Appendix:
The Surprising Effectiveness of Visual Odometry Techniques for Embodied PointGoal Navigation

This appendix is structured as follows:

- Sec. A provides the formal derivation of the geometric invariance loss described in Sec. 3.2.
- Sec. B describes the technical details to generate the egocentric top-down projection discussed in Sec. 3.4.
- Sec. C describes the navigation policy's architecture and hyperparameters used for training.
- Sec. D gives details about the visual odometry model's training and inference.
- Sec. E states implementation details of DeepVO as well as our model's performance on KITTI.
- Sec. F demonstrates that we cannot accurately estimate relative pose from depth due to sensor's noises.
- Sec. G provides more qualitative results to demonstrate the performance of our model.

A. Formal Derivation for Geometric Invariance Loss

Recall that we predict $\widehat{H}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \in SE(2)$ from two consecutive egocentric observations (I_t, I_{t+1}). Intuitively, invariance is obtained when observing (I_t, I_{t+1}) followed by (I_{t+1}, I_t). Due to the invertibility of transformations between coordinate systems $\mathcal{C}_t$ and $\mathcal{C}_{t+1}$, we have the following relation between ground-truth transformations:

$$H_{\mathcal{C}_t \to \mathcal{C}_{t+1}} H_{\mathcal{C}_{t+1} \to \mathcal{C}_t} = I_{3\times 3}, \tag{S1}$$

where $I_{3\times 3}$ is the three-dimensional identity matrix.

Meanwhile, an element from $SE(2)$ is defined as follows:

$$H_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \triangleq \begin{bmatrix} R_{\mathcal{C}_t \to \mathcal{C}_{t+1}} & \boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \\ & 1 \end{bmatrix}$$
$$\text{where } R_{\mathcal{C}_t \to \mathcal{C}_{t+1}} = \begin{bmatrix} \cos(\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}) & -\sin(\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}) \\ \sin(\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}) & \cos(\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}) \end{bmatrix}. \tag{S2}$$

Note, the rotation matrix can be computed via $R_{\mathcal{C}_t \to \mathcal{C}_{t+1}} = \exp\left(\texttt{alg}(\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}})\right)$, *i.e.*, by applying the exponential map $\exp$ on $\texttt{alg} : \mathbb{R} \mapsto \mathbb{R}^{2\times 2}$, the function that maps an angle from $\mathbb{R}$ to an element of the Lie algebra $\mathfrak{so}(2)$, namely $\texttt{alg}(\theta) = \theta \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$. When replacing the rotation matrix in Eq. (S2) with this representation and expanding the relation given in Eq. (S1), we obtain:

$$\begin{bmatrix} \exp\left(\texttt{alg}(\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}})\right) & \boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \\ & 1 \end{bmatrix} \cdot \begin{bmatrix} \exp\left(\texttt{alg}(\theta_{\mathcal{C}_{t+1} \to \mathcal{C}_t})\right) & \boldsymbol{\xi}_{\mathcal{C}_{t+1} \to \mathcal{C}_t} \\ & 1 \end{bmatrix} = I_{3\times 3}. \tag{S3}$$

After multiplying out the left-hand side we obtain the following system of equations:

$$\begin{cases} \exp\left(\texttt{alg}(\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}} + \theta_{\mathcal{C}_{t+1} \to \mathcal{C}_t})\right) = I_{2\times 2} \\ \exp\left(\texttt{alg}(\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}})\right) \cdot \boldsymbol{\xi}_{\mathcal{C}_{t+1} \to \mathcal{C}_t} + \boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} = \mathbf{0} \end{cases}. \tag{S4}$$

Upon simplification, this results in

$$\begin{cases} \theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}} + \theta_{\mathcal{C}_{t+1} \to \mathcal{C}_t} = 0 \\ \exp\left(\texttt{alg}(\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}})\right) \cdot \boldsymbol{\xi}_{\mathcal{C}_{t+1} \to \mathcal{C}_t} + \boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} = \mathbf{0} \end{cases}, \tag{S5}$$

which were used in Eq. (6) and Eq. (7) of the main manuscript to encourage the geometric invariance via the two losses:

$$\mathcal{L}^{\text{inv, rot}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \triangleq \left\| \widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} + \widehat{\theta}_{\mathcal{C}_{t+1} \to \mathcal{C}_t} \right\|_2^2. \tag{S6}$$

$$\begin{aligned} \mathcal{L}^{\text{inv, trans}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} &\triangleq \left\| \exp\left(\texttt{alg}(\widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}})\right) \cdot \widehat{\boldsymbol{\xi}}_{\mathcal{C}_{t+1} \to \mathcal{C}_t} + \widehat{\boldsymbol{\xi}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \right\|_2^2 \\ &= \left\| \widehat{R}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \cdot \widehat{\boldsymbol{\xi}}_{\mathcal{C}_{t+1} \to \mathcal{C}_t} + \widehat{\boldsymbol{\xi}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \right\|_2^2. \end{aligned} \tag{S7}$$

This concludes the derivation.

