This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

The Surprising Effectiveness of Visual Odometry Techniques for Embodied PointGoal Navigation

Xiaoming Zhao[†], Harsh Agrawal[‡], Dhruv Batra[‡,§], Alexander Schwing[†]
[†]University of Illinois, Urbana-Champaign [‡]Georgia Institute of Technology [§]Facebook AI Research
https://xiaoming-zhao.github.io/projects/pointnav-vo/

Abstract

It is fundamental for personal robots to reliably navigate to a specified goal. To study this task, PointGoal navigation has been introduced in simulated Embodied AI environments. Recent advances solve this PointGoal navigation task with near-perfect accuracy (99.6% success) in photo-realistically simulated environments, assuming noiseless egocentric vision, noiseless actuation and most importantly, perfect localization. However, under realistic noise models for visual sensors and actuation, and without access to a "GPS and Compass sensor," the 99.6%-success agents for PointGoal navigation only succeed with 0.3%.[1] *In this work, we demonstrate the surprising effectiveness of visual odometry for the task of PointGoal navigation in this realistic setting,* i.e.*, with realistic noise models for perception and actuation and without access to GPS and Compass sensors. We show that integrating visual odometry techniques into navigation policies improves the state-of-the-art on the popular Habitat PointNav benchmark by a large margin, improving success from 64.5% to 71.7% while executing 6.4 times faster.*

1. Introduction

The ability to navigate efficiently and accurately within an indoor environment is fundamental to personal robots and has been a focus of research in computer vision for many years [37]. To coalesce the community around a common framework and standard metrics, Anderson *et al*. [2] proposed the task of PointGoal navigation. In PointGoal navigation, an agent is randomly spawned in a previously unseen environment and has to navigate to a point goal specified relative to the agent's initial location and orientation, *e.g*., 'Go 5m north, 3m west relative to start'. The agent uses a discrete action space (*e.g*., `move_forward` 0.25m, `turn_left` or `turn_right` 30°, and `stop`) to navigate in the environment. Under the assumption of noiseless egocentric vision (noise-free RGB + depth sensors), noise-free actuation (*e.g*., `turn_left` will always turn exactly 30°) and perfect localization using `GPS+Compass` sensors, recent methods solve this task with near-perfect accuracy (99.6% success) [53].

[1] https://eval.ai/web/challenges/challenge-page/580/leaderboard/1631 (Habitat Team).

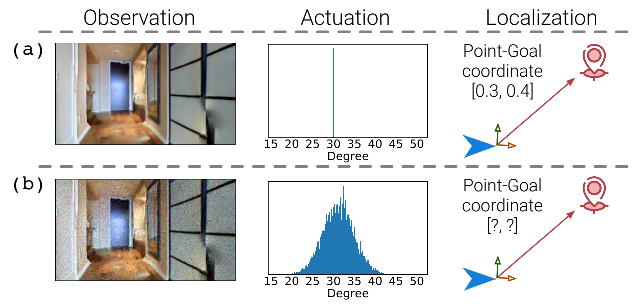


Figure 1: Noiseless (a) and noisy (b) PointGoal navigation. In the noisy setting, the agent observes: 1) sensor noises in egocentric observation; 2) actuation perturbations. The second column shows a histogram of orientation angle changes caused by a `turn_left` action; 3) no localization information. The agent's inaccurate localization results in uncertainty about the goal location.

However, these assumptions are unrealistic. Note that GPS sensors typically don't yield a precise location in indoor environments. In addition, perception and actuation of real robots often depend heavily on environment lighting and friction coefficients of surfaces. To study this more realistic setting, in a recent benchmark[2], PointGoal navigation was updated to include noisy actuation models from real robots [35]. For example, for a single `turn_left` action, the actual turn angle varies significantly as shown in column two of Fig. 1. Also, RGB and depth noise models from [9] were incorporated to simulate a real-world camera. Most importantly, as illustrated in column three of Fig. 1, the agent *does not* have access to `GPS+Compass` data and must navigate solely based on egocentric RGB + depth (RGB-D) measurements. Under such a more realistic setting, the performance of a policy that is near-perfect in noiseless scenarios [53] drops drastically to 0.3%. Improving upon it, prior state-of-the-art [24] incorporates particle SLAM into visual navigation and achieves a success rate of 64.5% under such a realistic setting. Compared to the 99.6% success rate on the noiseless version of the task, navigation with noisy perception and actuation as well as without localization information hence remains challenging.

[2] https://aihabitat.org/challenge/2020/

To better understand the challenges of navigation in this realistic setting, we study three visual odometry (VO) techniques. We find those VO techniques to be surprisingly effective for PointGoal navigation in this realistic setting. Specifically, we 1) leverage the geometric invariances of visual odometry; 2) incorporate discretization and ensembling to safeguard against noise; and 3) use top-down orthographic projection of depth information as an additional signal. For 1), we note that the estimated motion for a given pair of observations is related to the motion estimated for the permuted observation. Two loss terms encourage this relation. For 2) we study Dropout [46] in the last two layers of the visual odometry model to safeguard against uncertainty within the egomotion prediction, following [25]. We also find depth discretization to be effective. For 3), we infer an egocentric top-down projection from depth information at each *individual* step. We find that such a simple projection, which is *local* to each step, benefits egomotion estimation.

On the Habitat Challenge 2020 PointNav benchmark, we show that those three techniques are surprisingly effective, achieving a 71.7% success rate and a 52.5% SPL, which improves significantly upon the 64.5% and 37.7% SPL from prior state-of-the-art (SOTA). Moreover, using VO in a navigation policy also executes 6.4 times faster than prior SOTA. We perform exhaustive ablations to show the efficacy of each of the three techniques and find that *all the aforementioned techniques* contribute to a more accurate navigation.

Importantly, we train this visual odometry model separately instead of learning it online with the policy. Using the VO model as a drop-in replacement for a perfect `GPS+Compass` permits to **re-use** navigation policies that were learned with perfect localization information (*i.e.*, with `GPS+Compass` sensor) without any expensive re-training. Note that the visual odometry model can be trained for different environment dynamics using a static dataset of only a couple of million frames. In contrast, navigation policies are typically trained using over a billion frames collected using six-months of GPU-time [53].

To summarize, we study three techniques for realistic PointGoal navigation: 1) leveraging geometric invariances via losses; 2) incorporating discretization and ensembling; 3) using top-down projection of depth information.

We show: learning such a visual odometry model *offline* using only a couple of million frames and directly replacing the `GPS+Compass` input of a navigation policy achieves SOTA performance on the standard PointNav benchmark.

