Steerers: A framework for rotation equivariant keypoint descriptors

Supplementary Material

A. Supplementary theory

We provide further theoretical discussions that did not have room in the main text. First, Section A.1 contains a discussion of what having representations of C_4 or $\mathrm{SO}(2)$ on description space means. Section A.2 contains a proof of Theorem 5.1 and Section A.3 presents matching strategies that are considered in the extra ablations of Section B but were omitted from the main paper due to space limitations.

A.1. Disentangling description space

As explained following Theorem 4.1, any representation of C_4 or $\mathrm{SO}(2)$ can be block-diagonalized over the real numbers into blocks of size 1 and 2, called irreducible representations (irreps). We can think of these irreps as disentangling descriptions space [14], i.e. each eigenspace of the steerer is acted on by rotations in a specific way according to the respective irrep. This section explains the relevance of the different irreps to keypoint descriptors. For C_4, we have the following.

- 1×1 irreps $\begin{pmatrix}1\end{pmatrix}$ act by doing nothing. Hence, the corresponding dimensions in description space are invariant under rotations. For C_4, image features described by these dimensions could be crosses or blobs.
- 1×1 irreps $\begin{pmatrix}-1\end{pmatrix}$ correspond to dimensions that are invariant under $180°$ rotations, but not $90°$ rotations. Examples of such image features could be lines.
- 2×2 irreps $\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}$ correspond to pairs of dimensions that are not invariant under any rotation. Many image features should be of this type, *e.g.* corners.

For $\mathrm{SO}(2)$ we get the same $\begin{pmatrix}1\end{pmatrix}$ irrep, which in this case represents features invariant under all rotations such as blobs, and the 2×2 irreps in (8) which represent features rotating with j times the frequency of the image. E.g. lines rotate with frequency $j = 2$ since when we rotate the image by $180°$, the line returns to its original orientation.

The description for a keypoint does not lie solely in the dimensions of a single irrep. It will be a linear combination of quantities that transform according to the different irreps. The descriptions can then be viewed as a form of non-linear Fourier decomposition of the image features, as discussed in the literature for general image features. We will provide a short discussion in the next paragraph.

Example A.1. In Upright SIFT, the decomposition of the 128 description dimensions is equally split between the irreps, *i.e.*, there are 32 invariant dimensions, 32 dimensions that are invariant under $180°$ degree rotations and 64 dimensions which are not invariant under any rotation.

The connections to Fourier analysis of having a group representation acting on the latent space of a model were discussed in [14] for linear models and concurrently to this work for neural networks in [28]. We sketch the main idea here to give the reader some intuition. If we have a signal h on $\mathbb{R}^n$ and want to know how it transforms under cyclic permutations of the n coordinates, we can take the Discrete Fourier Transform (DFT). Each coordinate h_j can be written as a linear combination of Fourier basis functions: $h_j = \sum_{k=0}^{n-1} \hat{h}_k \exp(2\pi \mathbf{i} jk/n)$ and the DFT is simply the vector $\hat{h}$. When we cyclically permute h by J steps, it corresponds to multiplying each component $\hat{h}_k$ by $\exp(2\pi \mathbf{i} Jk/n)$. Thus, the cyclic permutation on h acts like a diagonal matrix on the DFT $\hat{h}$. The DFT is a linear transform of the signal h. In our setting, the signal consists of images and keypoints transformed by a neural network f to description space. As described in Theorem 4.1, rotations act by a diagonal matrix in description space (up to a change of basis Q). In the terminology of [28], we can think of the neural network f as doing a Neural Fourier Transform of the input.

The usefulness of having group representations act on latent spaces in neural networks has been considered in, for instance, [16, 36, 60]. In these works, the specific representation is fixed before training the network, similar to our Setting C. As far as we know, optimally choosing the representation remains an open question—the experiments in this paper showed that this is an important question. [48] and [25] considered using (9) as a loss term to obtain orthogonal representations on the latent space. This approach is also promising for keypoint descriptors, particularly for encoding transformations more complicated than rotations in description space, since it does not require knowledge of the representation theory of the transformation group in question.

A.2. Proof of Theorem 5.1

Here, we give a proof of Theorem 5.1.

