Autonomous Exploration Development Environment and the Planning Algorithms

Chao Cao, Hongbiao Zhu, Fan Yang, Yukun Xia, Howie Choset, Jean Oh, and Ji Zhang

Abstract—Autonomous Exploration Development Environment is an open-source repository released to facilitate development of high-level planning algorithms and integration of complete autonomous navigation systems. The repository contains representative simulation environment models, fundamental navigation modules, e.g., local planner, terrain traversability analysis, waypoint following, and visualization tools. Together with two of our high-level planner releases – TARE planner for exploration and FAR planner for route planning, we detail usage of the three open-source repositories and share experiences in integration of autonomous navigation systems. We use DARPA Subterranean Challenge as a use case where the repositories together form the main navigation system of the CMU-OSU Team. In the end, we discuss a few potential use cases in extended applications.

I. INTRODUCTION

Our Autonomous Exploration Development Environment, exploration planner, namely TARE planner [1], and route planner, namely FAR planner [2], form a full stack of geometry-based algorithms for navigation planning. Combining the three repositories, we provide a generic set of 3D Lidar-based navigation algorithms for ground vehicles. The software stack is beneficial to the research society as a platform for developing and deploying advanced navigation systems, and further meant to support state-of-the-art research in vision-based navigation.

The development environment is made to facilitate development of high-level planning algorithms. We provide a set of environment models for users to start algorithm development conveniently. The development environment is further compatible with photorealistic house models from Matterport3D [3]. We provide a set of fundamental navigation modules, e.g., local planner for collision avoidance, terrain traversability analysis, waypoint following. In a standard three-tier autonomous navigation system, the development environment functions as the mid-layer, interfacing with the state estimation and motion control on the bottom layer, and the high-level planner on the top layer. For providing state estimation to the system, we made available a list of compatible open-source state estimation methods.

Our navigation algorithms are created with favoring realistic system deployment kept in mind. We do not rely on any unrealistic information which can be made available only in simulation, e.g. semantic and terrain segmentation ground-truth. In particular, our terrain traversability is computed by a dedicated module that processes range data as in a real system. The basic concept is allowing users to develop high-level planning algorithms and conveniently migrate the repositories to the vehicle computer for deployment.

The repositories of the development environment, TARE planner, and FAR planner together form the main navigation system that the CMU-OSU Team uses to attend the DARPA Subterranean Challenge. In the final competition, the team received a “Most Sectors Explored Award” by conducting the most complete exploration (26 out of 28 sectors) among all teams. This paper uses the system as an example to detail usage of the repositories in autonomous exploration and navigation. We further discuss potential use cases in extended applications in simulated and real systems.

II. RELATED WORK

Autonomous navigation systems have been studied from multiple angles. The work described in this paper is based on key results in datasets, simulation environments, and navigation systems briefly discussed in this section.

Datasets: For outdoor settings, the most well-known dataset is KITTI Vision Benchmark Dataset [4] collected from a self-driving suite in road driving scenarios. The dataset contains sensor data from stereo cameras, 3D Lidar, and GPS/INS ground truth for benchmarking depth reconstruction, odometry estimation, object segmentation, etc. For localization purposes, the long-term localization benchmark [5], [6] includes a comprehensive list of datasets such as Aachen Day-Night dataset, Extended CMU Seasons dataset, RobotCar Seasons dataset, and SILDa Weather and Time of Day dataset. For indoor scenes, datasets such as NYU-Depth dataset [7], TUM SLAM dataset [8], inLoc dataset [9], MIT Stata center dataset [10], and KTH-INDOL dataset [11], [12] are available. Datasets are useful in developing and
benchmarking perception and planning algorithms providing users real sensor readings especially for those who don’t have access to integrated sensor suites.

Simulation environments: Carla [13] and AirSim [14] are two representative simulation environments for autonomous driving and flying. These simulators support various conditions such as lighting and weather changes, moving objects such as pedestrians, and incident scenes. For indoor navigation, iGibson [15], [16], Sapien [17], AI2Thor [18], Virtual Home [19], ThreeDWorld [20], MINOS [21], House3D [22] and CHALET [23] use synthetic scenes, while reconstructed scenes are also available in iGibson, AI Habitat [24], [25] and MINOS [21]. Compared to datasets, simulation environments have the advantage of providing access to ground truth data, e.g. vehicle pose and semantic segmentation to simplify the algorithm development and allowing full navigation system tests in closed control loop.

