 0001-0782, 1557-7317. DOI: 10/gg5vx8. URL: https://dl.acm.org/doi/10.1145/2812802. 8

[40] Minjie Wang et al. “Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks”. In: arXiv preprint arXiv:1909.01315 (2019). 20, 26

[41] Maurice Weiler and Gabriele Cesa. “General E(2)-Equivariant Steerable CNNs”. In: Advances in Neural Information Processing Systems. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/45d6637b718d0f24a237069fe41b0db4-Paper.pdf. 2, 4

[42] Jeffrey Wood and John Shawe-Taylor. “Representation theory and invariant neural networks”. en. In: Discrete Applied Mathematics 69.1-2 (Aug. 1996), pp. 33–60. ISSN: 0166218X. DOI: 10 / c3qmr6. URL: https://linkinghub.elsevier.com/retrieve/pii/0166218X95000753. 1, 2

[43] Kwang Moo Yi et al. “Learning to find good correspondences”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, pp. 2666–2674. 7–9, 22

[44] Manzil Zaheer et al. “Deep Sets”. In: arXiv:1703.06114 [cs, stat] (Apr. 2018). arXiv: 1703.06114. URL: http://arxiv.org/abs/1703.06114. 2

[45] Jiahui Zhang et al. “Learning Two-View Correspondences and Geometry Using Order-Aware Network”. In: arXiv:1908.04964 [cs] (Aug. 2019). arXiv: 1908.04964. URL: http://arxiv.org/abs/1908.04964. 7–9

[46] Zhen Zhong et al. “T-Net: Effective Permutation-Equivariant Network for Two-View Correspondence Learning”. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). Oct. 2021, pp. 1950–1959. URL: https://openaccess.thecvf.com/content/ICCV2021/papers/Zhong_T-Net_Effective_Permutation-Equivariant_Network_for_Two-View_Correspondence_Learning_ICCV_2021_paper.pdf. 2, 7–9, 24

12
In Search of Projectively Equivariant Neural Networks

| Symbol | Description |
|--------|-------------|
| P(V)   | Projective space of V, i.e. V/(F \ \{0\}) |
| Hom(V, W) | Space of linear maps from V to W |
| PGL(V) | Projective general linear group of V |
| \[n\] | \{0, 1, \ldots, n-1\} |
| \(\mathcal{A}_n\) | Permutations of signature 1 |
| SU(2)  | Unitary matrices in \(\mathbb{C}^2\) |
| \(\Pi_V\) | Projection map from V to P(V) |
| GL(V) | General linear group of V |
| \(\tilde{\rho}\) | Linear representation |
| \(\varphi\) | Group covering map |
| \(S_n\) | Permutation group of n elements |
| SO(3) | Rotation group in 3D |
| \(\text{id}\) | The identity matrix |

Table 2: Symbol glossary

A Omitted proofs

Here, we collect some proofs we left out in the main text.

A.1 Lemma 2.8

Proof of Lemma 2.8. Equivariance of the map \(A\) means that for all \(g \in G\) and \(v \in P(V)\), we have \(\rho_1(g)Av = A\rho_0(g)v\). Since all \(\rho_0(g)\) are invertible, this is equivalent to

\[\rho_1(g)A\rho_0(g)^{-1}v = Av\text{ for all }v \in V.\]

Now, the above equality is not an equality of elements in \(W\), but rather of elements in \(P(W)\). That is, it says that all vectors \(v \in V\) are solutions of the generalized eigenvalue problem \(\rho_1(g)A\rho_0(g)^{-1}v = \lambda Av\). The aim is to show that this implies that \(\rho_1(g)A\rho_0(g)^{-1} = \lambda A\) for some \(\lambda \in \mathbb{F}\). To show this, we proceed in two steps.

Claim Let \(M \in GL(V)\). If every vector \(v \in V\) is an eigenvector of \(M\), \(M\) is a multiple of the identity.

Proof Suppose not. Then, since all vectors are eigenvectors, there exists \(v \neq w \neq 0\) and \(\lambda \neq \mu\) with \(Mv = \lambda v, Mw = \mu w\). Now, again since all vectors are eigenvectors of \(M\), there must exist a third scalar \(\sigma\) with \(M(v+w) = \sigma(v+w)\). Now,

\[\sigma(v+w) = M(v+w) = Mv + Mw = \lambda v + \mu w \iff (\sigma - \mu)w = (\mu - \sigma)v.\]

Now, the final equation can only be true if \(\lambda - \mu = \mu - \sigma = 0\), i.e. \(\lambda = \mu = \sigma\), which is a contradiction.

Claim Let \(E, F \in Hom(V, W)\). If every \(v \in V\) is a solution of the generalized eigenvalue problem \(Ev = \lambda Fv\), \(E\) is a multiple of \(F\).

Proof First, \(E\) restricted to \(\ker F\) must be the zero map, since for all \(v \in \ker F\), \(Ev = \lambda Fv = \lambda \cdot 0 = 0\). Secondly, \(F^\circ : \ker F^\perp \to \text{ran } F^\perp\) is an isomorphism, so that the generalized eigenvalue problem of \(Ev = \lambda Fv\) on \(\ker F^\perp\) is equivalent to the eigenvalue problem of \(E(F^\circ)^{-1} : \text{ran } F \to \text{ran } F\). By the previous claim, \(E(F^\circ)^{-1} = \lambda \text{id}\) for some \(\lambda \in F\), i.e. \(E = \lambda F^\circ\) on \(\ker F^\perp\). Since both \(E\) and \(F\) additionally are equal to the zero map on \(\ker F\), the relation \(E = \lambda F\) remains true also there. The claim has been proven.

A.2 Lemma 2.13

Proof of Lemma 2.13. It is clear that \(\Pi_V(Mv) = \Pi_{GL(V)}(M)\Pi_V(v)\) for \(M \in GL(V)\) and \(v \in V\). Therefore, \((\text{Lin}_H)\) implies

\[\Pi_V(\chi) = \Pi_V(\tilde{\rho}(h)\chi) = \Pi_{GL(V)}(\tilde{\rho}(h))\Pi_V(\chi) = \rho(\varphi(h))\Pi_V(\chi)\text{ for all }h \in H,\]

where we used that \(\tilde{\rho}\) lifts \(\rho\) in the final step. It is now only left to note that \(\varphi\) is surjective to deduce that the above in fact implies \((\text{Proj}_G)\).

A.3 Induced projected representations

Let us here, for completeness, remark and prove a small statement that we use implicitly throughout the entire article.
Proposition A.1. Let $G$ be a group, $V$ and $W$ vector spaces, $\hat{\rho}_V$ and $\hat{\rho}_W$ be linear representations of $G$ on $V$ and $W$, respectively, and $\rho_V$ and $\rho_W$ their corresponding projected representations. Then, the induced projective representation $\rho$ on $\text{Hom}(V, W)$

$$\rho(g)A = \rho_W(g) \circ A \circ \rho_V(g)^{-1}$$

is the projection of the corresponding induced linear projection

$$\hat{\rho}(g)A = \hat{\rho}_W(g) \circ A \circ \hat{\rho}_V(g)^{-1}.$$  

Proof. What we need to prove is that if $M$ is equivalent to $\hat{\rho}_W(g)$ and $N$ is equivalent to $\hat{\rho}_V(g)$, the linear map

$$R_{MN} : \text{Hom}(V, W) \rightarrow \text{Hom}(V, W), A \mapsto MAN^{-1}$$

is equivalent to $\hat{\rho}(g)$. However, the stated equivalences mean that $M = \lambda \hat{\rho}_W(g)$ and $N = \mu \hat{\rho}_V(g)$ for some non-zero $\mu, \lambda$. This in turn shows that

$$R_{MN}(A) = \lambda \hat{\rho}_W(g)A(\mu \hat{\rho}_V(g))^{-1} = \lambda \mu^{-1} \hat{\rho}(g)A,$$

i.e., that $R_{MN}$ is equivalent to $\hat{\rho}(g)$. The claim has been proven. \hfill $\Box$

A.4 Group theoretic facts

Here we, out of convenience for the reader, present proofs of two well-known facts.