Theorem 5.1. [Adapted from Shakerinava et al. [48], Gupta et al. [25]] Assume that we have a function $f : V \to \mathbb{S}^{D-1}$ and a group G with representation ρ_{in} on V such that, for all $v, v' \in V$ and $g \in G$

$$\langle f(\rho_{\text{in}}(g)v), f(\rho_{\text{in}}(g)v')\rangle = \langle f(v), f(v')\rangle. \tag{9}$$

Then there exists an orthogonal representation $\rho(g)$, such that f is equivariant w.r.t. G with representations ρ_{in} and ρ.

Proof. Note that if $f(v) = f(v')$, then by (9), $f(\rho_{\text{in}}(g)v) = f(\rho_{\text{in}}(g)v')$. This means that we, for each $g \in G$, can define a map $\tilde{\varphi}_g : f(V) \to f(V)$ by

$$\tilde{\varphi}_g(w) = f(\rho_{\text{in}}(g) f^{-1}(w)), \tag{11}$$

where $f^{-1}(w)$ is any element that f maps to w. We next extend $\tilde{\varphi}_g$ to a map $\varphi_g : \mathbb{S}^{D-1} \to \mathbb{S}^{D-1}$. Start by writing any element $w \in \mathbb{S}^{D-1}$ as

$$w = w_\perp + \sum_{i=1}^{n} a_i w_i, \tag{12}$$

where $a_i \in \mathbb{R}$, the w_i's are of the form $f(v_i)$ for some $v_i \in V$ and form a basis of $\text{span}(f(V))$ and $w_\perp$ is orthogonal to $\text{span}(f(V))$. Define

$$\varphi_g(w) = w_\perp + \sum_{i=1}^{n} a_i \tilde{\varphi}_g(w_i). \tag{13}$$

We can now use (9) to show that φ_g is an isometry of the sphere $\mathbb{S}^{D-1}$, i.e. an orthogonal transformation:

$$\begin{aligned}
\langle \varphi_g(w), \varphi_g(w') \rangle &= \langle w_\perp, w'_\perp \rangle && (14)\\
&\quad + \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a'_j \langle \tilde{\varphi}_g(w_i), \tilde{\varphi}_g(w_j) \rangle && (15)\\
&\overset{(9)}{\underset{(11)}{=}} \langle w_\perp, w'_\perp \rangle + \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a'_j \langle w_i, w_j \rangle && (16)\\
&= \langle w, w' \rangle. && (17)
\end{aligned}$$

As it is an orthogonal transformation, we can write φ_g as being a matrix acting on vectors in $\mathbb{S}^{D-1}$ by matrix multiplication. Finally, we need to show that $\rho(g) = \varphi_g$ defines a representation of G, i.e. that $\varphi_g \varphi_{g'} = \varphi_{gg'}$ for all $g, g' \in G$. We begin by showing that φ_g and $\tilde{\varphi}_g$ are equal on $f(V)$, which now follows from linearity of φ_g as follows. Take a general $w \in f(V)$, and again write $w = \sum_{i=1}^{n} a_i w_i$, then

$$\begin{aligned}
\langle \varphi_g(w), \tilde{\varphi}_g(w) \rangle &= \left\langle \varphi_g \left(\sum_{i=1}^{n} a_i w_i \right), \tilde{\varphi}_g(w) \right\rangle && (18)\\
&= \sum_{i=1}^{n} a_i \langle \varphi_g(w_i), \tilde{\varphi}_g(w) \rangle && (19)\\
&\overset{(13)}{=} \sum_{i=1}^{n} a_i \langle \tilde{\varphi}_g(w_i), \tilde{\varphi}_g(w) \rangle && (20)\\
&\overset{(9)}{=} \sum_{i=1}^{n} a_i \langle w_i, w \rangle && (21)\\
&= \langle w, w \rangle && (22)\\
&= 1 && (23)
\end{aligned}$$

so that $\varphi_g(w) = \tilde{\varphi}_g(w)$. For the w_i's from before, we thus have

$$\begin{aligned}
\varphi_{gg'} w_i &= \tilde{\varphi}_{gg'}(w_i) && (24)\\
&= f(\rho_{\text{in}}(gg') v_i) && (25)\\
&= f(\rho_{\text{in}}(g) \rho_{\text{in}}(g') v_i) && (26)\\
&= \tilde{\varphi}_g(f(\rho_{\text{in}}(g') v_i)) && (27)\\
&= \tilde{\varphi}_g(\tilde{\varphi}_{g'}(f(v_i))) && (28)\\
&= \tilde{\varphi}_g(\tilde{\varphi}_{g'}(w_i)) && (29)\\
&= \tilde{\varphi}_g(\varphi_{g'}(w_i)) && (30)\\
&= \varphi_g \varphi_{g'} w_i. && (31)
\end{aligned}$$

