ExAug: Robot-Conditioned Navigation Policies via Geometric Experience Augmentation

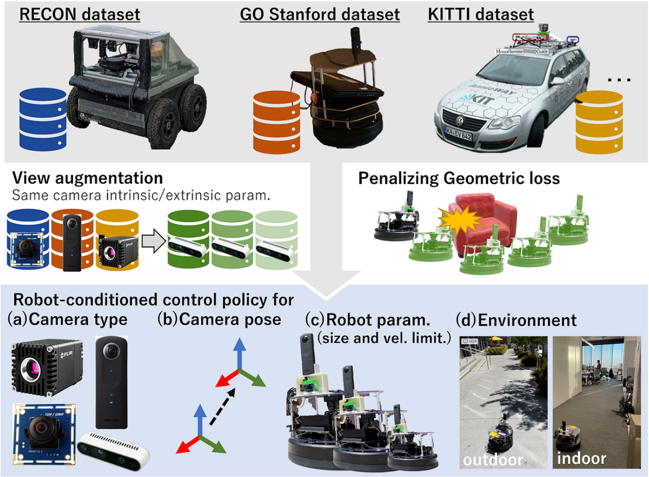
Noriaki Hirose[1,2], Dhruv Shah[1], Ajay Sridhar[1] and Sergey Levine[1]

***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).



https://www.amazon.com/ http://www.cvlibs.net/datasets/kitti/ https://www.flir.com/iis/machine-vision/

Fig. 1: **Learning navigation via 3D geometric experience augmentation.** Recovering a point cloud allows the system to imagine what the same situation would look like for a different robot, and what that different robot would need to do in that situation.

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

[1]UC Berkeley, [2]Toyota Motor North America
https://sites.google.com/view/exaug-nav

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 generalization 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 Q_s, 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 Q_s from image I as $Q_s = f_{\text{bproj}}(D)$, 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 Q_t, 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 Q_t to off-grid coordinates $[i_c, j_c]$, followed by weighted bilinear interpolation to obtain the target-domain pixel values [39], [40]. Mathematically, this transforms the image $I[i, j] \rightarrow I_p[i_c, j_c]$, 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

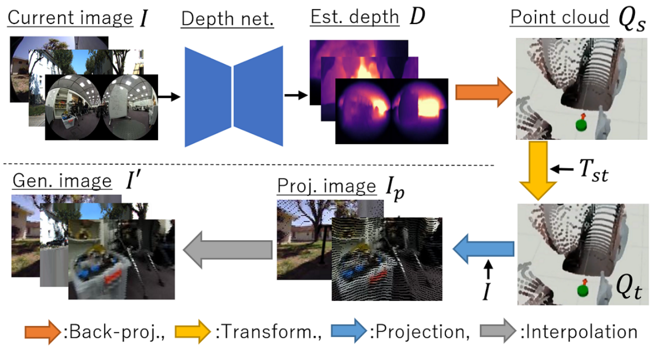


Fig. 2: **Overview of view synthesis.** We generate the images by projecting the estimated 3D points on the target image plane with interpolation.

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 π_θ that predicts velocity commands $\{v_i, \omega_i\}_{i=1 \dots N_s}$ and traversability estimates $\{t_i\}_{i=1 \dots N_s}$, 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_l, i.e., $\{v_i, \omega_i, t_i\}_{i=1 \dots N_s} = \pi_\theta(I_c, I_g, r_s, v_l)$, with the policy parameterized by neural network weights θ. 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 $\{p_i\}_{i=1 \dots N_s}$ in the frame of the local point cloud, and define a objective J to train the robot-conditioned policy π_θ encouraging goal-reaching while avoiding collisions:

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

Here w_g, 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:

$$J_{\text{geo}}(\theta) = \frac{1}{L} \sum_{k=1}^{N_s} \sum_{j=1}^{N_V} \sum_{i=1}^{N_U} g[i,j](r_s - d_k[i,j])^2, \tag{2}$$

