Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

Demonstrating Arena-Web: A Web-based Development and Benchmarking Platform for Autonomous Navigation Approaches

Linh Kästner[1,*], Reyk Carstens[1,*], Lena Nahrwold[1], Christopher Liebig[1], Volodymyr Shcherbyna[1], Subhin Lee[1], and Jens Lambrecht[1]

***Abstract*—In recent years, mobile robot navigation approaches have become increasingly important due to various application areas ranging from healthcare to warehouse logistics. In particular, Deep Reinforcement Learning approaches have gained popularity for robot navigation but are not easily accessible to non-experts and complex to develop. In recent years, efforts have been made to make these sophisticated approaches accessible to a wider audience. In this paper, we present Arena-Web, a web-based development and evaluation suite for developing, training, and testing DRL-based navigation planners for various robotic platforms and scenarios. The interface is designed to be intuitive and engaging to appeal to non-experts and make the technology accessible to a wider audience. With Arena-Web and its interface, training and developing Deep Reinforcement Learning agents is simplified and made easy without a single line of code. The web-app is free to use and openly available at https://app.arena-rosnav.com.**

I. Introduction

With recent advances in Deep Reinforcement Learning (DRL) for navigation and motion planning, several research works utilized DRL inside their approach [1], [2]. However, developing DRL agents comes along with a number of barriers and difficulties such as difficult training, complex setup, or expensive hardware thus, limited to experts and practitioners in the field. This, makes comparability difficult [3]. However, benchmarking and testing DRL approaches is crucial to compare them against classic approaches and assess their performance and feasibility. Aspirations and efforts to standardize the process of benchmarking were also made in recent years [4], [5], [6]. However, the application consists of multiple packages and repositories, which may not be easily accessible and understandable for laypersons. In addition, they also require an installation and setup process or specified hardware, which increases the barriers for laypeople or new practitioners. On the other hand, web-based applications have proved to facilitate a more intuitive experience and are more openly available. There have been several efforts to bring robotic applications and assist the user via web apps [7], [8], [9].

A web-app is easily accessible from all operating systems and does not require specific hardware in order to initiate complex tasks. Furthermore, there is no need to go through complicated installation procedures and face potential errors and issues, which may hamper the motivation to start in this field. Hence, an application that fosters easily accessible development would not only benefit laypeople aspiring to to work with DRL and motion planning approaches but also for expert scientists who are looking for a unified, uncomplicated platform to benchmark their approaches and save time and costs. On that account, we propose Arena-Web, a web-based application, which provides the possibility to develop DRL agents and benchmark these navigation approaches against classic ones using a variety of tools. Arena-web is acting as a wrapper around our training and testing platforms Arena-Rosnav [10] and Arena-Bench [5] (in the following denoted as Arena(to easily utilize and customize trainings runs using the Arena Platforms. In our previous works describing Arena, we have demonstrated the transferability of agents trained with the Arena platform towards real robots and conducted extensive tests to also

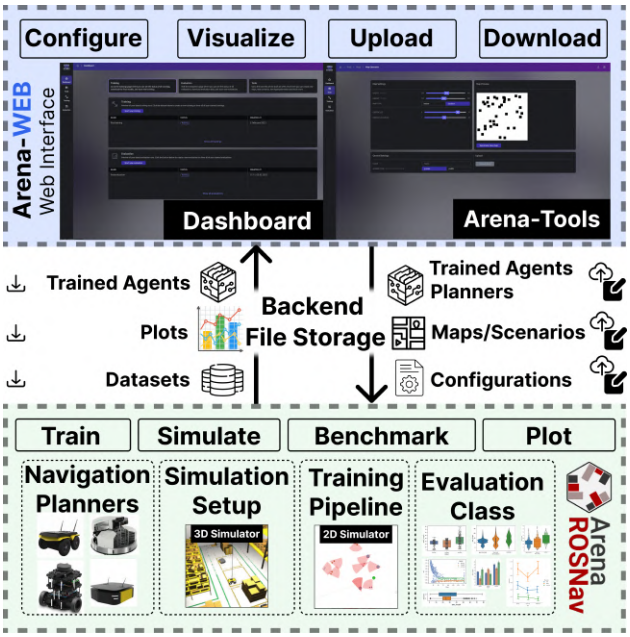

Fig. 1: Arena-Web provides a web-based interface to develop, train, and test navigation approaches conveniently on any computer. It provides interfaces to generate a number of utilities such as maps, scnearios, training parameters, and tasks in order to define training and testing runs. The UI is designed in an intuitive and appealing way and the user is guided through a number of predefined steps without sacrificing the high degree of custumization possibilities. The web-app is build on top of our previous work Arena-Rosnav which is run in the backend and contains all the functionalities to run the simulation and generate the data.

[1]Linh Kästner, Reyk Carstens, Christopher Liebig, Volodymyr Shcherbyna, Lena Nahrworld, Subhin Lee, and Jens Lambrecht are with the Chair Industry Grade Networks and Clouds, Faculty of Electrical Engineering, and Computer Science, Berlin Institute of Technology, Berlin, Germany linhdoan@tu-berlin.de

[*]Contributed equally

assess the sim2real gap. Thus, all agents trained using the web app can also be deployed towards real robots. The user has a high number of possibilities to set and generate different maps, worlds, and scenarios or task modes. All aspects along the pipeline are individually customizable, editable and can be specified by the user according to his or her needs. The user can customize task modes that the agent should be tested on, new worlds and scenarios, neural network architectures, etc. In particular the web-app is able to do the following: train DRL agents on custom maps, with custom reward values, and custom neural network architectures. Evaluation and benchmarking of the approaches in comparison to other learning-based and classic navigation planners. Download of the created models and evaluations files for further use, like plotting of the results or running the models on real robots. Furthermore, there are several editors to create scenarios, neural network architectures, set of hyperparameters, maps, and custom rewards. All functions are embedded into an intuitive and appealing web application for improved user experience, understanding, and accessibility. The main contributions of this work are the following:

- To the best of our knowledge, the first web-based DRL development and testing platform that lets the user create, train, and benchmark DRL agents with a variety of classic and learning-based navigation approaches. The web app is built on top of our previous works arena-bench, which is modular and can be extended with other planners and approaches.
- Possibility to customize the whole training pipeline from customized network architectures to hyperparameters, reward functions, and scenarios.
- Possibility to customize the whole evaluation pipeline from customized scenarios, task modes, and planners. A large number of qualitative and quantitative evaluation metrics and plotting functionalities are available.