Further, for any $w_\perp$ orthogonal to $\text{span}(f(V))$ we have

$$\varphi_{gg'} w_\perp = w_\perp = \varphi_g \varphi_{g'} w_\perp. \tag{32}$$

Thus by linearity $\varphi_{gg'} = \varphi_g \varphi_{g'}$. □

A.3. More matching strategies

We discuss more potential matching strategies. Their performance is shown in the large ablation Table 5.

Projecting to the invariant subspace. Given a steerer $\rho(\mathbf{g})$, we can project to the rotation invariant subspace of the descriptions by taking $\sum_{k=0}^{3} \rho(\mathbf{g})^k y/4$ as descriptions instead of y. Equivalently, one can project by decomposing $\rho(\mathbf{g})$ using (6). However, we will see that these invariant descriptions do not perform very well (but still better than just using y). This is likely because the invariant subspace is typically only a fourth of the descriptor space.

Subset matcher. We can estimate the best relative rotation between two images using the *max matches* matching strategy on only a subset of the keypoints in each image. The obtained rotation is then used to steer the descriptions of all keypoints. This *subset matcher* strategy gives lower runtime while not sacrificing performance much. In our experiments, we use $1,000$ keypoints.

Prototype Procrustes matcher. For frequency 1 descriptors, as a way to make the Procrustes matcher less computationally expensive, we propose, instead of aligning each description pair optimally, to align every description to a prototype description $\tilde{y} \in \mathbb{R}^{2\times(D/2)}$. Thus, we solve the Procrustes problem once per description in each image to obtain $2N$ rotation matrices $R_{1,m}$ and $R_{2,n}$ and form the matching matrix with elements $\langle \texttt{flatten}(R_{1,m} y_{1,m}), \texttt{flatten}(R_{2,n} y_{2,n}) \rangle$. $\tilde{y}$ can be obtained by optimizing it over a subset of the training set for a fixed frequency 1 descriptor. This strategy is similar to the group alignment proposed in RELF [31]; however, there, the alignment is done using a single feature in a permutation representation of C_{16}, whereas we look at the entire D-dimensional description. Similarly to RELF, we could use only specific dimensions of the descriptions for alignment and add a loss for this during training. However, we leave this and a careful examination of optimal alignment strategies for future work.

B. More experiments

We show further matching examples on AIMS in Figures 5 and 6. These examples were chosen by selecting pairs with 20-200 matches after RANSAC, corresponding to successful but challenging pairs. In the remainder of this section, we present ablations that did not have room in the main paper. A large results table is provided as Table 5. Further, we present an experiment in support of using Theorem 5.1 to motivate the existence of steerers in Section B.1 and a comparison to test time rotation augmentation in terms of performance and runtime in Section B.2

Figure 8 shows an example of the improvement obtained using a steerer for large rotations. Figure 7 shows the recall-precision curves for the experiments on AIMS from Section 6.3.

B.1. Equal rotation augmentation

We explained the spontaneous equivariance of DeDoDe-B by referring to Theorem 5.1 and saying that descriptors will be equivariant if the performance is equivalent for matching images I_1 and I_2 as matching jointly rotated images $P_{90}^k I_1$ and $P_{90}^k I_2$. To test this explanation, we can look at whether the equivariance of a keypoint descriptor relates to how good it is at matching jointly rotated images.

We experiment with four different descriptors, DeDoDe-B, DeDoDe-G, DISK [55] and a retrained DeDoDe-B with data augmentation where both images are rotated an equal multiple of 90°. This retrained version is denoted DeDoDe-B†. The results are shown in Table 6 and show that DeDoDe-B and DISK, for which the dropoff in performance between upright and jointly rotated images is relatively low, the steered performance is relatively high. Conversely, DeDoDe-G has a large dropoff in performance between upright and jointly rotated images, and it also has a worse-performing steerer. Finally, the retrained DeDoDe-B†, trained to perform well on jointly rotated images, has a more or less perfect steerer.

