


# Autonomous Ground Navigation in Highly Constrained Spaces

Lessons Learned From the Benchmark Autonomous Robot Navigation Challenge at ICRA 2022

By Xuesu Xiao , Zifan Xu, Zizhao Wang, Yunlong Song, Garrett Warnell, Peter Stone, Tingnan Zhang, Shravan Ravi, Gary Wang, Haresh Karnan, Joydeep Biswas, Nicholas Mohammad, Lauren Bramblett, Rahul Peddi, Nicola Bezzo, Zhanteng Xie, and Philip Dames

The Benchmark Autonomous Robot Navigation (BARN) Challenge took place at the 2022 IEEE International Conference on Robotics and Automation (ICRA), in Philadelphia, PA, USA. The aim of the challenge was to evaluate state-of-the-art autonomous ground navigation systems for moving robots through highly constrained environments in a safe and efficient manner. Specifically, the task was to navigate a standardized differential drive ground robot from a predefined start location to a goal location as quickly as possible without colliding with any obstacles, both in simulation and in the real world. Five teams from all over the world participated in the qualifying simulation competition, three of which were invited to compete with one another at a set of physical obstacle courses at the conference center in Philadelphia. The competition results suggest that autonomous ground navigation in highly constrained spaces, despite seeming simple for experienced roboticists, is actually far from being a solved problem. In this article, we discuss the challenge, the approaches used by the top three winning teams, and lessons learned to direct future research.

## BARN Challenge Overview

Designing autonomous robot navigation systems has been a topic of interest to the robotics community for decades.

Indeed, there currently exist many such systems that allow robots to move from one point to another in a collision-free manner [e.g., open source implementations in Robot Operating System (ROS) that have extensions to different vehicle types], which may create the perception that autonomous ground navigation is a solved problem. This perception may be reinforced by the fact that many mobile robot researchers have moved on to orthogonal navigation problems beyond the traditional metric (geometric) formulation and that focus only on path optimality and obstacle avoidance. These orthogonal problems include, among others, learning navigation systems in a data-driven manner, navigating in off-road and social contexts, and multirobot navigation.

However, autonomous mobile robots still struggle in many ostensibly simple scenarios, especially during real-world deployment. For example, even when the problem is simply formulated as traditional metric navigation so that the only requirement is to avoid obstacles on the way to the goal, robots still often get stuck and collide with obstacles when trying to navigate in naturally cluttered daily households; in constrained outdoor structures, including narrow walkways and ramps; and in congested social spaces, such as classrooms, offices, and cafeterias. In such scenarios, extensive engineering effort is typically required to deploy existing approaches, and this requirement presents a challenge for large-scale unsupervised real-world robot deployment. Overcoming this challenge requires sys-

tems that can both successfully and efficiently navigate a wide variety of environments with confidence.

The BARN Challenge was a competition at ICRA 2022, in Philadelphia, that aimed to evaluate the capability of state-of-the-art navigation systems to solve the previously mentioned challenge, especially in highly constrained environments where robots need to squeeze between obstacles to navigate to the goal. To compete in the BARN Challenge, each participating team needed to develop an entire software stack for navigation for a standardized and provided mobile robot. In particular, the competition provided a Clearpath Jackal with a 2D 270° field-of-view Hokuyo lidar for perception and a differential drive system with 2-m/s maximum speed for actuation. The aim of each team was to develop the navigation software stack needed to autonomously drive the robot from a given starting location through a dense obstacle field to a given goal and to accomplish this task without any collisions with obstacles and any human interventions.

The team whose system could best accomplish this task within the least amount of time would win the competition. The BARN Challenge had two phases: a qualifying phase evaluated in simulation and a final phase evaluated in a set of physical obstacle courses. The qualifying phase took place before the ICRA 2022 conference, using the BARN data set, which is composed of 300 obstacle courses in Gazebo simulation and randomly generated by cellular automata. The top three teams from the

simulation phase were then invited to compete in three different physical obstacle courses set up by the organizers at ICRA 2022, in the Philadelphia Convention Center. In this article, we report on the simulation qualifier and physical finals of the BARN Challenge at ICRA 2022, present the approaches used by the top three teams, and discuss lessons learned from the challenge that point out future research directions to solve the problem of autonomous ground navigation in highly constrained spaces.

### Simulation Qualifier

The BARN Challenge started on 29 March, two months before the ICRA 2022 conference, with a standardized simulation qualifier. The qualifier used the BARN data set, which consists of  $300 \times 5 \times 5$ -m obstacle environments randomly generated by cellular automata (see examples in Figure 1), each with a predefined start and goal. These obstacle environments range from relatively open spaces, where the robot barely needs to turn, to highly dense fields, where the robot needs to squeeze between obstacles with minimal clearance. The BARN environments are open to the public and were intended to be used by the participating teams to develop their navigation stack. Another 50 unseen environments, which are not available to the public, were generated to evaluate the teams' systems. A random BARN environment generator was also provided to teams so that they could generate their own unseen test environments (<https://github.com/dperille/jackal-map-creation>).

In addition to the 300 BARN environments, six baseline approaches were also provided for the participants' reference, ranging from classical sampling-based and optimization-based navigation systems to end-to-end machine learning methods and hybrid approaches. All baselines were implementations of different local planners used in conjunction with Dijkstra's search as the global planner in the ROS `move_base` navigation stack. To facilitate participation, a training pipeline capable of running the standardized Jackal robot in the Gazebo simulator with ROS Melodic (in Ubuntu 18.04), with the option of being containerized in Docker and Singularity containers for fast and standardized setup and evaluation, was also provided ([https://github.com/Daffan/ros\\_jackal](https://github.com/Daffan/ros_jackal)).