B. Technical Details for Generating Egocentric Top-Down Projection

We describe details on how to compute the egocentric top-down projection discussed in Sec. 3.4.

From depth map to 3D point. Given a pixel of the depth map $\texttt{depth}$ at image coordinates (u, v)[8], we obtain the 3D point's Cartesian coordinates in the camera coordinate system from:

$$\begin{aligned} (x, y, z)^T &= h(u, v, \texttt{depth}(u, v)) \\ &= (K^{-1} \cdot (u + 0.5, v + 0.5, 1)^T) \cdot \texttt{depth}(u, v), \end{aligned} \tag{S8}$$

where $h(\cdot, \cdot, \cdot)$ represents the function for generating 3D co-

[8] We follow common practice and let $+U$ point downward while $+V$ points to the right.

ordinates[9]. Here $K \in \mathbb{R}^{3\times3}$ is the intrinsic matrix assumed to be known and $\texttt{depth}(u, v)$ denotes the z-buffer value at (u, v). Note $u + 0.5$ and $v + 0.5$ are used to compute the 3D point from the center of the pixel and $(u + 0.5, v + 0.5, 1)$ is the homogeneous coordinate. Further, $z = \texttt{depth}(u, v)$.

Computing bounding box for point clouds. After generating 3D point clouds, we obtain a bounding box for those 3D points. Specifically, 1) for the Cartesian Z axis, we have z_{min} and z_{max}. They refer to the minimum and the maximum depth values, which are specified by the sensor; 2) for the Cartesian X axis, we have x_{min} and x_{max} which come from leftmost/rightmost pixels in depth observation `depth`. These values will be utilized to compute pixel coordinates in the next step.

Computing 2D pixel coordinates of top-down projection. As mentioned in Sec. 3.1, we assume that the agent's motion is planar. Therefore, we ignore coordinates in the direction perpendicular to the plane. Concretely, we use coordinates (x, z). Therefore, we obtain pixel coordinates in top-down projection for such a point as $(\texttt{row}, \texttt{col})$, where $\texttt{row} = \lfloor H \cdot \frac{z - z_{\text{min}}}{z_{\text{max}} - z_{\text{min}}} \rfloor$ and $\texttt{col} = \lfloor W \cdot \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \rfloor$, where $H \times W$ represents the top-down projection's resolution.

Generating *soft* top-down projection. 1) For every pixel in `depth`, we repeat the aforementioned steps to compute the corresponding pixel coordinates $(\texttt{row}, \texttt{col})$ in the top-down projection. 2) We count the number of points which fall into each $(\texttt{row}, \texttt{col})$ cell. A *soft* egocentric top-down projection `s-proj` is obtained by normalizing the count to the range of $[0, 1]$.

C. Navigative Policy Training Details

In Tab. S1, we provide training details of the navigation policy used in our experiments. We explain the strucutre of our policy in the following paragraphs.

Visual encoder. We use ResNet-18 [17] as our visual feature extractor to process an egocentric observation of size $341(\text{width}) \times 192(\text{height})$. Following [53], we replace every BatchNorm [20] layer with GroupNorm [54] to deal with highly-correlated trajectories in on-policy RL and massively distributed training. A `2x2-AvgPool` layer is added before ResNet-18 so that the effective resolution is 170×96. ResNet-18 produces a $256 \times 6 \times 3$ feature map, which is converted to a $114 \times 6 \times 3$ feature map through a `3x3-Conv` layer.

Point-Goal encoder. At each time step t, the agent receives the point-goal's relative position $\boldsymbol{v}_t^g$ or $\widehat{\boldsymbol{v}}_t^g$ in polar coordinate form. Similar to [53], we convert the polar coordinates into a magnitude and a unit vector to alleviate the discontinuity at the x-axis in polar coordinates. A subsequent fully-connected layer transforms it into a 32-dimensional representation.

[9]Following common practice, $+X$ points to the right, $+Y$ points upward and $+Z$ points backward.

Navigation Policy. The 2-layer LSTM in the navigation policy takes three inputs: 1) a 512-dimensional vector of egocentric observations, which is obtained by flattening the $114 \times 6 \times 3$ feature map from the visual encoder into a 2052-dimensional vector and then feeding it into a fully-connected layer; 2) a 32-dimensional output of the goal encoder; 3) a 32-dimensional embedding of the previous action (or the start-token when beginning a new episode). The output of the 2-layer LSTM is fed into a fully-connected layer, obtaining a distribution over the action space and an estimate of the value function.

Table S1: Hyperparameters.