Navigation systems: While countless algorithms have been developed for autonomous navigation, we only list the system-level efforts providing a comprehensive set of open-source perception and planning modules to be integrated as a navigation system. ROS (Robot Operating System) Navigation Stack [26] is an entry-level mobile robot navigation solution. The stack consists of packages including state estimation and planning modules to conduct navigation in 2D known/unknown environments. The open-source repositories from ETH Autonomous System Lab [27] and HKUST Aerial Robotics Group [28] provide packages ranging from sensor drivers to state estimation [29] [30], planning [31], [32], and exploration [33], [34] methods. Users can use these packages to integrate aerial navigation systems.

Our navigation system: Our system follows the three-tier architecture. Our development environment servers as the mid-layer in the system providing generic interfaces to state estimation and motion control modules as well as high-level planning algorithms. The development environment provides a set of fundamental navigation modules such as local planner for collision avoidance, terrain traversability analysis, and waypoint following, and supports common differential-drive platforms. The purpose is using the development environment to facilitate users to develop high-level planning algorithms and then port the entire repositories to the vehicle computer for deployment. Our environment models are both indoor and outdoor containing complex topology to stress exploration and navigation algorithms. In addition to our environment models, we support the photorealistic house models from Matterport3D [3]. Users are provided with scan data and RGB, depth, and semantic images rendered by AI Habitat [24], [25]. Both are widely used by the robotics and computation vision societies. We aim at making our system useful to support research in those societies.

III. DEVELOPMENT ENVIRONMENT

The development environment functions as a platform for developing and benchmarking high-level planning algorithms for ground vehicle navigation. In the development environment, we provide five environment models, fundamental navigation modules, and visualization and debugging tools.

![Environment Models](image)

**Fig. 2.** Five environment models. The characteristics of the environment models are listed in Table I.

A. Environment Models

The environment models resemble real-world settings where robotic systems are commonly deployed. Each of the environment models is distinctive with unique features and challenges. Fig. 2 gives an overview of the environment models and Table I summarizes their characteristics.

- **Campus** (340m × 340m): A large-scale environment as part of the Carnegie Mellon University campus, containing undulating terrains and convoluted layout.
- **Indoor** (130m × 100m): Consists of long and narrow corridors connected with lobby areas. Obstacles such as tables and columns are present.
- **Garage** (140m × 130m, 5 floors): An environment with multiple floors and sloping terrains to test autonomous navigation in a 3D environment.
- **Tunnel** (330m × 250m): A large-scale environment containing tunnels that form a network, provided by the Autonomous Robots Lab at University of Nevada, Reno.
- **Forest** (150m × 150m): Containing mostly trees and a couple of houses in a cluttered setting.

Our system also supports the photorealistic house models from Matterport3D [3]. Users are provided with scan data and RGB, depth, and semantic images rendered by AI Habitat [24], [25]. Users have the option of running AI Habitat side by side with our system or in post-processing. Detailed instructions on configuring our system to use Matterport3D house models and AI Habitat is available on our website.

| Environment Model Characteristics | Large Scale | Convoluted | Multi Terrain | Undulating Structure | Cluttered Obstacles | Thin Structure |
|----------------------------------|-------------|------------|---------------|----------------------|---------------------|---------------|
| Campus                           | X           | X          |               |                      |                     |               |
| Indoor                           | X           |            |               |                      |                     |               |
| Garage                           |             | X          |               |                      |                     |               |
| Tunnel                           |             | X          |               |                      |                     | X             |
| Forest                           |             |            |               |                      |                     |               |

TABLE I

| Environment Model | Characteristics |
|-------------------|-----------------|
| Campus            |                 |
| Indoor            |                 |
| Garage            |                 |
| Tunnel            |                 |
| Forest            |                 |
Fig. 3. (a) A Matterport3D house model and (b) corresponding RGB, depth, and semantic images rendered by AI Habitat.