Lemma A.2. $\text{SO}(3)$ is perfect.

Proof. Let $g \in \text{SO}(3)$ be arbitrary rotation, say of an angle $\theta \in [0, 2\pi]$ around an axis $v$. Let $h$ be the rotation about the same axis but of angle only $\omega = \theta/2$, so that $h^2 = g$. Now, let $k$ be a rotation with $kv = -v$. We claim that the rotation $\ell = kh^{-1}k^{-1}$ is equal to $h$. To notice this, let $w = v \times w$. Then, $k(w) = 2$ and $k(u) = -u$. Also, $w$ and $u$ are perpendicular to $v$, so that $h(w) = \cos(\theta)w \pm \sin(\theta)u$ and $h(u) = \cos(\theta)u \mp \sin(\theta)w$, without loss of generality $h(w) = \cos(\theta)w + \sin(\theta)u$. Then

$$\ell(v) = kh^{-1}k^{-1}(v) = kh^{-1}(-v) = k(-v) = v$$

$$\ell(w) = kh^{-1}k^{-1}(w) = kh^{-1}(w) = k(\cos(\theta)w - \sin(\theta)u) = \cos(\theta)w + \sin(\theta)u$$

$$\ell(u) = kh^{-1}k^{-1}(u) = kh^{-1}(-u) = k(-\cos(\theta)u - \sin(\theta)w) = \cos(\theta)u - \sin(\theta)w,$$

i.e., $\ell = h$. This implies that $g = h^2 = h\ell = hkh^{-1}k^{-1} \in \{\text{SO}(3), \text{SO}(3)\}$, and since $g$ was arbitrary, the claim. \hfill $\Box$

Lemma A.3. $\{S_n, S_n\} = A_n$.

Proof. This proof is also well known, and can be found in e.g. [29]. We again include it for convenience.

That $\{S_n, S_n\} \subseteq A_n$ follows from the fact that all commutators have signature 1:

$$\sigma(\tau \circ \pi \circ \tau^{-1} \circ \pi^{-1}) = \sigma(\tau) \sigma(\pi) \sigma(\tau)^{-1} \sigma(\pi)^{-1} = 1.$$

To prove the converse, let us first note that the cases $n = 1, 2$ are trivial – in both those cases, $A_n = \{\text{id}\}$, but $S_n$ is also abelian, so that $\{S_n, S_n\} = \{\text{id}\}$. For $n \geq 3$, we use the well-known fact that $A_n$ is generated by 3-cycles [23, Cor. 6.3] $(ijk)$. If we show that they are commutators, we are hence done. But

$$(ijk) = (jk)(ij)(jk)(ij),$$

so that the claim follows. \hfill $\Box$
A.5 Corollary 2.18

Proof of Corollary 2.18. The proof of the first part was in essence presented already in the main text, but let us carry out the details: First, that equivalence classes of solutions of \((\text{Lin}_{S_n})\) are solutions of \((\text{Proj}_{S_n})\) is Lemma 2.13. As for the other direction, Theorem 2.16 states that if \(x\) is in an equivalence class of a solution of \((\text{Proj}_{S_n})\), it is also a solution of \((S_n) = (\text{Lin}_{A_n})\), where the second equation is due to Lemma A.3. Now, we apply the results of [28, Prop. 6], which says that the solutions of \((\text{Lin}_{A_n})\) and \((\text{Lin}_{S_n})\) are equal.

It remains to show that the bound \(n \geq k + 2\) is tight. For \(k\) arbitrary, set \(n = k + 1\) and let \(I = (i_0, \ldots, i_{k-1})\) be a \(k\)-tuple. There are two alternatives: either at least two indices in \(I\) are equal, or we can define a permutation \(\pi_I \in S_n\) by setting \(\pi_I(\ell) = i_\ell\) for all \(\ell \in [k]\) and letting \(\pi_I(k)\) be the index in \([n]\) that is not equal to \(i_\ell\) for any \(\ell\). Now define a tensor \(T\) through

\[
T_I = \begin{cases} 
\text{sign}(\pi_I) & \text{if all indices in } I \text{ are different.} \\
0 & \text{else.}
\end{cases}
\]

We now show that for \(\sigma \in S_n\) arbitrary, \(\hat{\rho}(\sigma)T = \text{sign}(\sigma)(T)T\). To do so, notice that a \(k\)-tuple \(I\) has all indices different if and only if \(\sigma^{-1}(I)\) has. Thus \((\hat{\rho}(\sigma)T)_{I} = 0\) if and only if \(T_{I}\) is.

Now, if \(I\) does have all indices different, let \(J = (j_0, \ldots, j_{k-1}) = \sigma^{-1}(I) = (i_{\sigma^{-1}(0)}, \ldots, i_{\sigma^{-1}(k-1)})\) and \(\pi_J, \pi_I\) be defined as above. Then, \(\pi_J = \pi_I \circ \sigma^{-1}\), since

\[
(\pi_I \circ \sigma^{-1})(\ell) = \pi_I(\sigma^{-1}(\ell)) = i_{\sigma^{-1}(\ell)} = j_\ell.
\]

Hence,

\[
(\hat{\rho}(\sigma)T)_I = T_{\sigma^{-1}(I)} = T_J = \text{sign}(\pi_J) = \text{sign}(\pi_I \circ \sigma^{-1}) = \text{sign}(\pi_I)\text{sign}(\sigma^{-1}) = \text{sign}(\sigma)T_I.
\]

Since there exists \(\pi\) with \(\text{sign}(\pi) = -1\) \((n \geq k + 2 \geq 2)\), we conclude that \(T\) solves \((\text{Proj}_{S_n})\), but not \((\text{Lin}_{S_n})\).

\[\square\]

Remark A.4. Although we only need that they are equal for the above proof, let us for completeness describe the solutions of \((\text{Lin}_{S_n})\) and \((\text{Lin}_{A_n})\) following [27]. Define two indices \(I, J \in [n]^{k}\) as having the same equality pattern if \(i_\ell = j_\ell\) if and only if \(j_\ell = j_\ell\). The solutions of \((\text{Lin}_{S_n})\) are then tensors which are constant on the equivalence classes under the equality pattern relation.

A.6 Linear maps on \((\mathbb{R}^3)^m\) are almost never well defined on \(P(\mathbb{R}^3)^m\)

In the main text, we claimed that linear maps on \((\mathbb{R}^3)^m\) very seldom induce well-defined maps on \(P((\mathbb{R}^3)^m)\). Here, we state and prove a formal statement. First, let us note that we can view \(P((\mathbb{R}^3)^m)\) as a quotient of \(P(\mathbb{R}^3)^m)\) under the equivalence relation

\[
X \sim Y \iff \exists \lambda \in \mathbb{R}^m : x_i = \lambda_i y_i.
\]

Let \(V\) be a vector space and \(L : (\mathbb{R}^3)^m \to V\) a linear map. This map induces a map \(L_P : P((\mathbb{R}^3)^m) \to P(V)\). \(L\) is well-defined in \(P((\mathbb{R}^3)^m)\) if \(L_P(X)\) only depends on the equivalence class of \(X\) under the equivalence relation (5). We now show that this happens exactly when either \(L_P\) has a one-dimensional range, or only depends on one of the points.