The paper is structured as follows. Sec. II begins with related works. Subsequently, the methodology is presented in Sec. III. Sec. IV presents exemplary use cases that can be executed with the app. Finally, Sec. V will provides a conclusion and future works.

II. Related Works

DRL-based navigation approaches proved to be a promising alternative that has been successfully applied in various robotic applications with remarkable results. Thus, various research works incorporated DRL into their navigation systems [1], [2], [11], [12], [13], [14], [15], [10], [16].

However, most of these research works require at times tedious installation and setups in order to test and validate the approaches. Oftentimes, depreciated versions require manual adjustments and bug fixing, which consumes cost and time. Furthermore, standardized platforms to benchmark DRL-based approaches with classic ones side-by-side are rare and limited [3], [5]. There exist a number of platforms that aspire to provide a platform for benchmarking and developing these approaches [4], [5], [17], [6]. However, they also require an installation and setup process or specified hardware, which is not intuitive or possible for laypersons or new practitioners. On the other hand, web-based applications have proved to facilitate a more intuitive experience and are more openly available to a larger audience.

There has been a number of efforts to provide services from the robot operating system as web applications. For the past two decades, efforts have been made to combine robotics with web-based applications. In 2000, Schulz et al. [18] proposed web-based interfaces to remotely control robots via the internet. The researchers provided a web-based user interface in which the user could select a map and navigate a robot to a desired point within the map. Similarly, Simmons et al. proposed an extended and improved version with more functionalities and visualizations. Other web-based platforms for robot control and interaction include [7], [8], [19], [9].

Using web interfaces also prove practical for an improved human-robot-collaboration and -interaction since web services are easier to access and more openly available than command line tools or development setups. Thus, more recent robotic web-based applications range from robot assistance in healthcare [20], [21], [22], manufacturing [23], guiding robots [24], [25] or educational purposes [26], [27], [28], [29].

Furthermore, with the advent of the robot operating system (ROS) [30], tools have been proposed to combine ROS with web. Open-source libraries such as ros.js proposed by Osentoski et al. [31] or the Robot Web Tools proposed by Toris et al. [32], which provides major ROS functionalities via rosbridge as a web app, contribute to foster web-based robotic development and usage to broaden the development audience. More recently, tools such as Webviz [33] or Foxglove [34] aspire to shift ROS based visualization and monitoring entirely to the web.

On that account, and given the limitations of a unified evaluation and benchmarking platform, this paper proposes Arena-Web to facilitate the development of DRL-based approaches and evaluation against other learning-based and classic navigation planners. The platform is made available entirely via the web application to facilitate swift development and intuitive usage thus, accelerating the development cycle. The tool aspires to open DRL development and benchmarking of robot navigation approaches to a wider audience.

III. Functionalities

Arena-Web is the web interface that combines the functionalities of our previous works Arena-Rosnav [10] and Arena-Bench [5], adds several more functionalities to them, and serves as an openly accessible user interface. The main tasks of the application are divided into three categories: Tools, Training, and Evaluation. This section should provide an overview of the current functionalities of Arena-Web. It is to be noted, that more functionalities are constantly being added to the application.

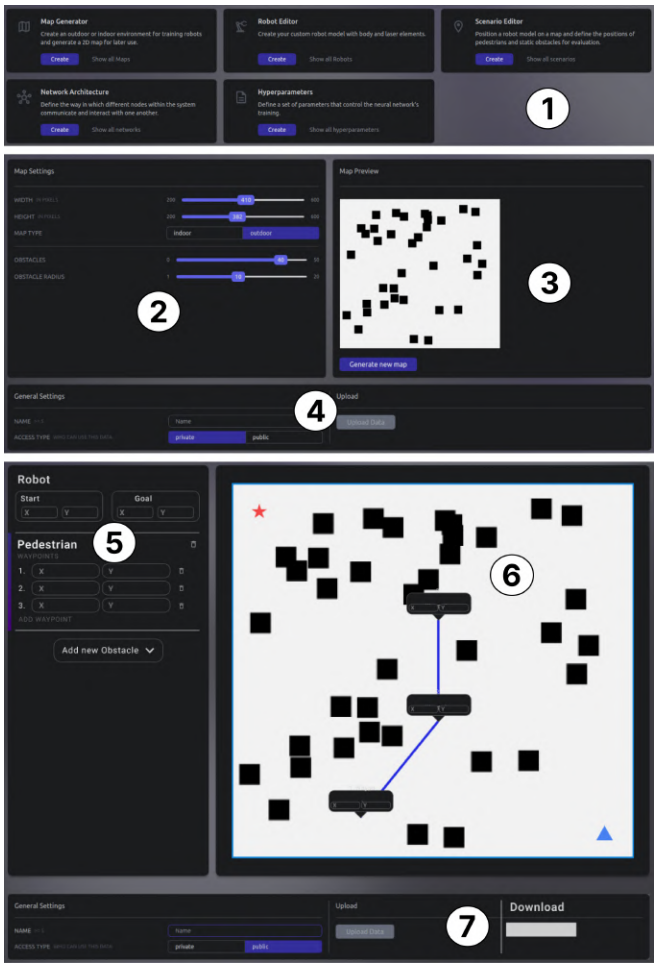

Fig. 2: We provide a number of editors/generators to generate utilities such as an occupancy map, scenarios with predefined obstacle numbers and behavior, training pipeline editors to generate new neural network architectures, select training hyperparameters, reward systems, etc. (1) The dashboard to select the tool, (2) Parameters regarding the map can be adjusted using sliders, (3) the map is previewed live so that the user can directly observe changes, (4) it is also possible to upload existing maps, (5) in the scenario editor the user can input positions of the robot and obstacles by drag-and-drop, the values will be displayed in the sidebar, (6) the scenario is previewed to the user, (7) the user can upload existing scenarios.

A. Tools

We provide tools to generate several important utilities necessary to conduct training or evaluation tasks. An overview of all currently available tools can be seen in Fig. 2.

