Cov2Pose: Leveraging Spatial Covariance for Direct Manifold-aware 6-DoF Object Pose Estimation

Supplementary Material

This supplementary material presents implementation details, detailed ablation studies, a pilot discussion on symmetry-aware training and extended experiments for Cov2Pose. We describe optimization and architecture choices, ablate key components of the proposed method, and report detailed per-object and cross-dataset results, as well as additional validation on Absolute camera Pose Regression (APR) on Cambridge Landmarks [10] and the more challenging spacecraft pose estimation task on SPEED+ [14], further demonstrating the applicability of the proposed method.

A. Additional Implementation Details

Ground-truth Preprocessing. As shown in (9) the 6D rotation and the 3D translation are encoded in the non-zero elements of the lower-triangular matrix $\mathbf{L}$. For rotation, we use the full $\mathbf{R}_{\text{gt}} \in \mathcal{SO}(3)$ matrix as the rotation target/objective. For translation, we compute per-axis statistics on the training split, *i.e.*, 99$^{\text{th}}$-percentile-based per-axis minima $\mathbf{t}_{\min}$ and per-axis ranges $\mathbf{t}_{\text{range}}$, and form normalized targets such that,

$$\mathbf{t}_{\text{gt}} = \big(\mathbf{t}_{\text{raw}} - \mathbf{t}_{\min}\big)/\mathbf{t}_{\text{range}}, \tag{a}$$

where $\mathbf{t}_{\text{raw}}$ are the original translation values from the dataset. We train with the normalized translation as objective and at inference, the network predicts $\mathbf{t}^{'}$ which we de-normalize with the same fixed statistics,

$$\mathbf{t} = \mathbf{t}^{'} \times \mathbf{t}_{\text{range}} + \mathbf{t}_{\min}. \tag{b}$$

Note that the division and product here are performed axis-wise. This preprocessing keeps targets well-scaled without changing the metric translation recovered at test time.

Stiefel–constrained Optimizer for BiMap Weights. We maintain each projection matrix at layer l on the Stiefel manifold $\mathbf{W}_l \in \mathcal{V}_n(\mathbb{R}^m)$ using a projected gradient step followed by a QR–based retraction. Given a gradient $\mathbf{G} = \nabla_{\mathbf{W}}\mathcal{L}_{\text{pose}}$, where $\mathcal{L}_{\text{pose}}$ is the training loss defined in (10), we first project this gradient onto the tangent space at $\mathbf{W}_l$,

$$\mathbf{G}_{\top} = \mathbf{G} - \mathbf{W}_l \,\text{sym}\big(\mathbf{W}_l^{\top}\mathbf{G}\big), \tag{c}$$

where sym denotes the symmetrization operator*. We then take a step along $-\mathbf{G}_{\top}$ with step size η,

$$\widetilde{\mathbf{W}}_l = \mathbf{W}_l - \eta\,\mathbf{G}_{\top}, \tag{d}$$

*For a square matrix $\mathbf{A}$, $\text{sym}(\mathbf{A}) = \frac{1}{2}(\mathbf{A} + \mathbf{A}^{\top})$.

and retract back to $\mathcal{V}_n(\mathbb{R}^m)$ via the $\mathbf{Q}$ factor of a reduced QR decomposition,

$$\widetilde{\mathbf{W}}_l = \mathbf{Q}_l\mathbf{R}_l, \qquad \mathbf{W}_l^{+} = \mathbf{Q}_l, \tag{e}$$

where $\mathbf{W}_l^{+}$ denotes the updated Stiefel-constrained weight at layer l after the retraction. In our implementation, we perform the QR decomposition in double precision when enabled, for improved numerical stability.

B. Additional Details on Ablation Studies

We provide additional details on the variants that we design in Section 5.5. Table S1 shows the results on LM-O [2] per-object of the ablation experiments.

Table S1. Per-object results on **LM-O**. We compare our full method with ablated variants of Cov2Pose to isolate the contribution of each component of the SPD head. This table is the detailed per-object version of Table 5.

