ExAug: Robot-Conditioned Navigation Policies via Geometric Experience Augmentation

Noriaki Hirose1,2, Dhruv Shah1, Ajay Sridhar1 and Sergey Levine1

Abstract—Machine learning techniques rely on large and diverse datasets for generalization. Computer vision, natural language processing, and other applications can often reuse public datasets to train many different models. However, due to differences in physical configurations, it is challenging to leverage public datasets for training robotic control policies on new robot platforms or for new tasks. In this work, we propose a novel framework, ExAug, to augment the experiences of different robot platforms from multiple datasets in diverse environments. ExAug leverages a simple principle: by extracting 3D information in the form of a point cloud, we can create much more complex and structured augmentations, utilizing both generating synthetic images and geometric-aware penalization that would have been suitable in the same situation for a different robot, with different size, turning radius, and camera placement. The trained policy is evaluated on two new robot platforms with three different cameras in indoor and outdoor environments with obstacles.

I. INTRODUCTION

Machine learning methods can be used to train effective models for visual perception [1]–[3], natural language processing [4], [5], and numerous other applications [6], [7]. However, broadly generalizable models typically rely on large and highly diverse datasets, which are usually collected once and then reused repeatedly for many different models and methods. In robotics, this presents a major challenge: every robot might have a different physical configuration, such that end-to-end learning of control policies usually requires specialized data collection for each robotic platform. This calls for developing techniques that can enable learning from experience collected across different robots and sensors.

In this work, we focus in particular on the problem of vision-based navigation, where heterogeneity between robots might include different cameras, sensor placement, sizes, etc., and yet there is structural similarity in the high-level objectives of collision-avoidance and goal-reaching. Training visual navigation policies across experience from multiple robots would require accounting for changes in these parameters, and learning navigation behavior for the target robot parameters — how do we train such a robot-conditioned policy? Akin to the use of data augmentation to learn invariances for generalization in computer vision and NLP [8], [9], we propose a mechanism for extending the capabilities of learning-based visual navigation pipelines by introducing experience augmentation, or ExAug: a new form of data augmentation to learn navigation policies that can generalize to a variety of different robot parameters, such as dynamics or camera intrinsics, and is robust to small variations (due to wear or quality control). In contrast to typical augmentation schemes that operate on single data points of images or language, since we are dealing with sequential data, ExAug performs augmentations at the level of robot trajectories. Our key insight is that we can generate such augmentations for free if we have access to the geometry of the scene: using scene geometry, we can simulate what the robot’s observations would appear from a different viewpoint, with a different camera hardware, and even what actions the robot should take if it had different physical properties (e.g., size or turning radius).

ExAug uses a combination of self-supervised depth estimation to recover scene geometry, novel view synthesis for generating counterfactual robot observations, and training a policy assuming different robot parameters, to augment the robot behavior in the public datasets — in essence, giving us infinite data from the training environments. We train a shared navigation policy on a combination of trajectories from three distinct datasets representing indoor navigation, off-road driving, and on-road autonomy, on vastly different robot platforms with different sensor stacks — augmented with ExAug (Fig. 1), and show that the trained policy can successfully navigate to visual goals from novel viewpoints, in novel environments, as well on novel robot platforms, by leveraging the various affordances depicted by the different datasets.

The primary contribution of this paper is ExAug, a novel framework to augment the robot experiences in the multiple datasets to train multi-robot navigation policies. We
successfully deploy policies trained with ExAug on two different robots in a variety of indoor and outdoor driving environments, and show that such policies can generalize across a range of robot parameters such as camera intrinsics (e.g. focal lengths, distortions), viewpoints, and dynamics (e.g. robot sizes or turning radii).

II. RELATED WORK

Geometric approaches to vision-based navigation have been well-studied in the robotics community for tasks such as visual servoing [10]–[12], model predictive control [13], [14], and visual SLAM [15], [16]. However, such geometric methods rely strongly on the geometric layout of the scene, often being too conservative [17], and are susceptible to changes in the map due to dynamic objects.