Proposition A.5. \(L_P\) is well-defined if and only if \(L\) has a one-dimensional range, or is of the form \(L(X) = \lambda(x_i)\) for some \(i \in [m]\) a linear map \(\lambda : \mathbb{R}^3 \to V\).

Proof. It is clear that every linear map \(L\) can be written as

\[
L(X) = \sum_{i \in [m]} \lambda_i(x_i)
\]

for some \(\lambda_i\). It is clear that if all but one of these are the zero map, \(L_P(X)\) is the same for all \(X\) in the same equivalence class. Also, if \(L\) has a one-dimensional range, \(L(X)\) is the same element in \(P(V)\) for all \(X\), and \(L_P\) is well defined.
This means that we may calculate
\[
\begin{align*}
\text{whereby the final quotient is a quotient of vector spaces.}
\end{align*}
\]
where the inequality is meant in a \(P(V)\)-sense.

\[L_P(X^1) = \lambda(x_i) + \lambda(x_j) \neq \lambda(x_i) + 2\lambda(x_j) = L_P(X^2),\]

where the inequality is meant in a \(P(V)\)-sense.

## B Lie Groups with Universal Covers

In this section, we would like to take a special look at the case of \(G\) being a Lie group (and in particular the underlying field being either \(\mathbb{R}\) or \(\mathbb{C}\)). Lie groups come with an additional smooth structure, which we can use to reformulate our theorems from the main text. The main aim of this section is to prove that if \(G\) admits a universal cover \(H\), projectively invariant elements necessarily fulfills a set of linear equations, just as in Theorem 2.16. Importantly, these linear equations can be found without explicit knowledge of \(H\), and in contrast to the ones in Theorem 2.16 does not involve a lifted representation \(\bar{\rho}\).

### B.1 \(\text{PGL}(V)\) as a Lie Group

To start, let us say some words about the set \(\text{PGL}(V)\). It surely is a group, but in fact even a Lie Group, i.e., a manifold. To see this, we use the Quotient Manifold Theorem [24, Th. 21.10], which says that if a Lie group \(K\) acts smoothly, freely and properly on a manifold \(M\), the space of equivalence classes \(M/K\) is a manifold, with a unique smooth structure that makes the projection \(\pi : M \to M/K\) a submersion. In our case, \(M = \text{GL}(V)\) is surely a manifold, on which the Lie Group \(\mathbb{F}\) is acting smoothly, freely and properly. Thus, \(\text{PGL}(V) = \text{GL}(V)/\mathbb{F}\) is a quotient manifold. Its tangent space at the unit element, i.e. its Lie Algebra, is

\[
\text{pgl}(V) = \text{gl}(V)/\ker\pi = \text{gl}(V)/\text{span}(id),
\]

whereby the final quotient is a quotient of vector spaces.

The Lie algebra \(\text{pgl}(V)\) comes equipped with a bracket \([\cdot,\cdot]_{\text{pgl}}\). Let us quickly prove that we may identify that bracket with the (commutator) bracket on \(\text{gl}(V)\).

**Lemma B.1.** For \(\hat{A}, \hat{B} \in \text{gl}(V)\), we have

\[
[\pi(\hat{A}), \pi(\hat{B})]_{\text{pgl}} = \pi([A, B]_{\text{gl}}).
\]

This means that we may calculate \([C, D]_{\text{pgl}}\) for \(C, D \in \text{pgl}(V)\) by simply calculating \([A, B]_{\text{gl}}\) for any \(A, B\) in the respective equivalence classes.

**Proof.** Let’s first recall some general Lie group theory. For an element \(g \in G\), we define the conjugation \(C(g) : G \to G, h \mapsto ghg^{-1}\). The differential of this map at \(e \in G\), \(dC(g) : g \to g\) is called \(\text{Ad}(g)\), and \(\text{Ad} : G \to \text{GL}(g)\) is the adjoint representation of \(G\) on \(g\). We may again consider the differential of \(\text{Ad}\) at \(e\) to define a map \(d\text{Ad}(e) = \text{ad} : g \to \text{End}(g)\). It turns out that (see [24, Th. 20.27]) \(\text{ad}\) is the adjoint (Lie Algebra) representation of \(g\) on itself:

\[
\text{ad}(X)Y = [X, Y].
\]

Now let us study these relations for the two groups \(\text{PGL}(V)\) and \(\text{GL}(V)\). We equip all maps on \(\text{GL}(V)\) with a circumflex – e.g., \(\hat{C}\) refers to the conjugation on \(\text{PGL}(V)\), whereas \(\hat{C}\) refers to the one on \(\text{GL}(V)\). This already allows us to write down the most important relation in this proof

\[
C_{\pi(M)} \circ \pi = \pi \circ \hat{C}_{\hat{M}} \text{ for all } \hat{M} \in \text{GL}(V).
\]
This relation is less trivial than it looks, and relies on the fact that \( \mathbb{F} \) commutes with all elements in \( \text{GL}(V) \). To be concrete, for any elements \( \lambda M, \mu N \) the \( \text{PGL}(V) \)-equivalence classes of \( M \) and \( N \), we have
\[
(\lambda M)\mu N (\lambda M)^{-1} = \mu M N M^{-1} = \mu \hat{C}_M(\hat{N}),
\]
which exactly means (6). Differentiating (6) at \( \text{id} \) reveals
\[
\text{Ad}_{\pi(\hat{M})} \circ \text{id} = d\pi \circ \hat{A}_d\hat{M} \text{ for all } \hat{M} \in \text{GL}(V).
\]
Differentiating again implies
\[
\text{ad}(d\pi \hat{A})(d\pi \hat{B}) = d\pi(\hat{A} \hat{B}) \text{ for all } \hat{A}, \hat{B} \in \text{gl}(V)
\]
\[
\iff [d\pi \hat{A}, d\pi \hat{B}]_{pgl} = d\pi([\hat{A}, \hat{B}]_{gl(V)}) \text{ for all } \hat{A}, \hat{B} \in \text{gl}(V),
\]
which was the claim.

\[\square\]

### B.2 Lie algebra representations

In [10], the authors use the *induced Lie algebra representation* of a linear representation \( \hat{\rho} \) to simply the problem of solving the invariance relations \( (\text{Lin}_H) \). Let us convince ourselves that we may do the same here. Recall that a *linear Lie algebra representation* is a Lie algebra homomorphism \( c : g \to \text{gl}(V) \), i.e. a map that fulfills
\[
[c(X), c(Y)]_{gl} = c([X, Y]_g).
\]
Given a representation \( \hat{\rho} : G \to \text{GL}(V) \), we immediately obtain a Lie algebra representation by considering the differential of \( \rho \) at the unit element \( e \), \( d\hat{\rho} \).

We can play exactly the same game for the projective representations.

**Definition B.2.** A *projective Lie algebra representation* is a Lie algebra homomorphism \( \xi : g \to \text{pgl}(V) \).

**Lemma B.3.** If \( \rho : G \to \text{PGL}(V) \) is a projective representation, \( \hat{d}\rho : g \to \text{pgl}(V) \) is a projective Lie algebra representation.