B.2. Comparison to test time augmentation

Given two images with an unknown relative rotation, the best obtainable matches from test time augmentation would be obtained when rotating the first image to have the same rotation from upright as the second, which is the case in the joint rotation benchmark. The joint rotation benchmark considered in the previous section hence gives an upper bound for how well test time augmentation can work. We also include the results of using C4-TTA with ordinary DeDoDe-B in Table 5. Therefore, Tables 6 and 5 show that using test time augmentation can give higher performance than a steerer in Setting A (Section 5.1) of optimizing a steerer given a fixed descriptor. The steerers obtained in Settings B and C, however, clearly outperform test time augmentation for the original DeDoDe networks (compare Table 5 and Table 6). Table 4 presents the improved runtime of using steerers.

C. Experimental details

We use the publicly available training code from DeDoDe [19] to train our models. In Setting A, we train the steerer for 10k iterations with a learning rate of 0.01. In Setting B, we set the learning rate of the steerer to $2 \cdot 10^{-4}$, which is the same as for the decoder in [19] and train for 100k iterations as in [19]. In Setting C, we also train for 100k iterations. All other hyperparameters are identical to [19].

C.1. How to initialize/fix the steerer

Theorem 5.1 tells us that the representation acting on the description space (*i.e.* the steerer) should be orthogonal. Further, since we match using cosine similarity, we can perform an orthogonal change of basis in description space without influencing matching. Thus, using the representation theory described in Section 4, we can always change the basis of description space so that the steerer is block-diagonal with blocks of size 1 and 2. Next, we describe the exact forms of steerers in our experiments when using different initializations or fixed steerers. The labels correspond to the ones described in Section 6.1. Again we have two different cases depending on whether we have a C_4 steerer $\rho(\mathbf{g})$ or a $\mathrm{SO}(2)$ steerer obtained from a Lie algebra generator $\mathrm{d}\varsigma$.
Inv. Here, the steerer is simply the identity matrix.
Freq1. We set the steerer $\rho(\mathbf{g})$ or the Lie algebra generator $\mathrm{d}\varsigma$ to

$$\bigoplus_{b=1}^{128} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{33}$$

Each block has eigenvalues $\pm\mathbf{i}$, so we get 128 of each.
Perm. We set the steerer $\rho(\mathbf{g})$ to

$$\bigoplus_{b=1}^{64} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{34}$$

Each block has eigenvalues $\pm 1, \pm\mathbf{i}$, so we get 64 of each.
Spread. We set the Lie algebra generator $\mathrm{d}\varsigma$ to

$$\left(\bigoplus_{b=1}^{40} (0) \right) \oplus \left(\bigoplus_{j=1}^{6} \left(\bigoplus_{b=1}^{18} \begin{pmatrix} 0 & -j \\ j & 0 \end{pmatrix} \right) \right) \tag{35}$$

Here, the first 40×40 zero matrix gives 40 eigenvalues 0, the remaining blocks give 18 eigenvalues of each $\pm j\mathbf{i}$ for $j = 1, 2, 3, 4, 5, 6$.