Map Generator: The map generator is able to generate 2D worlds for indoor and outdoor environments. The user can specify various parameters such as the height and width of the map, the amount and size of static obstacles, the width of passages between room and the size and amount of rooms in indoor maps. The map generator will generate an occupancy grid map.

Scenario Generator: The scenario editor allows the user to select a map and place dynamic obstacles inside, define their velocities, and define their behavior by setting waypoints. The robot's start and endpoint is also to be set. The user can edit all positions by drag-and-drop, which simplifies and improves the user experience. The scenario can later be used in the evaluation phase to offer the same environments for different planners and allow a consistent way of comparison.

Network Generator: Using the network generator, the user can create an individual neural network architecture for the training of DRL agents. The user can specify the in- and output-size of the different layers and is able to add and remove these layers. Currently, fully connected, convolutional, and Relu layers are possible, but our modular configuration makes it easy to add more layer types in near future.

Reward Generator: Since DRL is used to train the agent, rewards have to be defined for specific actions. Adapting the rewards can highly influence the outcome of the training. Therefore, with the reward generator, we offer a way for the user to adapt all rewards to their specific needs.

Hyperparamters Generator: Similar to the rewards, the hyperparameters are an essential and important component of the training process. Since setting up hyperparameters is oftentimes a trial-and-error process, with the hyperparameters generator we give the user the possibility to create their own set and try different values.

B. Training

One major task of our web-app is the ability to train a new DRL model for a robot. Therefore, we offer a preset of 10 robots to choose from. The training can be done on predefined or custom-created network architectures and maps. Furthermore, the user is able to specify a variety of hyperparameters such as the learning rate, the batch size, the evaluation frequency. While the training is running on the servers, the user is always able to directly download the current best model and see the log output to monitor the process. When the training has finished, the user will be notified. Since the training is always dependent on a robot, we offer a large variety of pre-available robots, covering all major robot kinematics. This includes the Turtlebot3, Robotino Festo, Carobot4, AGV Ota, Ridgeback, Kuka Youbot, Clearpath Jackal, Clearpath Husky, Dingo, TiaGo.

C. Evaluation

The evaluation task can be utilized to evaluate the trained agent or other existing planners within the Arena-Rosnav suite. All available planners are listed in Table I. We provide a large number of state-of-the-art learning-based as well as classic planners. By selecting predefined scenarios for the evaluation, the user can make sure to offer the same environment for all of the evaluations and have a consistent

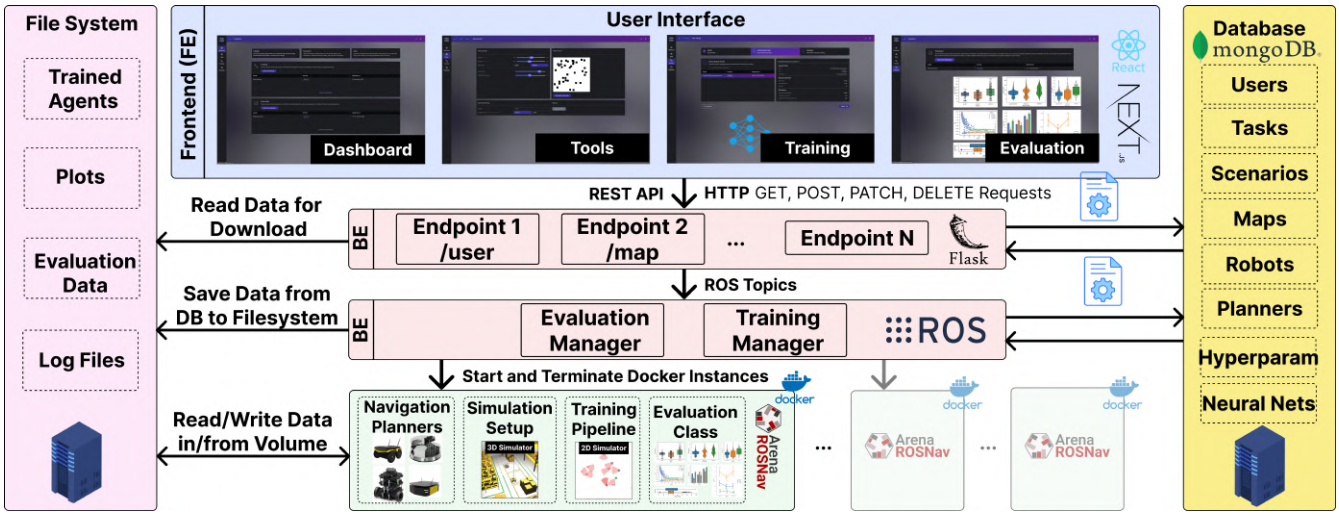

Fig. 3: System Design of Arena-Web. The frontend (FE) provides user interfaces to customize and setup training and evaluation runs and construct necessary utilities such as maps and scenarios. The web-app is build on top of Arena-Rosnav (see Fig. 4 for the system design of Arena-Rosnav), which provides tools to train and benchmark DRL planners against other learning-based or classic approaches and plot the results on different metrics. Therefore, the FE is communicating with the backend (BE) via a REST API and endpoints and an additional ROS layer consisting of two ROS nodes to create and manage the Arena-Rosnav dockers in the BE. The generated files are stored within a MongoDB database as well as a file system for user generated data. The backend and databases are run on our GPU servers.

comparison. At all times, the user is allowed to download the current evaluation data while the status of the evaluation run can be observed using the log files. Once the evaluation run is finished, the user can download the resulting .csv files to plot the results locally with our evaluation package. Users have the possibility to create their own plot declaration files, which defines what plots have to be created and what data to use for them. This allows even inexperienced users to quickly plot their results. Currently, we are working on integrating the plotting directly into the web app for an improved user experience.

D. Available Robots and Planners

There are a variety of already pre-available robots covering all major robot kinematics. These include the Turtlebot3, Robotino Festo, Carobot4, AGV Ota, Ridgeback, Kuka Youbot, Clearpath jackal, Clearpath Husky, Dingo, TiaGo. Furthermore, Table I lists all available planners.