**Proof.** Again using the notation \( C_g \) for the conjugation with \( g \in G \), \( \rho \) being a projective representation implies that
\[
C_{\rho(g)} \circ \rho = \rho \circ C_g \text{ for all } g \in G.
\]
Differentiating this relation twice reveals
\[
\text{Ad}(\rho(g))d\rho = d\rho \text{Ad}_g \text{ for all } g \in G.
\]
\[
\text{ad}(d\rho(X))d\rho(Y) = d\rho(\text{ad}(X)Y) \text{ for all } X, Y \in g.
\]
The final line exactly reads \( [d\rho(X), d\rho(Y)] = d\rho([X, Y]) \), which is what we wanted to prove. \(\square\)

Importantly, there is a very simple way to turn a projective Lie algebra representation to a linear Lie algebra representation. Since we will need this construction later, let us record it already now.

**Lemma B.4.** For \( A \in \text{pgl}(V) \), let \( A^o \in \text{gl}(V) \) be the unique element in \( A \) that has trace zero. Then, if \( \xi : G \to \text{pgl}(V) \) is a projective Lie algebra representation,
\[
\xi^o : g \to \text{gl}(V), \ X \mapsto \xi(A)^o
\]
is a linear Lie Algebra representation.

**Proof.** \( \xi \) being a projective linear representation means that
\[
[\xi(X), \xi(Y)]_{pgl(V)} = \xi([X, Y])
\]
in \( \text{pgl}(V) \). Since \( \xi(X)^o, \xi(Y)^o \) and \( \xi([X, Y])^o \) are representing elements of \( \xi(X), \xi(Y) \) and \( \xi([X, Y]) \), this together with Lemma B.1 means that
\[
\xi([X, Y])^o = [\xi(X)^o, \xi(Y)^o]_{gl} + \lambda \text{id} \quad (7)
\]
for some $\lambda \in \mathbb{F}$. The trace of the left hand side is however 0 by definition, and by the cyclic property of the trace,

$$\text{tr}([\xi(X)^\circ, \xi(Y)^\circ]_{g\mathfrak{t}}) = \text{tr}(\xi(X)^\circ \xi(Y)^\circ - \xi(Y)^\circ \xi(X)^\circ) = 0.$$ 

Hence, taking the trace of both sides of (7) reveals that $\lambda$ must be 0. But (7) with $\lambda = 0$ exactly means that $\xi^\circ$ is a linear Lie algebra representation.

For linear representations, the invariance relation $(\mathcal{P}_{\text{lin}}(G))$ can instantly be turned into the set of equations

$$d\rho(A)v = 0 \text{ for all } A \in \mathfrak{g},$$

by differentiating at the unity element. In fact, the above equations are in many important cases equivalent to $(\text{Proj}_G)$, as the following theorem shows.

**Theorem B.5.** (Special case of [10, Th. 1]) Let $G$ be a connected Lie group. Then $(\text{Proj}_G)$ and $(d\text{Lin}_\mathfrak{g})$ are equivalent.

We can apply the same reasoning for the projective representations, but need to take some more care in interpreting the resulting equations.

**Lemma B.6.** Let $G$ be a Lie group and $v \in \text{P}(V)$ an element which satisfies $(\text{Proj}_G)$. Then,

$$d\rho(A)v = 0 \text{ in } T_v \text{P}(V) \text{ for all } A \in \mathfrak{g},$$

where $T_v \text{P}(V) = V/\text{span}(v)$ is the tangent space of $\text{P}(V)$ at $v$.

**Proof.** Let $A \in \mathfrak{g}$ be arbitrary, and $\gamma : \mathbb{R} \to G$ a curve on $G$ with $\gamma_0 = e$, $\dot{\gamma}_0 = A$. Since $v$ solves $(\text{Proj}_G)$, we then have $\rho(\gamma_t)v = v$ for all $t$.

Differentiating this relation at $t = 0$ implies $0 = \frac{\partial}{\partial t}|_{t=0}\rho(\gamma_t)v$, whereby the relation is an equality in the tangent space of $\text{P}(V)$ at $v$. Now we have $\frac{\partial}{\partial t}|_{t=0}\rho(\gamma_t)v = d\rho_{\gamma_t}(\dot{\gamma}_t)|_{t=0}v = d\rho_x(A)v$, so that the claim follows.

Let us stress that (8) is not a linear equation, but rather an eigenvalue problem. To be concrete, (8) means that for every $M \in \mathfrak{gl}(V)$ in the equivalence class $d\rho(A) \in \mathfrak{pgl}(V)$ and $x$ in the equivalence class $v \in \text{P}(V)$, there exists a $\lambda \in \mathbb{F}$ with

$$Mx = \lambda Mx.$$

Here, we have highlighted that $\lambda$ depends on $M$, but not on $x$.

### B.3 A Lie algebra version of Theorem 2.16

We now formulate and prove a Lie algebra formulation of Theorem 2.16.

**Theorem B.7.** Let $G$ be a Lie Group, $\rho : G \to \text{PGL}(V)$ be a projective representation, and $v \in \text{P}(V)$ a solution of $(\text{Proj}_G)$. Then, every element $x \in V$ of the equivalence class $v$ solves the equations

$$d\rho^\circ(A)x = 0 \text{ for all } A \in [\mathfrak{g}, \mathfrak{g}].$$

Here, $[\mathfrak{g}, \mathfrak{g}]$ is the subalgebra of $\mathfrak{g}$ generated by commutators $\{[A, B] \mid A, B \in \mathfrak{g}\}$.

**Proof.** Lemma B.6 (together with the discussion just after it) implies that since $d\rho^\circ(A)$ is in the equivalence class of $d\rho$ and $x$ in the one of $v$, we must have

$$d\rho^\circ(A)x = \lambda_A x$$

for some scalars $\lambda_A$. Now, for $A, B \in \mathfrak{g}$ arbitrary we have, since $d\rho^\circ$ is a linear Lie algebra representation

$$d\rho^\circ([A, B])x = [d\rho^\circ(A), d\rho^\circ(B)]_{\mathfrak{g}}x
= (d\rho^\circ(A)d\rho^\circ(B) - d\rho^\circ(B)d\rho^\circ(A))x = \lambda_A \lambda_B x - \lambda_B \lambda_A x = 0.$$ 

This relation being true for commutators extends to all of $[\mathfrak{g}, \mathfrak{g}]$. The theorem has been proven.
Just as in the group case, this theorem has a special flavor in the case of a group with a perfect Lie algebra, i.e. one with \{\mathfrak{g}, \mathfrak{g}\} = \mathfrak{g}. In that case, \((d\text{Lin}_\rho)^\circ(A)\) immediately implies that \((d\rho^\circ(A))x = 0\) for all \(A \in \mathfrak{g}\), i.e. that a set of equations like \((d\text{Lin}_\rho)^\circ\) hold. This together with Theorem B.5 suggests that \((d\text{Lin}_\rho)^\circ\) in this case should be equivalent to \((\mathsf{Proj}_G)\). Since \(d\rho^\circ\) not necessarily is induced from a linear representation on \(G\), this is not immediate. However, we will still be able to prove the equivalence in the case of \(G\) admitting a universal cover.

**Definition B.8.** Let \(G\) be a Lie group. We say that a covering group \(H\) is a universal cover of \(G\) if \(H\) is simply connected.

**Example B.9.** We find an important example of a universal cover in \(SU(2)\) – it is a cover of \(SO(3)\), and also simply connected.