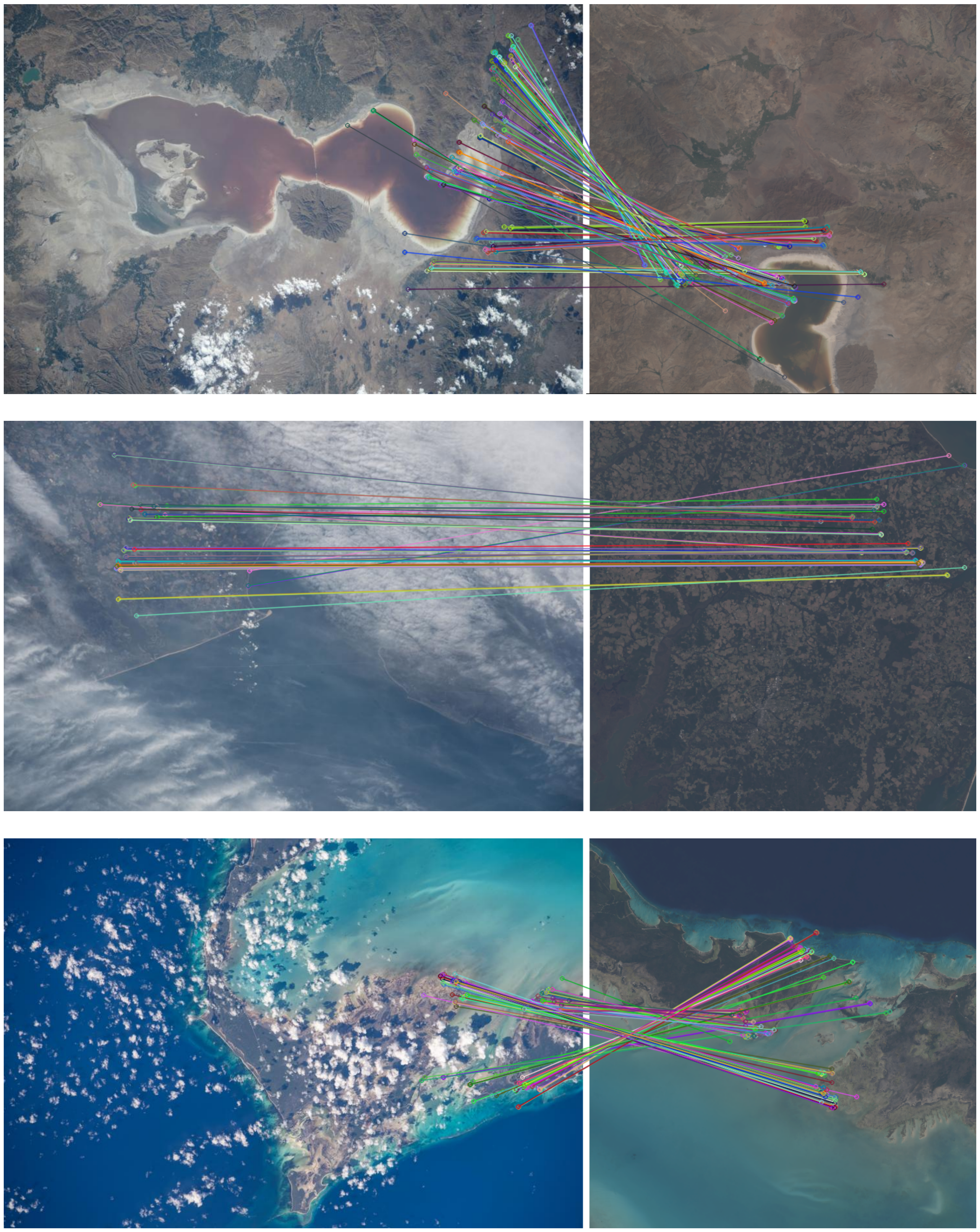

Figure 5. More qualitative challenging matching examples from the AIMS data.

C.2. AIMS details

In contrast to [49], we compute the average precision over the entire precision-recall curve instead of at fixed thresholds of the number of inliers. The thresholds used in [49] were chosen to approximately cover the precision-recall curve for the methods considered there. Still, we find using the complete precision-recall curve easier than rescaling the