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 $I[i,j]$,

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

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

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

$$\begin{aligned} J_{\text{pose}}(\theta) &= (p_{gt} - p_{N_s})^2, \\ J_{\text{trav}}(\theta) &= \sum_{i=1}^{N_s} (t_i^{gt} - t_i)^2, \\ J_{\text{diff}}(\theta) &= \sum_{i=1}^{N_s - 1} (v_{i+1} - v_i)^2 + (\omega_{i+1} - \omega_i)^2. \end{aligned} \tag{4}$$

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

To train π_θ 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 $r_s \in \{r_{\text{min}}, r_{\text{max}}\}$m to consider large data diversity in robot size. Besides, we set the robot height $h_s := (h_{\text{max}} - h_{\text{min}})$ 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_l, we provide an angular velocity constraint of $\omega_l \in [\omega_{\text{min}}, \omega_{\text{max}}]$ rad/s.

By feeding I_c, I_g, r_s and v_l, we can calculate the joint objective J in Eqn. 1. We train π_θ 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 $\{r_{\text{min}}, r_{\text{max}}\} = \{0.0,\ 1.0\}$ and $\{\omega_{\text{min}}, \omega_{\text{max}}\} = \{0.5,\ 1.5\}$ to adapt to new robots. We choose $N_p = 8$ steps and weights $\{w_g, w_d, w_t\}$ to be $\{5e3, 0.025, 0.25\}$ after running hyperparameter sweeps.

Figure 3 describes the neural network architecture of π_θ. 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 $\{v_i, \omega_i\}_{i=1 \dots N_s}$ from 3 fully-connected layers "FCv" are conditioned on the robot parameters $\{r_s, v_l\}$ and z. A scaled tanh activation

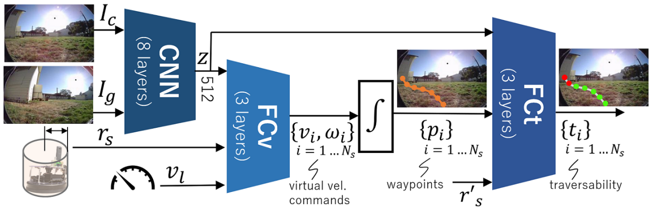


Fig. 3: **Network structure of our control policy π_θ.** "CNN" extracts image feature z from I_c and I_g. "FCv" generates the virtual velocity commands conditioned on goal and robot parameters. "FCt" estimates traversability at each virtual position.

is given to limit the output velocities as per the specified constraints v_l.

For estimating traversability $\{t_i\}_{i=1 \dots N_s}$, 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 size r'_s, followed by a sigmoid function to limit $t_i \in (0, 1)$. Although $r_s = r'_s$ in training, we found the flexibility of an independent $r'_s \neq r_s$ 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 $I_{g_{N_g}}$ by solely relying on egocentric visual observations I_c and searching for a sequence of subgoals $\{I_{g_i}\}_{i=1 \dots N_g}$ in the topological graph $\mathcal{M}$.

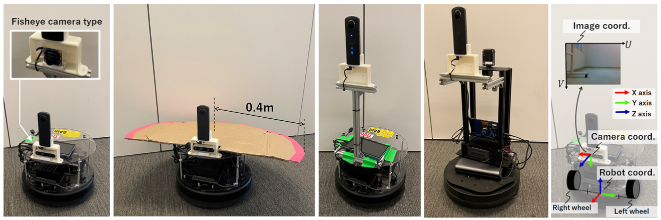
Following [21], [25], the navigation system comprises three "modules", tasked with (i) localization, (ii) low-level control, and (iii) safety. For (i), we follow the setup of Hirose et. al. [25], where the current observation is localized to the nearest node i_c from the $N_t = 5$ adjacent subgoal images, and pass the subgoal image $I_{g(i_c+1)}$ as the *next subgoal* to the policy module. The policy module uses π_θ 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.