Learning-based techniques have been shown to alleviate some of these challenges by operating on a topological, and not geometric, representation of the environment [18]–[20]. Other methods have also successfully used predictive models [21], [22], representation learning [23], [24], and probabilistic representations to improve navigation over long distances [25], [26]. While these approaches have been shown to generalize to environmental modifications, they strongly rely on the deployment trajectories (visual observations, robot dynamics, viewpoint, etc.) to be “in-distribution” with respect to their training datasets, which are usually collected on a single robot platform. Our key insight is that we can develop an experience augmentation technique that utilizes data from various robots and augments it to support training policies conditioned on robot parameters that generalize even to new robot configurations.

Data augmentation is a popular mechanism used to artificially increase the “span” of a dataset to encourage learning of “invariances”, hence avoiding over-fitting to a small training dataset [8], [9]. A popular use-case of augmentations in robotics is to use a simulated environment [27], [28], with randomized augmentations to boost data diversity. However, transferring policies from sim-to-real has a number of challenges due to the inability to simulate complex, real-world environments, and dynamic environmental changes [29]. Some recent work has also studied augmenting the training dataset with non-natural images [30]–[32]. ExAug proposes to use augmentations in a similar spirit to these works, but instead of applying simple image-space transformations, it applies augmentations at the level of entire trajectories by modifying both the observations as well as actions.

The closest concurrent works to ExAug are Ex-DoF [33], which studies synthetic view generation of spherical to augment training data but does not generalize across sensors and other physical robot parameters, and GNM [34], which studies direct generalization of simple goal-conditioned policies across different robots simply by training on highly diverse datasets. In contrast, ExAug studies how geometric understanding can be used to augment experience, in the form of counterfactual views and actions, and trains a policy conditioned on the robot’s configuration.

III. TRAINING GENERALIZABLE POLICIES WITH EXAUG

ExAug uses a geometric framework for data augmentation, where a local point cloud representation of the scene is reconstructed from monocular images, and then used to synthesize novel views to augment the dataset observations. This point cloud is also used to learn novel behavior via our proposed geometry-aware policy objective, providing supervision for the policy conditioned on counterfactual robot configurations (e.g., if a larger robot were in the same place, what would it do?). These two techniques together allow us to augment the robot experiences in multiple datasets to train the policy with counterfactual images and actions. Goal-conditioned policies trained with ExAug can thus be deployed on robots with novel parameters, such as new camera viewpoints, robot sizes, turning radii etc. By combining our method with a topological graph [18], [20], [35], we obtain a system that performs long-range navigation in diverse environments.

A. Novel View Synthesis via Recovered 3D Structure

Figure 2 overviews the synthetic image generation process to train the robot-conditioned policy: given a sequence of images I from a dataset, our goal is to transform them into a sequence I’ that matches the parameters of a different camera. For each image, we achieve this by (i) estimating the pixel-wise depth D, (ii) back-projecting to a 3D point cloud Qs, and (iii) projecting the point cloud into the image frame of the target camera, resulting in images I’. We formulate following process on standard image, camera, and robot coordinates (see. Fig. 4(e)).

To estimate depth from monocular RGB images, we use the self-supervised method proposed by [36]–[38] and train it on our diverse training dataset from multiple robots. Given access to this depth estimate D, we estimate a point cloud Qs from image I as Qs = f_{bproj}(D), where f_{bproj} is the learned camera back-projection function [38].

To convert this point cloud into a synthetic image I’ for a new camera (with different intrinsics and extrinsics), we first apply a coordinate-frame transformations T_{st} to obtain target-domain point cloud Qt, followed by the target camera projection function f_{proj}. Since the coordinate shift can induce mixed pixels, we use a technique of 3D-warping to project the points Qs to off-grid coordinates [i, j]c, followed by weighted bilinear interpolation to obtain the target-domain pixel values [39], [40]. Mathematically, this transforms the image I[i, j] → I_p[i, j]c, which is then resampled to obtain the discretized image I’[i, j]. To generate synthetic images corresponding to the target camera on the target robot, we only measure f_{proj} and T_{st} from the target robot and apply it in the above process.