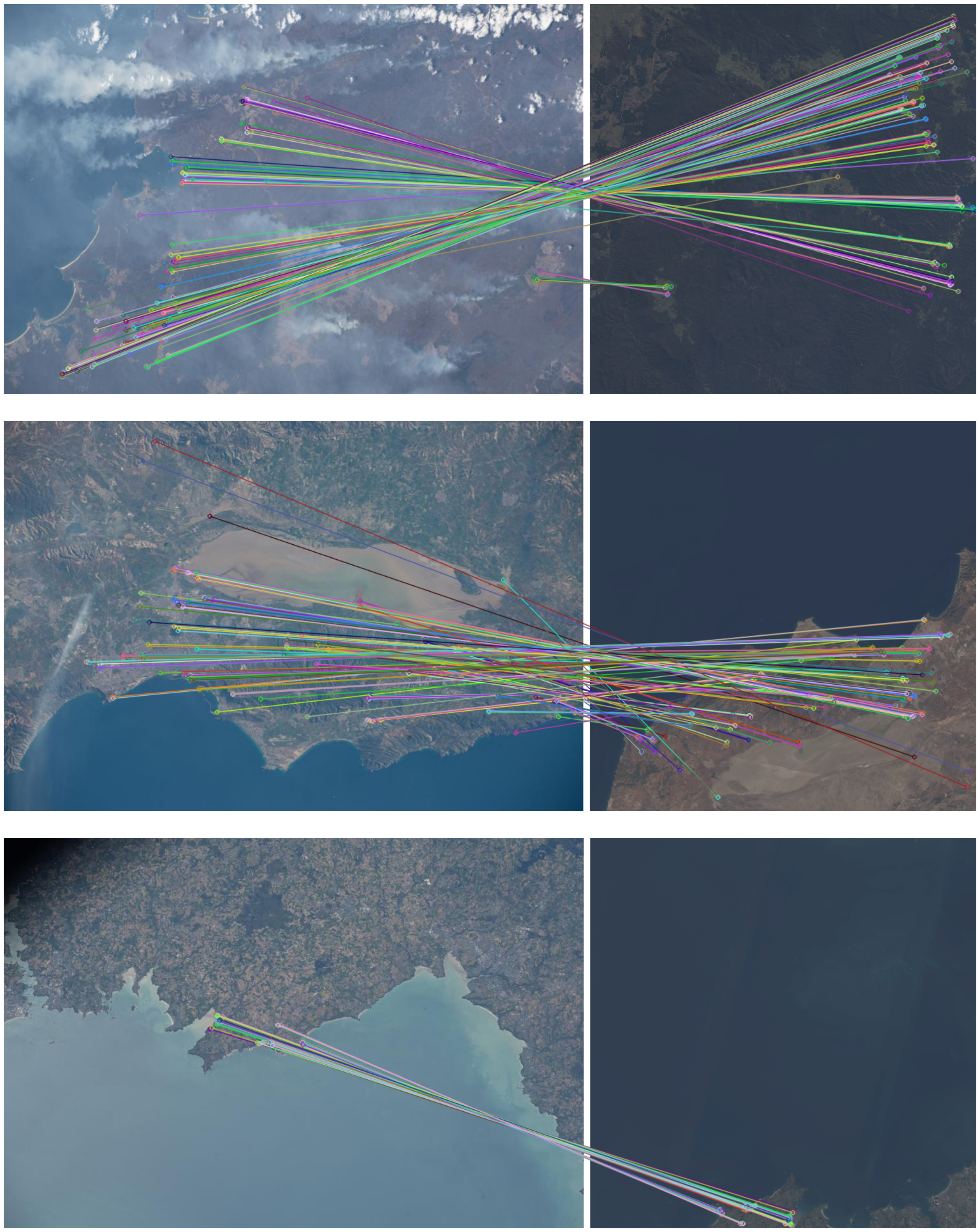

Figure 6. More qualitative challenging matching examples from the AIMS data.

thresholds for our methods. Furthermore, in [49], the average precision was computed with a maximum of 100 negative satellite images per astronaut photo. Instead, for each astronaut photo, we use all associated satellite images in the AIMS to get more accurate precision scores. These changes were agreed upon with the authors of [49].

We use only the Scale-1 subset of AIMS as we aim to evaluate rotational robustness. Following [49], we resize all images so that the smallest side is 576px during matching. For homography estimation, we use OpenCV with flag `USAC_MAGSAC` [5] with confidence 0.999, max iterations 10,000 and inlier threshold 5 pixels. These settings were given to us by the authors of [49]. We rerun SE2-LoFTR to compute the average precision as described above. We confirmed that we get approximately the same score using the old evaluation protocol for SE2-LoFTR as reported in [49] (they report 0.62 on Upright and 0.51 on All Others, while we get 0.60 and 0.52 respectively).

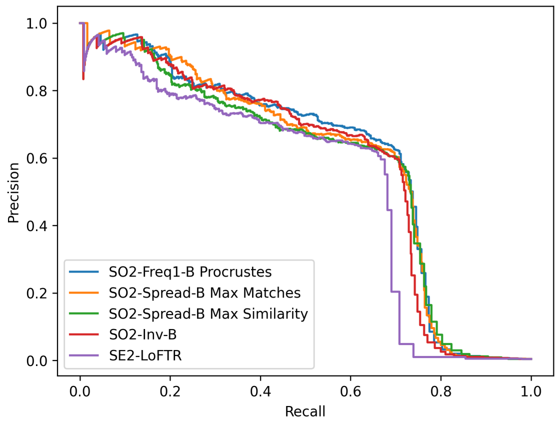

Figure 7. **Precision-recall on AIMS.** We plot precision-recall curves over the complete AIMS.