[a] original (radius=0.2 m) [b] double-size (radius=0.4 m) [c] higher camera pose [d] LoCoBot with spherical camera [e] Coordinate definition

Fig. 4: **The robots in our evaluation.** [a] shows the Vizbot with a spherical camera and a narrow FoV camera. We can replace the narrow FoV camera with the wide FoV camera. [b] shows a double-size Vizbot with a cardboard body. [c] shows the Vizbot with the spherical camera at a higher pose, [d] shows the LoCoBot with a spherical camera, and [e] shows the coordinate frame.

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.

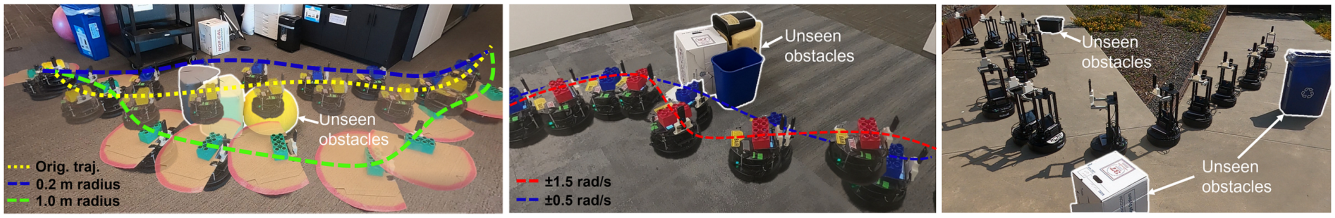
B. Implementation Details: View Synthesis

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 hypothetical 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 128×128 image frame for each target camera, obtained by using its camera parameters.

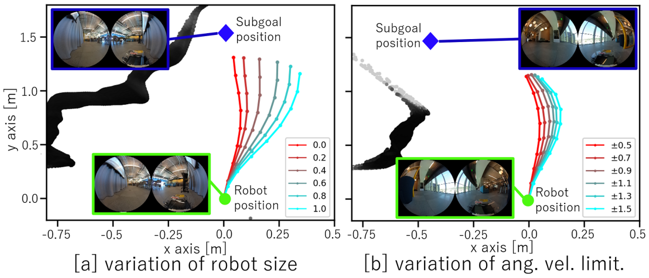
C. Evaluation Setup

We evaluate policies trained with ExAug on two *new* robot platforms with a variety of modifications to evaluate different



[a] Variation of robot size [b] Variation of angular vel. limit. [c] Navigation using LoCoBot

Fig. 5: **Qualitative behavior of ExAug policies.** "Orig. traj." is the teleoperated trajectory for collecting the graph. [a] Our control policy with $r_s = 1.0$ makes a wider turn to avoid collision, instead of going through the narrow path with $r_s = 0.2$. [b] Our control policy with larger angular velocity limits makes a sharper turn to preserve clearance. [c] Our method controls a new robot, the LocoBot, in an outdoor setting. "Unseen obstacles" are placed obstacles after collecting the graph along as a sequence of subgoals.



[a] variation of robot size [b] variation of ang. vel. limit.

Fig. 6: **Policy outputs for different parameters.** We visualize the generated trajectory from our policy for different robot sizes in [a] and angular velocity constraints in [b].

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 ℓ_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.