With a universal cover at our disposal, we may lift the linear Lie algebra representation \(d\rho^\circ\) to a linear representation of \(H\).

**Lemma B.10.** Let \(G\) be a Lie group with a universal covering group \(H\), and \(\rho : G \to \text{PGL}(V)\) a projective representation of \(G\). Then, there exists a linear representation \(\hat{\rho} : H \to \text{GL}(V)\) with the following properties

1. The induced representation \(d\hat{\rho} : \mathfrak{h} \to \text{gl}(V)\) is \(d\rho^\circ \circ d\varphi_e\), where \(\varphi : H \to G\) is a covering map.
2. \(\hat{\rho}\) is a lift of \(\rho\).

**Proof.** Since \(\varphi : H \to G\) is a covering map, \(d\varphi_e : \mathfrak{h} \to \mathfrak{g}\) is a Lie algebra homomorphism. This implies that \(c = d\rho^\circ \circ d\varphi_e\) defines a linear Lie algebra representation:

\[
[c(A), c(B)]_{\text{gl}} = [d\rho^\circ (d\varphi_e(A)), d\rho^\circ (d\varphi_e(B))] = d\rho^\circ ([d\varphi_e(A), d\varphi_e(B)]).
\]

Now, since \(H\) is simply connected, there exists a linear representation \(\hat{\rho} : H \to \text{GL}(V)\) whose induced representation is \(c = d\hat{\rho}^\circ \circ d\varphi_e\) [5, Th. 5.33].

It remains to prove that this representation is a lift of \(\rho\), i.e. that \(\rho \circ \varphi = \Pi_{\text{End}(V)} \circ \hat{\rho}\). Since \(H\) is connected, there elements in any neighborhood \(U\) of the identity that generates \(H\) [24, Prop. 7.14], and if we choose this neighborhood small enough, we can represent any elements in \(U\) as \(\exp(A_i)\) for some \(A_i \in \mathfrak{h}\). Since \(\varphi, \rho\) and \(\hat{\rho}\) are group homomorphism, it therefore suffices to show

\[
\rho(\varphi(\exp(A))) = \Pi_{\text{End}(V)}(\hat{\rho}(\exp(A)))\text{ for all }A \in \mathfrak{h},
\]

i.e. that \(\hat{\rho}(\exp(A))\) is in the equivalence class \(\rho(\varphi(\exp(A)))\). However, it is well known that (see e.g. [24, Prop 20.8] if \(K\) and \(L\) are Lie groups and \(\sigma : K \to L\) is a Lie group homomorphism, we have

\[
\exp(d\sigma(X)) = \sigma(\exp(X))\text{ for all }X \in \mathfrak{k}.
\]

Applying this for \(\rho, \hat{\rho}\) and \(\varphi\), we obtain

\[
\rho(\varphi(\exp(A))) = \rho(\exp(d\varphi_e(A)))
\]

\[
\hat{\rho}(\exp(A)) = \exp(d\hat{\rho}^\circ (d\varphi_e(A))) = \rho^\circ(\exp(d\varphi_e(A))),
\]

and since \(\rho^\circ(B)\) (by definition) is in the equivalence class of \(\rho(B)\) for every \(B\), the claim follows.

\[\square\]

We may now easily derive the second main theorem of this section.

**Theorem B.11.** Let \(G\) be a Lie group which admits a universal cover and has a perfect Lie algebra, and \(\rho : G \to \text{PGL}(V)\) be a projective representation. Then, a vector \(x \in V\) solves

\[(d\rho^\circ(A))x = 0\text{ for all }A \in \mathfrak{g}\]

if and only if its equivalence class \(v \in \text{Proj}(V)\) solves \((\mathsf{Proj}_G)\).

**Proof.** First, the fact that the Lie algebra is perfect together with Theorem B.7 shows that \((\mathsf{Proj}_G)\) implies \((d\text{Lin}_\rho)^\circ\). It remains to prove the other direction.
To do so, consider the linear representation \( \tilde{\rho} \) from Lemma B.10. By that Lemma, we know that its induced representation is \( d\rho^\circ d\varphi_e \). Since the covering map \( \varphi \) is a local diffeomorphism around \( e \in H \), \( d\varphi_e : h \to g \) is an isomorphism. Therefore, \( (\text{dLin}_\rho) \) immediately implies that
\[
d\tilde{\rho}(B)x = 0 \text{ for all } B \in \mathfrak{h}.
\]
Since \( H \) is connected, this together with Theorem B.5 shows that \( x \) is a solution of
\[
\tilde{\rho}(h)x = x \text{ for all } h \in H.
\]
Since \( \tilde{\rho} \) is a lift of \( \rho \), this however implies \( \text{(Proj}_G \) by Lemma 2.13.

\[\square\]

C The EGNN model

As we choose to use it, an EGNN-layer transforms a graph \( G \), where each node has feature \( h_k^\ell \in \mathbb{R}^f \) and coordinate-feature \( \xi_k^\ell \in \mathbb{R}^D \) according to the following message-passing procedure
\[
m_{ij} = \phi_e(h_i^\ell, h_j^\ell, ||\xi_i^\ell - \xi_j^\ell||^2) \quad \xi_i^{\ell+1} = \sum_{j \in N_i} (x_i^\ell - x_j^\ell)\phi_\xi(m_{ij})
\]
\[m_i = \sum_{j \in N_i} m_{ij} \quad h_i^{\ell+1} = \phi_h(h_i^\ell, m_i).
\]
Here, \( N_i \) is the neighborhood of the node \( i \), and \( \phi_e, \phi_\xi \) and \( \phi_h \) are 2-layer MLP:s. Provided the feature \( h_k^\ell \) is invariant to rotations, \( h_i^{\ell+1} \) and \( \xi_i^{\ell+1} \) will be invariant and equivariant to rotations, respectively. Note that the EGNN networks are defined for any dimension \( D \), and is equivariant to the group of Euclidean transformations \( \text{SE}(D) \), in particular to \( \text{SO}(3) \).

Remark C.1. We would like to point out that the EGNN layers are not technically equivariant MLP’s, i.e. iterative, alternating concatenations of equivariant linear layers and activation functions. This is upon first inspection somewhat disappointing, since our theory is the most directly applicable to the equivariant MLP’s.

However, our theory is still relevant: Message passing networks can be arbitrarily well approximated with linearly permutation-equivariant MLP’s [27], and our results imply that the linear layers of the latter look exactly the same as they could would we only assume a projective equivariance. Hence, to some extent, the models we consider here are not more expressive than projectively equivariant MLP’s.

D Details and further results for line fitting

Equivariant model. We build a network with a varying number (3, 4 or 5) EGNN layers. We let the intermediate \( m_{ij} \) have dimension 64, and all MLP:s have 64 hidden features. We use the EGNN implementation from the DGL-package [40]. We made only one minor adjustment: The original code calculated an expression of the form \( x/\text{sqrt}(x*x)+\epsilon\text{ps1on} \) to approximate a normalization procedure. This ran into problems when it was fed zero vectors, given the non-differentiability of \( \text{sqrt} \) in zero. We fixed this by shifting the security margin \( \epsilon\text{ps1on} \) into the square root.

Non-equivariant model. As a comparison, we use a standard neural message passing architecture [13]. This architecture is based on a slightly different message passing procedure compared to the EGNN networks. As we use it, it transforms features \( g_k^\ell \in \mathbb{R}^{f_\ell} \) according to
\[
g_k^\ell = \nu \left( \sum_{j \in N_i} \phi_\theta(g_i^\ell, g_j^\ell)g_j^\ell \right)
\]
Here, \( \phi_\theta \) is an MLP which outputs a matrix in \( \mathbb{R}^{f_{\ell+1} \cdot f_\ell} \), \( \nu \) is the normalisation function \( \nu(x) = \frac{x}{||x||} \).