Variant	(A)	(B)	(C)	Cov2Pose
Ape	27.2	56.1	62.6	64.0
Can	40.8	93.8	94.4	96.4
Cat	15.7	70.0	66.1	74.1
Driller	26.7	88.1	91.9	96.3
Duck	16.7	49.3	48.1	55.8
Eggbox	47.5	48.0	53.6	57.9
Glue	42.4	91.7	87.0	93.6
Holep.	30.9	70.1	74.7	76.8
Avg. ADD(-S)↑	31.0	70.9	72.3	**76.8**

(A) Euclidean Regression Head. We replace the regression head with a purely Euclidean baseline composed of a two–layer MLP as illustrated in Figure S1. From spatial covariance $\hat{\mathbf{\Sigma}} \in \mathcal{S}_{++}^{N}$ we form a vector $\text{vec}(\hat{\mathbf{\Sigma}}) \in \mathbb{R}^{\frac{N(N+1)}{2}}$ by stacking its upper–triangular entries *i.e.* unique elements. The MLP maps $\hat{\mathbf{\Sigma}} \to \mathbb{R}^{256} \to \mathbb{R}^{9}$, directly regressing the 6D rotation parameters $(\mathbf{u}, \mathbf{v})$ and the 3D translation $\mathbf{t}$. As reported in Table 5, this setup significantly decreases performance because learning in Euclidean space ignores the Riemannian geometry of the SPD manifold and collapses the second-order spatial structure.

(B) Channel-wise Covariance. Given backbone features $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$, we flatten the spatial grid to obtain $\mathbf{X} \in \mathbb{R}^{C\times N}$ with $N{=}HW$, where each row is a channel and each column a spatial location. We remove the spatial mean per channel to obtain $\tilde{\mathbf{X}}$ and form the channel-wise covariance,

$$\hat{\mathbf{\Sigma}}_{\text{ch}} = \tfrac{1}{N-1}\,\tilde{\mathbf{X}}\,\tilde{\mathbf{X}}^{\top} \in \mathcal{S}_{++}^{C}. \tag{f}$$

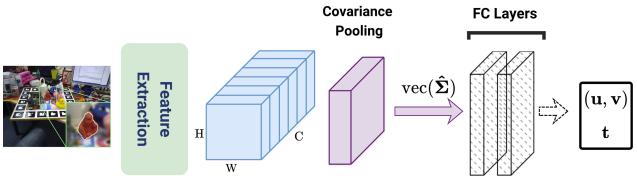

Figure S1. **Variant (A)**. The spatial covariance matrix $\hat{\mathbf{\Sigma}}$ is vectorized and fed to a two-layer MLP that regresses pose purely in Euclidean space.

This descriptor summarizes second-order co-activations between channels aggregated over all spatial locations. Table 5 shows that spatial covariance empirically surpasses channel covariance, as it preserves pairwise relations between image regions that vary with viewpoint, whereas channel covariance collapses the spatial layout and cannot distinguish configurations with different spatial arrangements. It also requires more operations to construct; Forming channel covariance costs $\mathcal{O}(C^2N)$, while the spatial covariance costs $\mathcal{O}(N^2C)$. Since in late backbone stages $C \gg N$ (in our setup C=2304, $H\times W$=17×17 $\Rightarrow$ N=289), channel covariance is markedly heavier. Empirically, on the same device, it runs at 168 ms per image, versus 46.9 ms for spatial covariance, further supporting the use of spatial covariance.

(C) Training on the log–tangent space of the SPD manifold. This variant keeps the SPD head and removes the Cholesky pose decoder. Let $\mathbf{Z}_L \in \mathcal{S}^4_{++}$ be the SPD output of the head composed of L layers, and let $\mathbf{Z}_{\text{gt}} \in \mathcal{S}^4_{++}$ be the precomputed SPD pose label obtained from the ground-truth pose in a manner similar to Equation (9). We train by aligning both in the tangent space using a Log–Euclidean map and use a Frobenius loss

$$\mathcal{L}_{\text{Fro}} = \left\| \operatorname{LogEig}(\mathbf{Z}_L) - \operatorname{LogEig}(\mathbf{Z}_{\text{gt}}) \right\|_F^2 , \qquad \text{(g)}$$

where LogEig is the matrix logarithm operation [7]. No pose loss is used here; this design shows the relevance of using our Cholesky decoder to learn the pose parameters. The process is illustrated in Figure S2.