### Rules

Each participating team was required to submit its developed navigation system as a (collection of) launchable ROS node(s). The challenge utilized a standardized evaluation pipeline (<https://github.com/Daffan/nav-competition-icra2022>) to run each team's navigation system and compute a standardized performance metric that considered the navigation success rate (collisions and not reaching the goal counted as failure), actual traversal time, and environment difficulty (measured by the optimal traversal time). Specifically, the score  $s$  for navigating each environment  $i$  was computed as

$$s_i = 1_i^{\text{success}} \times \frac{OT_i}{\text{clip}(AT_i, 4OT_i, 8OT_i)}$$

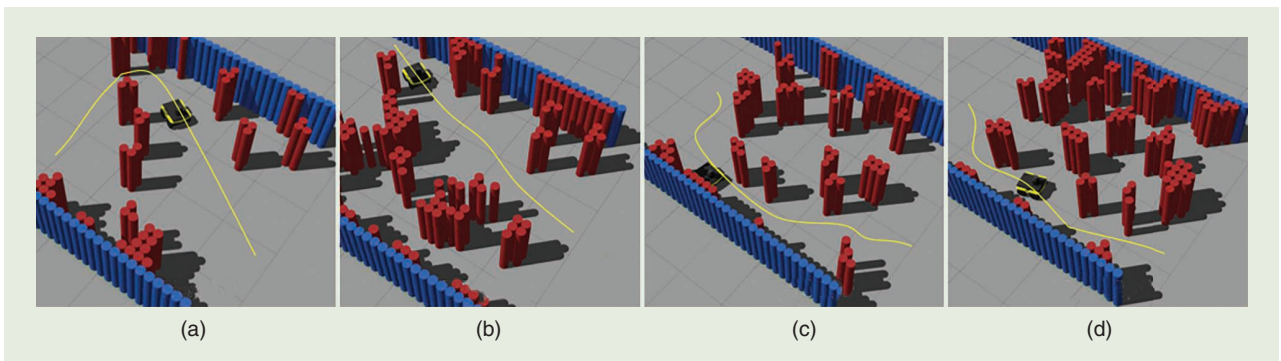
where the indicator function  $1_{\text{success}}$  evaluates to one if the robot reaches the navigation goal without any collisions and evaluates to zero otherwise. Here,  $AT$  denotes the actual traversal time, while  $OT$  denotes the optimal traversal time as an indicator of the environment difficulty and measured by the shortest traversal time, assuming the robot always travels at its maximum speed (2 m/s):

$$OT_i = \frac{\text{Path Length}_i}{\text{Maximal Speed}}$$

The path length is provided by the BARN data set, based on Dijkstra's search from the given start to goal. The clip function clips  $AT$  within  $4OT$  and  $8OT$  to assure that navigating extremely quickly or slowly in easy or difficult environments, respectively, will not disproportionately scale the score. The overall score of each team is the score averaged over all 50 unseen test BARN environments, with 10 trials in each environment. Higher scores indicate better navigation performance. The six baseline scores were between 0.1627 and 0.2334. The maximum possible score based on our metric was 0.25.

### Results

The simulation qualifier started on 29 March and lasted through 22 May. In total, five teams from all over the world submitted their navigation systems. The performance of each submission was evaluated by a standard evaluation pipeline, and the results are shown in Table 1. All methods



**Figure 1.** Four example BARN environments in the Gazebo simulator (ordered by the ascending relative difficulty level): (a) World 0, (b) World 99, (c) World 199, and (d) World 299.

outperformed the baseline dynamic window approach (DWA), with both a 2- and 0.5-m/s maximum speed, the latter of which is the default local planner for the Jackal robot. However, only one approach (from Temple University) outperformed all baselines. The top three teams from the simulation qualifier, i.e., Temple Robotics and Artificial Intelligence Lab (TRAIL), from Temple University; Autonomous Mobile Robotics Laboratory (AMRL), from the University of

Texas at Austin (UT Austin); and Autonomous Mobile Robots (AMR) Lab, from the University of Virginia (UVA), were invited to the physical finals at ICRA 2022.

### Physical Finals

The physical finals took place at ICRA 2022, in the Philadelphia Convention Center, on 25 and 26 May. Two physical Jackal robots with the same sensors and actuators were provided by the competition sponsor, Clearpath Robotics.

### Rules

Physical obstacle courses were set up using approximately 200 cardboard boxes in the convention center (Figure 2). Because the goal of the challenge was to test a navigation system's ability to perform local planning, all three physical obstacle courses had an obvious passage that connected the start and goal locations (i.e., the robot should not have been confused by global planning at all), but the overall obstacle clearance when traversing this passage was designed to be very constrained, e.g., a few centimeters around the robot.