2. Related work

Navigation for embodied tasks. Recently, there has been a renewed interest in the field of Embodied AI. The community has built several indoor navigation simulators [41, 57, 40, 27] on top of photo-realistic scans of 3D environments [27, 6, 47, 56, 55]. To test a robot's ability to perceive, navigate and interact with the environment, the community has also introduced several tasks [57, 5, 45, 10, 52, 36, 3, 28, 48, 22, 21, 51, 16, 34, 33, 31, 32] and benchmarks. Specifically, Batra *et al*. [5] introduce evaluation details for the task of *Object Navigation*, requiring the agent to navigate to a given object class instead of a final point-goal. Similarly, *Room Navigation* [36] requires the agent to navigate to a given room type. More recently, Krantz *et al*. [45, 28, 48] extend the navigation task to utilize instructions in natural language. VLN [2, 28] and ALFRED [45] require the agent to follow a sequence of natural language instructions in order to reach the specified goal. Thomason *et al*. [48] introduce Vision-and-Dialog Navigation that requires back-and-forth communication in order to reach the desired location. Jain *et al*. [22, 21] develop FurnLift and FurnMove to study visual multi-agent navigation. While these tasks differ in their setup, each of them requires the agent to navigate accurately in an environment. Towards this, the agent's navigation policy assumes perfect knowledge of an agent's location and orientation (for example by using a perfect `GPS+Compass` sensor). Recently, to alleviate this unrealistic assumption, Datta *et al*. [11] propose to estimate egomotion from a pair of depth maps. Like them, we also conduct egomotion estimation from visual observation. However, differently, we study components that improve robustness. As we show in Sec. 4.3, without improving robustness to observation and actuation noise, the model yields inferior results.

Camera pose estimation and visual odometry (VO). Camera pose estimation is related to localization estimation. *E.g*., direct use of a convolutional neural net (CNN) to estimate relative camera pose was studied [59, 30], following the aforementioned egomotion estimation [11]. These models don't usually consider robustness. Meanwhile, in the last few decades, a number of methods have been developed for VO [42, 14]. The pipeline typically consists of several steps from camera calibration, feature selection and matching to motion estimation from correspondences, outlier detection, and bundle adjustment. More recently, various deep-learning-based architectures have been proposed for VO. For instance, Wang *et al*. [49] proposed a CNN + recurrent neural net (RNN) to estimate VO in an outdoor environment from RGB input. Because three successive frames in indoor navigation have little overlap, we find sequential training with an RNN to not help. In contrast, we use a faster ResNet-18 [17] architecture to learn VO from a noisy RGB-D input pair. Wang *et al*. [50] leverage the mathematical group property of the rigid motion to learn a VO model for outdoor navigation. Similarly, we also utilize geometric invariance constraints as a self-supervisory signal during training. In addition, we deliberately utilize representations that make the model robust to observation noise.

To model the agent's uncertainty about its egomotion prediction, Kendall *et al*. [25] used Dropout [46] after each

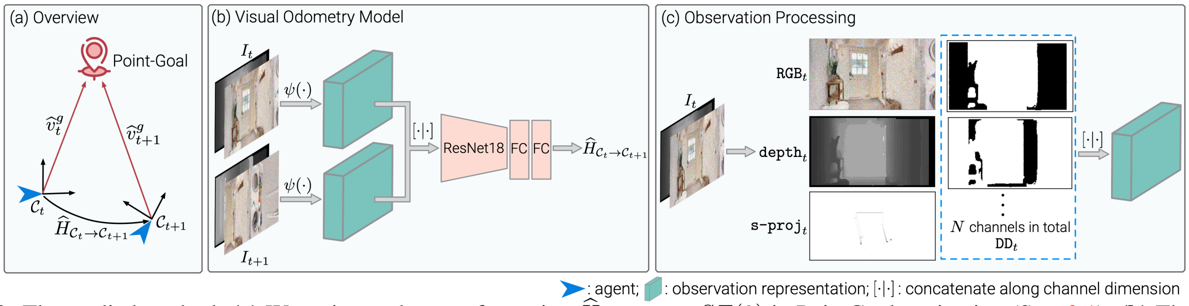


Figure 2: The studied method. **(a)** We estimate the transformation $\widehat{H}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \in SE(2)$ in PointGoal navigation (Sec. 3.1). **(b)** The visual odometry (VO) operates on two consecutive egocentric observations (I_t, I_{t+1}) and yields $\widehat{H}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ (Sec. 3.5). **(c)** Illustration for $\psi(\cdot)$. To deal with noise, besides raw $\texttt{RGB}_t$ and $\texttt{depth}_t$, we find discretization $\texttt{d-depth}_t$ (Sec. 3.3) and top-down projection $\texttt{s-proj}_t$ (Sec. 3.4) to help.

convolution layer and the penultimate linear layer. At test time, their model uses 40 random samples to get a robust estimate of the egomotion. 40 forward passes of the model at every time step is prohibitively expensive when used as input to a navigation policy. Moreover, since the input to the VO model is already noisy, adding Dropout to the CNN architecture provides little benefit. Instead, we add Dropout to the *last two* layers of the model, and approximate the effect of averaging the predictions from multiple models by scaling the parameters of the last two layers. This permits robust estimation with a single forward pass.

3. Approach

We study a simple but effective visual odometry (VO) model, suitable for Embodied AI tasks that predict egomotion from a pair of noisy RGB-D frames. This VO model, which is based solely on classical components, can be used as a drop-in replacement for a perfect `GPS+Compass` sensor in a downstream navigation task. In the following, an overview is provided before the components are discussed.

3.1. Overview

The model is illustrated in Fig. 2. PointGoal navigation [2] requires an agent to navigate to a point goal $\boldsymbol{v}_t^g$, which is specified relative to the agent's current location at each time step t. After the first move, due to noise, the agent only has an estimate $\widehat{\boldsymbol{v}}_t^g$ of the relative position.

Based on the estimated relative coordinates $\widehat{\boldsymbol{v}}_t^g$ as well as egocentric observations $I_{\leq t}$ until time t, *e.g.*, measurements from an RGB-D sensor, the agent chooses the next action towards the goal. For this, the agent computes a distribution over an action space $\mathcal{A} = \{\texttt{turn\_left}, \texttt{turn\_right}, \dots\}$, *i.e.*, a policy $\pi(\cdot|\widehat{\boldsymbol{v}}_t^g, I_{\leq t})$. Upon executing action $a_t \in \mathcal{A}$, the agent's position and orientation change. This results in a change of the agent's local coordinate system from $\mathcal{C}_t$ to $\mathcal{C}_{t+1}$. Any point's location in coordinate system $\mathcal{C}_t$ can be transformed to that of coordinate system $\mathcal{C}_{t+1}$ using a transformation $H_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$, which is an element of the group of rigid transformations in the 2D plane, *i.e.*, $SE(2)$. This assumes that the agent's motion is planar which holds because an episode is defined on a single floor. Note, all techniques can be extended easily to $SE(3)$ if required.

However, transformation $H_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ is not available because perfect location change measurements are not accessible. Hence, we need to estimate $\widehat{H}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \in SE(2)$ given the agent's egocentric observations. Using the transformation $\widehat{H}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$, the agent computes the goal's relative position at time $t+1$ from its prior estimate $\widehat{\boldsymbol{v}}_t^g$ via

$$\widehat{\boldsymbol{v}}_{t+1}^g = \widehat{H}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \cdot \widehat{\boldsymbol{v}}_t^g. \tag{1}$$

Sec. 3.2 discusses how to estimate the transformation $\widehat{H}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ from egocentric observations by using geometric invariances. Sec. 3.3 explains a simple way to make a visual odometry model robust to uncertainty in egomotion estimates. Next, Sec. 3.4 discusses a simple method to utilize a top-down projection from egocentric observation as an additional signal. Finally, Sec. 3.5 details training.