It should be noted that our approach of view synthesis using self-supervised depth estimation is one of many different ways to synthesize novel views for augmenting the dataset [41]–[44], and the proposed experience augmentation framework is compatible with any of these alternatives.

B. Geometry-Aware Policy Learning

The above process generates synthetic images via perceptual augmentation. We now discuss how the control policy
is trained, leveraging the same point cloud representation that we used for augmentation to instead provide supervision suitable for a variety of robot types. The actions in the dataset are specific to the robot that collected each trajectory, and might not be appropriate for a robot with a different size or speed constraint. To train a policy that can generalize effectively over robot configurations, we aim to augment this data and provide supervision to the policy that indicates what a different robot would have done in the same situation. To this end, we derive a policy objective that incorporates this experience augmentation via a control objective guided by the estimated point cloud.

We design a policy architecture $\pi_\theta$ that predicts velocity commands $\{v_i, \omega_i\}_{i=1,...,N_v}$ and traversability estimates $\{t_i\}_{i=1,...,N_t}$, corresponding to the inverse probability of collision. We condition this policy on the current and goal observations $I_c, I_g$, and robot parameters corresponding to size $r_s$ and velocity constraints $v_i$, i.e., $\{v_i, \omega_i, t_i\}_{i=1,...,N_t} = \pi_\theta(I_c, I_g, r_s, v_i)$, with the policy parameterized by neural network weights $\theta$. Note that $I_c$ and $I_g$ are synthetic images generated as per Section III-A during training, though at test time the policy uses raw images from the target robot’s actual camera. We parameterize the robot as a cylinder with radius $r_s$ and a finite height $h_s$, which is used for estimating collisions with point cloud from original image of $I_c$. We use integrated velocities to estimate future robot poses $\{p_i\}_{i=1,...,N_v}$ in the frame of the local point cloud, and define a objective $J$ to train the robot-conditioned policy $\pi_\theta$ encouraging goal-reaching while avoiding collisions:

$$J(\theta) := J_{\text{pose}}(\theta) + w_y J_{\text{geo}}(\theta) + w_d J_{\text{diff}}(\theta) + w_t J_{\text{trav}}(\theta).$$  (1)

Here $w_y$, $w_d$, and $w_t$ are weighting factors. To encourage collision-avoidance, we penalize positions that lead to collisions between the cylindrical robot body and the environment points:

$$J_{\text{geo}}(\theta) = \frac{1}{L} \sum_{k=1}^{N_v} \sum_{l=1}^{N_v} \sum_{i=1}^{N_t} g[i,j](r_s - d_k[i,j])^2,$$  (2)

where $L$ is a normalization factor, $g[i,j]$ is a masked normalization weight that penalizes points inside the robot’s body and accounts for the point cloud density, and $d_k[i,j]$ is distance on the horizontal plane between $k$-th predicted waypoint $p_k$ and the back-projection of $f[i,j]$.

$$d_k = \sqrt{(Q_k^X)^2 + (Q_k^Y)^2}.$$  (3)

The point $Q_k := \{Q_k^X, Q_k^Y\}$ on the robot coordinate of $k$-th waypoint is calculated as $Q_k = T_st Q_s$, using a coordinate frame transform.