Hyperparameter	**Value**
PPO (DD-PPO)	
Clip parameter [44]	0.2
Rollout timesteps	128
Epochs	2
Value loss coefficient	0.5
Discount factor (γ)	0.99
GAE parameter (λ) [43]	0.95
Normalize advantage	False
Preemption threshold [53]	0.6
Training	
Optimizer	Adam [26]
(β_1, β_2) for Adam	$(0.9, 0.999)$
Learning rate	$2.5e^{-4}$
Gradient clip norm	0.2

D. VO Model Training and Inference Details

D.1. Environment Details

Consistent with [41], in Tab. S2, we show the inventory of all scenes from Gibson [56] that were used in our experiments. Each of them is rated with quality level 4 or above as described in [41]. From the 72 scenes of the `train` split, we create a training dataset $\mathcal{D}$ with one million data points as described in Sec. 4.1. Similarly, a validation dataset $\mathcal{D}_{\text{val}}$ with 50,000 data points is generated from 14 scenes of the `val` split.

D.2. VO Dataset Statistics

Tab. S3 summarizes the statistics of our visual odometry (VO) training dataset $\mathcal{D}$. As mentioned in Sec. 4.1, since our training data is sampled from shortest-path trajectories, the ratio of actions roughly represents the percentage of actions that appeared in actual navigation tasks.

Tab. S3 provides another reason to use a separate model per action (SepAct) in a visual odometry model. Since the

Table S2: Gibson-4+ scene split.

Split	Scenes
`Train`	Adrian, Applewold, Bolton, Cooperstown, Goffs, Hominy, Mobridge, Nuevo, Quantico, Roxboro, Silas, Stanleyville, Albertville, Arkansaw, Bowlus, Crandon, Hainesburg, Kerrtown, Monson, Oyens, Rancocas, Sanctuary, Sodaville, Stilwell, Anaheim, Avonia, Brevort, Delton, Hambleton, Maryhill, Mosinee, Parole, Reyno, Sasakwa, Soldier, Stokes, Andover, Azusa, Capistrano, Dryville, Haxtun, Mesic, Nemacolin, Pettigrew, Roane, Sawpit, Spencerville, Sumas, Angiola, Ballou, Colebrook, Dunmor, Hillsdale, Micanopy, Nicut, Placida, Roeville, Seward, Spotswood, Superior, Annawan, Beach, Convoy, Eagerville, Hometown, Mifflintown, Nimmons, Pleasant, Rosser, Shelbiana, Springhill, Woonsocket
`Val`	Cantwell, Denmark, Eastville, Edgemere, Elmira, Eudora, Greigsville, Mosquito, Pablo, Ribera, Sands, Scioto, Sisters, Swormville

dataset is imbalanced with respect to the type of action, a unified model across all actions needs to deal with imbalanced training data. Empirically, we find that a unified model overfits for `turn_left` and `turn_right` while the performance of `move_forward` has not converged yet. The SepAct design overcomes this issue. More discussion is presented in Sec. D.4.

In Fig. S1, we illustrate the distribution of translation and rotation caused by each action. We note that for each of the actions, the distribution of the translation changes has a peak around $0m$, which is caused by the agent getting stuck when encountering collisions.

D.3. Qualitative Examples from $\mathcal{D}$

Fig. S2 shows qualitative examples of translation and rotation changes resulting from each action. Apart from the noisy egocentric observations, the complexity of estimating the $SE(2)$ transformation also stems from similar translation and rotation changes across different actions. For example, $\xi^x_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ in all six figures is extremely similar.

D.4. VO Model Evaluation

Fig. S3 shows the evaluation curve on $\mathcal{D}_{\text{val}}$ for the *Unified* and *SepAct* models, namely the VO model of Row 6 and Row 8 in Tab. 2. We define `sys_error` as the average absolute difference between ground-truth and estimated values if the VO model always predicts the mean of the training data in Fig. S1. For example, if we let $\mathcal{D}^{\text{forward}} \subset \mathcal{D}$ and $\mathcal{D}^{\text{forward}}_{\text{val}} \subset \mathcal{D}_{\text{val}}$ be datasets whose data points are generated by the `move_forward` action, we compute the `sys_error` of `move_forward` on $\xi^x_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ as:

$$\texttt{sys\_error} = \frac{1}{|\mathcal{D}^{\text{forward}}_{\text{val}}|} \sum_{d_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \in \mathcal{D}^{\text{forward}}_{\text{val}}} |\xi^x_{\mathcal{C}_t \to \mathcal{C}_{t+1}} - \mu|,$$

$$\text{where } \mu = \frac{1}{|\mathcal{D}^{\text{forward}}|} \sum_{d_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \in \mathcal{D}^{\text{forward}}} \xi^x_{\mathcal{C}_t \to \mathcal{C}_{t+1}}. \quad \text{(S9)}$$

Here $d_{\mathcal{C}_t \to \mathcal{C}_{t+1}} = \big((I_t, I_{t+1}), \boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}, \theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}\big)$ and $\mu = 0.018$ from the first histogram of Fig. S1a. Note, `sys_error` is computed equivalently for $\xi^z_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$, $\xi^z_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$, and $\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ of all three actions. The closer the evaluation curve is to `sys_error`, the less useful the information that the VO model learns. Apparently, the SepAct model learns more helpful information as its curve is further away from the `sys_error` line.