While it was the organizers' original intention to run exactly the same navigation systems submitted by the three top teams and use the same scoring metric in the simulation qualifiers in the physical finals, these systems suffered from (surprisingly) poor navigation performance in the real world (not even being able to finish one single trial without any collisions). Therefore, the organizers decided to change the rules by giving each team 30 min before competing in each of the three physical obstacle courses to fine-tune their navigation systems. After all three teams had this chance to set up for a particular obstacle course, the actual physical finals started as a 30-min timed session for each team. In each 30-min session, a team tested its navigation system in the obstacle course and notified the organizers when it was ready to time a competition trial. Each team had the opportunity to run five timed trials (after notifying the organizers). The fastest three out of the five timed trials were counted, and the team that had the most successful trials (reaching the goal without any collisions) was the winner. In the case of a tie, the team with the fastest average traversal time would be declared the winner.

### Results

The physical finals took place on 25 and 26 May (see the final award ceremony in Figure 3). The three teams' navigation performance is provided in Table 2. Since all navigation systems navigated at roughly the same speed, the final results were determined solely by the success



**Figure 2.** One of the three physical obstacle courses during the finals.

**Table 1. The simulation results.**

Rank	Team/Method (University)	Score
1	Temple Robotics and Artificial Intelligence Lab (Temple University)	0.2415
2	LfLH (baseline)	0.2334
3	Autonomous Mobile Robotics Laboratory (University of Texas at Austin)	0.231
4	Autonomous Mobile Robots (University of Virginia)	0.22
5	E band (baseline)	0.2053
6	End to end (baseline)	0.2042
7	APPLR-DWA (baseline)	0.1979
8	Yiyuiii (Nanjing University)	0.1969
9	NavBot (Indian Institute of Science)	0.1733
10	Fast (2.0 m/s) DWA (baseline)	0.1709
11	Default (0.5 m/s) DWA (baseline)	0.1627

DWA: dynamic window approach; LfLH: Learning from Learned Hallucination; APPLR: Adaptive Planner Parameter Learning.





**Figure 3.** (From left) The competition sponsor (Clearpath Robotics), competition organizers, and Temple, UVA, and UT Austin teams.

rate of the best three out of five timed trials for each team. Surprisingly, the best system in simulation, by Temple University, exhibited the lowest success rate, while UT Austin's system enjoyed the highest rate of success.

### Top Three Teams and Approaches

In this section, we report the approaches used by the three winning teams.

#### UT Austin

To enable robust, repeatable, and safe navigation in the constrained spaces frequently found in BARN, the UT Austin team, AMRL (<https://amrl.cs.utexas.edu/>), utilized state-of-the-art classical approaches to handle localization, planning, and control along with an automated pipeline to visualize and debug continuous integration. To plan feasible paths to reach the goal location while avoiding obstacles, a medium-horizon kinematic planner from ROS `move_base` was used, combined with a discrete path rollout greedy planner for local kinodynamic planning from AMRL's graph navigation stack. This two-stage hierarchical planning generated safe motion plans for the robot to make progress toward the goal while reactively avoiding obstacles along its path by using lidar scans.

Additionally, since the environment contained tight spaces that were challenging to navigate through, it was observed that accurate motion estima-

tion of the robot was crucial to deploying a planning-based navigation controller in an unmapped environment. When executing sharp turns in constrained environments, poor estimates of the robot's motion negatively interfered with costmap updates in `move_base` and often prevented the midlevel planner from discovering any feasible path to the goal.

Toward addressing this problem, episodic non-Markov localization (EnML) was utilized, which fuses the lidar range scans with wheel odometry through non-Markov Bayesian updates. Combining EnML with two-stage hierarchical planning proved to be useful in safely handling constrained spaces. Additionally, the UT Austin team developed custom automated tools to generate visualizations for debugging that helped identify failure cases easily, perform manual hyperparameter tuning, and accelerate bug fixes during the competition.

While classical approaches helped solve a majority of the environments in the BARN challenge, significant challenges still remain for navigation in extremely constrained spaces. For example, the two-stage hierarchical planning module does not explore unobserved regions of the environment before committing to a kinematically feasible path. This sometimes leads to suboptimal paths causing a longer time to be taken to reach the goal. We posit that a learnable midlevel planner with the ability to

actively explore the environment appropriately to plan the optimal path may be a promising future direction of research to improve autonomous navigation in constrained spaces.

#### UVA

To quickly and robustly navigate through the unknown cluttered BARN challenge environments, the UVA team, AMR (<https://www.bezzorobotics.com/>), developed a mapless "follow-the-gap" planning scheme that 1) detects open gaps for the robot to follow to reach a final goal and 2) plans local goals to reach those open gaps without colliding with intermediate obstacles. The framework expands upon the UVA AMR lab's previous work. Figure 4(a) illustrates the framework, displaying the laser scan point cloud of a world from the BARN data set along with the detected intermediate gaps  $g_1, g_2,$  and  $g_3$ ; vehicle position  $x_r \in \mathbb{R}^2$ ; and final goal position  $x^* \in \mathbb{R}^2$ .

Figure 4(b) describes the local planner, which provides course corrections

**Table 2. The physical results.**

Rank	Team/Method (University)	Success/Total Trials
1	AMRL (UT Austin)	8/9
2	AMR (UVA)	4/9
3	TRAIL (Temple University)	2/9

for the robot to avoid obstacles while reaching a selected gap goal. The approach takes advantage of the fact that gaps start and end at discontinuities in the laser scan and leverages this principle to find intermediate gap goals for navigation. Let  $p_i, p_{i+1} \in \mathbb{R}^2$  be adjacent points in the laser scan and  $R$  denote the maximum sensing range of the lidar. The first discontinuity, referred to as *type 1*, occurs when the distance between the adjacent readings is larger than the circumscribed diameter  $d_r$  of the robot:  $\|p_i - p_{i+1}\|_2 > d_r$ . The second discontinuity, type 2, occurs when one of the two readings is outside the lidar's sensing range:  $\|p_i - x_r\|_2 \geq R \oplus \|p_{i+1} - x_r\|_2 \geq R$ . If  $\|p_{i+1} - x_r\|_2 > \|p_i - x_r\|_2$ , the discontinuity is referred to as *rising*; otherwise, it is *falling*. In the following, we describe how to leverage these discontinuities to identify gaps.