The other components of $J$ correspond to reaching the ground-truth pose $p_{gt}$, predicting the ground-truth traversability $\{t_{gt}\}_{i=1,...,N_t}$, and a smoothness term:

$$J_{\text{pose}}(\theta) = (p_{gt} - p_{ns})^2,$$
$$J_{\text{trav}}(\theta) = \sum_{i=1}^{N_t} (t_{gt} - t_i)^2,$$
$$J_{\text{diff}}(\theta) = \sum_{i=1}^{N_t} (v_{i+1} - v_i)^2 + (\omega_{i+1} - \omega_i)^2.$$  (4)

Here, $p_{ns}$ is $N_v$-th estimated robot pose by integrating $\{v_i, \omega_i\}_{i=1,...,N_v}$, $\{t_{gt}\}_{i=1,...,N_t}$ is calculated by collision check between the cylindrical robot model and estimated point clouds, similar to $J_{\text{geo}}$.

To train $\pi_\theta$ we select pairs of observations $I_c$ and $I_g$ from synthetic images that are less than $N_p$ steps apart. $N_p$ is selected for each dataset based on the robot’s speed and dataset’s frame rate, to ensure that they are close enough to establish correspondence between the images.

We assign randomized radii $r_s \in \{r_{min}, r_{max}\} \text{m}$ to consider large data diversity in robot size. Besides, we set the robot height $h_s := (h_{max} - h_{min})$ to a fixed 0.45 m, with the base $h_{min}$ set at 0.2 m to mask out noisy point cloud corresponding to the ground plane. To condition on robot velocity limitation $v_i$, we provide an angular velocity constraint of $\omega_i \in [\omega_{min}, \omega_{max}] \text{rad/s}$.

By feeding $I_c, I_g, r_s$ and $v_i$, we can calculate the joint objective $J$ in Eqn. 1. We train $\pi_\theta$ by minimizing $J$ using the Adam optimizer with the learning rate $10^{-3}$. This training process enables us to train robot-conditioned policies that can accept different robot parameters.

IV. IMPLEMENTING EXAUG IN A NAVIGATION SYSTEM

We now instantiate ExAug in a navigation system by first implementing the low-level robot-conditioned policy and then integrating it with a navigation system based on topological graphs.

A. Implementation Details: Policy Learning

For our evaluation systems (see Section V-C), we set the limits of variation of the robot radii and the angular velocity as $\{r_{min}, r_{max}\} = \{0.0, 1.0\}$ and $\{\omega_{min}, \omega_{max}\} = \{0.5, 1.5\}$ to adapt to new robots. We choose $N_p = 8$ steps and weights $\{w_y, w_d, w_t\}$ to be $\{5e3, 0.025, 0.25\}$ after running hyperparameter sweeps.

Figure 3 describes the neural network architecture of $\pi_\theta$. An 8-layer CNN is used to extract the image features $z$ from $I_c$ and $I_g$, with each layer using BatchNorm and ReLu activations. The predicted velocity commands $\{v_i, \omega_i\}_{i=1,...,N_v}$ from 3 fully-connected layers “FCv” are conditioned on the robot parameters $\{r_s, v_i\}$ and $z$. A scaled tanh activation
is given to limit the output velocities as per the specified constraints \(v_i\).

For estimating traversability \(\{t_i\}_{i=1,\ldots,N_f}\), we integrate the velocities to obtain waypoints predictions and feed them to a set of fully-connected layers “FCt” along with the observation embedding \(z\) and target robot pose \(r_g\) for \(i\in (0,1)\). Although \(r_s = r'\) in training, we found the flexibility of an independent \(r_s \neq r'\) crucial to the collision-avoidance performance of our system in inference: e.g. setting \(r_s = 0.3\) m dictates the conservativeness of the action commands, whereas \(r_s = 0.2\) dictates the collision predictions, which can be used as a hard safety constraint for triggering an emergency stop.

B. Navigation system

Since our policy does not allow us to feed the goal image at far position, we construct a mobile robot system that uses the trained policy at the low level, coupled with a topological graph for planning to evaluate in challenging long navigation scenarios [18]. For the image-goal navigation task, the objective is to navigate to the desired goal image \(I_{g_{\text{goal}}}\) by solely relying on egocentric visual observations \(I_c\) and searching for a sequence of subgoals \(\{I_{g_i}\}_{i=1,\ldots,N_g}\) in the topological graph \(\mathcal{M}\).