Meanwhile, as discussed in Sec. D.2, the training dataset, which represents the actual path's action distribution, is imbalanced. A unified model may encounter overfitting on one action while yielding unsatisfactory prediction on another. Specifically, in the first plot of Fig. S3a, the performances on `turn_left` and `turn_right` encounters overfitting at around the 30^{th} epoch, while the performance on `move_forward` improves even at the 120^{th} epoch. This issue does not arise in SepAct's evaluation curve in Fig. S3b, verifying the efficacy of SepAct.

E. DeepVO and KITTI

In this section we discuss implementation details of DeepVO as well as our model's performance on KITTI.

DeepVO implementation. There isn't an official code of DeepVO and the most-starred public one yields incorrect results (Tab. S4's Col. 2)[10]. Our re-implementation of DeepVO (Col. 3 in Tab. S4) matches the numbers reported in the original DeepVO paper (Col. 1)[11]. Therefore, we apply our implemented DeepVO in the PointGoal navigation task.

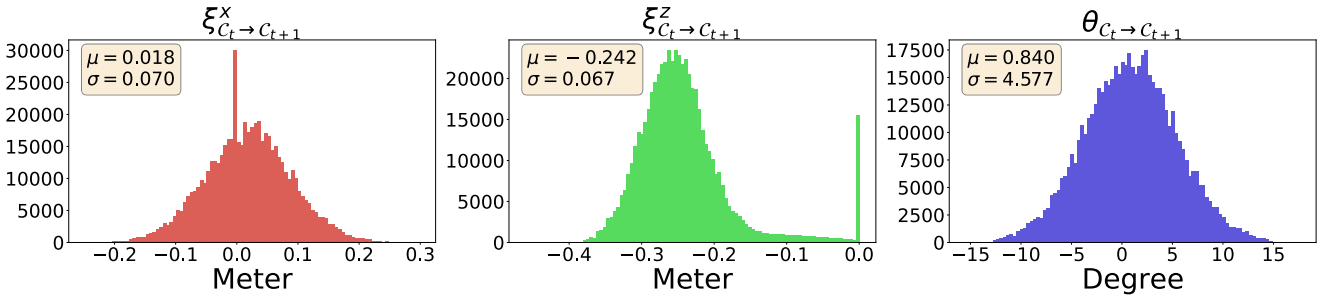
Our VO module on KITTI [15]. In order to run our VO module on KITTI, we need depth information. We use one of the best entries (DeepPruner [13]) in Tab. 3 from [29] to obtain a depth estimate. As can be inferred from Tab. S4's Col. 3 *vs.* 4 and Tab. 2's Row 0 *vs.* Row 18, outdoor and indoor tasks have their own challenges.

[10] https://github.com/ChiWeiHsiao/DeepVO-pytorch

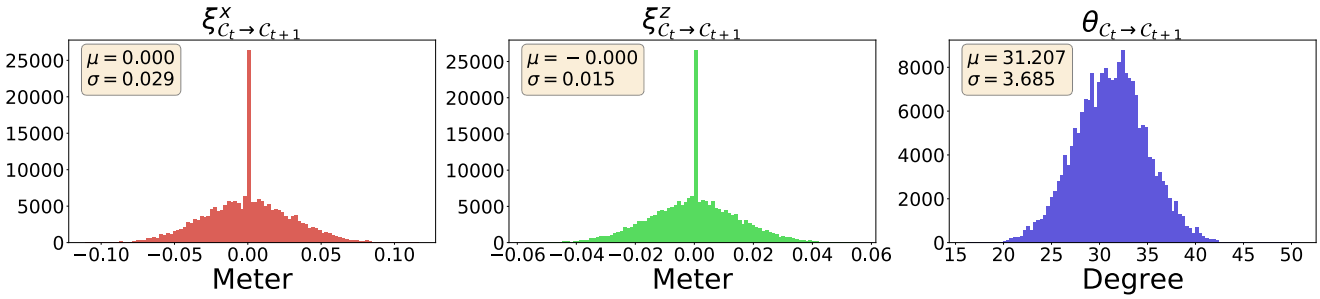
[11] Differences are due to the rare train/test split in the DeepVO paper while we train on Seq00-08 and evaluate on Seq09/10 as Tab. 1 in [8].

Table S3: Visual odometry training dataset statistics.