TABLE I: Available Navigation Planners

Classic	Hybrid	Learning-based
TEB [35]	Applr [36]	ROSNavRL [10]
DWA [37]	LfLH [3]	RLCA [38]
MPC [39]	Dragon	Crowdnav [11]
Cohan [40]	TRAIL [41]	SARL [42]
		Arena [43]
		CADRL [2]

IV. Methodology

A. System Design Overview

A schematic overview of the whole system is shown in Fig. 3. The user interacts with the user interface in the frontend, which is implemented in React.js. The frontend can be segregated into three main segments, the tools, the training, and the evaluation. The frontend communicates with the backend over a REST API. The backend is written in flask and offers multiple different endpoints for all functionalities of the web app. For persistent and efficient storage, a MongoDB database is used, which is chosen over a relational database because of its possibility to handle unstructured data. When starting new training and evaluations, Arena-Rosnav is run as a Docker on the server to process the task. The system design of Arena-Rosnav is illustrated in Fig. 4. Since the process should be customizable by the user's created tools and the output and created files should also be available to the user, parts of the server's file system are virtually loaded as volumes into the docker to save and load files from outside of the docker. For an extra layer of security and easier managing of the different docker containers, a small layer of ROS-Nodes is implemented between the flask backend and dockers.

B. Frontend

The Frontend offers user interfaces for starting new training and evaluation runs as well as creating new utility files for use in either training or evaluation with the tools as described in Section III. For security and management reasons, the whole frontend is only available to already registered users, which are logged in. This allows to always relate user-created data and tasks to the specific user and therefore keep the whole interaction private. We have decided to uncouple the direct connection between utility files and specific tasks so that created files can be used in combination with other utilities and for different and

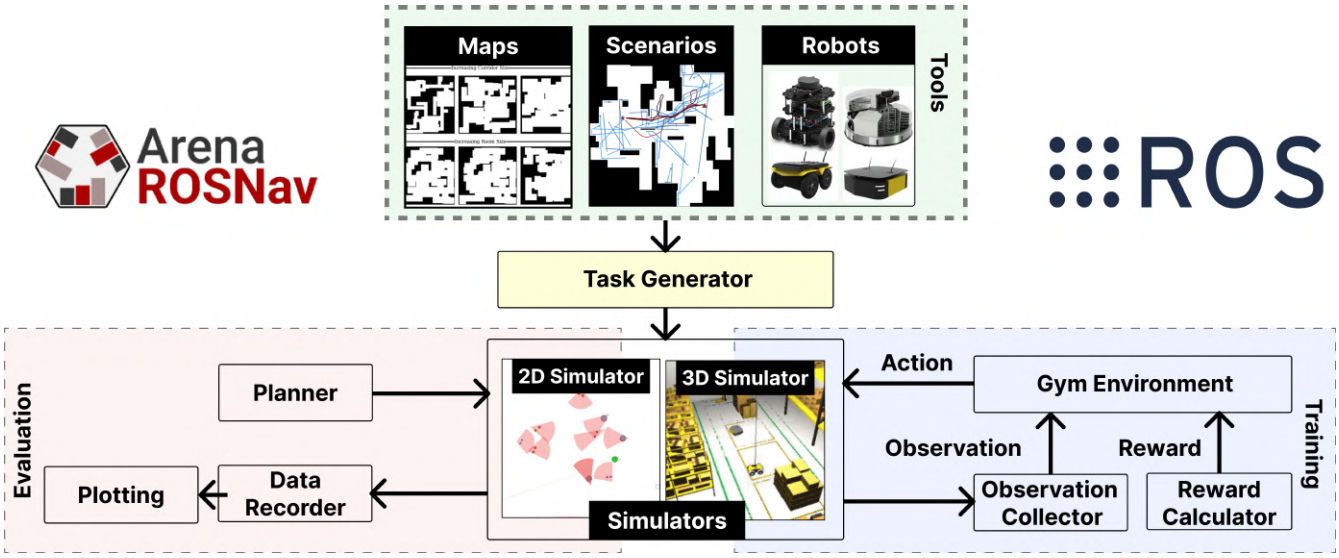

Fig. 4: System design of Arena-Rosnav. It provides tools to develop, train, and evaluate DRL planners on different environments and scenarios, especially in highly dynamic environments. Therefore, it includes multiple simulation environments such as Flatland (2D), Gazebo (3D), or Unity (3D). Note that currently the user can only choose simulation within the 2D environment. We are currently working on enabling 3D environments to be choosen. The training pipeline conducts a DRL training, which can be customized by the user. Subsequently, the trained agent can be benchmarked against other navigation approaches. Arena-Rosnav provides a suite of different state-of-the-art planners listed in Table I and an evaluation pipeline including a data recorder and a plotting class. The data recorder records all necessary data during the evaluation run and the plotting class can utilize these files to plot the results on a number of navigational metrics such as navigational efficiency, saftey, and smoothness.

multiple tasks. This allows for modularity and experiment with even less work.

In the frontend, the different tools are used to create the requisites for customized training or evaluations. As such, a map-, scenario-, network architecture-, hyperparameter-, and reward generator were implemented, which have already been presented in section III. We have decided to make it possible to create all utility files used in training and evaluation separately from specific tasks in order to reuse them for different and multiple runs instead of having to repeatedly creating them again for different training or evaluation runs.

C. Backend

The backend offers a variety of different REST Endpoints to compute every functionality that is possible in the frontend. The backend uses a MongoDB database for persistent and efficient storage of all data created using the tools. All created utility files, which are created with the tools, as well as training and evaluation infos are stored inside of the database for easier manipulation and consistent access from all parts of the backend. Beside the flask backend the server runs a small topology of ros nodes to manage docker containers, in which the training and evaluation tasks are run. Therefore, when starting new tasks the flask backend tells the ros topology to start the defined task with specified configurations included in the request. The Ros Nodes will then create the necessary utility files and store them in the filesystem of the server. Subsequently, a new docker container for the task is started and the utility files are loaded into the docker using docker volumes. To allow the user to download created data, the output directories of files in the docker are also loaded as volumes to be visible in the server's file system and be usable by the backend.

Using the docker volumes not only allows to use of created data later on but also allows the use of the same docker for different tasks and with different utility files. Using this extra layer of ros nodes has multiple advantages. At first, it offers another layer of security because it is not attached to the backend and therefore, failures in or attacks on the flask backend will not affect the docker containers. Furthermore, it allows to be independent by the flask backend and makes possible further integration into other applications way easier. And at last, it leads to a much quicker response to for the user because the endpoint itself only has send a single rosmessage instead storing all files and starting the docker on its own. This way, the computation can be conducted asynchronously.