The first step in gap detection is to perform a forward pass from  $p_0$  to  $p_{n-1}$  in the laser point cloud scan for rising type 1 and type 2 discontinuities. Let  $p_i$  denote the location of the first rising discontinuity and  $\mathcal{L}^+ = \{i+1, \dots, n-1\}$ . This point becomes the beginning of the gap. To determine the end, we find the point  $p_j$  closest to  $p_i$  such that  $j \in \mathcal{L}^+$ . That is,

$$p_j = \operatorname{argmin}_{j \in \mathcal{L}^+} \|p_i - p_j\|_2. \quad (1)$$

The process continues, starting from  $p_{j+1}$ . Once the forward pass is complete, a mirrored backward pass from

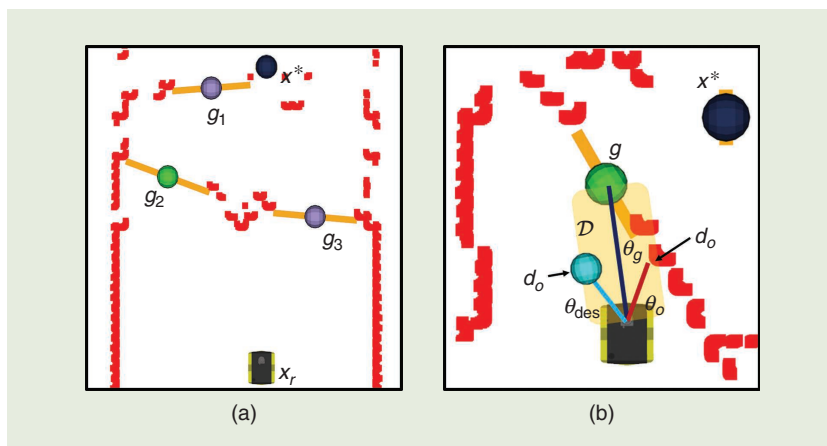
$p_{n-1}$  to  $p_0$  is done to find gaps via falling discontinuities. At each detected gap, defined as  $g_i = (a_i, b_i)$ , a tuple of the start and end points is added to a quadtree  $\mathcal{T}_g$ , which keeps track of where all previously identified and visited gaps are located. If any gap already exists in the tree, it is ignored.

Once  $\mathcal{T}_g$  is updated, a gap  $g^* \in \mathcal{T}_g$  is selected to be the intermediate goal if it is determined that the final goal  $x^*$  is not admissible. In this context, admissibility is determined by checking whether a given goal is navigable; that is, from the laser scan data, a path is known to exist from the robot position to the goal. The check is done by using a similar algorithm, as discussed by Minguez and Montano, which, given any start point  $x_a$  and endpoint  $x_b$ , ensures that no inflated obstacles block the robot along the line connecting the two points.

The process to select the gap goal from  $\mathcal{T}_g$  when  $x^*$  is inadmissible is outlined in Algorithm 1. At each iteration, the algorithm finds the closest gap  $g^*$  to the final goal  $x^*$ . If  $g^*$  is inadmissible from the robot's current position, properties of quadtree queries are utilized to find all gaps  $G' \subseteq \mathcal{T}_g$  that must be passed as the robot drives from  $x_r$  to  $g^*$ . The algorithm then iteratively finds the closest admissible gap  $g \in G'$  to the robot that is also admissible to  $g^*$ , meaning that the robot knows that a feasible path from  $x_r$  to  $g$  and from  $g$  to  $g^*$  exists. If no  $g$  satisfies this constraint for the given  $g^*$ , the process repeats,

with  $g^*$  as the next-closest gap to  $x^*$ , and terminates once an admissible gap is found. For clarity, Figure 4(a) gives an example of the goal selection process. The final goal  $x^*$  is not admissible, nor is the closest gap to it,  $g_1$ . However,  $x_r$  to  $g_2$  is admissible as well as  $g_2$  to  $g_1$ . Thus,  $g_2$  is selected as the intermediate goal, and the selection process repeats once the robot reaches  $g_2$ .

Even though the selected gap goal is admissible, a direct path to it may not be feasible given the configuration of the obstacles within an environment. For example, a robot navigating directly to  $g$  in Figure 4(b) will collide with the obstacles shown by the laser scan data. To prevent such issues from arising, a local planner is utilized that replans the mobile robot's trajectory at every time step if collision is imminent. The direct path to the goal is formulated as a region  $\mathcal{D}$ , which accounts for the relative heading to the goal,  $\theta_g$ , and the diameter  $d_r$  of the robot. The region  $\mathcal{D}$  is checked against the laser scan points for any obstacles. Let  $\mathbf{p}$  represent all obstacle coordinates within region  $\mathcal{D}$ . If no obstacles are in  $\mathcal{D}$ , that is,  $\mathbf{p} = \emptyset$ , the robot is sent directly to the gap goal,  $g$ . If there are multiple obstacles within  $\mathcal{D}$ , the one closest to the robot is selected. Let  $d_o$  represent the distance to the closest obstacle and  $\theta_o$  represent the direction of the obstacle with respect to the robot's heading. The new desired heading is then computed by accounting for the offset between the



**Figure 4.** Examples of the UVA team's (a) detected gaps in a simulated BARN environment and (b) local planner obstacle avoidance.