B. Local Planner

The local planner [35] warrants safety in reaching waypoints that are sent by high-level planners. It computes and follows collision-free paths that lead to the waypoint. The module pre-computes a motion primitive library and associates the motion primitives to 3D locations in the vicinity of the vehicle. The motion primitives are modeled as Monte Carlo samples and organized in groups. In real-time, when a location is occupied by obstacles, the module can determine motion primitives that are collided with the obstacle within milliseconds. The module then selects the group of motion primitives with the maximum likelihood toward the waypoint. In Fig. 4 the red paths represent the collision-free motion primitives. For ground vehicles, the traversability of the vehicle is determined by the terrain characteristics. The local planner takes in the terrain map from the terrain analysis module, detailed in the next section. The module also has interface to take in additional range data for collision avoidance as an extension option.

C. Terrain Traversability Analysis

The terrain analysis module examines the traversability of the local terrain surrounding the vehicle. The module builds a cost map where each point on the map is associated with a traversal cost. The cost is determined by the local smoothness of the terrain. We use a voxel grid to represent the environment and analyze the distributions of data points in adjacent voxels to estimate the ground height. The points are associated with higher traversal costs if they are further apart from the ground. Fig. 5 gives an example terrain map covering a 40m x 40m area with the vehicle in the center. The green points are traversable and the red points are non-traversable. In addition, the terrain analysis module can handle negative obstacles that often result in empty areas with no data points on the terrain map. When negative obstacle handling is turned on, the module treats those areas as non-traversable.

D. Visualization and Debugging Tools

To facilitate algorithm development, we provide a set of tools to visualize the algorithm performance. The visualization tools display the overall map, explored areas, and vehicle trajectory. Metrics such as explored volume, traveling distance, and algorithm runtime are plotted and logged to files. Further, the system supports using a joystick controller to interfere with the navigation, switching among multiple operation modes to ease the process of system debugging. Detailed information is available on the project website.

IV. HIGH-LEVEL PLANNERS

We discuss two high-level planners, TARE planner for exploration and FAR planner for route planning. The high-level planners take in the state estimation output and generate task-specific waypoints, which are executed by the local planner in closed-control-loop navigation. Typically, the high-level planners re-plan at a lower frequency and provide long-distance routes, while the low-level navigation modules react instantaneously to follow the route and avoid obstacles.

A. TARE Planner for Exploration

TARE planner is a hierarchical framework that utilizes a two-layered representation of the environment to plan the exploration path in a multi-resolution manner. As illustrated in Fig. 6, the planner uses low-resolution information to plan a coarse path at the global level. In the local area surrounding the vehicle, the planner plans a detailed path using high-resolution information. The method optimizes the overall path by solving a traveling salesman problem at each level. Compared to existing methods relying on greedy strategies, the planner can effectively adapt to the structural environment and produce an approximately optimal exploration path that avoids redundant revisiting.

TARE planner is evaluated in several large and complex environments in both simulation and real world. It is compared to state-of-the-art methods, namely, NBVP [33], GBP [36], and MBP [37]. Results indicate that TARE planner
produces significantly higher efficiency in exploration and computation. Fig. 7 shows a representative result where the ground vehicle in Fig. 4(a) is used to explore a four-storage garage and a connected patio. The vehicle starts from the entrance of the garage on the top floor and explores the whole environment before reporting completion after 1839m of travel in 1907s. Fig. 7 shows the resulting map and vehicle trajectory. Due to space limitations, we omit the experiment details. Please refer to our paper [1] and website for detailed evaluations of the exploration performance.

B. FAR Planner for Route Planning

FAR planner is a visibility graph-based planner that dynamically builds and maintains a reduced visibility graph along with the navigation. The planner can handle both known and unknown environments. In a known environment, it uses a prior map to plan the route. In an unknown environment, however, it attempts multiple ways to guide the vehicle to the goal and picks up the environment layout during the navigation. FAR planner models obstacles in the environment as polygons. It extracts edge points around the obstacles and converts the edge points into a set of polygons. The polygons are then merged over sensor data frames, and from which, the visibility graph is developed.