In comparison to the previous training and evaluation capabilities of Arena-Bench, a number improvements have been made. To allow the modularity with which different neural network architectures, rewards, maps, and scenarios can be used, we defined completely new formats to store the data and integrated efficient parsers to read and use the data in *Arena-Bench*. Additionally all of the data is validated in

the frontend and in the backend to detect possible errors as soon as possible and reduce errors.

V. Exemplary Use Case Workflows

In this chapter, we illustrate how the application can be utilized for different use cases and needs. In particular, there are three main use cases, which are covered using the web-app: creating new utility files like maps, scenarios, rewards, and hyperparamters, training DRL agents, and evaluating existing planners or trained DRL agents, which have been created with the web-app by the user. For each of these use cases, we define an example use case and show the user journey through the web-app and specify the required tools and interactions. The exemplary use cases and respective required tools are illustrated in Fig. 5. The example use cases are as follows:

- Creation of a new scenario
- Starting a new training and downloading the model.
- Starting a new evaluation with a trained DRL agent, downloading the evaluation data and plot the results.

A demo video showcasing all of the functionalities is given in the supplementary materials.

A. Creation of a New Scenario

The scenario serves as a fixed environment in the evaluation for better comparison of different planner approaches. It specifies the start and goal position of a robot as well as different dynamic obstacles like pedestrians. Furthermore, the exact behavior of the obstacles can be specified by waypoints and the velocity of the obstacles can be set. In order to create a new scenario, the user has to select a map, which can be created by the map generator tool.
After logging in, the user is on the Dashboard, which shows a small overview over the latest tasks. From here, the user can directly jump to the tool selection and start creating a new map. After selecting the parameters, the map can be uploaded. Afterwards, the user jumps back to the tool selection and selects to create a new scenario. Here, the user will be asked to select one of the maps the user has access to. Subsequently, the user will be directed to the actual scenario generator page where the map is visualized on the right side. On the left side the user can set the different options in a list and has the possibility to drag and drop the goal of the robot and start position onto the map or directly input X and Y coordinates. Furthermore, the user will find a dropdown menu beneath it from which the type of a new dynamic obstacle, which is then added to the scenario, can be selected. For each obstacle, the velocity can be changed by an input field and new waypoints can be added, which can be changed by dragging them in the map or directly passing in the coordinate values. After finishing the scenario the user is prompted to give the scenario a name and select whether other users are allowed to see and use the created scenario. After hitting upload the process is finished and the scenario can either be downloaded directly or be used for a new evaluation.

B. Starting a New Training Task

The training can be started with a variety of customization possibilities, since the network architecture, hyperparameters, map, and rewards can be selected and specified. Due to the high number of possibilities and combinations, which could make the user experience overwhelming, we designed an interactive and straightforward pipeline consisting of four steps for selecting the different utilities and starting the training. For all steps, we already offer a variety of openly accessible files to use, and have defined a pre-selected default values.

1) ***Selecting the Map and Robot****:* After entering the app and being located at the dashboard, the user can directly jump to the page where a new training is created. There, the user is prompted to select the map to train on. Either the user selects a map from high number of pre-existing maps or from self-generated or uploaded ones. After selecting a map, the user will need to select a robot on which the trained model should run. Based on this decision, the neural network architecture step will consider the observation and action space depending on the robot since all robots have different observation spaces.

2) ***Creating Sets of Hyperparameters****:* As stated beforehand, sets of hyperparameters necessary for training should be configured in the UI. The UI is depicted in Fig. 6).
The first step “Settings”, where the user is able to set each parameter to the desired value (see Fig. 6). As the hyperparameter generation should be more intuitive than simply putting numbers or strings into an input field, buttons and sliders are displayed instead. A small text below each hyperparameter describes the usage of such. The usage of sliders and buttons also avoids parameters being set to values out of bounds. While parameters with buttons as options can only be set to the displayed values, sliders have a range of what the parameters can be set to. Initially before user configuration, all hyperparameters are set to predefined values as displayed in Fig. 6.

After configuring the hyperparameters, choosing their access type is the user’s next step in "General Information" either as public, to enable public usage, or private. Based on this decision, other users might see the created hyperparameters. In this step, the user is also asked to name the created hyperparameters. This is important to avoid confusion on the user side. Finally, the user is given an overview of the chosen settings and access type and can submit hyperparameters.

3) ***Creating Neural Network Architectures****:* For simplicity, the process of creating custom NN architectures is similar to creating a set of hyperparameters. The NN architecture generator is structured in analogous three steps. The first step "Architecture" guides the user through the creation of the network architecture. This part, however, is more complex than in the hyperparameter generator. The NN architecture configured at any time is saved in an array. The UI can