3.2. Geometric Invariances for Visual Odometry

The goal is to learn a convolutional neural net (CNN) that estimates the transformation $\widehat{H}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \in SE(2)$ from a given pair of egocentric observations (I_t, I_{t+1}). Formally, an element of $SE(2)$ is specified by a translation $\widehat{\boldsymbol{\xi}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \in \mathbb{R}^2$ in the ground plane and an angle $\widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \in \mathbb{R}$, *i.e.*,

$$\widehat{H}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} = \begin{bmatrix} \widehat{R}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} & \widehat{\boldsymbol{\xi}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \\ & 1 \end{bmatrix}, \tag{2}$$

with $\widehat{R}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} = \begin{bmatrix} \cos(\widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}) & -\sin(\widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}) \\ \sin(\widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}) & \cos(\widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}) \end{bmatrix} \in SO(2)$ denoting the estimated rotation matrix from the special orthogonal group. Given this parameterization, we found $SE(2)$ estimation via regression to be effective when using the following loss: $\mathcal{L}^{\text{reg}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \triangleq$

$$\|\boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} - \widehat{\boldsymbol{\xi}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}\|_2^2 + \|\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}} - \widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}\|_2^2. \tag{3}$$

Here, $\boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ and $\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ are grounth-truth $SE(2)$ components while $\widehat{\boldsymbol{\xi}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ and $\widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ are estimates of the model f_ϕ illustrated in Fig. 2(b), *i.e.*,

$$\left(\widehat{\boldsymbol{\xi}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}, \widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}\right) = f_\phi\left((\psi(I_t), \psi(I_{t+1}))\right). \tag{4}$$

Further, ϕ refers to parameters of the VO model and ψ denotes a function that processes egocentric observations. The architecture of the model will be presented in Sec. 3.5.

Note, use of the loss given in Eq. (3) is common for learning the parameters of a VO model which often exhibits the structure given in Eq. (4), *e.g.*, [49, 11]. However, as we show in Sec. 4.3, without specifically accounting for perceptual and actuation noise, pure regression does not work well. We discuss robustness improvements next.

Beyond regressing to ground truth data via the loss given in Eq. (3), more information is available in a pair of observations (I_t, I_{t+1}). To see this, suppose the agent observes (I_t, I_{t+1}) followed by (I_{t+1}, I_t). In this case we know that, in general, the agent returned to its original location. This is more formally described via the $SE(2)$ invariance $H_{\mathcal{C}_t \to \mathcal{C}_{t+1}} H_{\mathcal{C}_{t+1} \to \mathcal{C}_t} = I_{3\times 3}$. Such geometric invariances are ubiquitous. To exploit them, in addition to the regression loss given in Eq. (3), we found two additional losses during training of a VO model to help:

$$\mathcal{L}^{\text{inv}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \triangleq \mathcal{L}^{\text{inv, rot}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} + \mathcal{L}^{\text{inv, trans}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}. \tag{5}$$

$\mathcal{L}^{\text{inv, rot}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ and $\mathcal{L}^{\text{inv, trans}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ are the rotation and translation invariance loss, which are explained next.

Rotation invariance. Intuitively, if a rotation with angle $\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ transforms coordinates in $\mathcal{C}_t$ to ones in $\mathcal{C}_{t+1}$, then the inverse coordinate transformation from $\mathcal{C}_{t+1}$ to $\mathcal{C}_t$ will be achieved via a rotation with angle $-\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$, *i.e.*, $\theta_{\mathcal{C}_{t+1} \to \mathcal{C}_t} = -\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$. Consequently, a VO model which receives egocentric observations (I_t, I_{t+1}) followed by observations (I_{t+1}, I_t) should be encouraged to predict $\widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} + \widehat{\theta}_{\mathcal{C}_{t+1} \to \mathcal{C}_t} = 0$. This is achieved via the self-supervised learning loss

$$\mathcal{L}^{\text{inv, rot}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \triangleq \left\| \widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} + \widehat{\theta}_{\mathcal{C}_{t+1} \to \mathcal{C}_t} \right\|_2^2. \tag{6}$$

Translation invariance. The translation invariance property is intuitively similar to the one for rotation. If the transformation from $\mathcal{C}_t$ to $\mathcal{C}_{t+1}$ consists of pure translation $\boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$, then the reverse transformation from $\mathcal{C}_{t+1}$ to $\mathcal{C}_t$ is simply another translation with $\boldsymbol{\xi}_{\mathcal{C}_{t+1} \to \mathcal{C}_t} = -\boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$. This results in the loss $\|\widehat{\boldsymbol{\xi}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} + \widehat{\boldsymbol{\xi}}_{\mathcal{C}_{t+1} \to \mathcal{C}_t}\|_2^2$. The relation is slightly more involved when the transformation consists of both rotation and translation. We obtain

$$\mathcal{L}^{\text{inv, trans}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \triangleq \left\| \widehat{\boldsymbol{\xi}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} + \widehat{R}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \cdot \widehat{\boldsymbol{\xi}}_{\mathcal{C}_{t+1} \to \mathcal{C}_t} \right\|_2^2. \tag{7}$$

We provide the formal derivation of the losses in Eq. (6) and Eq. (7) in the appendix.

3.3. Robustness to Uncertainty

In addition to leveraging geometric invariances, we found it was important to further increase robustness of the model's $SE(2)$ estimation. This is important because measurements are noisy: 1) visual observations differ even if the camera position and orientation are identical because of observation noises. This makes the processing of observations brittle; 2) perturbations in actuation influence the VO model's prediction since they increase the variance of rotation and translation. For robustness we use two classical techniques:

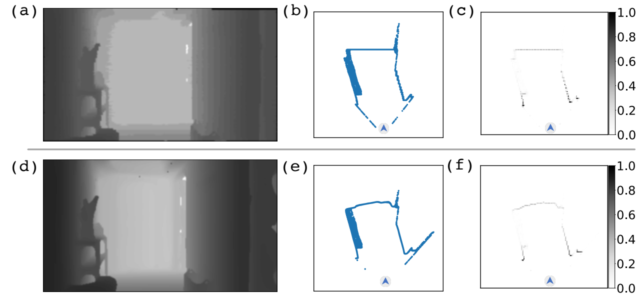


Figure 3: Steps to infer an egocentric top-down projection from depth. Top and bottom rows show inferred top-down projections from noisy and noiseless depth image at the same location. (b,e): top-down scatter plot. (c,f): the *soft* top-down projection. As can be seen, after processing, (c) and (f) share more similarities than (b) and (e), making the representation more robust to depth noises.