### Algorithm 1: Find Gap Goal (UVA Team)

```

1: Input: quadtree  $\mathcal{T}_g$ , robot position  $x_r$ , and final goal  $x^*$ 
2: Output: gap goal  $g^*$ 
3: while  $\mathcal{T}_g \neq \emptyset$  & !isAdmissible( $x_r, g^*$ )
   do
4:    $g^* \leftarrow \operatorname{argmin}_{g^* \in \mathcal{T}_g} \|x^* - g^*\|_2$ 
5:    $\mathcal{T}_g \leftarrow \mathcal{T}_g \setminus \{g^*\}$ 
6:   # Returns children in descending order of dist to  $x_r$ 
7:    $G' \leftarrow \operatorname{getChildren}(g^*, x_r, \mathcal{T}_g)$ 
8:   for  $g \in G'$  do
9:     if isAdmissible( $x_r, g$ ) & isAdmissible( $g, g^*$ ) then
10:       $g^* = g$ 
11:     end if
12:   end for
13: end while
14: return  $g^*$ 

```

goal and obstacle to the gap goal,  $\theta_{des} = \theta_g + (\theta_g - \theta_o)$ , and the local goal is placed at a distance of  $d_o$  in this desired direction [shown in teal in Figure 4(b)].

The inputs to the robot are angular and linear velocities and are determined using proportional controllers:

$$\begin{cases} \omega = \min(k_t(\theta_{des} - \theta_r), \omega_{max}) \\ v = k_v v_{max} \left(1 - \alpha \frac{|\omega|}{\omega_{max}}\right) \end{cases} \quad (2)$$

where  $k_t$ ,  $k_v$ , and  $\alpha$  are constant proportional gains,  $\theta_r$  is the current heading of the robot, and  $\omega_{max}$  and  $v_{max}$  are the maximum angular and linear velocities, respectively.

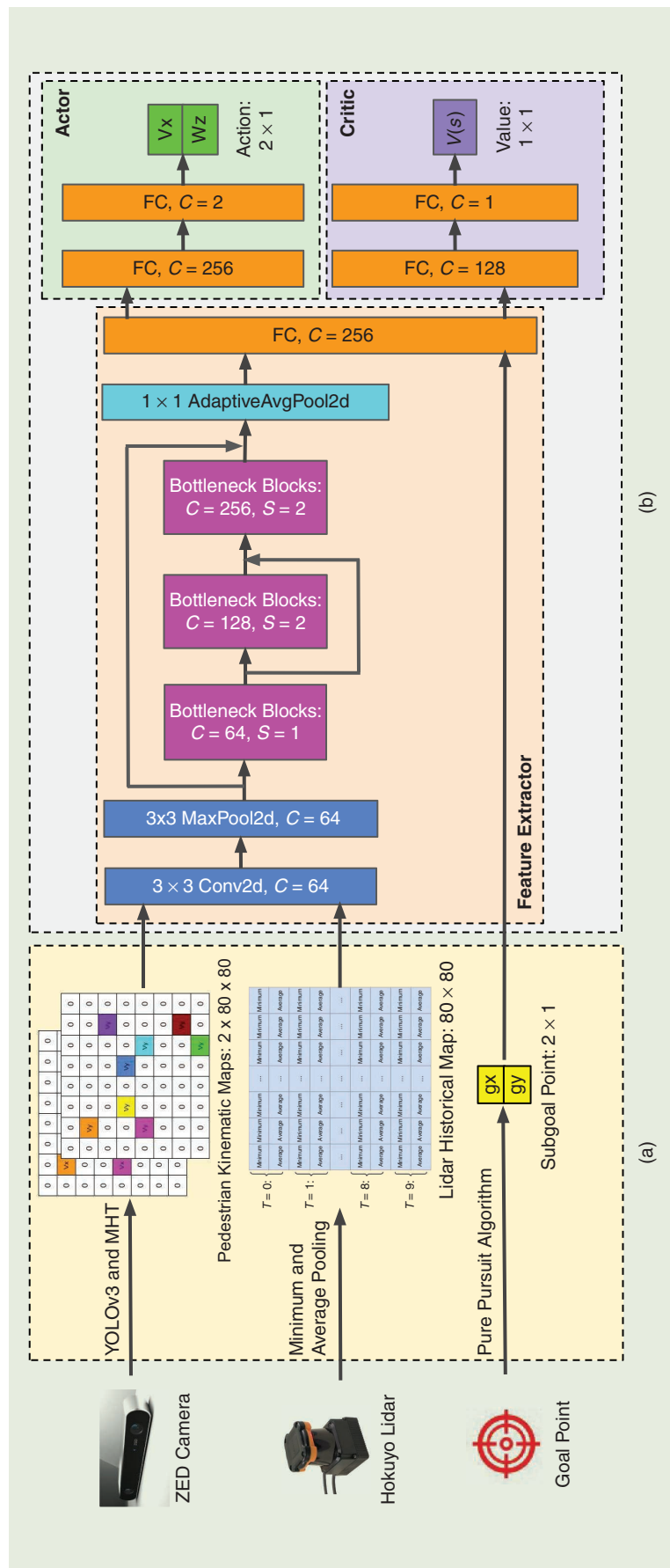
### Temple University

The team at Temple (<https://sites.temple.edu/trail/>) used a deep reinforcement learning (DRL)-based control policy, called DRL-velocity obstacles (VO), originally designed for safe and efficient navigation through crowded dynamic environments. The system architecture of the VO control policy, provided in Figure 5, is divided into two modules: preprocessing and the DRL network.