[a] **Comparison with baselines in offline data**

Method	PGA	PCF	PTA	$J^\dagger$
Random	0.351	0.939	-	1.483
Imitation Learning	0.751	0.888	-	0.414
DVMPC	0.805	0.889	-	0.358
ExAug wo data aug.	0.801	0.929	**0.897**	0.367
ExAug wo geo. loss	0.799	0.919	0.848	0.333
ExAug (full)	**0.833**	**0.945**	0.858	**0.319**

$J^\dagger := J_{\text{pose}} + w_g J_{\text{geo}} + w_d J_{\text{diff}}$
(since the baselines don't predict traversability)

[b] **Closed-loop navigation using a real robot.**

Env.	Method	Cam.type	TC	GA	CF
Indoor	DVMPC	spherical	0.678	0.389	0.611
	ExAug	spherical	0.939	0.889	0.944
		wide FoV	0.817	0.556	0.944
		narrow FoV	0.689	0.500	0.722
Outdoor	DVMPC	spherical	0.528	0.278	0.889
	ExAug	spherical	0.881	0.722	0.944
		wide FoV	0.853	0.722	0.833
		narrow FoV	0.817	0.500	0.833

[c] **Ablation study of dataset in offline data**

GS	RECON	KITTI	PGA	PCF	PTA	J
✓	✓	✓	**0.674**	**0.944**	**0.820**	**0.655**
✓	✓		**0.674**	**0.950**	**0.815**	0.680
✓		✓	**0.669**	**0.953**	0.804	0.688
	✓	✓	0.387	0.775	0.767	1,170
✓			0.647	**0.954**	**0.816**	0.713
	✓		0.229	0.747	0.756	1.067
		✓	0.206	0.751	0.759	1.177

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

Method	Configuration	TC	GA	CF
DVMPC	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

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 that are 30cm apart. For ExAug, we train two separate policies, one for each target camera height, and compare against DVMPC. Table II shows that, while DVMPC struggles with goal-reaching and collision avoidance due to the out-of-distribution observations, ExAug continues to perform strongly and suffers no degradation.

We also perturb the robot size r_s and dynamics constraints on angular velocity ω_l and find that the trained policies can account for these differences. Fig. 5[a] shows the qualitative behavior of ExAug under different size parameters: when its radius is increased to 1m, it takes an alternative path around the obstacle because the shortest path would lead to collision with its large footprint, and it successfully reaches the goal. Fig. 6[a] shows the output of our policy for different values of r_s, overlaid on a point cloud cross-section, showing that large r_s leads to increasingly conservative trajectories. Fig. 5[b] shows the qualitative behavior of the robot under different dynamics constraints: when the angular velocity is limited (blue), the robot cannot take sharp maneuvers and instead takes a smoother trajectory to avoid the obstacles. Fig.6[b] shows the outputs of our policy for different ω_l, suggesting that the policy can account for different constraints.

Lastly, Fig. 6[c] shows a timelapse of ExAug deployed on a LoCoBot. Despite having a higher camera pose and smaller angular velocity range, our method can successfully guide a LoCoBot to the goal while avoiding unseen obstacles in a challenging outdoor environment.

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 I[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.

APPENDIX

A. Masked normalization weight in J_{geo}

Our geometric-aware objective, J_{geo} can penalize the robot positions that collide with the environment. In J_{geo} of eq.(2), we give the masked normalization weight $g[i,j] = m^g_k[i,j] \cdot w^g_k[i,j]$. In this section, we explain our implementation of $m^g_k[i,j]$ and $w^g_k[i,j]$, respectively. $m^g_k[i,j]$ is the binary value to remove the effect of $(r_s - d_k[i,j])^2$ in the cases where the 3D point $Q_k[i,j]$ is outside of the robot's area.

$$m^g_k[i,j] = \begin{cases} 1, & \text{(inside of robot area)} \\ 0, & \text{(outside of robot area)} \end{cases} \tag{5}$$

$w^g_k[i,j]$ is the weighting variable to balance the geometric cost according to the sparsity of the 3D points.

$$\begin{aligned} w^g_k[i,j] \quad = \quad & f_{dist}(Q_k[i-1,j], Q_k[i+1,j]) \\ & \times f_{dist}(Q_k[i,j-1], Q_k[i,j+1]). \end{aligned} \tag{6}$$

Assuming the corresponding 3D point is representative point in the area formed by the adjacent four points, we dictate $w^g_k[i,j]$ by its approximate area. Here $f_{dist}(A,B)$ is the function to calculate the distance between A and B. Without w^g_k, J_{geo} gives larger penalization to obstacles with dense 3D points, and it causes the robot to collide with obstacles with sparse 3D points.