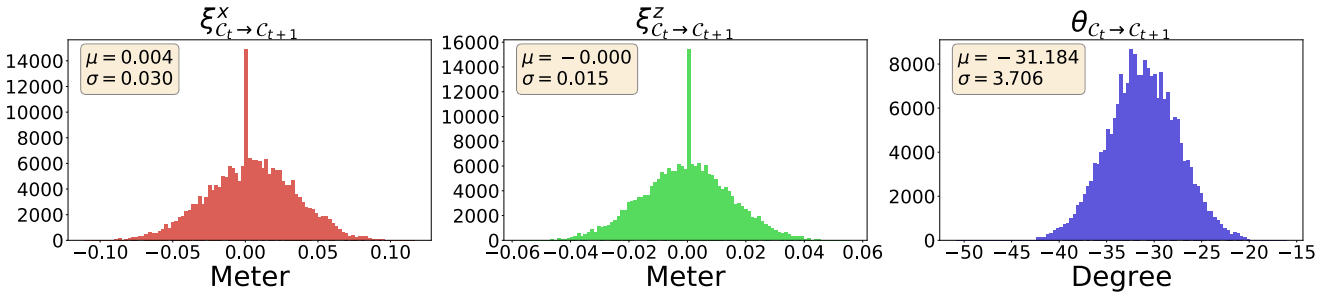
Category \ Action	move_forward	turn_left	turn_right	Total
Non-collided	503,890 (87.90%)	186,291 (87.32%)	197,318 (92.49%)	887,499 (88.75%)
Collided	69,342 (12.10%)	27,143 (12.68%)	16,016 (7.51%)	112,501 (11.25%)
Total	573,232	213,434	213,334	1,000,000



(a) Action move_forward.



(b) Action turn_left.



(c) Action turn_right.

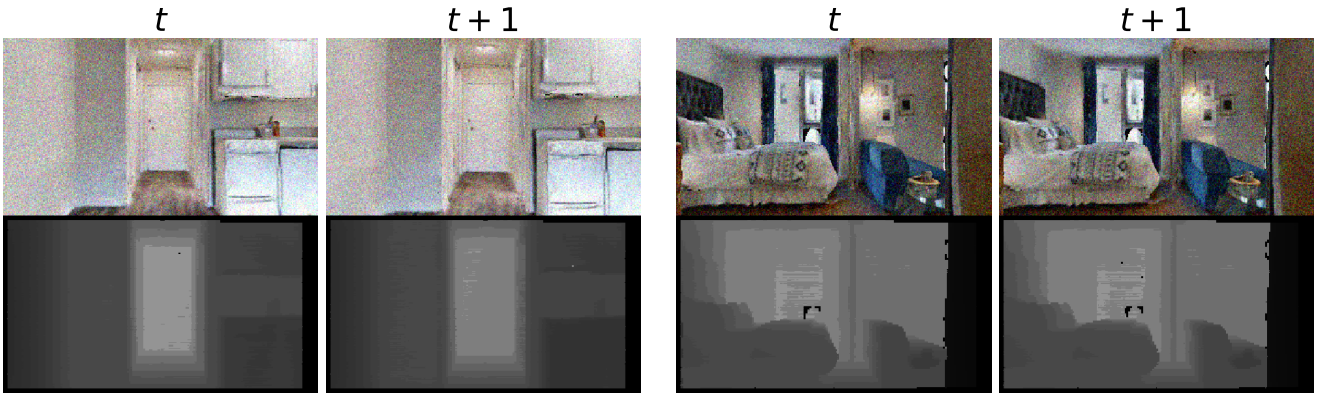
Figure S1: Translation and rotation distribution histogram of each action in our VO training dataset. Because the simulator aligns the forward direction with the negative direction of the axis, most of the $\xi^z_{C_t \to C_{t+1}}$ values for move_forward are negative.

F. Estimate Relative Pose from Depth

Because depth is noisy as mentioned in Sec. 3, it prevents reliable estimation of relative pose. To verify, we experiment with the following pipeline.

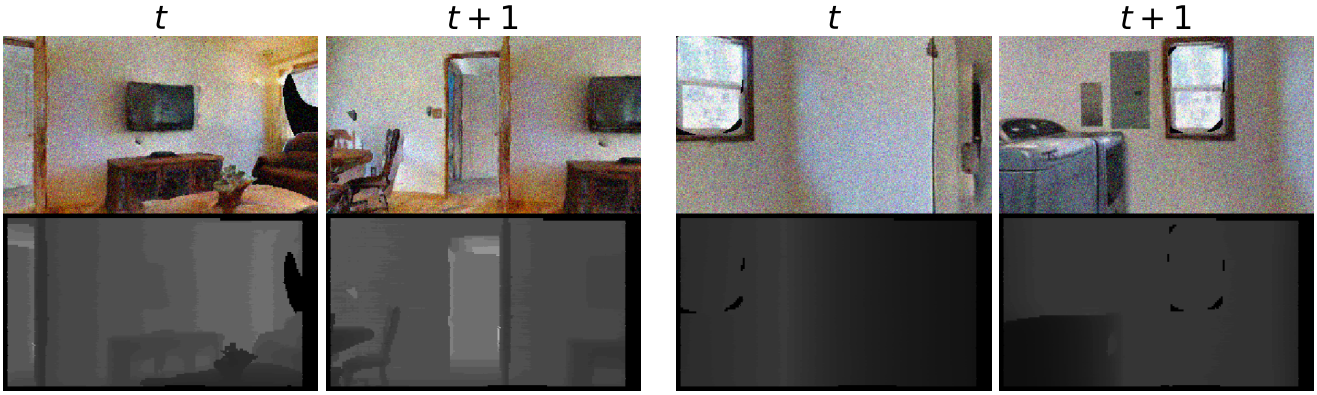
1) Find matching points. To extract and match point descriptors in adjacent RGB frames, we use the recent SuperPoint-SuperGlue (SPSG) [12, 39] which was shown to improve over traditional hand-engineered methods. Qualitatively, Fig. S4a verifies high-quality matches.