Table 4. **Runtime comparison**. We report the mean runtime over 100 random image tensors for description and matching on a single A100 GPU. I.e. the time for loading images and detection of keypoints is not measured. We use resolution 784×784 and 5,000 keypoints throughout.

Descriptor	Matching strategy	Time [ms]
DeDoDe-B	Dual softmax	63.4 ± 0.03
DeDoDe-B	Dual softmax + TTAx4	254.2 ± 0.10
DeDoDe-B	Dual softmax + TTAx8	513.0 ± 0.09
DeDoDe-B-C4	Max matches C4-steered	96.5 ± 0.05
DeDoDe-B-C4	Subset C4-steered	68.4 ± 0.10
DeDoDe-B-C4	Max similarity C4-steered	66.9 ± 0.02
DeDoDe-B-SO2	Max matches C8-steered	141.4 ± 0.05
DeDoDe-B-SO2	Subset C8-steered	73.4 ± 0.13
DeDoDe-B-SO2	Max similarity C8-steered	72.4 ± 0.02
DeDoDe-B-SO2	Procrustes	95.0 ± 0.04
DeDoDe-B-SO2	Prototype Procrustes	63.9 ± 0.02
DeDoDe-G	Dual softmax	217.3 ± 0.11
DeDoDe-G	Dual softmax + TTAx4	872.5 ± 0.11
DeDoDe-G-C4	Max matches C4-steered	250.4 ± 0.09
DeDoDe-G-C4	Subset C4-steered	222.9 ± 0.08
DeDoDe-G-C4	Max similarity C4-steered	221.3 ± 0.05

Table 5. **Evaluation on MegaDepth extended**. This is a larger version of Table 3. The first section shows Setting A where we only optimize the steerer, the second section shows Setting B where we jointly optimize the descriptor and steerer and the third section shows Setting C where we predefine the steerer and optimize only the descriptor. For MegaDepth-1500 we always use dual softmax matcher to evaluate the descriptors on upright images, except when the matching strategy is marked by *, in which case we use the specified matching strategy for MegaDepth-1500 as well. We use $20,000$ keypoints throughout. The best values for B- and G-models are highlighed in each column. See Section 6.1 for shorthand explanations for our models.

Detector DeDoDe	Descriptor DeDoDe	Matching strategy	AUC @	MegaDepth-1500 5°	10°	20°	MegaDepth-C4 5°	10°	20°	MegaDepth-SO2 5°	10°	20°
Original	B	Dual softmax		49	65	77	12	17	20	12	16	20
Original	B	Dual softmax + TTA C4		-"-	-"-	-"-	46	61	73	34	49	61
Original	B	Max matches C4-steered		-"-	-"-	-"-	43	60	73	30	44	56
C4	B	Max matches C4-steered		50	66	78	43	60	74	30	44	56
C4	B	Project to invariant subspace*		39	55	68	33	49	62	18	31	45
SO2	B	Max matches C4-steered		50	66	78	44	61	74	30	45	58
SO2	B	Max matches C8-steered		50	66	78	40	57	70	34	51	65
Original	G	Dual softmax		**52**	**69**	**81**	13	17	21	16	22	28
Original	G	Max matches C4-steered		-"-	-"-	-"-	31	45	57	26	39	50
C4	C4-B	Max matches C4-steered		**51**	**67**	**79**	**50**	**67**	**79**	39	55	68
C4	C4-B	Subset C4-steered		-"-	-"-	-"-	50	66	78	39	54	68
C4	C4-B	Max similarity C4-steered		50	**67**	**79**	49	65	78	35	50	62
SO2	SO2-B	Max matches C8-steered		47	63	76	47	63	76	44	61	74
SO2	SO2-Spread-B	Max matches C8-steered		50	66	**79**	49	66	78	**46**	**63**	**76**
SO2	SO2-Spread-B	Subset C8-steered		-"-	-"-	-"-	49	65	78	**46**	62	75
SO2	SO2-Spread-B	Max similarity C8-steered		49	66	78	47	64	77	43	61	74
C4	C4-Inv-B	Dual softmax		48	64	76	47	63	76	39	55	69
C4	C4-Perm-B	Max matches C4-steered		50	**67**	**79**	**50**	66	**79**	39	54	67
C4	C4-Freq1-B	Max matches C4-steered		49	66	78	49	65	78	36	51	64
SO2	SO2-Inv-B	Dual softmax		46	62	75	45	61	74	43	60	73
SO2	SO2-Freq1-B	Max matches C8-steered		47	64	77	47	64	76	45	62	75
SO2	SO2-Freq1-B	Procrustes		47	64	76	46	62	75	45	61	74
SO2	SO2-Freq1-B	Prototype Procrustes		44	61	74	43	60	73	41	58	72
C4	C4-Perm-G	Max matches C4-steered		**52**	**69**	**81**	**53**	**69**	**82**	**44**	**61**	**74**
C4	C4-Perm-G	Subset C4-steered		-"-	-"-	-"-	52	**69**	81	43	60	73