We added this step to keep the features from exploding with a growing number of layers. In our experiments, we use \( f_\ell = 16 \) for all intermediate layers, and a number of hidden features in the \( MLP \phi_\theta \) to let the equivariant and non-equivariant architecture have roughly the same number of total parameters. Note that this architecture does not discriminate between coordinate-features and regular features. Accordingly, we let the final layer output layers in \( \mathbb{R}^3 \), and use a pooled version of this, again normalised to the 3D unit sphere, as our estimate of \( w(X) \).
Data generation. Similarly to [37], we generate data randomly ‘on the fly’ by generating 100 points in \([-1, 1]^2 \subseteq \mathbb{R}^2\) uniformly randomly, project them to a line defined by two random points with a probability \(p\) (which we in these experiments set to 0.6) and thereafter add a small amount of Gaussian noise (\(\sigma = .01\)) to each point. Note that we are hence in a probabilistic data setting with practically infinitely amount of data. In Figure 3, an example of a generated cloud is provided.

![Figure 3: An example of a generated cloud, and the way we provide it to the equivariant networks as a point cloud on the unit sphere.](image)

Augmentation. Let us describe the data augmentation scheme in some more detail. To generate the random scene rotations, we first rotate the cloud arbitrarily in 2D and then apply a rotation about the \(e_1\)-axis in 3D, uniformly chosen in \([0, \theta_{\text{max}}]\), where \(\theta_{\text{max}}\) is a parameter we vary in our experiments. Note that these rotations are applied to the homogeneous coordinates of the points. For the non-equivariant models with input in \(\mathbb{R}^2\), this procedure moves the points away from the origin. This leads to coordinate magnitudes at test time that were unseen during training. To compensate for this for the nonequivalent models, we check whether 90% of the transformed \(y_i\) coordinates still lie in \([-1, 1]\). If not, we renormalize them \(\tilde{y}_i = f y_i\) to obey that property.

Training. All networks are implemented in PyTorch [30] and trained using the Adam algorithm for 100 epochs, each consisting of 2500 examples, using a batch size of 64, with a constant training rate of \(10^{-4}\).

D.1 Further results

In Figure 4 we report the results we presented in Figure 2 in table form, along with the standard deviations over the ten runs. An observation which deserves to be made is that the scores of equivariant model consistently have low standard deviation compared to the augmented non-equivariant ones. However, given the very small scope of the experiment, one should not draw any general conclusions from this.

![Figure 4: Robust linefitting results. The 'aug'-columns refer to nonequivalent models. For each experiment, we report averages and standard deviations over 10 runs.](image)

D.2 Robust line extraction II: Invariant features

We perform an experiment to test the effect of providing the robust line extraction methods with invariant feature, similarly to ET-Net. We use exactly the same models as in the main experiments, but additionally provide a ‘local svd’ feature: the singular values of the matrix

\[ M_i = \sum_{j \text{ in nghb. of } i} \frac{(p_i - p_j)(p_i - p_j)^*}{|p_i - p_j|^2}, \]

whereby \(p_i = x_i\) for the equivariant models, whereas \(p_i = y_i\) for the non-equivariant models. The hope is that these features can detect local linear structures, which is the aim of the model. Note that for the equivariant models, these are invariant to scene rotations, leaving the whole network equivariant. We will refer to this feature as ‘local SVD’.
In our experiment, we consider three 'flavours' of models: equivariant models, nonequivariant non-augmented models and non-equivariant augmented models, with \( \theta_{\max} = 30^\circ \). Each of these are tested with and without the local svd feature, making up a total of six different models. The models not using the local svds are using the same architecture but instead fed an initial constant node feature. We use exactly the same training parameter setup as in the main experiments.

| Layers | 0° at test | 30° at test | Rel. diff. |
|--------|------------|-------------|------------|
|        | mean     | σ           | mean     | σ           |            |
| 3 layers |          |             |           |             |            |
| Equiv + svd | 7.84e-02 | 7.29e-03 | 7.73e-02 | 5.32e-03 | +1.35% |
| Equiv | 8.28e-02 | 5.39e-03 | 8.29e-02 | 4.24e-03 | -0.19% |
| 0° + svd | 4.15e-02 | 5.79e-03 | 1.13e-01 | 9.87e-03 | -172% |
| 0° | 4.80e-02 | 1.22e-02 | 1.27e-01 | 2.01e-02 | -165% |
| 30° + svd | 4.16e-02 | 4.50e-03 | 7.12e-02 | 3.09e-03 | -71.2% |
| 30° | 5.14e-02 | 1.06e-02 | 8.40e-02 | 1.73e-02 | -63.6% |

| 4 layers | 0° at test | 30° at test | Rel. diff. |
|----------|------------|-------------|------------|
| mean     | σ           | mean     | σ           |            |
| Equiv + svd | 6.49e-02 | 1.60e-03 | 6.48e-02 | 1.86e-03 | +0.16% |
| Equiv | 6.97e-02 | 2.89e-03 | 6.93e-02 | 2.63e-03 | +0.58% |
| 0° + svd | 3.47e-02 | 2.42e-03 | 1.05e-01 | 3.32e-03 | -202% |
| 0° | 4.04e-02 | 1.87e-02 | 1.17e-01 | 1.87e-02 | -190% |
| 30° + svd | 3.43e-02 | 2.69e-03 | 5.56e-02 | 3.27e-03 | -61.7% |
| 30° | 4.13e-02 | 1.50e-02 | 6.40e-02 | 2.22e-02 | -54.8% |

| 5 layers | 0° at test | 30° at test | Rel. diff. |
|----------|------------|-------------|------------|
| mean     | σ           | mean     | σ           |            |
| Equiv + svd | 5.94e-02 | 1.75e-03 | 5.93e-02 | 2.19e-03 | 0.25% |
| Equiv | 6.57e-02 | 2.72e-03 | 6.51e-02 | 2.75e-03 | 0.93% |
| 0° + svd | 4.20e-02 | 1.02e-02 | 1.12e-01 | 1.40e-02 | -166% |
| 0° | 4.10e-02 | 1.17e-02 | 1.10e-01 | 1.24e-02 | -168% |
| 30° + svd | 2.77e-02 | 2.57e-03 | 4.64e-02 | 3.49e-03 | -67.8% |
| 30° | 3.63e-02 | 1.36e-02 | 5.06e-02 | 2.36e-02 | -56.7% |

Table 3: Difference in performance of models with and without local svds.

Results. We evaluate the models in the same ways as in the main experiment. Results averaged over ten runs are presented in table 3.

In large, the results follow the same trend as the GraTNet experiment: Including the svd features in general improve the models, but do not make the non-equivariant ones more equivariant. We again include the standard deviation over the runs, and observe that the svd features seem to make all models more stable.

The observant reader may have observed that the models without added svd-features have slightly different performances compared to the previous experiments. All deviations are however within a bit more than one standard deviation, most of them lower than that.