FAR planner is evaluated in large and convoluted environments. It is compared to RRT* [38], RRT-connect [39], A* [40], and D* Lite [41] planners and demonstrates its strength in fast re-planning over long distances. The planner uses ~15% of a single CPU thread on an i7 computer for expanding the visibility graph and a second CPU thread for path search. A path is found within 0.3ms in all of our experiments. Fig. 8 shows a representative result using a Matterport3D house model as the environment. In Fig. 8(a), the planner does not use any prior information of the environment and treats the environment as unknown. The planner attempts to guide the vehicle to the goal by dynamically registering obstacles in the environment (red polygons) and building a visibility graph (cyan lines) during the navigation. In Fig. 8(a), it re-plans after discovering a better route. In (b), the planner uses the visibility graph from a prior map to plan the route.

V. Best Practices

Safety margin: The local planner uses a planning horizon at the distance between the vehicle and waypoint. This ensures that the vehicle can stop at waypoints relatively close to obstacles - collision check does not consider obstacles further than the waypoint. However, if the vehicle is not expected to stop at the waypoint, it is preferable for the high-level planner to keep the waypoint a distance away (≥ 3.75m as the default planning horizon) from the vehicle. If the waypoint is closer, users can project the waypoint further away and keep the waypoint in the same direction w.r.t. the vehicle to fully use the safety margin. On the other hand, if the vehicle needs to navigate through narrow openings, reducing the distance helps the local planner find collision-free motion primitives through the openings.

Sharp turns: Typically, a high-level planner selects the waypoint along the path that is a distance, namely look-ahead distance, ahead of the vehicle and sends the waypoint to the local planner (possibly after projecting the waypoint further
away from the vehicle as discussed above). When handling sharp turns ($\geq 90$ deg), the look-ahead distance needs to be properly set or the waypoint may jump to the back of the vehicle, causing the vehicle to osculate back-and-forth. We recommend to select the waypoint on the starting segment of the path that is in line-of-sight from the vehicle.

Dynamic obstacles: The terrain analysis module eliminates dynamic obstacles from the terrain map by ray-tracing after the dynamic obstacles move away. This is implemented in the vicinity of the vehicle ($\leq 5m$ to the vehicle) due to the fact that range data becomes sparse further away and the trade-off between eliminating dynamic obstacles and thin structures is hard to set. Depending on the use case, users are advised to perform task-specific dynamical obstacle handling. Both TARE and FAR planners contain such a step.

VI. SYSTEM INTEGRATION

A. Basic System

Our basic system uses a wheelchair-based platform shown in Fig. 9(a). The vehicle is equipped with a Velodyne Puck Lidar, a camera at $640 \times 360$ resolution, and a MEMS-based IMU. A 4.1GHz i7 computer handles processing onboard. Fig. 10 shows the basic system diagram. The development environment functions as the mid-layer where it takes in the output from state estimation module and sends command velocity to the motion control module on the bottom layer. It interfaces with the high-level planner (TARE or FAR planner) at the top layer by providing terrain map and taking in waypoint, navigation boundary, and speed. Such a system setup allows users to develop high-level planning algorithms without the necessity of understanding the interfaces among low-level navigation modules.

Our system is compatible with several open-source 3D Lidar-based odometry/SLAM methods as candidates of the state estimation module, including LOAM [42], A-LOAM [43], LeGO-LOAM [44], LIO-SAM [45] and LIO-mapping [46]. Instructions on setting up these methods can be found on our website. Specifically, our system requires the state estimation module to output scan data registered in the world frame. Using registered scans has a few advantages, i.e., it makes processing less sensitive to the time synchronization between scan data and state estimation. Further, multiple registered scans can be stacked together to extract rich geometric information from the environment. However, if scan data associated with the sensor frame is needed, our system also provides it with the corresponding synchronized state estimation. Our system supports generic differential-drive platforms (including skid-steer mechanisms), which is the most commonly used mobile robot kinematic model. Omnidirectional and legged platforms can still use our system but their mobility is not fully exploited, i.e., they move as a differential-drive platform without lateral movement. Our system currently does not support car-like platforms.

B. DARPA Subterranean Challenge System

DARPA Subterranean Challenge highlights autonomous navigation and exploration in underground, GPS-denied environments. The challenge involves three types of environments: tunnel systems, urban underground, and cave networks. Teams deploy a fleet of autonomous vehicles to search for artifacts (backpack, cellphone, etc) and report their locations. The challenge allows a human operator to command the vehicles from the entrance of the competition site over intermittent wireless networks.