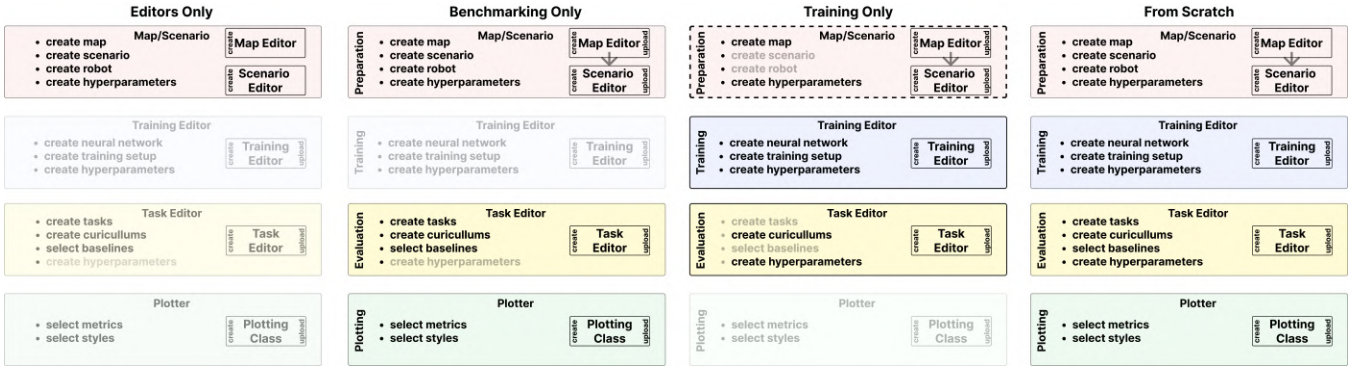

Fig. 5: Exemplary workflows and their necessary modules (blended in for the respective use case). Using Arena-Web can serve different use cases and needs. The user can utilize the variety of editors to create ROS occupancy grid maps or different scenarios using the intuitive interface to simplify the generation process of these files for working with a local instance of Arena-Rosnav. The user can also use the web-app to initiate a DRL training from scratch and benchmark the resulting agent against several planners. Other possibilities include to utilize the app for conducting only a training using existing maps and agents or conducting only evaluation tasks with preexisting agents, planners, and scenarios without accessing the development PC. This could be useful e.g. for teams to conduct multiple evaluation runs without requiring the exact PC setup from anywhere. We are currently working on a mobile version to also facilitate starting or monitoring these tasks from anywhere, which makes the development process as well as collaborations more flexible and efficient.

Fig. 6: Hyperparameter setting for the training tasks. The user can choose from a number parameters such as the task mode, training step size, evaluation step size, etc.

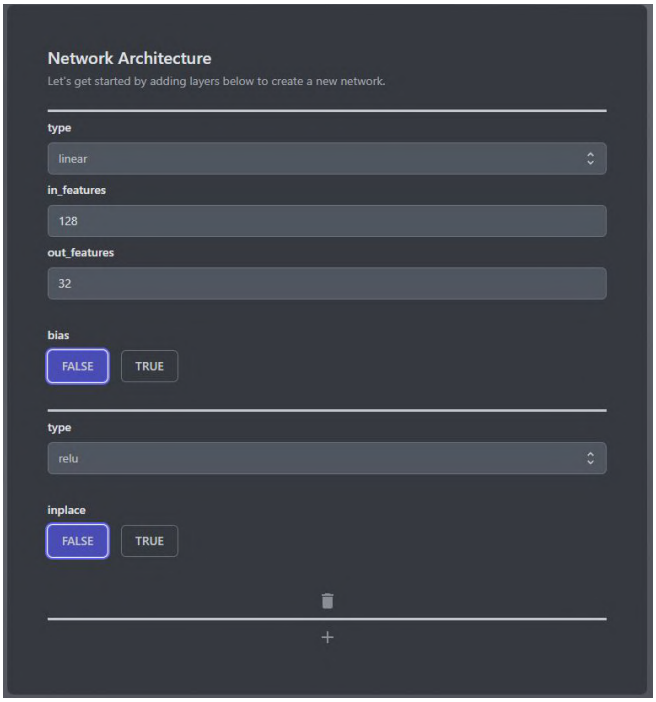

Fig. 7: The user interface to define the neural network architecture of the DRL agent. The user can define the type of layers, the input- and output size, or the activation function. The architecture will be visualized for improved understanding. Furthermore, based on the robot, there may be error messages if the in- and output size do not match with the robots specifications.

be seen in Fig. 7. By clicking on an add button, the user extends the architecture with a new module and thereby adds another module to the existing array. The last module added can also be deleted using the trash icon. A module consists of a type and the parameters needed for the type of module. In this example, it can be seen that linear layers have the parameters input features, output features and bias. When adding a new module, initially, it is a linear layer but its type can be changed to all available modules. Once finished with the architecture, the user has the possibility to make it public or keep it private and set its name in the "General Information" step. In the "Summary" step the user can see the types of modules combined and the access type once more before submitting it to the backend.

*4) **Specifying the Reward System and Starting a Training**:* In the final step, the user should specify the reward system by specifying values for a number of preset penalties and rewards using the reward generator. A default reward system is already set so that the user can also skip this step and start the training immediately. To start the training once all necessary input was provided, the user can submit the training task and a JSON object including the name of the training and the ids of the selected robot, network architecture, and hyperparameters will be sent to the backend. The backend will start *Arena-bench* and the training based on the selected configurations. After starting the training the user will be redirected to a page where the log output of the training, and the current best model can be downloaded.

C. Starting a New Evaluation Task

Another important feature of the web-app is the possibility to assess the performance of navigation approaches. It is possible to utilize the application for pure benchmarking of several planners against one another. The user can choose which metrics should be recorded and the implemented data recorder will record only the topics, necessary to calculate the metrics. Similar to the training task, starting a new evaluation is coupled with the selection of a high number of different configurations, which can be overwhelming for the user. Therefore, our task editor provides a straightforward user path with specified steps to be taken. From the dashboard, the user can directly select to create a new evaluation task. In the first step, a scenario file is be selected. Although, we highly recommend using scenarios because it allows for a more stable and consistent comparison between multiple evaluation runs, we also offer the possibility to evaluate on random tasks, where obstacles as well as start and goal position of the robot are placed randomly on the map. Therefore, the user has the possibility to toggle between random and scenario mode in the first step. Furthermore, the user can specify the number of evaluation episodes. In the second step, the user selects the robot that should be used for the evaluation. Based on this selection, a planner must be selected in the third step. The user can select between a variety of different planners listed in Table I or select one of the trained DRL agents, which has been created using the web app. In the last step, the user is again prompted to name the evaluation for better identification later on.

After starting the evaluation the user is redirected to a new page, where the log output of the evaluation is shown. Both, log files and the latest .csv evaluation files can be downloaded. We offer an evaluation class in form of a Jupyter notebook in which pre-existing code and explanations guide the user and allow for individual plotting of the results. This ensure a high degree of customization possibilities for the plots depending on the individual user. Exemplary plots generated with our evaluation class are illustrated in Fig. 9 In the future an additional live display of the plots inside the web-app is aspired. For the evaluation, a high number of quantitative and qualitative metrics presented in Table II are provided. The required workflow and screens of the web-app are illustrated in Fig. 8.