C. Rotation Representation using Euler Angles

To assess the benefit of using a continuous rotation representation in $\mathcal{SO}(3)$, we also design an alternative encoding using a lower–dimensional variant that encodes the rotation using Euler angles which are known to be discontinuous due to Gimbal lock [21]. In this variant, we regress $\mathbf{Z}_L \in \mathcal{S}^3_{++}$ and the mapping from (8) becomes

$$\begin{array}{rccc} \mathbf{\Psi}_- & : & \mathcal{S}^3_{++} & \longrightarrow & \mathcal{P}_- \\ & & \mathbf{Z}_L & \longmapsto & \mathbf{\Psi}_-(\mathbf{Z}_L) = \mathbf{P} \,, \end{array} \qquad \text{(h)}$$

where $\mathbf{P} = (\boldsymbol{\theta}, \mathbf{t})$ is the pose with $\mathbf{t} = (t_x, t_y, t_z) \in \mathbb{R}^3$ the translation vector and $\boldsymbol{\theta} = (\theta_x, \theta_y, \theta_z) \in [0, 2\pi)^3$ the three Euler angles (roll, pitch and yaw) that represent the object's rotation in 3D space w.r.t. the camera frame. The corresponding pose space is $\mathcal{P}_- = [0, 2\pi)^3 \times \mathbb{R}^3$ which is a discontinuous pose representation due to the wrap–around of the Euler angles (at $0/2\pi$) and the gimbal–lock singularities. For encoding the 3D rotation representation (3-dimensional) as well as the translation vector (3-dimensional) we need a matrix with at least 3 non-zero, thus fixing n to 3, the Cholesky decomposition such that $\mathbf{Z}_L = \mathbf{L}\mathbf{L}^\top \in \mathcal{S}^3_{++}$ gives a 3×3 lower-triangular matrix that we use to encode the pose parameters such that,

$$\mathbf{L} = \begin{bmatrix} e^{\theta_x} & 0 & 0 \\ t_x & e^{\theta_y} & 0 \\ t_y & t_z & e^{\theta_z} \end{bmatrix} . \qquad \text{(i)}$$

For training, we keep an identical configuration as the default 6D + differentiable GS we propose for our method, the results comparing the two rotation parameterizations are shown in Table 4 of the main paper.

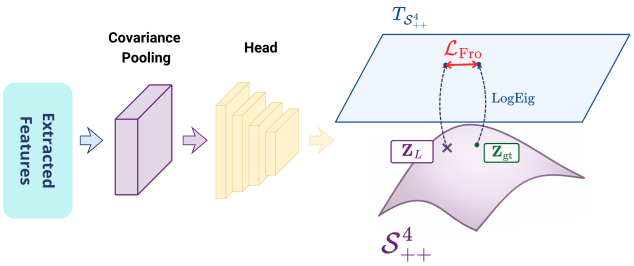

Figure S2. **Variant (C)**. We remove the Cholesky-based pose decoder and train on the log-tangent space of the SPD manifold. $\mathbf{Z}_L$ and $\mathbf{Z}_{\text{gt}}$ are mapped via LogEig to the log–tangent space. For clarity, a single tangent space $T_{\mathcal{S}^4_{++}}$ is depicted.

D. Computation and memory requirements

We further provide a brief analysis of the scaling trade-offs between the spatial covariance size and the spatial resolution ($N\times N$). As discussed in Section B, CovPool has time complexity $\mathcal{O}(N^2C)$ and memory complexity $\mathcal{O}(N^2)$. Table S2 shows that the CovPool runtime is negligible (less than 1 ms), while the SPD head dominates the computation and increases with N. Therefore, we use, and recommend using, late, low-resolution features, or keeping the spatial resolution moderate via downsampling/pooling.