Following [21], [25], the navigation system comprises three “modules”, tasked with (i) localization, (ii) low-level control, and (iii) safety. For (i), we follow the setup of Hirose et. al. [25], where the current observation is localized to the nearest node \(i_c\) from the \(N_i = 5\) adjacent subgoal images, and pass the subgoal image \(I_{g_{i_c+1}}\) as the next subgoal to the policy module. The policy module uses \(\pi_{\theta}\) to obtain velocity commands, which are used in a receding-horizon manner to control the robot, and traversability estimates, which are used by the safety module for emergency stoppage.

V. Multi-Robot Datasets and Setup

In this section, we describe the datasets used for training the point cloud estimator and our policy, and the robot platforms used for evaluation.

A. Training Datasets

In order to train a generalizable policy that can leverage diverse datasets, we pick three publicly available navigation datasets that vary in their collection platform, visual sensors, and dynamics. This allows us to train policies that can learn shared representations across these widely varying datasets, and generalize to new environments (both indoors and outdoors) and new robots.

GO Stanford (GS): The GS dataset [21] consists of 10 hours of tele-operated trajectories collected across multiple buildings in a university campus. The data was collected on a TurtleBot2 platform equipped with the Ricoh Theta S spherical camera.

RECON: The RECON dataset [45] consists of 30 hours of self-supervised trajectories collected in an off-road environment. This data was collected on a Clearpath Jackal UGV equipped with an ELP fisheye camera.

KITTI: KITTI [46] is a dataset collected on the roads of Germany, with trajectories recorded from a narrow FoV camera mounted on a driving car. We use a subset of 10 sequences from the KITTI Odometry dataset for training [47].

In training, we set the maximum interval \(N_p\) between \(I_c\) and \(I_g\) as 12 for GS and RECON and 2 for KITTI due to the maximum speed of each platform and the frame rate.

B. Implementation Details: View Synthesis

Since each dataset contains actions from a single camera, we use the view synthesis process described in Sec. III-A to augment the datasets with novel viewpoints. We synthesize views for three target cameras: Intel RealSense (narrow FoV), ELP Fisheye (wide FoV) and Ricoh Theta S (spherical) at a hypothesetical camera pose \([0.2,0.0,0.3]\) on the robot coordinate to train three policies for each target camera.

We use a self-supervised depth and pose estimation pipeline [36], [48]. We resize the images for each dataset to a standard size of \(416 \times 128 = (N_U \times N_V)\); for data with a spherical camera, we discard the rear-facing camera due to robot body occlusions. Following Hirose et. al. [38], we introduce the learnable camera model to use the dataset without camera parameters and provide supervision for pose estimation during training to resolve scale ambiguities in depth estimates. Please refer to the original paper [38] for further implementation details. To generate novel views, we project the 2.5D reconstruction of the scene into a \(128 \times 128\) image frame for each target camera, obtained by using its camera parameters.

C. Evaluation Setup

We evaluate policies trained with ExAug on two new robot platforms with a variety of modifications to evaluate different
robot sizes, camera hardware, viewpoints etc., as shown in Figure 4: [a-c] show a custom robot platform, Vizbot [49], which is a new robot without any training data. We deploy the Vizbot in multiple different configurations, such as with a spherical, narrow, or wide FoV camera, artificially boosted robot size (with a cardboard cut-out), and modified camera pose (by mounting the camera on a raised platform). We also evaluate our policies on the LoCoBot platform (Figure 4[d]), with a different robot base and camera pose.

In addition to closed-loop evaluation, we also perform offline evaluation using one hour’s worth of navigation trajectories collected with a Vizbot by teleoperating it in both indoor and outdoor environments. Offline evaluation serves as a proxy for closed-loop evaluation during the training process to identify the best hyperparameters.

VI. EVALUATION

Our experiments aim to study the following questions:

Q1. Can ExAug augment experience from multiple datasets to train a control policy that generalizes to new robot configurations and robots?