The other parameters N_U and N_V in eq.(2) are the number of pixels on U and V axis. And L is defined as $L = \sum_{k=1}^{N_s} \sum_{j=1}^{N_V} \sum_{i=1}^{N_U} m^g_k[i,j]$ to calculate the mean of the effective 3D points inside of the robot's area.

B. Process of view synthesis

Our view synthesis approach generates synthetic images via an estimated point cloud. As shown in Section III-A, we project the point clouds onto the image plane. Then, we interpolate the projected images to fill in the pixels without a projected color value. Here, we explain the detail implementation of both the projection and interpolation process.

By projecting the point cloud Q_t on to the image plane using intrinsic parameters measured from the target camera, we can get the corresponding pixel position $[i_c, j_c]$ in target image coordinates as follows:

$$[i_c, j_c] = f_{\text{proj}}(Q_t[i,j]), \tag{7}$$

where $f_{\text{proj}}()$ is the projection function using measured camera intrinsic parameters. Here, i_c and j_c are scalar values (not integer values). Hence, we can have four candidate pixel positions to project the pixel value of $I[i,j]$. To fill in many pixels, we merge four intermediate images $\{I_{p_i}\}_{k=1\ldots4}$ in the projection process. The four intermediate images can be given as follows:

$$\begin{aligned} & I_{p_1}[\lceil i_c \rceil, \lceil j_c \rceil] = I[i,j], I_{p_2}[\lceil i_c \rceil, \lfloor j_c \rfloor] = I[i,j], \\ & I_{p_3}[\lfloor i_c \rfloor, \lceil j_c \rceil] = I[i,j], I_{p_4}[\lfloor i_c \rfloor, \lfloor j_c \rfloor] = I[i,j], \end{aligned} \tag{8}$$

where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ indicate ceil and floor functions, respectively. $\{I_{p_k}\}_{k=1\ldots4}$ can be obtained by executing eq.(8) for all pixels, starting from the one with the largest depth to consider occlusion. The projected image I_p can be calculated by a weighted sum of $\{I_{p_k}\}_{k=1\ldots4}$ as follows:

$$I_p = \left(\sum_{k=1}^{4} w_{p_k} I_{p_k} \right) / \sum_{k=1}^{4} w_{p_k} \tag{9}$$

where $\{w_{p_k}\}_{k=1\ldots4}$ are weight matrices that give larger weight to the pixels that are closer to $[i_c, j_c]$,

$$w_{p_s} = \left(\sum_{k=1}^{4} l_k - l_s \right) / \sum_{k=1}^{4} l_k \tag{10}$$

where $\{l_k\}_{k=1\ldots4}$ are the lengths between $[i_c, j_c]$ and the corresponding pixel position.

$$\begin{aligned} l_1 &= \sqrt{(\lceil i_c \rceil - i_c)^2 + (\lceil j_c \rceil - j_c)^2} \\ l_2 &= \sqrt{(\lceil i_c \rceil - i_c)^2 + (\lfloor j_c \rfloor - j_c)^2} \\ l_3 &= \sqrt{(\lfloor i_c \rfloor - i_c)^2 + (\lceil j_c \rceil - j_c)^2} \\ l_4 &= \sqrt{(\lfloor i_c \rfloor - i_c)^2 + (\lfloor j_c \rfloor - j_c)^2} \end{aligned} \tag{11}$$

The last process in our view augmentation is interpolation of I_p. Although we can fill a lot of pixel positions by merging in eq.(9), there are still a lot of blank pixels in I_p. Nearest neighbors is one of the simplest methods to fill them. However, it makes the generated images noisy. We fill the blank pixels with a weighted sum of the closest four pixels to generate I', similar to eq.(9).