Table S2. CovPool layer and SPD head latencies across different spatial resolutions.

$H\times W$	CovPool (ms)	SPD Head (ms)
8^2	0.38	7.8
17^2	**0.50**	**23.8**
25^2	0.61	68.6

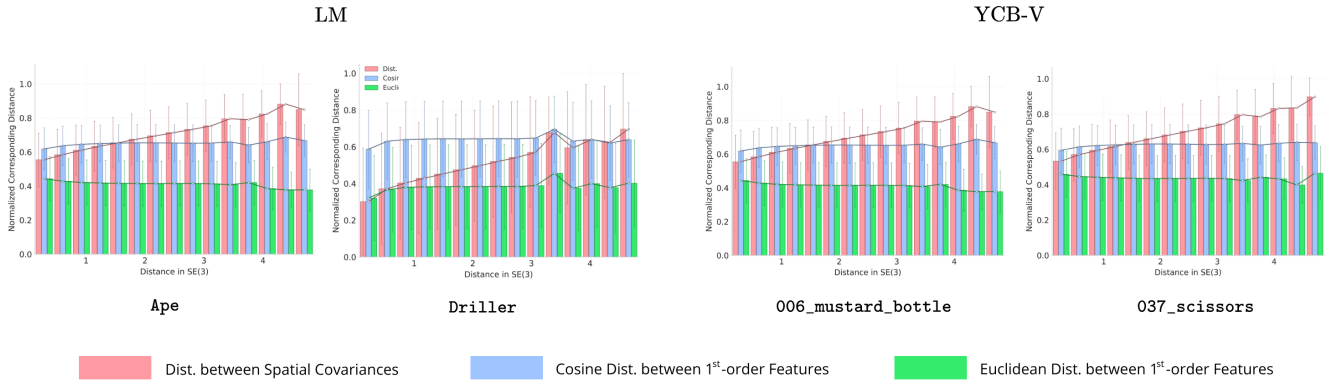

Figure S3. **Additional Pose–representation alignment.** As in Figure 1, image pairs are sorted by ground-truth SE(3) distance. We plot Log–Euclidean distances between spatial covariances and, for comparison, cosine and Euclidean distances between flattened features. Covariance distances grow with distances in SE(3), while flattened-feature distances remain nearly constant, reinforcing the hypothesis that spatial covariances are more correlated with true pose.

Regarding memory usage, the same setup described in Section B requires ≈2.54 MB to store $\mathbf{X}$ and ≈0.32 MB to store $\mathbf{\Sigma}$ per input (both in fp32). During backpropagation, activations are also retained, resulting in an overall memory scaling of $\mathcal{O}(CN + N^2)$. Additional buffers (e.g., centered $\mathbf{X}$ and eigen/QR workspaces) have the same asymptotic order.

E. Generality beyond CNN backbones

We show that `Cov2Pose` also extends beyond CNN backbones. To illustrate this, we use a ViT_b_16 backbone [4]. Instead of computing the covariance over spatial locations (pixels), we compute it over the patch tokens extracted by the ViT backbone. Importantly, because ViTs use global self-attention, the resulting SPD matrix does not represent purely "spatial" covariance. Nevertheless, a small pilot experiment with ViT_b_16 indicates that a token-covariance + SPD head still outperforms a ViT_b_16 baseline that regresses pose parameters using an MLP. The results are shown in Table S3.

Table S3. Pilot comparison on *Ape* (LM-O) with a ViT_b_16 backbone. Results reported in ADD.