Q2. Can ExAug outperform prior methods, as well as policies trained on single-robot datasets?

A. COMPARATIVE ANALYSIS

We implement ExAug on a Vizbot and compare it against competitive baselines in a number of challenging indoor and outdoor environments. All our baselines use privileged omni-directional observations from a spherical camera, and we compare the downstream navigation performance against ExAug with access to different camera observations, including narrow FoV cameras. Despite this, ExAug consistently outperforms baselines, with a higher success rate and fewer collisions.

Random: A control policy that randomly generates velocities between same upper and lower velocity boundaries.

Imitation Learning: A control policy to imitate the expert velocity commands by using $\ell_2$ regression.

DVMPC [21]: A controller based on a velocity-conditioned predictive model that is trained via minimizing the image difference between the predicted images and the goal image.

Imitation Learning and DVMPC use raw images from the best single-robot dataset (GS), since sharing heterogeneous data leads to worse performance.

In evaluation on an offline dataset of real-world trajectories, Table I reports the predicted goal arrival, collision-free rates, traversability accuracy, and test loss values of each method. From Table I[a], we observe that ExAug consistently achieves higher goal-arrival rates and collision-free rates, as well as a lower loss estimate. We also ablate the augmentation and geometric loss components of ExAug. ExAug (full) with synthetic images at target camera pose can outperform the ablation of view augmentation with raw images at different camera pose. In addition, our geometric loss performs to avoid collision.

We further compare ExAug against DVMPC in 12 challenging real-world indoor and outdoor environments for the task of goal-reaching, with 3 trials per environment. Note that DVMPC expects spherical images due to its strong reliance on the geometry of the scene, whereas our method can work with any camera type. In each environment, we collect a topological map by manually teleoperating the robot from start to goal, saving “image nodes” at 0.5Hz. We randomly place novel obstacles on or between subgoal positions, after subgoal collection, in half of the environments.

Table I[b] presents the performance of the different variants of ExAug and DVMPC. Comparing performance with the same camera (spherical), our method vastly outperforms DVMPC: with over a 100% improvement in goal-arrival rates (GA) and 75% reduction in collisions. The performance gap is most significant in challenging environments sharp, dynamic maneuvers and novel obstacles, where DVMPC fails to avoid the obstacle and loses track of the goal. Furthermore, we find that ExAug continues to perform strongly from cameras with limited FoV, like the RealSense camera, outperforming DVMPC from a spherical camera.
TABLE I: Quantitative comparisons. In offline evaluation [a], we report the predicted goal-arrival rate (PGA), predicted collision-free rate (PCF), predicted traversability accuracy (PTA), and the full policy objective $J$. In closed-loop evaluation [b], we report the task completion rate (TC), goal arrival rate (GA), and collision-free rate (CF). [a] and [c] use a spherical camera and a RealSense camera for evaluation, respectively. In [b], we evaluate three different cameras. ExAug consistently outperforms the baseline methods in most cases, and in comparing the method with different datasets [c], we find that including all datasets leads to best performance.

| Method           | PGA  | PCF  | PTA  | $J$  |
|------------------|------|------|------|------|
| Random           | 0.351| 0.939| -    | 1.483|
| Imitation Learning | 0.751| 0.888| -    | 0.844|
| DVMPM            | 0.805| 0.889| -    | 0.558|
| ExAug wo geo. loss | 0.801| 0.929| 0.897| 0.367|
| ExAug (full)     | 0.833| 0.945| 0.858| 0.319|

B. Can ExAug Control Different Robots?

Next, we qualitatively show the above behaviors in a closed-loop evaluation with novel robot configurations (Fig. 4). We systematically evaluate ExAug on modified versions of a Vizbot with (i) different camera mounting heights, (ii) with different physical sizes, and (iii) with different dynamics constraints. We also show that the same policy can control a LoCoBot, a new robot absent in the training datasets.