Ensemble. To improve robustness, one can train an ensemble of models. Averaging predictions over an ensemble typically reduces variance. However, reinforcement learning (RL) based navigation systems need billions of samples to train a good policy [53]. Since the policy relies on the VO model to provide the agent's current location estimate, it is important to increase the inference speed and avoid unnecessary computations. Therefore, instead of ensembling multiple models, we found it helpful to train one CNN architecture while adding Dropout [46] to the last two fully-connected (FC) layers. This economically resembles the behavior of training a large number of ensembles [4, 18]. During training, Dropout randomly disables hidden units in the FC layer with a probability p, essentially sampling from a collection of sub-networks. During inference, every hidden unit in the FC layer is scaled with the same factor p to mimic the averaging of predictions from multiple sub-networks.

Depth discretization. In addition, we found depth discretization to yield a more robust representation of the egocentric observation of a range sensor. Specifically, a single-channel depth map `depth` is discretized into representation `d-depth` with N channels using a one-hot encoding. Given a pixel of `depth` at image coordinates (x, y) we obtain the value of the i-th channel of `d-depth` via

$$\texttt{d-depth}_i(x, y) = \mathbb{1}\left\{\texttt{depth}(x, y) \in [z_{i-1}, z_i)\right\}, \tag{8}$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function and $\{z_{i-1}, z_i\}$ are endpoints of discretization intervals. Intuitively, this increases the absolute tolerance of the depth uncertainty to $\min_i \frac{|z_i - z_{i-1}|}{2}$ since the same representation will be generated unless a depth entry crosses the interval boundary. Empirically we find an equidistant discretization into N intervals using end-points $z_i = i \cdot (z_{\max} - z_{\min})/N$ to work well. Here, $z_{\max}$ and $z_{\min}$ are the maximum ($10m$) and minimum depth ($0m$) value respectively.

3.4. Top-Down Projection as Additional Signal

Intuitively a map should further improve model robustness. However, the key challenge in our setting: noise in the depth sensor is fairly subtle and often hardly visible (see Fig. 3(a,d)). But once projected to a 2D layout, the noise manifests itself in gross deviations, holes, and blockages as apparent in Fig. 3(b,e). To address this challenge we use a normalized *soft* projection. Normalized *soft* projection $\texttt{s-proj}_t$, shown in Fig. 3(c,f), resembles the room layout given by the depth maps. Note that they also share more similarities than the projection given in Fig. 3(b,e).

We obtain the *soft* projection by 1) mapping depth observations into 3D point clouds, 2) using a 2D top-down orthographic projection, and 3) normalizing the projection with respect to the number of points within each pixel. *Soft* projections are provided as input to the end-to-end trained VO model which learns to use it appropriately. Details of how to compute soft projections are presented in appendix.

3.5. VO Model Architecture, Training Details, and Integration with Navigation Policy

Model Architecture. The visual odometry model f_ϕ in Eq. (4) employs a ResNet-18 [17] backbone to extract visual features. For this we first compute representations from egocentric observation as sketched in Fig. 2(c) via

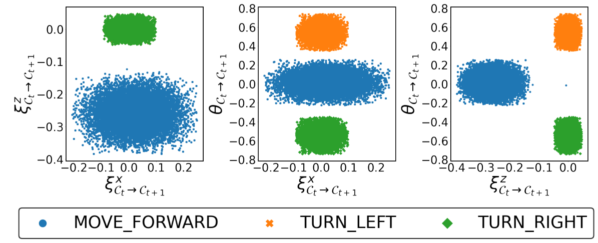
$$\psi(I_t) \triangleq (\texttt{RGB}_t, \texttt{depth}_t, \texttt{d-depth}_t, \texttt{s-proj}_t). \tag{9}$$

Then, we stack $(\psi(I_t), \psi(I_{t+1}))$ along the channel dimension to obtain the ResNet-18 input. Since $\texttt{RGB}_t$, $\texttt{depth}_t$, $\texttt{d-depth}_t$ and $\texttt{s-proj}_t$ have three, one, N and one channels respectively, the input to the ResNet-18 is a tensor with $(2N + 10)$ channels. To estimate $\widehat{H}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$, we use two Fully Connected (FC) layers with Dropout on top of the ResNet-18 feature extractor. These FC layers operate on 512-dimensional features and produce the output $\left(\hat{\xi}^x_{\mathcal{C}_t \to \mathcal{C}_{t+1}}, \hat{\xi}^z_{\mathcal{C}_t \to \mathcal{C}_{t+1}}, \widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}\right)$. Here $\hat{\xi}^z_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ refers to the translation in the agent's forward direction while $\hat{\xi}^x_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ refers to the translation in the direction perpendicular to the forward motion on the ground plane.

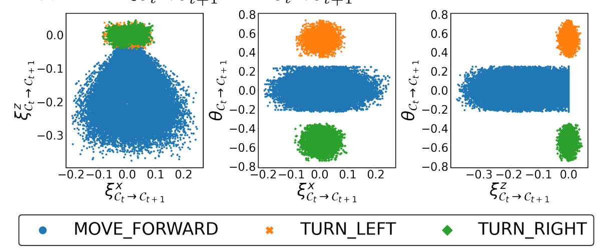
VO training. We train the visual odometry model f_ϕ on a dataset $\mathcal{D}_{\text{train}} = \left\{\left((I_t, I_{t+1}), \boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}, \theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}\right)\right\} \triangleq \left\{d_{\mathcal{C}_t \to \mathcal{C}_{t+1}}\right\}$. Each data point consists of a pair of egocentric observations as well as ground-truth translation and rotation angle. The model is optimized to jointly minimize the regression loss and geometric invariance loss defined in Eq. (3) and Eq. (5), *i.e.*, we address $\min_\phi \mathcal{L}_{\text{VO}} \triangleq$

$$\sum_{d_{\mathcal{C}_t \to \mathcal{C}_{t+1}} \in \mathcal{D}_{\text{train}}} \left[\lambda_{\text{reg}} \mathcal{L}^{\text{reg}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} + \lambda^{\text{trans}}_{\text{inv}} \mathcal{L}^{\text{inv, trans}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}} + \lambda^{\text{rot}}_{\text{inv}} \mathcal{L}^{\text{inv, rot}}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}\right],$$

where λ_{reg}, $\lambda^{\text{trans}}_{\text{inv}}$ and $\lambda^{\text{rot}}_{\text{inv}}$ are user-specified hyper-parameters. We set them to 1.0 in our experiments. We optimize the VO model with Adam [26] using a learning rate of 2.5×10^{-4}. The dropout factor is $p = 0.2$ during training.



(a) Plot of $\boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ and $\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ from data *w/o* collisions.



(b) Plot of $\boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ and $\theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ from data *with* collisions.

Figure 4: Three-drawing plot of VO training data $\mathcal{D}_{\text{train}}$ described in Sec. 4.1. Different actions have obviously distinct $SE(2)$ distributions, which we find cannot be well-learnt with a unifed model.