Class (Ape; LM-O)	ADD↑
ViT+MLP	33.3
ViT+Cov+SPD head	**40.6**

F. Pilot study on symmetry handling and point-based losses

Object symmetries remain a well-known challenge in 6D object pose estimation. `Cov2Pose` does not explicitly model symmetries, as its main design targets a CAD-free, RGB-only, geometry-agnostic, and fully differentiable formulation. Nevertheless, in the main results we did not observe a specific degradation on the symmetric LM/LM-O objects *eggbox* and *glue*, motivating a small pilot study to assess whether adding symmetry-oriented supervision improves performance.

A natural way to address symmetric ambiguities is to use point-based losses such as ADD-S, or symmetry-aware rotation losses [12, 18]. However, these losses require CAD model points and/or symmetry annotations during training. This changes the training assumptions of `Cov2Pose`.

Pilot experiment. We trained variants of `Cov2Pose` on the symmetric objects of LM-O by replacing the pose supervision with: (i) an ADD-S-inspired point-based loss that matches predicted and ground-truth transformed 3D points by nearest neighbors, and (ii) a symmetry-aware rotation loss from prior work similar to [18]. We evaluate on the symmetric objects *Eggbox* and *Glue* using ADD-S, following the standard BOP protocol.

Results. Table S4 summarizes the pilot results. The ADD-S-inspired loss does not improve performance in this setting and can degrade results, while the symmetry-aware loss matches or only slightly improves the baseline. In particular, on *Eggbox*/*Glue*, where `Cov2Pose` achieves 57.9/93.6 ADD-S, the ADD-S-inspired variant reaches 55.0/84.8, and the symmetry-aware loss reaches 57.8/93.8.

Discussion. These pilot results suggest that simply plugging in point-based or symmetry-aware losses is not sufficient to consistently improve `Cov2Pose`, despite introducing stronger geometric supervision. Moreover, the two pilot variants depart from the original formulation by relying on operations/supervision that break the fully differentiable

end-to-end design of Cov2Pose and its CAD-free training assumptions. We therefore view explicit symmetry handling as an important but non-trivial extension, especially under the constraints targeted by our framework. Designing symmetry-aware training/inference strategies that preserve the CAD-free and fully differentiable nature of Cov2Pose, as in [3, 16], is a promising future direction.

Table S4. Pilot study on symmetric LM-O objects using point-based and symmetry-aware losses.

Method	CAD?	*Egg.* ADD-S↑	*Glu.* ADD-S↑
Cov2Pose	×	**57.9**	93.6
Cov2Pose+ $\mathcal{L}_{\text{ADD-S}}$	✓	55.0	84.8
Cov2Pose+ $\mathcal{L}_{\mathbf{R},\text{sym}}$ [18]	✓	57.8	**93.8**

G. Additional Results & Analysis for Figure 1

In Figure 1, we show that Log–Euclidean distances between spatial covariances of CNN-extracted feature maps grow with ground-truth $\mathrm{SE}(3)$ pose distance, whereas cosine distances between flattened features remain nearly constant. This reinforces the hypothesis that a spatial–covariance representation is more pose-sensitive than flattened activations, and thus more suitable for direct regression.

To achieve the result represented in Figure 1, we use an EfficientNet-B6 [17] backbone pretrained on ImageNet, with no pose-specific finetuning and no augmentation to avoid task leakage and isolate the representational contrast between spatial covariance and flattened features.

Figure S3 presents analogous experiments on additional classes from LineMod [5] and YCB-V [20] training data. We report cosine distances, and additionally Euclidean distances, between flattened features, both of which remain nearly constant across pose separations, while Log Euclidean distances between spatial covariances increase, confirming that spatial covariance is a pose-sensitive representation and more correlated with the poses.

H. Additional Results on APR

To assess generality beyond object pose, we evaluate Cov2Pose on the task of Absolute camera Pose Regression (APR) using the Cambridge Landmarks benchmark [10]. The dataset comprises four outdoor scenes (King's College, Old Hospital, Shop Facade and St Mary's Church) with labeled video frames and standard train/test splits. We keep the same pipeline using spatial covariance pooling, BiMap+ReEig SPD head, and the SPD(4) Cholesky decoder. Following the PoseNet protocol, we report median translation [m] and rotation [°] errors. As shown in Table S5, compared with the PoseNet family of baselines (PoseNet [10], Bayesian PoseNet [8], and GPoseNet [9]), Cov2Pose achieves lower median translation and rotation errors across all scenes and on average, indicating that spatial second-order cues and manifold-aware learning transfer effectively to scene-centric APR.