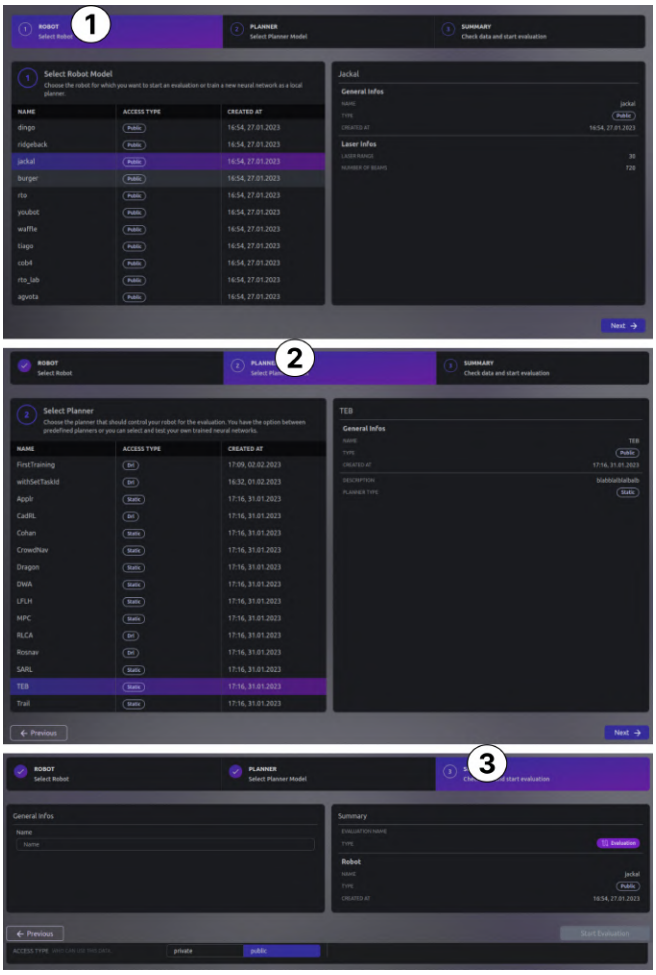

Fig. 8: Exemplary pipeline when conducting an evaluation run. It consists of three steps: (1) choosing the robot model, (2) choosing the planner, and (3) choosing the evaluation tasks.

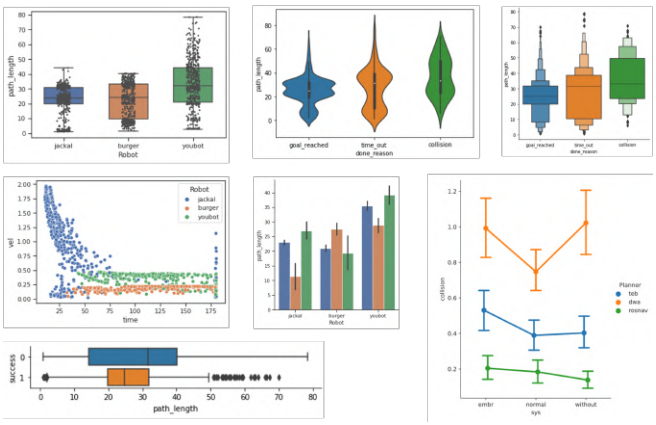

Fig. 9: Exemplary Plots created with the evaluation class.

TABLE II: Evaluation metrics

Hyperparameter	Value	Explanation	Calculation
Safety			
Collision[1,2]	-	Total number of collisions	
Clearing Dist. [2]	-	Distance to obstacles	
Robustness			
Success Rate[2]	%	Fraction of episodes with <2 collisions and no timeout	
Variance[2]	-	Variance between all measurements	$v = \sqrt{\sum_0^N \frac{(s_N - \bar{s}}{N}}$
Efficiency			
Path Length[1,2]	$[m]$	Path length in m	
Time to reach goal[2]	$[s]$	Time required to reach the goal	
Smoothness			
Movement Jerk[2]	$[\frac{m}{s^3}]$	Average acceleration	$j = \lvert\frac{\Delta a}{\Delta t}\rvert$, $a = \frac{\Delta v}{\Delta t}$
Roughness[2]	-	Inverse trajectory smoothness	$R(\mathrm{x}_i) = \frac{\text{Triangle Area}(\mathrm{x}_i, \mathrm{x}_{i+1}, \mathrm{x}_{i+2})}{\lvert\mathrm{x}_{i+2} - \mathrm{x}_i\rvert^2}$
Angle[2]	$[\frac{1}{m^2}]$	Angle over the path length	$c_{norm}(\mathrm{x}_i) = \frac{c(\mathrm{x}_i)}{\lvert\mathrm{x}_{i+1} - \mathrm{x}_i\rvert + \lvert\mathrm{x}_{i+2} - \mathrm{x}_{i+1}\rvert}$

VI. Conclusion

In this paper, we proposed Arena-Web, a web-based application to develop, train, and benchmark DRL-based navigation planners on different robotic systems. Therefore, the web app offers a highly open accessible user interface for a state-of-the-art mobile robot benchmarking framework. With our appealing and well-designed user interface, we offer even inexperienced users to dive into training and benchmarking of navigational planners without being overwhelmed by the amount of possibilities the app offers for experienced users, therefore, bringing the field of mobile robotics to a very large group and lower the barrier to entry. Arena-Web provides intuitive tools for generating new maps, scenarios, neural network architectures, reward values, and hyperparameters that allows for highly customized training and evaluation processes. The web-app provides a high number of pre-available robots and planners as well as predefined maps and scenarios. The evaluation class includes a variety of navigational metrics. The use of modern software frameworks and a well-structured system design makes the system not only very secure, but also offers a lot of potential for extensions and new features. By allowing the users to make their data publicly available, we offer the possibility of creating a large community where everyone benefits from others' creations. By providing possibilities to download all created data, the user is not limited to any of the functionalities of *Arena-Web*, and can also use small parts of the app separately. Future works aspire to include more visualizations and live videos (e.g. of the training and evaluation runs) inside the web application to improve user understanding and experience even more. As mentioned, ongoing works include the integration of the plotting into the web app, which will also enhance simplicity and usability even more. Furthermore, we currently work on more functionalities, such as robot modeling, and integration of multi-agent benchmarking. The web-app is free to use and openly available under the link stated in the supplementary materials.

References

[1] D. Dugas, J. Nieto, R. Siegwart, and J. J. Chung, "Navrep: Unsupervised representations for reinforcement learning of robot navigation in dynamic human environments," *arXiv preprint arXiv:2012.04406*, 2020.