It should be noted that alternative methods are possible in projection and interpolation [39], [40]. We employ the above process because it is fast on GPU.

C. Visualization of data augmentation

Figure. 7 shows synthetic images of our view augmentations and raw images of the target camera to visualize our view augmentations. We mounted the spherical camera and the narrow FoV camera at different positions on same robot platform. We generated synthetic images corresponding to the narrow FoV camera from the spherical camera images. Although the generated images are a bit blurry and noisy when comparing them to the raw images, there are enough details to learn the positional relationships between the images.

References

[1] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255. 1

[2] K. He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1

[3] B. Mildenhall *et al.*, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021. 1

[4] A. Wang *et al.*, "Glue: A multi-task benchmark and analysis platform for natural language understanding," in *7th International Conference on Learning Representations, ICLR 2019*, 2019. 1

[5] M. Lewis *et al.*, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019. 1

[6] J. A. Sidey-Gibbons and C. J. Sidey-Gibbons, "Machine learning in medicine: a practical introduction," *BMC medical research methodology*, vol. 19, no. 1, pp. 1–18, 2019. 1

[7] T. Davenport and R. Kalakota, "The potential for artificial intelligence in healthcare," *Future healthcare journal*, vol. 6, no. 2, p. 94, 2019. 1

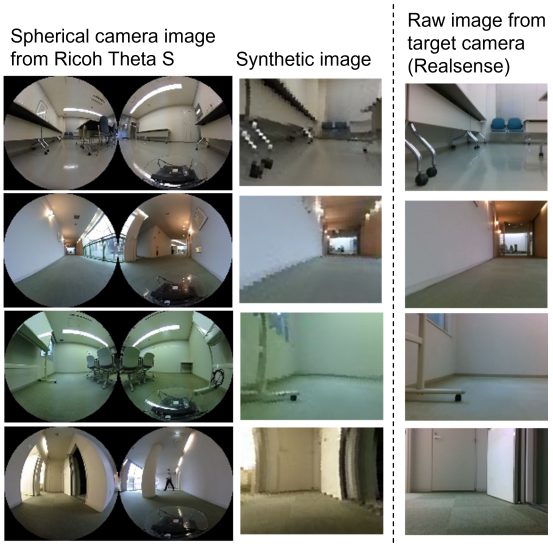

Spherical camera image from Ricoh Theta S
Synthetic image
Raw image from target camera (Realsense)

Fig. 7: **Comparison of the synthetic images with the raw images of the target camera, Realsense.** We mount the spherical camera and the Realsense at different positions on our robot platform. We generate synthetic images corresponding to the Realsense from the spherical camera images.

[8] C. Shorten *et al.*, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, p. 60, 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0 1, 2

[9] S. Y. Feng *et al.*, "A survey of data augmentation approaches for nlp," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 968–988. 1, 2

[10] S. Hutchinson *et al.*, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996. 2

[11] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006. 2

[12] ——, "Visual servo control. ii. advanced approaches [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007. 2

[13] Z. Li *et al.*, "Vision-based model predictive control for steering of a nonholonomic mobile robot," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 2, pp. 553–564, 2015. 2

[14] M. Sauvée *et al.*, "Image based visual servoing through nonlinear model predictive control," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 1776–1781. 2

[15] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017. 2

[16] A. Kim and R. M. Eustice, "Perception-driven navigation: Active visual slam for robotic area coverage," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3196–3203. 2

[17] G. Kahn *et al.*, "Badgr: An autonomous self-supervised learning-based navigation system," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1312–1319, 2021. 2

[18] N. Savinov *et al.*, "Semi-parametric topological memory for navigation," in *International Conference on Learning Representations*, 2018. 2, 4

[19] K. Chen *et al.*, "A behavioral approach to visual navigation with graph localization networks," *arXiv preprint arXiv:1903.00445*, 2019. 2