2) Compute relative pose. We use findEssentialMat and recoverPose from OpenCV to recover rotation $\widehat{\theta}_{C_t \to C_{t+1}}$ and *direction* of translation. Fig. S4b shows inliers for Fig. S4a found by OpenCV. High-quality inliers ease the analysis as the final VO prediction error unlikely stems from mismatched points.



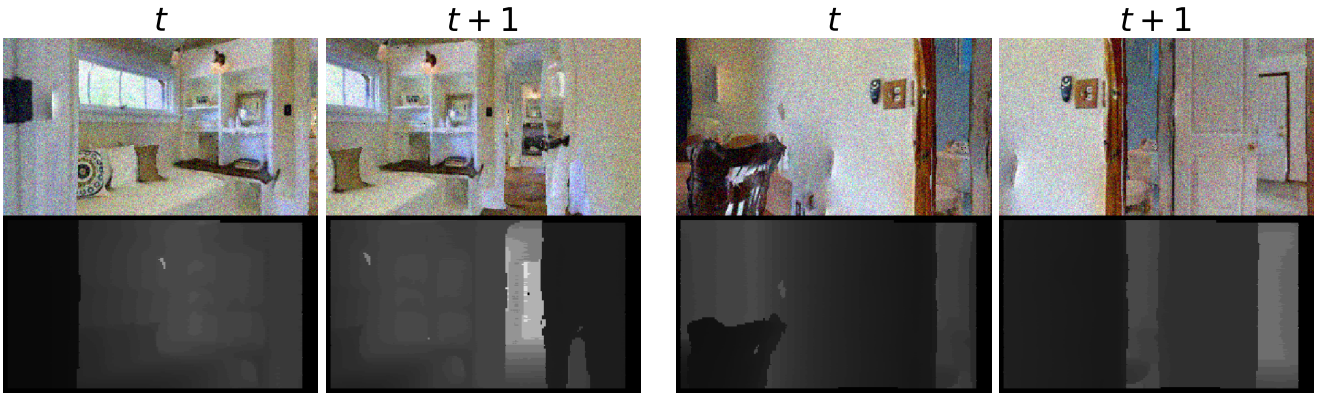
(a) `move_forward`, no collision, $(0.05, 0.19, -0.26)$.

(b) `move_forward`, with collision, $(0.00, 0.00, -0.20)$.



(c) `turn_left`, no collision, $(-0.04, -0.01, 32.8)$.

(d) `turn_left`, with collision, $(0.00, 0.00, 38.6)$.



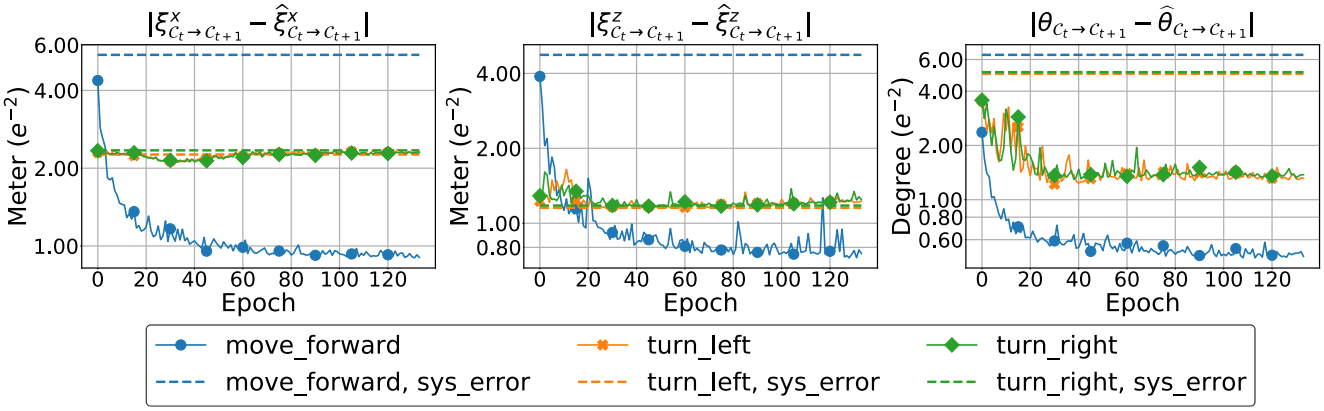
(e) `turn_right`, no collision, $(-0.02, 0.00, -28.4)$.

(f) `turn_right`, with collision, $(0.06, 0.00, -29.4)$.