Navigation policy training. The focus of our work is PointGoal navigation under realistic conditions, *i.e.*, noisy observations and actuation as well as no access to GPU+Compass sensors. In order to demonstrate that VO techniques can be a simple drop-in replacement for a ground truth GPS+Compass sensor, we directly use the navigation policy from [53]. Specifically, the navigation policy π consists of a 2-layer LSTM [19] and uses a ResNet-18 [17] backbone to process the visual observations. The policy is *learned independently* of the visual odometry model and has access to perfect location data. During training, at each time step t, the policy π operates on egocentric observations $I_{\leq t}$, the ground-truth point goal $\boldsymbol{v}^g_t$ as well as prior actions $a_{\leq t-1}$, and computes a distribution over the action space $\mathcal{A}$. To learn the policy we use DD-PPO [53], a distributed version of PPO [44]. We use the same set of hyper-parameters and reward shaping settings [53], which we discuss more in the appendix.

Visual odometry for navigation. During inference, at every time $t + 1$, the agent obtains an egocentric observation I_{t+1}. Together with the previous egocentric observation I_t, the VO model f_ϕ computes the $SE(2)$ estimate $\widehat{H}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ using Eq. (4). Given the relative position estimate $\widehat{\boldsymbol{v}}^g_t$ from the previous time t, the agent updates the current estimate $\widehat{\boldsymbol{v}}^g_{t+1}$ via Eq. (1) and uses it as policy input.

4. Experiments

We strive to answer the following questions: 1) to what extent does such a visual odometry (VO) model help navigation? 2) what contributes to its performance? We report results on the online Habitat Challenge test split in Sec. 4.2 and conduct ablation on the offline validation split in Sec. 4.3.

4.1. Experimental Setup

Simulator specification. All experiments are conducted using the Habitat simulator [41] and we follow the Habitat PointNav Challenge [1] guidelines for all studies. We summarize them here and defer details to the appendix:

Dataset. We utilize the training data released as part of the Habitat Challenge. It consists of 72 scenes from the Gibson dataset [58] with a rating of 4 or above (Gibson-4+). The offline validation split consists of 14 different scenes which are not part of the training dataset.

Observations. Similar to a LoCoBot[3], the agent is equipped with an RGB-D camera mounted at a height of $0.88m$. It has a $70°$ field of view and records egocentric observations of resolution $341(\text{width}) \times 192(\text{height})$. The visual observations incorporate a noise model [9].

Actuation. The action space $\mathcal{A}$ consists of four actions: `move_forward` which moves the agent forward by $\sim 25cm$, `turn_left` and `turn_right` which rotate the agent by $\sim 30°$, and `stop`. The agent exhibits actuation noise modeled after the LoCoBot robot [35]. During collisions, the 'sliding' behavior that allows the agent to *slide* along the obstacle instead of stopping is disabled. This more accurately mimics the movement of a real robot [23]. Fig. 4 shows how actuation noise and collisions affect an agent's ground-truth translation and rotation for each action type.

VO dataset. To train the VO model, we create a dataset $\mathcal{D}_{\text{train}}$ of one million data points from 24,286 trajectories uniformly sampled from 72 training scenes.[4] As described in Sec. 3.5, each data point $d_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ consists of a pair of observations as well as ground-truth translation and rotation: $\left((I_t, I_{t+1}), \boldsymbol{\xi}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}, \theta_{\mathcal{C}_t \to \mathcal{C}_{t+1}}\right)$. We generate data points from each scene by repeating the following three-step procedure: 1) randomly sample a starting position and orientation of the agent and a navigable PointGoal in the scene; 2) follow the shortest path to navigate from starting point to the point goal; and 3) randomly sample data points $d_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$ along the trajectory. We find that due to actuation noise, the action leads to collisions approximately 11.25% of the time. The distribution of the ground-truth translation and rotation in this VO dataset $\mathcal{D}_{\text{train}}$ is illustrated in Fig. 4. We observe `move_forward`, `turn_left`, and `turn_right` to have distinct distributions. This finding motivates to train action-specific models, which is effective for this task.

Metrics. PointGoal Navigation is evaluated on several criteria, summarized by Anderson *et al.* [2]. An episode is considered successful ($S = 1$) if the agent stops within $0.36m$ ($2\times$ the agent radius) of the target global coordinate, otherwise the episode is marked as failed ($S = 0$). Using the length of the shortest-path trajectory l and the length of an agent's path l_a for an episode, Success Weighted by Path Length (SPL) is defined as $S\frac{l}{\max(l_a, l)}$. SPL intuitively captures how closely the agent followed the shortest path and successfully completed the episode. Distance to goal (d_G) captures the geodesic distance between the agent and the goal upon episode termination averaged across all episodes. Finally, the challenge also introduced the new SoftSPL metric [11]: using the starting geodesic distance to the goal d_{init} and the termination geodesic distance d_G, SoftSPL is defined as $(1 - \frac{d_G}{d_{\text{init}}})\frac{l}{\max(l_a, l)}$. It replaces the binary success S with a progress indicator that measures how close the agent gets to the target global coordinate at episode termination.

[3] http://www.locobot.org/

[4] Trajectories are shortest paths computed on ground-truth layout map.

Table 1: Online evaluation as of 1:30 am CST, Mar. 17th, 2021. S, SPL, and SoftSPL are reported in %.

Rank	Team	$S\uparrow$	SPL$\uparrow$	$d_G\downarrow$	SoftSPL$\uparrow$	Time $(h)\downarrow$
1-1	`Ours w/ finetuning`	**71.7**	**52.5**	0.802	**66.5**	5.83
1-2	`Ours w/o finetuning`	**69.8**	**52.0**	0.823	**65.7**	6.63
2	Karkus *et al.* [24]	64.5	37.7	**0.697**	52.1	37.50
3	Ramakrishnan *et al.* [38]	29.0	22.0	2.567	47.3	11.06
4	Information Bottleneck	16.3	12.2	2.075	56.1	2.73
5	Datta *et al.* [11]	15.7	11.9	2.232	58.6	**2.31**
6	cogmodel_team (39)	1.3	0.9	4.879	30.4	5.47
7	cso	1.2	0.7	4.632	24.7	5.57
8	UCULab	0.8	0.5	6.555	10.4	15.12
9	`Habitat Team`	0.3	0.0	6.929	3.8	-

4.2. Results on the Online Leaderboard

Tab. 1 shows the results from the online leaderboard on the test-standard split[5] of the Habitat Challenge PointNav Benchmark 2020 (we will call it Challenge hereafter). The 2020 winners achieved a success of 29.0% by integrating occupancy anticipation [38] into active neural SLAM [7] (Rank 3 in Tab. 1). Karkus *et al.* [24] proposed an end-to-end particle SLAM-net to generate a global occupancy map and utilized D* to plan the path, pushing SOTA to 64.5% in Nov. 2020 (Rank 2 in Tab. 1). Our approach of training a visual odometry model taking into account robustness as discussed in Sec. 3 and aforementioned action-specific design *improves SOTA to 71.7%*. Specifically, we evaluate the VO model quality in two settings: 1) direct integration into a pre-trained navigation policy as a drop-in module; 2) fine-tuning of a pre-trained policy w.r.t. the VO using a small budget.[6] Rank 1-1 and 1-2 in Tab. 1 verify that combining all of the discussed techniques achieves state-of-the-art performance on three out of four metrics, irrespective of fine-tuning. Besides success rate, it improves SPL by 14.8 points (from 37.7% to 52.5%). Regarding SoftSPL, it improves 7.9 points (from 58.6% of Rank 5 to 66.5%). Note, VO in the navigation policy executes evaluation 6.4 times faster than Rank 2 [24] (5.83 *vs.* 37.50 hours) and 1.9 times faster than Rank 3 [38] (5.83 *vs.* 11.06 hours).