### Preprocessing Module

Instead of directly feeding the raw sensor data into deep neural networks like other works, the DRL-VO control policy utilizes preprocessed data as the network input. There are three types of inputs that capture different aspects of the scene:

- 1) *Pedestrians*: To track pedestrians, the raw red-green-blue image data and point cloud data from a ZED camera are fed into the You Only Look Once v3 object detector to get pedestrian detections. These detections are passed into a multiple hypothesis tracker to estimate the number of pedestrians and their kinematics (i.e., position and velocity). These pedestrian kinematics are encoded into two  $80 \times 80$  occupancy grid-style maps.
- 2) *Scene geometry*: To track the geometry, the past 10 scans (0.5 s) of lidar data are collected. Each lidar scan is downsampled using a combination of minimum pooling and average pooling,



**Figure 5.** The system architecture of the Temple team's DRL-VO control policy. Raw sensor data from the ZED camera and Hokuyo lidar as well as the goal point are fed into a preprocessing module to create intermediate data representations. These low-level intermediate features are fused in a feature extractor network to obtain high-level abstract features. The actor network uses these abstract features to generate steering commands to control the robot, while the critic network outputs the state value for training the policy. The (a) preprocessing module and (b) DRL network module early fusion architecture. YOLO: You Only Look Once; MHT: multiple hypothesis tracker.



data are then reshaped and stacked to create an  $80 \times 80$  lidar map.

- 3) *Goal location*: To inform the robot of where to go, the final goal point and its corresponding nominal path are fed into the pure pursuit algorithm to extract the subgoal point, which is fed into the DRL-VO network.

This novel preprocessed data representation is one key idea of the DRL-VO control policy, allowing it to bridge the simulated-to-real gap and generalize to new scenarios better than other end-to-end policies.

### DRL Network Module

The DRL-VO control policy uses an early fusion network architecture to combine the pedestrian and lidar data at the input layer to obtain high-level abstract feature maps. This early fusion architecture facilitates the design of small networks with fewer parameters than late fusion works, which is the key to deploying them on resource-constrained robots. These high-level feature maps are combined with the subgoal point and fed into the actor and critic networks to generate suitable control inputs and a state value, respectively.

### Training

To ensure that the DRL-VO policy leads the robot to navigate safely and efficiently, the team at Temple developed a new multiobjective reward function that rewards travel toward the goal, avoiding collisions with static objects, having a smooth path, and actively avoiding future collisions with pedestrians. This final term, which utilizes the concept of velocity obstacles, is key to the success of the DRL-VO control policy. With this reward function, the DRL-VO policy is trained via the proximal policy optimization algorithm in a 3D lobby Gazebo simulation environment with 34 moving pedestrians, using a Turtlebot2 robot with a ZED camera and a 2D Hokuyo lidar.

### Deployment

The Temple team directly deployed the DRL-VO policy trained on a Turtlebot2 in the BARN Challenge without any model

fine-tuning. To achieve this, the team had to account for three key differences:

- 1) *Unknown map*: During development, the DRL-VO policy used a known occupancy grid map of the static environment for localization, which was not available in the BARN challenge. To account for this, the localization module (`amcl`) was removed from the software stack and replaced with a laser odometry module.
- 2) *Static environment*: The DRL-VO policy was designed to function in dynamic environments. To account for the fact that the environments in the BARN Challenge were all static and highly constrained, the pedestrian map was set to all zeros.
- 3) *Different robot model*: The DRL-VO policy was trained on a Turtlebot2, which has a different maximum speed and footprint compared to the Jackal platform. In the BARN Challenge, the maximum speed of the robot was modified based on the robot's proximity to obstacles. This led to the robot moving slowly (0.5 m/s, the same speed as the Turtlebot2) when near obstacles and quickly (up to 2 m/s, the maximum speed of the Jackal) when in an open area.

### Discussions

Based on each team's approach and the navigation performance observed during the competition, we now discuss lessons learned from the BARN Challenge and point out promising future research directions to push the boundaries of efficient mobile robot navigation in highly constrained spaces.

### Generalizability of Learning-Based Systems

One surprising discrepancy between the simulation qualifier and the physical finals is the contrasting performance of the DRL-VO approach by Temple University, which outperformed all baselines and other participants in simulation but suffers from frequent collisions with obstacles in the real world. Despite the fact that the organizers modified the rules during the physical finals to allow the teams to make last-minute modifica-

tions to their navigation systems, DRL-VO still did not perform well in all three physical obstacle courses. The TRAIL team believes this was due to two types of gaps between the simulator and the real world: 1) the real-world environments were all highly constrained, whereas the simulator environments contained both constrained and unconstrained environments and 2) the DRL-VO policy was learned on a Turtlebot2 model (which has a smaller physical footprint than a Jackal). Most of the collisions during the hardware tests were light grazes on the side, so a robot with a smaller size may have remained collision-free.