For evaluating invariance to viewpoint change, we evaluate the two methods on a robot with two different camera height configurations. The results show that ExAug is robust to viewpoint changes.

C. Does Training on Multiple Datasets Help?

A central hypothesis in our work is that, by leveraging datasets from different platforms in different environments and combining them with experience augmentation, we can train generalizable policies that can control robots with different sensors and physical properties. In this section, we evaluate the relative contribution of each portion of our combined dataset to overall performance, so as to ascertain whether more diverse data actually improves performance.

Towards this, we train policies with ExAug using subsets of the three datasets and evaluate the policies with RealSense on offline real-world trajectories. Table II(c) shows the 3-dataset policy consistently outperforming other variants, with the lowest loss value (Eqn. 1). We hypothesize that training on larger, more diverse datasets encourages the model to learn a more generalizable representation of the observations, and results in better downstream navigation performance.

VII. CONCLUSIONS

We proposed a framework for training a robot-conditioned policy for vision-based navigation by performing experience augmentation on heterogeneous multi-robot datasets. We propose ExAug, a technique that uses point clouds recovered from monocular images to construct synthetic views and counterfactual action labels to supervise the policy with trajectories that other robots would have taken in the same situation. The resulting policy can be conditioned on robot parameters, such as size and velocity constraints, and deployed on new robots and in new environments without additional training. We demonstrate our method on two new robots and in two new environments.

Our method does have a number of limitations. First, we must manually select the robot parameters to condition on. Second, we still require the robots to be structurally similar – e.g., all robots have forward-facing cameras. An exciting direction for future work would be to utilize the point clouds to also construct synthetic readings for other types of sensors, such as radar, and further extend the range of robots the system can control, potentially with learned latent robot embeddings.

TABLE II: Comparing performances with perturbed viewpoint. We evaluate the performance of the two methods on two different camera height configurations. The results show that ExAug is robust to viewpoint changes.

| Method         | Configuration | TC   | GA   | CF   |
|----------------|---------------|------|------|------|
| DVMPM          | High (0.6m)   | 0.831| 0.722| 0.722|
|                | Low (0.3m)    | 0.678| 0.389| 0.611|
| ExAug          | High (0.6m)   | 0.961| 0.889| 0.889|
|                | Low (0.3m)    | 0.939| 0.889| 0.944|

VIII. Ablation study of dataset in offline data

| Method | Cam.type | TC   | GA   | CF   | GS  | RECON | KITTI | PGA  | PCF  | PTA  | J   |
|--------|----------|------|------|------|-----|-------|-------|------|------|------|-----|
| DVMPC  | spherical| 0.678| 0.389| 0.844| ✓  | ✚    | ✚    | 0.647 | 0.954 | 0.816 | 0.713 |
| ExAug  | spherical| 0.690| 0.500| 0.722| ✚ | ✓    | ✓    | 0.674 | 0.944 | 0.820 | 0.665 |
|        | wide FoV | 0.817| 0.556| 0.944| ✓  |       |       | 0.669 | 0.953 | 0.804 | 0.688 |
|        | narrow FoV| 0.605| 0.487| 0.944| ✓  |       |       | 0.387 | 0.775 | 0.767 | 1.170 |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
[45] D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine, “Rapid exploration for open-world navigation with latent goal models,” arXiv preprint arXiv:2104.05859, 2021.

[46] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in 2012 IEEE conference on computer vision and pattern recognition. IEEE, 2012, pp. 3354–3361.

[47] “Kitti visual odometry benchmark 2012,” https://www.cvlibs.net/datasets/kitti/eval_odometry.php.

[48] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, “Digging into self-supervised monocular depth prediction,” The International Conference on Computer Vision (ICCV), October 2019.

[49] T. Niwa, S. Taguchi, and N. Hirose, “Spatio-temporal graph localization networks for image-based navigation,” arXiv preprint arXiv:2204.13237, 2022.