4.3. Ablations

To better understand the role of each technique, we perform an extensive ablation study (Row 1 - 19) in Tab. 2. Specifically, we ablate over all combinations of: 1) visual

[5] https://evalai.cloudcv.org/web/challenges/challenge-page/580/leaderboard/1631

[6] We finetuned the policy using 14.7 million frames, instead of billions of frames required to train a policy.

Table 2: Evaluation on the Gibson-4+ validation split. VO prediction errors are presented in the order of $(\hat{\xi}^{x}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}, \hat{\xi}^{z}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}, \widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}})$. Results are reported from three evaluations with different seeds. We use D as abbreviation for depth. S, SPL, and SoftSPL are reported in %.

	VO								Policy Tune	$S\uparrow$	SPL↑	$d_G\downarrow$	SoftSPL↑	Pred Error per Step (e^{-2}) ↓
	Visual	DD	S-Proj	Dropout	ActInfo	DataAug	GeoInv	#param (M)						
0	DeepVO [49]							100.49		50±1	39±1	0.93±0.02	65±0	(2.40, 1.83, 1.62)±(0.00, 0.00, 0.01)
1	RGB							3.92		52±1	39±1	0.94±0.01	64±1	(1.96, 1.62, 1.37)±(0.02, 0.02, 0.01)
2	D							3.92		54±2	40±1	1.21±0.04	61±1	(1.88, 1.53, 1.38)±(0.01, 0.02, 0.02)
3	RGB-D							3.93		61±1	46±1	1.14±0.05	62±1	(1.72, 1.10, 1.23)±(0.04, 0.00, 0.00)
4	RGB-D			✓				3.93		68±1	51±1	0.78±0.03	66±0	(1.42, 0.98, 1.03)±(0.01, 0.01, 0.02)
5	RGB-D			✓(rnd10)				3.93		42±1	31±1	1.64±0.07	57±0	(1.71, 1.35, 1.84)±(0.00, 0.01, 0.01)
6	RGB-D			✓				12.4		70±1	52±1	0.89±0.04	65±0	(1.39, 1.02, 1.01)±(0.01, 0.01, 0.01)
7	RGB-D			✓	Embed			12.4		72±0	53±0	0.83±0.10	65±0	(1.36, 0.89, 0.93)±(0.02, 0.01, 0.01)
8	RGB-D			✓	SepAct			3×3.93		75±0	56±0	0.68±0.06	66±0	(1.24, 0.86, 0.82)±(0.00, 0.00, 0.01)
9	RGB-D			✓	SepAct	✓		3×3.93		75±2	56±1	0.67±0.03	66±0	(1.15, 0.85, 0.78)±(0.00, 0.00, 0.01)
10	RGB-D			✓	SepAct	✓	✓	3×3.93		77±1	57±0	0.65±0.04	67±0	(1.13, 0.85, 0.76)±(0.01, 0.00, 0.01)
11	RGB-D	5		✓	SepAct	✓	✓	3×3.96		74±2	57±1	0.70±0.05	68±0	(1.07, 1.03, 0.69)±(0.01, 0.01, 0.01)
12	RGB-D	10		✓	SepAct	✓	✓	3×3.96		79±1	60±1	0.54±0.00	69±0	(1.08, 0.90, 0.67)±(0.00, 0.00, 0.00)
13	RGB-D	20		✓	SepAct	✓	✓	3×3.96		79±0	60±0	0.52±0.03	69±0	(1.06, 0.85, 0.67)±(0.00, 0.00, 0.01)
14	D	10	✓	✓	SepAct	✓	✓	3×3.95		72±1	55±1	0.72±0.01	68±0	(1.40, 0.84, 0.86)±(0.00, 0.00, 0.00)
15	RGB-D		✓	✓	SepAct	✓	✓	3×3.93		77±1	59±1	0.54±0.04	70±0	(1.12, 0.91, 0.72)±(0.00, 0.00, 0.00)
16	RGB	10	✓	✓	SepAct	✓	✓	3×3.96		79±1	61±1	0.52±0.02	69±0	(1.18, 0.78, 0.75)±(0.00, 0.00, 0.01)
17	RGB			✓	SepAct	✓	✓	3×3.92		59±2	45±1	0.74±0.05	67±0	(2.02, 1.73, 1.15)±(0.01, 0.00, 0.01)
18	RGB-D	10	✓	✓	SepAct	✓	✓	3×3.96		**81**±1	**62**±1	**0.51**±0.03	**70**±0	(1.10, 0.84, 0.68)±(0.00, 0.00, 0.01)
19	RGB-D	10	✓	✓	SepAct	✓	✓	3×3.96	✓	**82**±1	**63**±1	**0.48**±0.00	**71**±0	(1.08, 0.85, 0.65)±(0.01, 0.01, 0.00)
20	Ground-Truth									97±0	71±0	0.42±0.02	70±0	

sensors (RGB and/or depth); 2) geometric invariance learning discussed in Sec. 3.2; 3) dropout and depth discretization detailed in Sec. 3.3; 4) soft egocentric projection described in Sec. 3.4; 5) use of action-specific models mentioned in Sec. 4.1. Note, the VO is a *drop-in replacement* in a pretrained navigation policy in Row 1 - 18 (no fine-tuning).

Evaluation is conducted on 994 episodes from 14 validation scenes, each of which provides 71 episodes. We abbreviate the discretized depth `d-depth` defined in Eq. (8) via *DD* and use *S-Proj* to indicate use of the top-down projection discussed in Sec. 3.4. In addition to the aforementioned metrics, we also report the VO prediction absolute error per navigation step for $\hat{\xi}^{x}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$, $\hat{\xi}^{z}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$, and $\widehat{\theta}_{\mathcal{C}_t \to \mathcal{C}_{t+1}}$, discussed in Sec. 3.5.

Note, prior work showed that without `GPS+Compass` sensor, the policy achieves 0 SPL after 100-million-frame training and 15% SPL after 2.5-billion-frame training [53].[7] In contrast, when evaluated with perfect `GPS+Compass` sensors under noisy observations and actuations (Row 19 in Tab. 2), the policy obtains 71% SPL with 97% success rate. We now discuss to what extent each of the techniques detailed in Sec. 3 and Sec. 4.1 shrinks this gap.