Table S5. APR results on **Cambridge Landmarks** [10]. Median translation [m] and rotation [°] errors per scene and their average (Lower is better). Cov2Pose achieves the best performance across all scenes and on average.

Method	K. College	Hospital	Shop Facade	St. Mary	Avg.↓
PoseNet [10]	1.92, 5.40°	2.31, 5.38°	1.46, 8.08°	2.65, 8.48°	2.08, 6.83°
BayesianPN [8]	1.74, 4.06°	2.57, 5.14°	1.25, 7.54°	2.11, 8.38°	1.91, 6.28°
GPoseNet [9]	1.61, 2.29°	2.62, 3.89°	1.14, 5.73°	2.93, 6.46°	2.07, 4.59°
Ours	**1.57**, **1.81°**	**2.08**, **2.35°**	**1.11**, **5.33°**	**2.06**, **5.00°**	**1.70**, **3.56°**

I. YCB-V Dataset Per-object Evaluation

In Table S6, we report a more detailed overview of the results per-object on the YCB-V dataset [20]. For the sake of space, we report only the results of End-to-End pose learning methods.

J. Additional Results on SPEED+

We further evaluate Cov2Pose on SPEED+ [14], a spacecraft pose estimation dataset where PnP-based solutions currently predominate [1, 13, 15, 19]. Cov2Pose is trained on the synthetic train split and evaluated on its test set. To assess cross-domain generalization, we report inference results on the additional test domains of SPEED+, namely, lightbox, simulating diffuse orbital illumination, and sunlamp, replicating direct sunlight via a specialized lamp setup. As summarized in Table S7, Cov2Pose surpasses SPNv2 [13] across all metrics while being faster at inference. It also outperforms the PnP-based YOLOv8s-pose [1] on the lightbox split and achieves superior translation accuracy on sunlamp. These results further support the applicability of Cov2Pose to spacecraft pose estimation. Qualitative results are shown in Figure S4.

Table S7. **Results on SPEED+.** Performance in terms of Translation E_T, Rotation E_R and Pose E_{pose} errors as defined in [13]. The inference time of SPNv2 is measured on the same device and at the same input resolution (768×512) as ours.

		YOLOv8s-pose [1]	†SPNv2 [13]	**Ours**
	Latency (ms)	—	80.25	**60.46**
Synthetic	E_T [m]	—	—	**0.184**
	E_R [°]	—	—	**3.462**
	E_{pose}	—	—	**0.089**
Lightbox	E_T [m]	0.758	0.368	**0.300**
	E_R [°]	18.00	20.258	**15.096**
	E_{pose}	0.432	0.411	**0.3101**
Sunlamp	E_T [m]	0.518	0.457	**0.430**
	E_R [°]	**19.80**	34.916	29.128
	E_{pose}	**0.432**	0.682	0.574

†Results from the direct pose regression head of SPNv2.

Table S6. **YCB-V** per-object results in terms of ADD(-S) and AUC-S/AUC(-S).“—” indicates values not reported in the original paper. Classes denoted with * are symmetric.