The ground vehicles used by the CMU-OSU Team are 4-wheel-drive and skid-steer platforms as shown in Fig. 9(b). The navigation system is an extended version of our basic system. The system diagram is shown in Fig. 11. The state estimation module can detect and introduce loop closures. The module outputs state estimation in the odometry frame containing accumulated drift. When loop closure is triggered, it outputs loop closure adjustments to globally relax the vehicle trajectory and corresponding maps. Loop closure adjustments are used by the high-level planners since they are in charge of planning at the global scale. Modules such as local planner and terrain analysis only care about the local environment surrounding the vehicle and work in the odometry frame.
The local planner and terrain analysis modules are extended to handle complex terrains including negative obstacles such as cliffs, pits, and water puddles with a downward-looking depth sensor. The TARE planner, FAR planner, and other planners (for stuck recovery, etc) are run in parallel for tasks such as exploration, go-to waypoint, and return home. On top of these planners, behavior executive and multi-robot coordination modules are built specifically for the challenge. The modules share explored and unexplored areas across multi-robots and call TARE and FAR planners cooperatively. In particular, when a long-distance transition is determined due to new areas containing artifacts are discovered or operator waypoints are received, the behavior executive switches to FAR planner for relocating the vehicle. During the relocation, FAR planner uses a sparse roadmap merged from multi-robots for high-level guidance. After the relocation is complete, TARE planner takes place for exploration.

Fig. 12 shows a representative exploration result from a competition run in Urban Circuit held at Satsop Nuclear Plant, WA. Our vehicle uses TARE planner to explore the environment fully autonomously, traveling over 886m in 1458s. Fig. 13 shows the result from the final competition in Louisville Mega Cavern, KY. Three vehicles with red, green, and blue trajectories are deployed running TARE and FAR planners together. The segments between A and B on the red trajectory and between the start point (blue dot) and C on the blue trajectory use FAR planner to transit. Other segments on the trajectories use TARE planner to explore.

Self-supervised learning for visual navigation: Learning-based navigation in the context of exploring a new environment, traversing to a goal point, or searching for a specific object can utilize our system to collect training data. For example, users can set up our Lidar-based system as a way of self-supervision for training a vision-based deep network, and then, during test time, the method uses only a camera to navigate. Since we support photorealistic environment models such as Matterport3D, our simulation environment is designed to facilitate the simulation to real adaptation.

Mixed-initiative navigation with human instructions: This can benefit from the Matterport3D house models which have semantic ground truth provided. By interpolating natural language, e.g., “go through the door in the front and then turn left,” users can use our system together with the semantic ground truth to locate the door, navigate the vehicle through the door, and then turn left. The algorithm being developed does not have access to the semantic ground truth but is trained by the reference trajectories from our system.

Social navigation in crowded environments: We use Gazebo simulator which makes modifications to the simulator configuration convenient. Users can insert custom human avatars into the simulation environment to help develop social navigation algorithms. Once the human avatars are inserted, users have access to the human pose ground truth, which can potentially simplify the algorithm development by skipping the human pose detection module in simulation.

VIII. Discussion and Future Activities

The purpose of this work is to provide a generic navigation system to the society to support a variety of multi-disciplinary research. To this end, we choose to work with ground vehicles instead of aerial vehicles due to their capacity to carry heavy sensor-computer payloads and extended battery life, since contemporary AI research often involves sophisticated sensors and computers with powerful GPUs. In the future, we plan to extend our system to multi-robot coordination and exploration. We also plan to organize tutorials on future ICRA and CVPR conferences to better prepare users to utilize our system in facilitating their work.
[43] T. Qin and S. Cao, “Advanced implementation of LOAM,” HKUST Aerial Robotics Group. [Online]. Available: https://github.com/HKUST-Aerial-Robotics/A-LOAM

[44] T. Shan and B. Englot, “LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, October 2018. [Online]. Available: https://github.com/RobustFieldAutonomyLab/LeGO-LOAM

[45] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, “LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, Nevada, October 2020. [Online]. Available: https://github.com/TixiaoShan/LIO-SAM

[46] H. Ye, Y. Chen, and W. Liu, “Tightly coupled 3D lidar inertial odometry and mapping,” in International Conference on Robotics and Automation (ICRA), Montreal, Canada, May 2019. [Online]. Available: https://github.com/hyye/lio-mapping