Both RGB and Depth observations help visual odometry. Row 1 - 3 study the role of visual modalities for visual odometry. We find that the RGB-D model (Row 3) has lower per-step prediction error and higher navigation success rate compared to RGB-only (Row 1) and depth-only (Row 2) VO models. This finding overturns the accepted conventional wisdom in this sub-field [53, 11] that RGB models overfit and depth-only models outperform RGB-D models. We find that both RGB and depth observations are important for training a visual odometry model. We hypothesize that RGB enables better feature matching between frames. In addition, this result highlights the advantage of separately training VO model and navigation policy as they capture different features of the input observations.

Adding Dropout in the VO model learns a more robust egomotion estimator. We find significant performance improvements when using Dropout to economically mimic an ensemble for more robust egomotion prediction. Empirical results demonstrate the effectiveness of this design as success rate and SPL improve 7 and 5 points respectively (Row 3 *vs*. 4 in Tab. 2). To demonstrate the advantage of a single forward pass over multiple ones during inference, we conduct additional experiments (Row 5). We randomly select hidden units with ratio p at test time and average results of 10 forward passes. Apart from the apparent inferior results (success 42% *vs*. 68% for Row 5 *vs*. 4), the VO model's throughput drastically decreases from 118.8 FPS (frames per second) for Row 4 to 8.45 FPS for Row 5.

Learning action-specific models helps. As mentioned in Sec. 4.1, action-specific model design (SepAct) improves the navigation's success rate from 68% (Tab. 2 Row 4) to 75% (Row 8) while improving other metrics as well. Furthermore, SepAct increases the accuracy of VO prediction for all three components. To validate that this improvement is due to SepAct and not from an increased parameter count, we add two more ablations (Row 6 and 7): 1) in Row 6, a VO model was trained with 3× more parameters (12.4M) than the single-action model (3.93M) by increasing the ResNet-

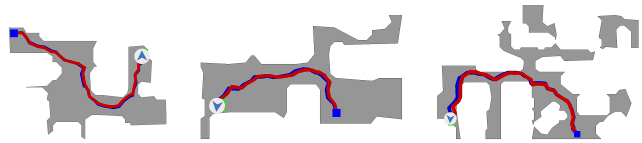
[7] Note, [53] do not train with observation and actuation noise, 15% SPL is hence an upper bound.

18 layer width twofold. Note, we observed that wider models work better than deeper ones for PointGoal navigation. Comparing Row 8 to Row 6, we can see that simply adding more parameters performs worse in success rate (75% to 70%), SPL (56% to 52%) as well as VO prediction; 2) in Row 7, instead of training separate models, we exposed a *unified* model to action information via an action embedding. Performance increases from Row 6 to Row 7 on success rate (70% to 72%), SPL (52% to 53%) and VO prediction, establishing that action information is important for such a task. However, the worse results compared to Row 8 (success and SPL both drop 3 points) confirm the effectiveness of SepAct.

Encouraging geometric invariance in the egomotion predictions is helpful. As discussed in Sec. 3.2, the VO model can benefit from exploiting the geometric invariance properties. Row 8 *vs*. Row 10 in Tab. 2 confirms the effectiveness of this technique: success rate and SPL improves two and one points respectively. To verify that this improvement indeed stems from the self-supervised signal instead of data augmentation, we conduct an ablation with a simple data augmentation for invertible actions like `turn_left` and `turn_right`. Specifically, when training the VO model for `turn_left`, apart from using the original pair of frames collected for `turn_left`, we also utilize the frames collected for the `turn_right` action by reversing the pair of observations and computing the corresponding ground-truth $SE(2)$. Similar processing is applied when training the VO model for `turn_right`. We do not apply data augmentation to `move_forward` since there do not exist situations where agents move backward. Tab. 2 shows that sole data augmentation does not help navigation performance (success and SPL remain the same across Tab. 2 Row 9 *vs*. Row 8).

Depth discretization and top-down projection account for more satisfactory results. As shown in Sec. 3.3, we add depth discretization `d-depth` to obtain a more robust egomotion estimation. Indeed, use of `d-depth` increases success rate from 77% to 79% and SPL from 57% to 60% (Tab. 2 Row 10 *vs*. Row 12). To understand whether the performance is robust to the number of `d-depth`'s channels, we ablate over 5, 10, and 20 channels in Row 11 - 13. The results verify that coarse discretization harms the navigation performance (Row 11 *vs*. Row 12). However, when the granularity increases (20 channels instead of 10), the gains from adding more channels are not significant (Row 12 *vs*. Row 13). Meanwhile, use of the soft projection discussed in Sec. 3.4 benefits PointGoal navigation improving success and SPL by two points (Row 12 *vs*. Row 18 in Tab. 2).

Every representation feature is indispensable for VO. To verify that *every input feature is required*, we conduct ablations by removing each feature (RGB, D, DD, S-Proj) from the VO model. Specifically, if we ignore the RGB representation, success drops from 81% to 72% (Row 14 *vs*. 18 in Tab. 2). Trends are similar for depth (success drops two



(a) SPL 85%. (b) SPL 80%. (c) SPL 85%.

Figure 5: Qualitative results. 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.

points from Row 16 *vs*. 18), depth discretization (success drops 4 points from Row 15 *vs*. 18), and egocentric top-down projection (Row 12 *vs*. 18). Moreover, we train our VO without any depth-related parts, *i.e.*, depth, DD, and S-Proj (Row 17). Row 17 *vs*. 18 again verifies the importance of depth. Note, the difference between Row 1 and Row 17 is that Row 17 uses Dropout, SepAct, DataAug, and GeoInv. the 7-point success rate improvement validates those technique's usefulness (Row 1 *vs*. Row 17).

Tuning RL policy with VO further improves performance. The VO model's efficiency (36 FPS for Row 18 in Tab. 2 on a 3.10GHz Intel Xeon Gold 6254 CPU and an Nvidia GeForce RTX 2080 Ti GPU) permits fine-tuning of the RL policy with respect to the VO module. In Tab. 2's Row 19, we observe overall best performance across all criteria after tuning the RL policy with only 14.7 million frames, which is much more affordable than billions of frames [53].

Comparison to other VO methods. We further compare to DeepVO [49], a supervised RNN-based VO, on PointGoal Navigation. Please see the appendix for implementation details. We train DeepVO on our collected dataset. We found DeepVO to fall short of the simplest VO model as success rate drops from our 52% to 50% (Row 0 *vs*. 1 in Tab. 2). We hypothesize that the RNN does not perform well due to little overlap between consecutive frames.

4.4. Qualitative Results

Fig. 5 shows several successful trajectories that overlay the ground-truth top-down map. We show that integrating VO techniques into a navigation policy permits to accurately guide the agent towards the point goal. For example, in Fig. 5c, the VO model is able to precisely estimate $SE(2)$ around corners and in case of collisions. More examples and failure cases are available in the appendix.

5. Conclusion

To conclude, we find classical visual odometry techniques to be surprisingly effective and yield a very strong baseline for Embodied PointGoal Navigation in a realistic setting (noisy actuation and perception; no localization sensor).

Acknowledgements: Work supported in part by NSF grants #1718221, 2008387, 2045586, MRI #1725729, and NIFA 2020-67021-32799, UIUC, Samsung, Amazon and Cisco Systems Inc. (award 1377144 - thanks for access to Arcetri).