D.3 Occupancy function models

In the robust line estimation task, the main problem is to decide which points in the noisy cloud \( X \) are inliers – when that is known, the estimation of the line parameters is surely a trivial task. It should therefore be wise to not design a network for inferring the line parameters \( w(X) \) directly, but rather learn an occupancy function \( \Omega : P(V) \to [0,1]^m \) detecting which points are located on the line and which are not, and subsequently solving a weighted least squares problem

\[
\min_{|w| = 1} \sum_{i \in [m]} \Omega(X)_i |\langle w, x_i \rangle|^2.
\] (9)

If \( \Omega \) is invariant to the relevant group actions, the solution of the above problem will also be. This is an approach taken in works on e.g. context normalization [37, 43], which is better suited for the 'robust' portion of the robust line estimation problem. In this section, we perform a small experiment to test which projective invariance can have for these types of models.

Similarly as for the more direct experiments, we build a small neural network by feeding features in to a three-layer EGNN (with 32,32 and 15 features per layer, respectively) that applies attentive context normalization between the layers, as in [37]. The final attentive context weights \( \omega_{0}(i) \) are then used to calculate line parameters \( w_0 \) via the program (9). The weights \( \omega_{0} \), along with signed distances \( \langle w_0, x_i \rangle \), are then used along with the output features of the EGNN as input features to...
a new three-layer EGNN, in a fashion similar to OANet. We use a total of three EGNN in such an iterative fashion.

Here, we make an experiment to test to which extent invariance to scene rotations can boost performance of the network. To do so, we consider three variants of inputs features: (1) the ‘local SVD’ feature defined in the main paper, (2) the radius of the 2D-portion of each point \(x_i\), that is, \(|(x_i(0), x_i(1))|\) (‘norm’) and (3) the 2D-coordinates \((x_i, x_1)\) ‘coords’. Note that these are increasingly non-invariant to scene rotations: The local SVD is completely invariant, the norm is not invariant to all rotations, but at least the one around the principal axis \([0, 0, 1]\) and the coords is is ‘fully non-invariant’ to scene rotations.

We consider six different networks. They all use an EGNN backbone. Three of them use clouds where each input point is normalized to lie on the unit sphere, and three of them input clouds where the points are normalized to lie on the \(x_2 = 1\)-plane. For ease of exposition, we will refer to the two variants of networks as projective and planar networks.

For each of the two variants, we consider three subvariants: the first step calculate different features. All use the local SVD, but one of them additionally use the norm, and one the coords. We will refer to these as the equivariant, norm-breaking and total-breaking architectures. Note that the projective equivariant architecture is equivariant to scene rotations, whereas the equivariant planar architecture is equivariant to \(\Omega(2)\). The norm-breaking architectures have some equivariant bias left (they are equivariant to rotations about the principal axis \([0, 0, 1]\)), but can differentiate between points depending on their distance to the principal axis. On the other hand, the total-breaking architectures do not have any apparent bias. Note that, importantly, the non-equivariant architectures are not stripped of the local SVD-features, but a priori have the potential to use only that feature. All networks use the loss function

\[
\sum_{i=0}^{2} \alpha_i \text{CE}(\omega_i, \omega^0) + \beta_i |w_i - w^0|^2,
\]

where \(\text{CE}\) denotes the cross-entropy function, \(\omega^0\) and \(w^0\) are ground truth occupancies and line parameters, and \(\alpha_i\) and \(\beta_i\) are real parameters.

We test the three architectures by generating simulated data just as in the main paper. However, we now choose to set the outlier, and vary it as in [37] – we test the values \(\{0.6, 0.7, 0.8, 0.85, 0.9\}\) and train them for 50 epochs with an initial learning rate of \(10^{-4}\) that we drop by a factor of 0.1 after five and twenty epochs. We change the values of the parameters \(\alpha_i, \beta_i\) as in Table 4 alongside the training – the idea here is to initially supervise intermediate weights \(\omega_i\) the subnetworks, move over to focus on the final layer and finally explicitly penalize the line parameter \(w_2\) output. As a performance metric, we measure the difference in \(L_2\)-norm between the output \(w_2(X)\) of the network and the ground truth \(w^0\). The experiment is repeated for all of the six network types 10 times. The results are presented in Table 6.

| Epoch 0 − 5 | (\(\alpha_0, \beta_0\)) | (\(\alpha_1, \beta_1\)) | (\(\alpha_2, \beta_2\)) |
|------------|----------------|----------------|----------------|
| Epoch 5 − 20 | \(0.0\) | \(0.0\) | \(0.0\) |
| Epoch 21 − 49 | \(0.0\) | \(0.0\) | \(0.2, 1\) |

**Table 4:** Evolution of the \(\alpha\) and \(\beta\)-parameters in the loss function

The results clearly suggest that for this particular task, the inductive \(\text{SE}(2)\)-invariance of the weights \(\Omega(X)\) is more powerful than the scene rotation invariance of them. We speculate that restricting to invariance of the whole group of scene rotations restricts the networks too much. After all, the application of some scene rotations will result in very singular point clouds, whereas every \(\text{SE}(2)\)-transformation results in a plausible 2D-cloud.

We can however also compare the different ‘flavours’ of the architectures within each group. Given the very similar structure of the networks, it is not surprising that the differences in performance are relatively small in the ‘internal comparisons’ – the scores often do not differ more than a standard deviation. However – both in the planar and in the projective cases – there is a clear trend of the norm-break models performing best – we speculate that this ‘partial break’ is a good balance between utilizing the symmetry and using the information the absolute coordinates provide. Also note that
in almost all cases, the equivariant models perform better than the total-break ones, suggesting that equivariance can help the models to perform better. In contrast to the previous experiments, there is no clear trend in the comparisons of the standard deviations.

| Plan. equiv | Plan. norm break | Plan. total break |
|-------------|------------------|------------------|
| mean        | σ               | mean            | σ          | mean            | σ          |
| 60%         | 1.940e-02       | 8.139e-03       | 1.352e-02  | 5.738e-03       | 2.030e-02  | 7.851e-03 |
| 70%         | 4.240e-02       | 1.092e-02       | 4.227e-02  | 1.164e-02       | 4.991e-02  | 4.465e-03 |
| 80%         | 1.491e-01       | 1.070e-03       | 1.495e-01  | 1.186e-02       | 1.569e-01  | 1.908e-02 |
| 85%         | 3.027e-01       | 8.590e-03       | 3.029e-01  | 8.478e-03       | 3.326e-01  | 1.155e-02 |
| 90%         | 5.302e-01       | 9.071e-03       | 5.334e-01  | 1.033e-02       | 5.420e-01  | 1.015e-02 |

Table 5: Results for the planar models for different fractions of outliers.

| Proj. equiv | Proj. norm break | Proj. total break |
|-------------|------------------|------------------|
| mean        | σ               | mean            | σ          | mean            | σ          |
| 60%         | 1.639e-02       | 7.202e-03       | 1.966e-02  | 5.738e-03       | 2.151e-02  | 7.424e-03 |
| 70%         | 5.210e-02       | 9.749e-03       | 4.917e-02  | 1.147e-02       | 5.388e-02  | 5.120e-03 |
| 80%         | 1.790e-01       | 1.850e-02       | 1.569e-01  | 1.210e-02       | 1.775e-01  | 8.650e-03 |
| 85%         | 3.258e-01       | 1.976e-02       | 3.149e-01  | 1.228e-02       | 3.401e-01  | 2.231e-02 |
| 90%         | 5.564e-01       | 1.067e-02       | 5.420e-01  | 1.086e-02       | 5.563e-01  | 1.020e-02 |

Table 6: Results for the planar models for different fractions of outliers.