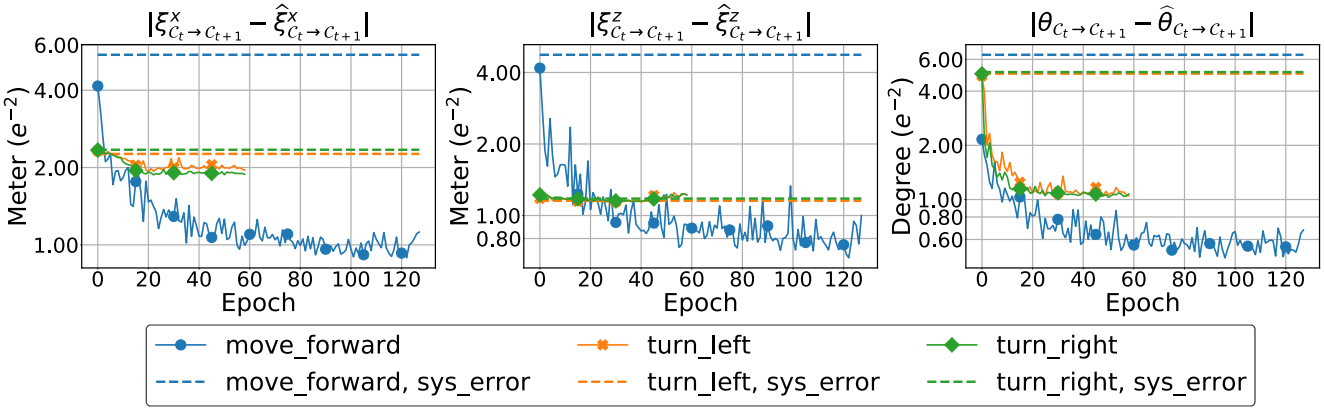
Figure S2: Qualitative examples of translation and rotation changes caused by each action. The changes are presented in the order of $(\xi^x_{\mathcal{C}_t \to \mathcal{C}_{t+1}}, \xi^z_{\mathcal{C}_t \to \mathcal{C}_{t+1}}, \theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}})$.

Table S4: Results on KITTI. Values are $r_{\text{rel}}(^\circ)/t_{\text{rel}}(\%)$.

	1.DeepVO[†]	2.DeepVO[‡]	3.DeepVO[§]	4.RGB-D-DD-S-Proj
Seq09	N/A	33.37 / 92.97	4.016 / 11.14	7.062 / 19.22
Seq10	8.83 / 8.11	38.68 / 90.22	4.498 / 11.24	9.298 / 15.80



(a) Unified model.



(b) SepAct model.

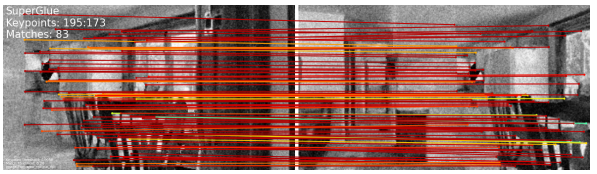
Figure S3: Evaluation of VO models on generated validation dataset $\mathcal{D}_{\text{val}}$. We show the average absolute difference between ground-truth value and prediction from VO models. The y-axis uses log-scale. The `sys_error` is defined in Sec. D.4.

3) Resolve scale ambiguity. 3).a With depth, we compute 3D coordinates of inliers in two camera coordinate systems. **3).b** We rotate 3D coordinates in one frame with $\widehat{\theta}_{C_t\to C_{t+1}}$. **3).c** We compute the scale as the averaged norm between the rotated coordinates and the coordinates in the other frame. To obtain the final translation $(\hat{\xi}^x_{C_t\to C_{t+1}}, \hat{\xi}^z_{C_t\to C_{t+1}})$, we rescale the *direction* produced by `OpenCV`. The obtained VO error (`E1`) is much larger than ours (`E3`) (Tab. S5) and prevents successful navigation.

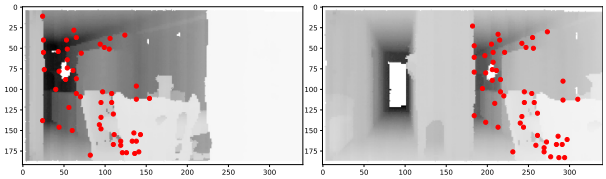
4) Additional oracle experiment. We conduct an oracle experiment using *ground-truth* rotation $\theta_{C_t\to C_{t+1}}$ in **3).b**. From `E2` *vs*. `E3` (ours): directly estimating relative pose from depth is inferior. Note, the validation set scenes are not used for training our VO model (Sec. 4.1).

G. More Qualitative Results

In Fig. S5, we provide additional qualitative results when integrating the navigation policy with our VO model.



(a) Matched points from SPSG.