Table 6. **Performance on jointly rotated images vs steerer performance**. We evaluate three descriptors on image pairs where both images are rotated an equal multiple of 90° from upright. This gives an upper bound on how good the performance of test time augmentation can be. We compare to the performance of a steerer trained for the fixed descriptor (Setting A). Finally we show the performance of a descriptor DeDoDe-B† which is trained with data augmentation with jointly rotated images. For DeDoDe-B† we also use Setting A, so it is trained without a steerer and then a steerer is trained with the fixed descriptor. We use $20,000$ DeDoDe keypoints throughout.

Descriptor	AUC @	MegaDepth-1500 5°	10°	20°	MegaDepth-1500 joint rotation 5°	10°	20°	MegaDepth-C4 with steerer 5°	10°	20°
DeDoDe-B [19]		49	65	77	46	62	74	43	60	73
DeDoDe-G [19]		52	69	81	45	61	74	31	45	57
DISK [55]		34	49	62	29	45	58	26	41	54
DeDoDe-B†		50	66	78	50	66	78	50	66	78

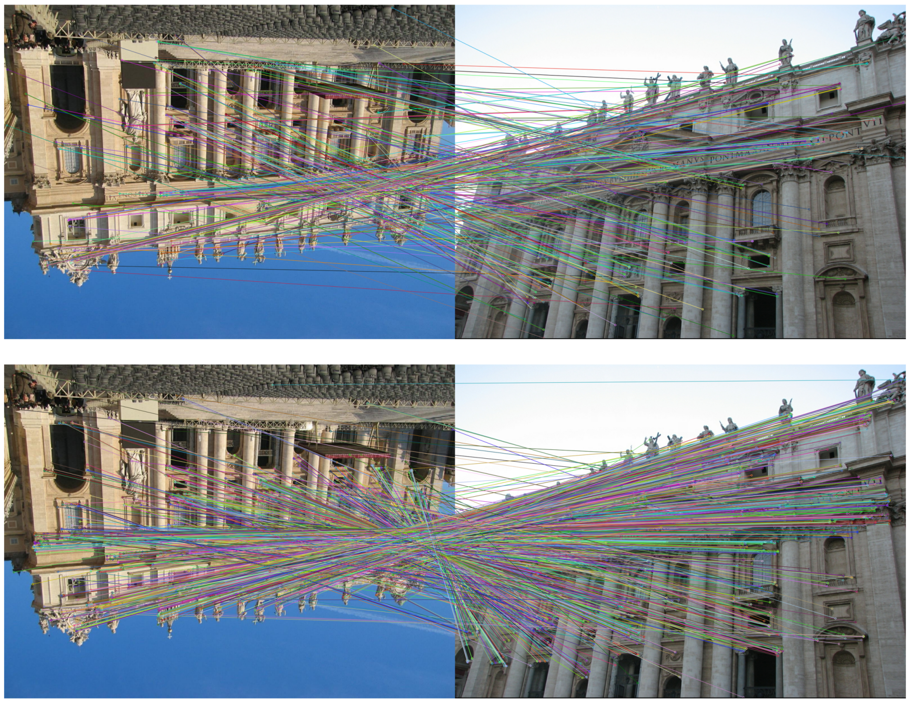

Figure 8. **Steering DeDoDe descriptions under half turn rotations.** We replicate [19, Figure 7] but with a steerer. In the upper image pair, we match the ordinary DeDoDe-descriptions. In the lower image pair, we instead modify the descriptions of the keypoints in the right image by multiplying them by a steering matrix $\rho(\mathbf{g})^2$. This corresponds to setting A, where we, for a fixed descriptor, have optimized a steerer.