The stark performance contrast between simulation and the real world suggests a generalizability gap for the reinforcement learning approach. It was not practical for the team to retrain a new system on site during the competition, given the impractically massive amount of training data required for reinforcement learning. How to address this generalizability gap and make a navigation policy trained in simulation generalizable to the real world and different robot/sensor configurations remains to be investigated, even for such a simple static obstacle avoidance problem.

Another potential way to address such inevitable generalizability gaps is to seek help from a secondary classical planner that identifies out-of-distribution scenarios in the real world and recovers from them through rule-based heuristics. In fact, for the last two physical courses, the Temple team tried to implement just such a "recovery planner" as a backup for DRL-VO: when a potential collision is detected (i.e., the robot faces an obstacle that is too close), the robot rotates in place to head toward an empty space in an attempt to better match the real-world distribution to that in the simulation during training. Although such a recovery planner did help in some scenarios, it was difficult for it to cover every difficult scenario and navigate through the entire obstacle course.

Indeed, the Temple team spent time during the 30-min timed sessions to

fine-tune the parameters of the recovery planner but found it difficult to find a single set of parameters to recover the robot from all out-of-distribution scenarios while not accidentally driving the robot into such scenarios throughout the entire course. On one hand, the simple nature of the recovery planner designed on site during the competition contributed to the failure. On the other hand, tuning the parameters of a planner to cover as many scenarios as possible remains a difficult problem and will be discussed further in the following.

### ***Tunability of Classical Systems***

Similar to Temple's rule-based recovery planner, the UT Austin team's entire navigation system relies on classical methods: EnML localization, a medium-horizon kinematic planner, and a local rollout-based kinodynamic planner. Inevitably, these classical approaches have numerous tuning parameters, which need to be correctly tuned to cover as many scenarios as possible. A natural disadvantage of relying on a single set of parameters to cover all different difficult scenarios in the BARN Challenge (e.g., dense obstacle fields, narrow curving hallways, and relatively open spaces) is the inevitable tradeoff, or compromise, to sacrifice performance in some scenarios to succeed in others and to decrease speed for better safety.

Indeed, the UT Austin team's strategy in the physical finals was to spend the first 20 min in the 30-min timed session to fine-tune the system parameters until a good set of parameters that allowed successful navigation through the entire obstacle course was found, then finish three successful "safety trials" first and, finally, retune the system to enable faster and more aggressive but riskier navigation behaviors to reduce the average traversal time. Although most such "speed trials" failed, luckily for the UT Austin team, other teams' inability to safely finish three collision-free trials to the goal made it the winner of the BARN Challenge but only by having a higher success rate (not faster navigation).

Two orthogonal future research directions can potentially help with the tunability of navigation systems: 1) developing planners free of tunable parameters on site during deployment, such as end-to-end learning approaches, but, as mentioned previously, with significantly better simulation-to-real transfer and generalizability and 2) enabling more intelligent parameter tuning of classical systems, rather than laborious manual tuning, for example, through automatic tuning and even dynamic parameter policies learned from teleoperated demonstration, corrective interventions, evaluative feedback, and reinforcement learning.

### ***Getting "Unstuck"***

Although most of the failed trials during the physical finals were due to collisions with obstacles, there were also many trials that did not succeed because the robot got stuck in some densely populated obstacle areas. In those places, the robot kept repeating the same behaviors multiple times, e.g., detecting imminent collisions with obstacles, rotating in place, backing up, resuming navigation, detecting the same imminent collision again, and so on. Such behavior sometimes led to a collision with an obstacle and sometimes got the robot stuck forever, although it might also have succeeded on rare occasions. All three teams experienced such behaviors, with the UT Austin and UVA teams being able to fix them by tuning parameters and the Temple team changing the threshold between DRL-VO and the recovery planner.

Similarly, in real-world autonomous robot navigation, how to get "unstuck" safely remains a common and challenging problem. No matter how intelligent an autonomous mobile robot is, it may still make mistakes in the real world, e.g., when facing scenarios out of the training distribution, corner cases not considered by the system developer, and situations where the current parameter set is not appropriate. It is very likely that the robot will repeat the same mistake over and over,

e.g., getting stuck at the same place, which needs to be avoided. Future research should investigate ways to identify such "stuck" situations, balance the tradeoff between exploitation and exploration (i.e., when to keep trying the previous way versus when to try out new ways to get unstuck), utilize previous successful exploratory experiences in future similar scenarios to not get stuck again, and leverage offline computation to correct such failure cases in the future.

### ***Tradeoff Between Safety and Speed***

While the BARN Challenge was originally designed to test existing navigation systems' speed of maneuvering through highly constrained obstacle environments, given the safety constraint of being collision-free, it ended up being a competition about safety alone. The UT Austin team won the competition simply by safely navigating eight out of nine physical trials, not by doing so with the fastest speed. All the teams except the UT Austin team, after it figured out an effective set of parameters for each physical obstacle course, struggled with simply reaching the goal without any collision. The challenge organizers also deployed the widely used DWA planner in the ROS `move_base` navigation stack in the physical obstacle courses, only to find out that, despite being relatively safe compared to the participating teams' methods, it struggled with many narrow spaces and got stuck in those places very often. Such a fact shows that the current autonomous mobile robot navigation capability still lags further behind than one may expect.