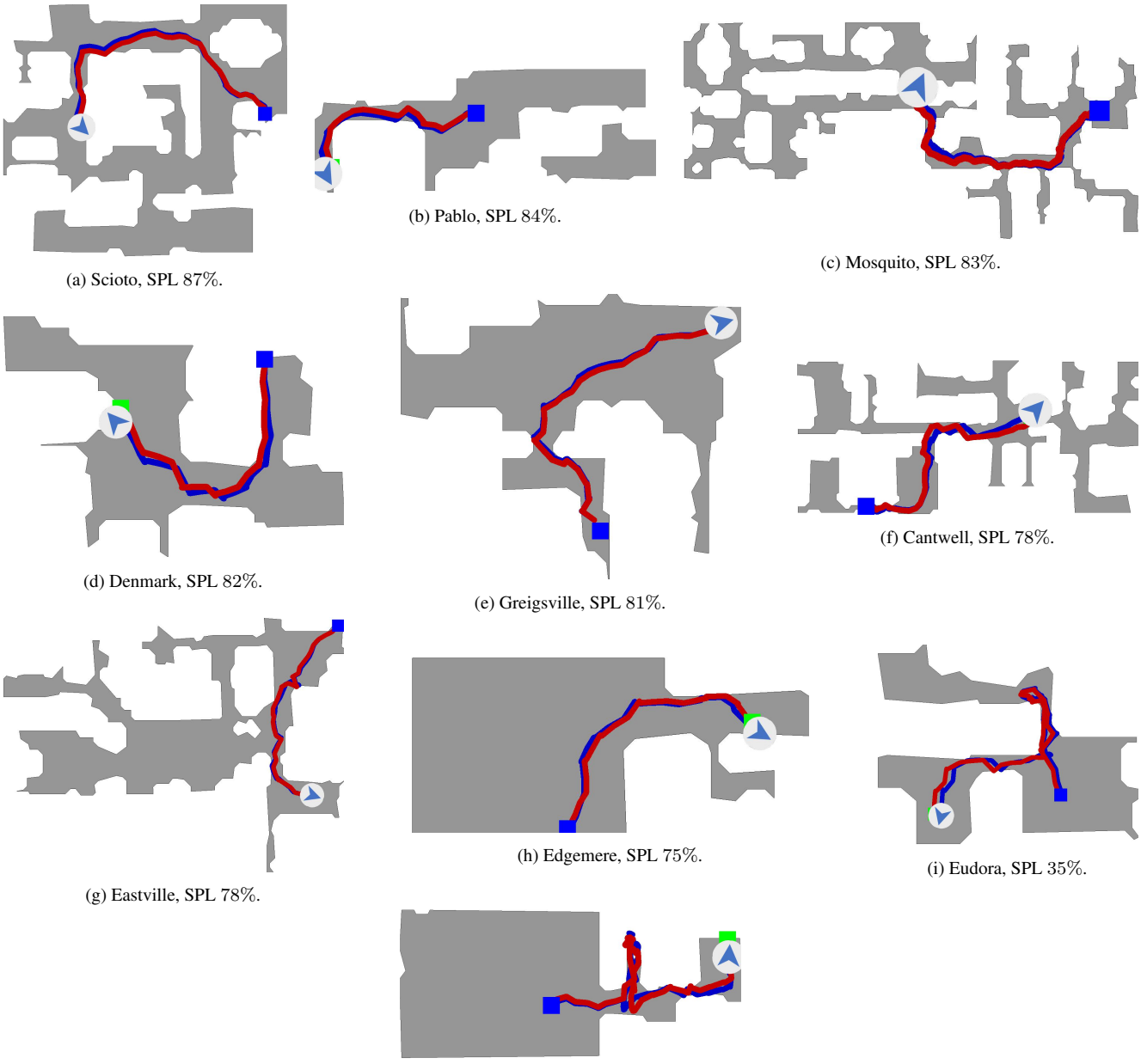


(b) Inliers from `OpenCV`.

Figure S4: Qualitative examples for feature matching.

E1 (e^{-2})	E2 (e^{-2})	E3 (e^{-2})
(15.9, 21.3, 8.51)	(3.97, 10.6, 0.00)	(1.22, 0.86, 0.66)

Table S5: **VO prediction error on $\mathcal{D}_{\text{val}}$** (50000 entries, see Sec. D.1). Lower is better. Following Tab. 2, we report $(\hat{\xi}^x_{C_t\to C_{t+1}}, \hat{\xi}^z_{C_t\to C_{t+1}}, \widehat{\theta}_{C_t\to C_{t+1}})$. `E1`: Feature Matching; `E2`: Feature Matching Oracle; `E3`: our result.



(a) Scioto, SPL 87%.

(b) Pablo, SPL 84%.

(c) Mosquito, SPL 83%.

(d) Denmark, SPL 82%.

(e) Greigsville, SPL 81%.

(f) Cantwell, SPL 78%.

(g) Eastville, SPL 78%.

(h) Edgemere, SPL 75%.

(i) Eudora, SPL 35%.

(j) Sisters, fail.

Figure S5: Qualitative results (best viewed in color). Agent is asked to navigate from blue square to green square. Blue curve is the actual path the agent takes while red curve is based on the agent's estimate of its location from the VO model by integrating over $SE(2)$ estimation of each step.