Method	PoseCNN [20]			Single-Stage [6]†			DeepIM [11]			GDR-Net [18]†			**Ours**		
	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)
002_master_chef_can	—	84.0	50.9	—	—	—	—	—	—	41.5	96.3	65.2	61.0	94.7	63.1
003_cracker_box	—	76.9	51.7	—	—	—	—	—	—	83.2	97.0	88.8	86.7	91.2	79.8
004_sugar_box	—	84.3	68.6	—	—	—	—	—	—	91.5	98.9	95.0	94.5	94.7	89.8
005_tomato_soup_can	—	80.9	66.0	—	—	—	—	—	—	65.9	96.5	91.9	73.8	95.6	89.7
006_mustard_bottle	—	90.2	79.9	—	—	—	—	—	—	90.2	100.0	92.8	93.3	93.2	84.2
007_tuna_fish_can	—	87.9	70.4	—	—	—	—	—	—	44.2	99.4	94.2	68.7	97.1	94.2
008_pudding_box	—	79.0	62.9	—	—	—	—	—	—	2.8	64.6	44.7	98.7	96.5	93.2
009_gelatin_box	—	87.1	75.2	—	—	—	—	—	—	61.7	97.1	92.5	96.0	97.0	93.6
010_potted_meat_can	—	78.5	59.6	—	—	—	—	—	—	64.9	86.0	80.2	67.5	87.9	78.0
011_banana	—	85.9	72.3	—	—	—	—	—	—	64.1	96.3	85.8	54.7	90.1	75.4
019_pitcher_base	—	76.8	52.5	—	—	—	—	—	—	99.0	99.9	98.5	90.2	95.9	89.2
021_bleach_cleanser	—	71.9	50.5	—	—	—	—	—	—	73.8	94.2	84.3	75.7	92.3	81.9
024_bowl*	—	69.7	69.7	—	—	—	—	—	—	37.7	85.7	85.7	10.7	73.6	73.6
025_mug	—	78.0	57.7	—	—	—	—	—	—	61.5	99.6	94.0	63.5	90.5	76.4
035_power_drill	—	72.8	55.1	—	—	—	—	—	—	78.5	97.5	90.1	83.7	94.7	85.0
036_wood_block*	—	65.8	65.8	—	—	—	—	—	—	59.5	82.5	82.5	63.6	85.7	85.7
037_scissors	—	56.2	35.8	—	—	—	—	—	—	3.9	63.8	49.5	46.6	85.3	70.1
040_large_marker	—	71.4	58.0	—	—	—	—	—	—	7.4	88.0	76.1	16.0	78.8	68.3
051_large_clamp*	—	49.9	49.9	—	—	—	—	—	—	69.8	89.3	89.3	72.0	84.1	84.1
052_extra_large_clamp*	—	47.0	47.0	—	—	—	—	—	—	90.0	93.5	93.5	88.0	92.5	92.5
061_foam_brick*	—	87.8	87.8	—	—	—	—	—	—	71.9	96.9	96.9	58.6	78.3	78.3
Avg. ↑	—	75.9	61.3	53.9	—	—	—	88.1	81.9	60.1	91.6	84.3	69.7	90.0	82.2

† Method is End-to-End by using a differentiable PnP strategy.

References

[1] Michele Bechini and Michèle Lavagna. Robust and efficient single-cnn-based spacecraft relative pose estimation from monocular images. *Acta Astronautica*, 233:198–217, 2025. 4

[2] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *Computer Vision – ECCV 2014*, pages 536–551, Cham, 2014. Springer International Publishing. 1

[3] Dingding Cai, Janne Heikkilä, and Esa Rahtu. Sc6d: Symmetry-agnostic and correspondence-free 6d object pose estimation. In *2022 International Conference on 3D Vision (3DV)*, pages 536–546. IEEE, 2022. 4

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3

[5] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Computer Vision – ACCV 2012*, pages 548–562, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 4

[6] Yinlin Hu, Pascal Fua, Wei Wang, and Mathieu Salzmann. Single-stage 6d object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2930–2939, 2020. 5

[7] Zhiwu Huang and Luc Van Gool. A riemannian network for spd matrix learning. In *Proceedings of the AAAI conference on artificial intelligence*, 2017. 2

[8] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016. 4

[9] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5974–5983, 2017. 4

[10] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 1, 4

[11] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 683–698, 2018. 5

[12] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7668–7677, 2019. 3

[13] Tae Ha Park and Simone D'Amico. Robust multi-task learning and online refinement for spacecraft pose estimation across domain gap. *Advances in Space Research*, 73(11): 5726–5740, 2024. 4