### ***Latency Compensation for High Speed***

Only the UT Austin team attempted to pursue higher-speed navigation ( $>0.5$  m/s), doing so after an appropriate parameter set was found for the particular physical course and three successful "safety trials" had been achieved. However, most "speed trials" ended in collision. One contributing factor to such failure was improper



latency compensation for various high speeds. The UT Austin team was the only team that explicitly considered latency compensation in its AMRL stack, through a latency parameter. During high-speed maneuvers, the robot inevitably needs to aggressively change its navigation speed to swerve around obstacles and accelerate in open spaces. System latency caused by sensing, processing, computation, communication, and actuation will likely invalidate previously feasible plans. While simply tuning the latency parameter value can help to certain extent, a more intelligent and adaptive way to calculate and compensate system latency is necessary for the robot to take full advantage of its computing power before executing aggressive maneuvers.

### **Navigation Is More Than Planning**

To plan agile navigation maneuvers through highly constrained obstacle environments, the robot first needs to accurately perceive its configuration with respect to the obstacles. Inaccurate localization and odometry during fast maneuvers with significant angular velocity usually produce significant drift, causing previously valid plans to become infeasible. While all three teams' local planners rely on raw perception to minimize such an adverse effect, e.g., by using high-frequency laser scans and directly planning with respect to these raw features, their global planner usually depends on the results of localization and odometry techniques. For example, the Temple team used Dijkstra's global planner in `move_base`. An erroneous localization will cause an erroneous global plan, which, in turn, will affect the quality of the local plan.

Such an adverse effect will diminish when the navigation speed is low because localization techniques may recover from drift over time. During high-speed navigation, however, the planner needs to quickly plan actions regardless of whether the drift has been fixed or not. As mentioned previously, latency will start to play a role, as well, because a good latency compensation

technique will depend on an accurate localization and odometry model of the robot, i.e., being able to predict where the robot will be based on where the robot is and what action will be executed. Techniques for better odometry, localization, and kinodynamic models during high-speed navigation will be necessary to allow mobile robots to move both fast and accurately at the same time.

### **Conclusions**

The results of the BARN Challenge at ICRA 2022 suggest that, contrary to the perception of many in the field, autonomous metric ground robot navigation cannot yet be considered a solved problem. Indeed, even the competition organizers initially assumed that obstacle avoidance alone was too simple a goal and therefore emphasized navigation speed before the physical competition. However, each of the finalist teams experienced difficulty performing collision-free navigation, and this ultimately led the organizers to modify the competition rules to focus more on collision avoidance. This result suggests that state-of-the-art navigation systems still suffer from suboptimal performance potentially due to many aspects of the full navigation system (as discussed in the "Discussions" section). Therefore, while it is worthwhile to extend navigation research in directions orthogonal to metric navigation (e.g., purely vision-based, off-road, and social navigation), the community should not overlook the problems that still remain in this space, especially when robots are expected to be extensively and reliably deployed in the real world.

**Xuesu Xiao**, George Mason University and Everyday Robots, Fairfax, VA 22030 USA. E-mail: xiao@gmu.edu.

**Zifan Xu**, The University of Texas at Austin, Austin, TX 78712 USA. E-mail: zfxu@utexas.edu.

**Zizhao Wang**, The University of Texas at Austin, Austin, TX 78712 USA. E-mail: zizhao.wang@utexas.edu.

**Yunlong Song**, University of Zurich, Zurich 8050, Switzerland. E-mail: song@ifi.uzh.ch.

**Garrett Warnell**, Army Research Laboratory and The University of Texas at Austin, Austin, TX 78712 USA. E-mail: garrett.a.warnell.civ@mail.mil.

**Peter Stone**, The University of Texas at Austin and Sony AI, Austin, TX 78712 USA. E-mail: pstone@cs.utexas.edu.

**Tingnan Zhang**, Robotics@Google, Mountain View, CA 94043, USA. E-mail: tingnan@google.com.

**Shravan Ravi**, The University of Texas at Austin, Austin, TX 78712 USA. E-mail: shravan.ravi@utexas.edu.

**Gary Wang**, The University of Texas at Austin, Austin, TX 78712 USA. E-mail: gary\_wang@utexas.edu.

**Haresh Karnan**, The University of Texas at Austin, Austin, TX 78712 USA. E-mail: haresh.miriyala@utexas.edu.

**Joydeep Biswas**, The University of Texas at Austin, Austin, TX, 78712 USA. E-mail: joydeepb@cs.utexas.edu.

**Nicholas Mohammad**, University of Virginia, Charlottesville, VA 22903 USA. E-mail: nm9ur@virginia.edu.

**Lauren Bramblett**, University of Virginia, Charlottesville, VA 22903 USA. E-mail: qbr5kx@virginia.edu.

**Rahul Peddi**, University of Virginia, Charlottesville, VA 22903 USA. E-mail: rp3cy@virginia.edu.

**Nicola Bezzo**, University of Virginia, Charlottesville, VA 22903 USA. E-mail: nbezzo@virginia.edu.

**Zhanteng Xie**, Temple University, Philadelphia, PA 19122, USA. E-mail: zhan teng.xie@temple.edu.

**Philip Dames**, Temple University, Philadelphia, PA 19122, USA. E-mail: pdames@temple.edu. 