E Details and further results for correspondence filtering

Figure 5 illustrates the data used in the correspondence filtering process, as well as the rotated version used for evaluating rotation equivariance. In Section E.1 we present additional results. In Sections E.2 and E.3 we present details about the GraT-Net and ET-Net implementations respectively.

E.1 Further results

We show the best performing rerun results in Table 7.

| AUC @ 5° | Test time rotation 0° | Test time rotation 15° |
|----------|----------------------|-----------------------|
|          | 5°       | 10°     | 15°    | 20°     | 5°       | 10°     | 15°     | 20°     |
| RANSAC   | /8.5     | /13.8   | /18.4  | /22.3   | /8.5     | /13.9   | /18.5   | /22.4   |
| GraT-Net2| 2.5/40.3 | 6.4/50.1 | 10.3/56.3 | 14.4/60.7 | 2.5/40.5 | 6.0/50.3 | 10.1/56.8 | 14.2/61.3 |
| GraT-Net3| 2.1/38   | 6.0/47.3 | 9.9/53.4 | 13.7/57.8 | 2.2/38.0 | 6.0/47.4 | 9.9/53.7 | 13.7/58.1 |
| GraT-Net2| 25.8/51.0 | 39.8/61.9 | 49.0/67.3 | 55.8/71.6 | 16.2/47.7 | 27.6/58.4 | 36.4/64.7 | 43.2/69.2 |
| GraT-Net3| 26.8/51.0 | 39.1/60.7 | 47.9/66.9 | 54.6/71.0 | 15.3/46.4 | 25.6/57.3 | 33.8/64.1 | 40.3/68.5 |
| ET-Net2  | 49.2/56.5 | 62.5/66.9 | 70.1/72.9 | 75.0/76.9 | 35.0/53.5 | 49.9/64.0 | 59.1/70.1 | 65.2/74.2 |
| ET-Net3  | 50.4/57.7 | 63.4/67.8 | 70.9/73.6 | 75.5/77.3 | 35.3/53.7 | 50.2/64.5 | 59.4/70.8 | 65.4/74.9 |
| T-Net (retrained) | 48.2/55.3 | 61.0/65.7 | 69.1/71.7 | 73.8/75.6 | 35.4/52.8 | 49.4/63.1 | 58.1/69.3 | 63.9/73.2 |
| T-Net (paper) | 52.3/56.1 | -        | -        | 75.8/75.5 | -        | -        | -        | -        |

Table 7: Here we report the maximum scores of three reruns instead of the median scores that were reported in Table 1. Note that only ET-Net and T-Net were rerun. Scores are without/with RANSAC.

E.2 GraT-Net

The GraT-Net model replaces each T-Block (or “sub-network”) of a T-Net [46] by an EGGNN block [35] which we call GraphBlock. We consider as input to the network a set of correspondences \( \{(x_i, y_i)\} \) of keypoints \( x_i \) in the first image and \( y_i \) in the second image – the coordinates can be given either in 2D (calibrated image coordinates) or 3D (spherical image coordinates as described in Section 3). We build two graphs as follows. The first graph has node coordinates \( \{x_i\} \) while the second has node coordinates \( \{y_i\} \). The graphs have the same edge structure, generated in two steps:

1. The \( i \)'th node has edges to each of the \( K = 12 \) nearest neighbours in the metric \( \|x_i - x_j\| \).
Figure 5: Example of the correspondence data. **First row:** SIFT matches extracted from two YFCC100M images of the Taj Mahal. We show only 20 matches instead of the full 2000, for ease of viewing. Note that there are several outlier matches present. **Second row:** Just the coordinate matches without image information, as is input into the network. **Third row:** Slight rotations applied to both cameras, this is the data we use for testing rotation equivariance. **Fourth row:** For completeness, the corresponding warped images are shown as well. The third and fourth rows are not to scale with the first and second rows.

2. the \( i \)'th node also has edges to each of the \( K = 12 \) nearest neighbours in the metric \( \| y_i - y_j \| \).

Thus a node in either graph is connected not only to the closest neighbours in its own image, but also to the closest neighbours of the corresponding node in the other image. The two graphs share edges and node features but have different node coordinates.

### E.2.1 Architecture

We use two GraphBlocks followed by the “\( \mid \)”-part of T-Net. A GraphBlock consists of the following.

1. In the GraT-Net-- case, the node features are initialized to constant 1. In the GraT-Net case, the node features are a linear transformation of the keypoint coordinates.
2. A first EGGNNConv layer is applied to each of the two graphs separately. This yields new node coordinates and new node features, which may now be different between the two graphs.
3. Maxpooling of the node features over corresponding nodes in the two graphs, so that the two graphs have the same node features again.
4. A second EGGNNConv layer, as in step 2.
5. Maxpooling as in step 3.
6. A linear layer followed by ReLU and tanh transforming the node features into inlier weights.
7. A weighted least squares solver for the essential matrix $E$.

In T-Net three T-Blocks (or “sub-networks”) are used, the reason that we use only two blocks here is to be able to fit the more memory demanding graph layers onto a single GPU. We also use gradient checkpointing of the EGNNConv layers for the same reason.

E.2.2 Modifications to the EGNN model

For an overview of EGNN see Section C. Here we use the default EGNNConv implementation in the DGL package [40], with the following modifications.

1. We introduce a normalization in the edge, node and coordinate MLP:s, after the first activation functions. The normalization is implemented as a Pytorch BatchNorm1d layer, which due to the data representation in DGL means that we normalize the channels over the graphs in each batch as well as the nodes/edges in each graph. Without this normalization training worked extremely poorly.
2. We clamp the radial edge data to minimum $10^{-8}$ and maximum 10 to avoid numerical issues.

E.2.3 Training details

We train using the same hyperparameters as T-Net, for 7 days on a single NVIDIA Tesla T4 GPU. This means that the total training time is similar as for T-Net and ET-Net. The GraT-Net-- models trained for 500000 iterations just like T-Net and ET-Net, but the GraT-Net models required more time per forward pass and so are only trained for around 350000 iterations. Their performance after was so far below the performance of T-Net after 350000 iterations that we deemed it a waste of resources to train the full 500000 iterations – and we suggest that the more fair comparison might be in terms of time trained anyway.

E.3 T-Net and ET-Net

We train ET-Net and retrain T-Net aiming to use the same hyperparameters as in the original T-Net paper. The batch size is 32, the optimizer is Adam, the learning rate $10^{-3}$. We train for 500000 iterations, which takes almost 7 days on a single NVIDIA Tesla T4 GPU. The first 20000 iterations are ran with only classification loss on the inlier weights and after that the loss on the essential matrix is added, with half the weight of the classification loss. Slight modifications were made to the official T-Net implementation to remove errors that were obtained when trying to run it and a few further modifications were made for aesthetic reasons. We will publish our code on Github upon publication of the paper.

The rotation invariant features that are used in ET-Net are produced as follows:

1. For each correspondence $(x_i, y_i)$ (in either 2D or 3D as described in Section 3), calculate the $K = 12$ nearest neighbours in both images.
2. Stack the distances to these neighbours into a vector $a \in \mathbb{R}^{48}$. It has 48 entries because we have the distances $||x_i - x_j||$ as well as $||y_i - y_j||$ for each of the $2 \cdot 12 = 24$ neighbours. We stack first the sorted $x$-distances followed by the sorted $y$-distances.
3. Feed $a$ through a two layer MLP with ReLU activations to produce a feature $b \in \mathbb{R}^{6}$, which is then stacked with the input coordinates to form the input feature to T-Net.