[14] Tae Ha Park, Marcus Märtens, Gurvan Lecuyer, Dario Izzo, and Simone D'Amico. Speed+: Next-generation dataset for spacecraft pose estimation across domain gap. In *2022 IEEE aerospace conference (AERO)*, pages 1–15. IEEE, 2022. 1, 4, 6

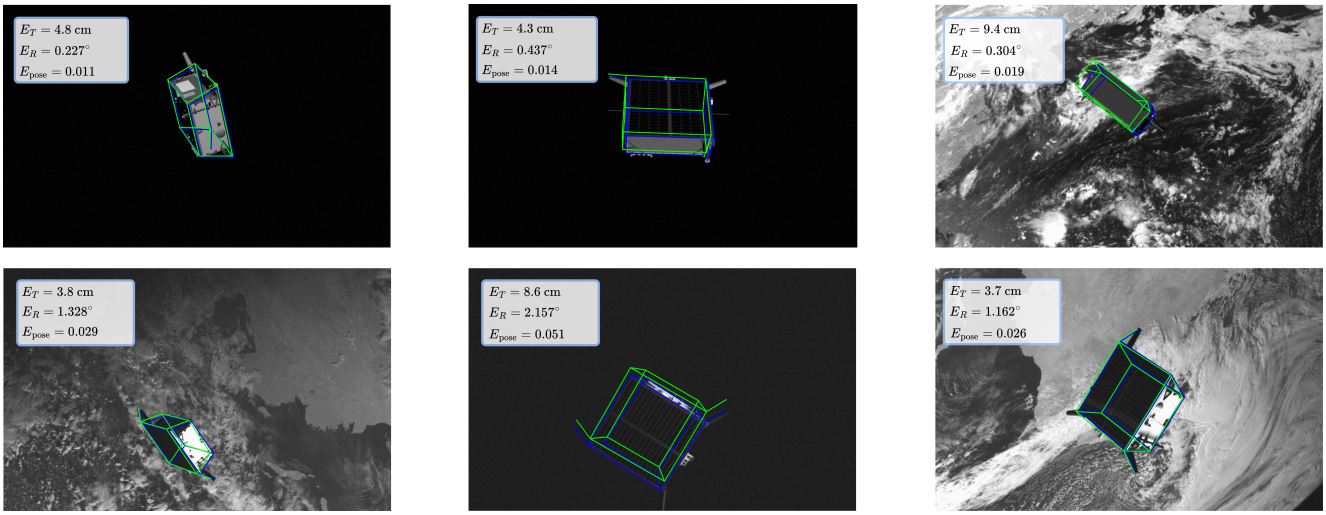

Figure S4. **Qualitative results on SPEED+.** Examples of accurate pose predictions from `Cov2Pose` on the `synthetic` (1st row) and `lightbox` (2nd row) subsets of SPEED+ [14]. Green bounding boxes show predicted poses, while blue bounding boxes indicate ground-truth poses.

[15] Tae Ha Park, Marcus Märtens, Mohsi Jawaid, Zi Wang, Bo Chen, Tat-Jun Chin, Dario Izzo, and Simone D'Amico. Satellite pose estimation competition 2021: Results and analyses. *Acta Astronautica*, 204:640–665, 2023. 4

[16] Giorgia Pitteri, Michaël Ramamonjisoa, Slobodan Ilic, and Vincent Lepetit. On object symmetries and 6d pose estimation from images. In *2019 International conference on 3D vision (3DV)*, pages 614–622. IEEE, 2019. 4

[17] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 4

[18] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16611–16621, 2021. 3, 4, 5

[19] Zi Wang, Minglin Chen, Yulan Guo, Zhang Li, and Qifeng Yu. Bridging the domain gap in satellite pose estimation: A self-training approach based on geometrical constraints. *IEEE transactions on aerospace and electronic systems*, 60 (3):2500–2514, 2023. 4

[20] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2018. 4, 5

[21] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5745–5753, 2019. 2