[20] D. Shah *et al.*, "Ving: Learning open-world navigation with visual goals," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 215–13 222. 2

[21] N. Hirose *et al.*, "Deep visual mpc-policy learning for navigation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3184–3191, 2019. 2, 4, 5

[22] D. Pathak *et al.*, "Zero-shot visual imitation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 2050–2053. 2

[23] Y. Zhu *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3357–3364. 2

[24] G. Kahn *et al.*, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5129–5136. 2

[25] N. Hirose *et al.*, "Probabilistic visual navigation with bidirectional image prediction," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1539–1546. 2, 4

[26] D. Shah and S. Levine, "Viking: Vision-based kilometer-scale navigation with geographic hints," *arXiv preprint arXiv:2202.11271*, 2022. 2

[27] F. Xia *et al.*, "Gibson env: Real-world perception for embodied agents," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9068–9079. 2

[28] M. Savva *et al.*, "Habitat: A platform for embodied ai research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9339–9347. 2

[29] A. Kadian *et al.*, "Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance?" *IEEE Robotics and Automation Letters*, 2020. 2

[30] H. Kataoka *et al.*, "Pre-training without natural images," in *Proceedings of the Asian Conference on Computer Vision*, 2020. 2

[31] K. Nakashima *et al.*, "Can vision transformers learn without natural images?" in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, 2022, pp. 1990–1998. 2

[32] Y. Wang and C.-Y. Ko, "Visual pre-training for navigation: What can we learn from noise?" *arXiv preprint arXiv:2207.00052*, 2022. 2

[33] K. Tahara and N. Hirose, "Ex-dof: Expansion of action degree-of-freedom with virtual camera rotation for omnidirectional image," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 10 382–10 389. 2

[34] D. Shah *et al.*, "GNM: A General Navigation Model to Drive Any Robot," in *arXiV*, 2022. [Online]. Available: https://arxiv.org/abs/2210.03370 2

[35] X. Meng *et al.*, "Scaling local control to large-scale topological navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 672–678. 2

[36] T. Zhou *et al.*, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1851–1858. 2, 4

[37] C. Godard *et al.*, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3828–3838. 2

[38] N. Hirose and K. Tahara, "Depth360: Monocular depth estimation using learnable axisymmetric camera model for spherical camera image," *arXiv preprint arXiv:2110.10415*, 2021. 2, 4

[39] H.-Y. Huang and S.-Y. Huang, "Fast hole filling for view synthesis in free viewpoint video," *Electronics*, vol. 9, no. 6, p. 906, 2020. 2, 7

[40] I. Daribo and B. Pesquet-Popescu, "Depth-aided image inpainting for novel view synthesis," in *2010 IEEE International Workshop on Multimedia Signal Processing*, 2010, pp. 167–170. 2, 7

[41] T. Zhou *et al.*, "View synthesis by appearance flow," in *European conference on computer vision*. Springer, 2016, pp. 286–301. 2

[42] G. Riegler and V. Koltun, "Free view synthesis," in *European Conference on Computer Vision*. Springer, 2020, pp. 623–640. 2

[43] B. Mildenhall *et al.*, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European conference on computer vision*, 2020. 2

[44] O. Wiles *et al.*, "Synsin: End-to-end view synthesis from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7467–7477. 2

[45] D. Shah *et al.*, "Rapid exploration for open-world navigation with latent goal models," *arXiv preprint arXiv:2104.05859*, 2021. 4

[46] A. Geiger *et al.*, "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. 4

[47] "Kitti visual odometry benchmark 2012," https://www.cvlibs.net/datasets/kitti/eval_odometry.php). 4

[48] C. Godard *et al.*, "Digging into self-supervised monocular depth prediction," *The International Conference on Computer Vision (ICCV)*, October 2019. 4

[49] T. Niwa *et al.*, "Spatio-temporal graph localization networks for image-based navigation," *arXiv preprint arXiv:2204.13237*, 2022. 5