References

[1] *Habitat Challenge 2020*, 2020. 6

[2] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *ArXiv*, 2018. 1, 2, 3, 6

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *CVPR*, 2018. 2

[4] Pierre Baldi and Peter Sadowski. Understanding dropout. *NIPS*, 2013. 4

[5] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects. *ArXiv*, 2020. 2

[6] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from rgb-d data in indoor environments. *3DV*, 2017. 2

[7] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and R. Salakhutdinov. Learning to Explore using Active Neural SLAM. *ICLR*, 2020. 6

[8] C. Chen and *et al*. A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence. *arxiv/2006.12567*, 2020. 13

[9] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. *CVPR*, 2015. 1, 6

[10] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. *CVPR*, 2018. 2

[11] Samyak Datta, Oleksandr Maksymets, Judy Hoffman, Stefan Lee, Dhruv Batra, and Devi Parikh. Integrating Egocentric Localization for More Realistic Point-Goal Navigation Agents. *CoRL*, 2020. 2, 4, 6, 7

[12] Daniel DeTone and *et al*. Superpoint: Self-supervised interest point detection and description. *CVPRW*, 2018. 14

[13] Shivam Duggal, Shenlong Wang, W. Ma, R. Hu, and R. Urtasun. DeepPruner: Learning Efficient Stereo Matching via Differentiable PatchMatch. *ICCV*, 2019. 13

[14] F. Fraundorfer and D. Scaramuzza. Visual odometry : Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics Automation Magazine*, 19(2), 2012. 2

[15] Andreas Geiger, Philip Lenz, and R. Urtasun. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. *CVPR*, 2012. 13

[16] Saurabh Gupta, Varun Tolani, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *IJCV*, 2019. 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016. 2, 5, 12

[18] Geoffrey E. Hinton, Nitish Srivastava, A. Krizhevsky, Ilya Sutskever, and R. Salakhutdinov. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. *ArXiv*, 2012. 4

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997. 5

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, 2015. 12

[21] U. Jain*, L. Weihs*, E. Kolve, A. Farhadi, S. Lazebnik, A. Kembhavi, and A. G. Schwing. A Cordial Sync: Going Beyond Marginal Policies For Multi-Agent Embodied Tasks. *ECCV*, 2020. 2

[22] U. Jain*, L. Weihs*, E. Kolve, M. Rastegrari, S. Lazebnik, A. Farhadi, A. G. Schwing, and A. Kembhavi. Two Body Problem: Collaborative Visual Task Completion. *CVPR*, 2019. 2

[23] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, M. Savva, S. Chernova, and Dhruv Batra. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? *IROS*, 2020. 6

[24] Peter Karkus, Shaojun Cai, and David Hsu. Particle SLAM-Net for Visual Navigation. *CVPR*, 2021. 1, 6

[25] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. *ICRA*, 2016. 2

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ArXiv*, 2015. 5, 12

[27] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *ArXiv*, 2017. 2

[28] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. *ECCV*, 2020. 2

[29] Hamid Laga, Laurent Valentin Jospin, Farid Boussaïd, and M. Bennamoun. A survey on deep learning techniques for stereo-based depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 13

[30] Zakaria Laskar, Iaroslav Melekhov, Surya Kalia, and Juho Kannala. Camera Relocalization by Computing Pairwise Relative Poses Using Convolutional Neural Network. *ICCV Workshop*, 2017. 2

[31] I.-J. Liu, U. Jain, R. Yeh, and A. G. Schwing. Cooperative Exploration for Multi-Agent Deep Reinforcement Learning. In *Proc. ICML*, 2021. 2

[32] I.-J. Liu, Z. Ren, R. Yeh, and A. G. Schwing. Semantic Tracklets: An Object-Centric Representation for Visual Multi-Agent Reinforcement Learning. In *Proc. IROS*, 2021. 2

[33] I.-J. Liu, R. Yeh, and A. G. Schwing. High-Throughput Synchronous Deep RL. In *Proc. NeurIPS*, 2020. 2

[34] I.-J. Liu*, R. Yeh*, and A. G. Schwing. PIC: Permutation Invariant Critic for Multi-Agent Deep Reinforcement Learning. In *Proc. CORL*, 2019. * equal contribution. 2

[35] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. PyRobot: An open-source robotics framework for research and benchmarking. *ArXiv*, 2019. 1, 6

[36] Medhini Narasimhan, Erik Wijmans, Xinlei Chen, Trevor Darrell, Dhruv Batra, Devi Parikh, and Amanpreet Singh. Seeing the un-scene: Learning amodal semantic maps for room navigation. *ECCV*, 2020. 2

[37] N. Nilsson. Shakey the Robot. 1984. 1

[38] Santhosh K. Ramakrishnan, Z. Al-Halah, and K. Grauman. Occupancy Anticipation for Efficient Exploration and Navigation. *ECCV*, 2020. 6

[39] Paul-Edouard Sarlin and *et al*. Superglue: Learning feature matching with graph neural networks. *CVPR*, 2020. 14

[40] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas A. Funkhouser, and Vladlen Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *ArXiv*, 2017. 2

[41] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. *ICCV*, 2019. 2, 6, 12

[42] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics Automation Magazine*, 18(4), 2011. 2

[43] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *ArXiv*, 2016. 12

[44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, 2017. 5, 12

[45] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *ArXiv*, 2019. 2

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014. 2, 4

[47] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard A. Newcombe. The Replica Dataset: A Digital Replica of Indoor Spaces. *ArXiv*, 2019. 2

[48] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. *ArXiv*, 2019. 2

[49] Sen Wang, Ronald Clark, Hongkai Wen, and Agathoniki Trigoni. DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. *ICRA*, 2017. 2, 4, 7, 8

[50] Xiangwei Wang, Daniel Maturana, Shichao Yang, Wenshan Wang, Qijun Chen, and Sebastian A. Scherer. Improving learning-based ego-motion estimation with homomorphism-based losses and drift correction. *IROS*, 2019. 2

[51] David Watkins-Valls, Jingxi Xu, Nicholas R. Waytowich, and Peter K. Allen. Learning your way without map or compass: Panoramic target driven visual navigation. *ArXiv*, 2019. 2

[52] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. *CVPR*, 2019. 2

[53] Erik Wijmans, Abhishek Kadian, Ari S. Morcos, Stefan Lee, Irfan Essa, D. Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. *ICLR*, 2020. 1, 2, 4, 5, 7, 8, 12

[54] Yuxin Wu and Kaiming He. Group Normalization. *ECCV*, 2018. 12

[55] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *ArXiv*, 2018. 2

[56] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: real-world perception for embodied agents. *CVPR*, 2018. 2, 12

[57] Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 2020. 2

[58] Fei Xia, Amir Roshan Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. *CVPR*, 2018. 6

[59] A. Zamir, T. Wekel, Pulkit Agrawal, Colin Wei, Jitendra Malik, and S. Savarese. Generic 3D Representation via Pose Estimation and Matching. *ECCV*, 2016. 2