[2] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[3] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: a survey," *Autonomous Robots*, pp. 1–29, 2022.

[4] E. Heiden, L. Palmieri, L. Bruns, K. O. Arras, G. S. Sukhatme, and S. Koenig, "Bench-mr: A motion planning benchmark for wheeled mobile robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4536–4543, 2021.

[5] L. Kästner, T. Bhuiyan, T. A. Le, E. Treis, J. Cox, B. Meinardus, J. Kmiecik, R. Carstens, D. Pichel, B. Fatloun *et al.*, "Arena-bench: A benchmarking suite for obstacle avoidance approaches in highly dynamic environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9477–9484, 2022.

[6] N. Tsoi, A. Xiang, P. Yu, S. S. Sohn, G. Schwartz, S. Ramesh, M. Hussein, A. W. Gupta, M. Kapadia, and M. Vázquez, "Sean 2.0: Formalizing and generating social situations for robot navigation," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 047–11 054, 2022.

[7] R. Marin, P. J. Sanz, and J. S. Sánchez, "A very high level interface to teleoperate a robot via web including augmented reality," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 3. IEEE, 2002, pp. 2725–2730.

[8] R. Siegwart and P. Saucy, "Interacting mobile robots on the web," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[9] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte *et al.*, "Minerva: A second-generation museum tour-guide robot," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 3. IEEE, 1999.

[10] L. Kästner, T. Buiyan, X. Zhao, L. Jiao, Z. Shen, and J. Lambrecht, "Towards deployment of deep-reinforcement-learning-based obstacle avoidance into conventional autonomous navigation systems," *arXiv preprint arXiv:2104.03616*, 2021.

[11] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6015–6022.

[12] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5113–5120.

[13] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[14] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[15] R. Guldenring, M. Görner, N. Hendrich, N. J. Jacobsen, and J. Zhang, "Learning local planners for human-aware navigation in indoor environments," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 6053–6060.

[16] L. Kästner, X. Zhao, T. Buiyan, J. Li, Z. Shen, J. Lambrecht, and C. Marx, "Connecting deep-reinforcement-learning-based obstacle avoidance with conventional global planners using waypoint generators," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1213–1220.

[17] N. Tsoi, M. Hussein, J. Espinoza, X. Ruiz, and M. Vázquez, "Sean: Social environment for autonomous navigation," in *Proceedings of the 8th International Conference on Human-Agent Interaction*, 2020, pp. 281–283.

[18] D. Schulz, W. Burgard, D. Fox, S. Thrun, and A. B. Cremers, "Web interfaces for mobile robots in public places," *IEEE Robotics & Automation Magazine*, vol. 7, no. 1, pp. 48–56, 2000.

[19] J. Koch, M. Reichardt, and K. Berns, "Universal web interfaces for robot control frameworks," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 2336–2341.

[20] T. Bhattacharjee, E. K. Gordon, R. Scalise, M. E. Cabrera, A. Caspi, M. Cakmak, and S. S. Srinivasa, "Is more autonomy always better? exploring preferences of users with mobility impairments in robot-assisted feeding," in *Proceedings of the 2020 ACM/IEEE international conference on human-robot interaction*, 2020, pp. 181–190.

[21] A. Di Nuovo, F. Broz, T. Belpaeme, A. Cangelosi, F. Cavallo, R. Esposito, and P. Dario, "A web based multi-modal interface for elderly users of the robot-era multi-robot services," in *2014 IEEE international conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2014, pp. 2186–2191.

[22] S. Coşar, M. Fernandez-Carmona, R. Agrigoroaie, J. Pages, F. Ferland, F. Zhao, S. Yue, N. Bellotto, and A. Tapus, "Enrichme: Perception and interaction of an assistive robot for the elderly at home," *International Journal of Social Robotics*, vol. 12, pp. 779–805, 2020.

[23] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. B. Beck, M. Haage, M. R. Hansen *et al.*, "Smerobotics: Smart robots for flexible manufacturing," *IEEE Robotics & Automation Magazine*, vol. 26, no. 1, pp. 78–90, 2019.

[24] F. Del Duchetto, P. Baxter, and M. Hanheide, "Lindsey the tour guide robot-usage patterns in a museum long-term deployment," in *2019 28th IEEE international conference on robot and human interactive communication (RO-MAN)*. IEEE, 2019, pp. 1–8.

[25] J. Bačík, P. Tkáč, L. Hric, S. Alexovič, K. Kyslan, R. Olexa, and D. Perduková, "Phollower—the universal autonomous mobile robot for industry and civil environments with covid-19 germicide addon meeting safety requirements," *Applied Sciences*, vol. 10, no. 21, p. 7682, 2020.

[26] J. M. Cañas, E. Perdices, L. García-Pérez, and J. Fernández-Conde, "A ros-based open tool for intelligent robotics education," *Applied Sciences*, vol. 10, no. 21, p. 7419, 2020.

[27] L. A. Hormaza, W. M. Mohammed, B. R. Ferrer, R. Bejarano, and J. L. M. Lastra, "On-line training and monitoring of robot tasks through virtual reality," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1. IEEE, 2019, pp. 841–846.

[28] A. Faina, B. Nejati, and K. Stoy, "Evobot: An open-source, modular, liquid handling robot for scientific experiments," *Applied Sciences*, vol. 10, no. 3, p. 814, 2020.

[29] R. Bejarano, B. R. Ferrer, W. M. Mohammed, and J. L. M. Lastra, "Implementing a human-robot collaborative assembly workstation," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1. IEEE, 2019, pp. 557–564.

[30] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[31] S. Osentoski, G. Jay, C. Crick, B. Pitzer, C. DuHadway, and O. C. Jenkins, "Robots as web services: Reproducible experimentation and application development using rosjs," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 6078–6083.

[32] R. Toris, J. Kammerl, D. V. Lu, J. Lee, O. C. Jenkins, S. Osentoski, M. Wills, and S. Chernova, "Robot web tools: Efficient messaging for cloud robotics," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 4530–4537.

[33] "Webviz," https://webviz.io/, accessed: 2023-02-03.

[34] "Foxglove," https://foxglove.dev/, accessed: 2023-02-03.

[35] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *2015 european control conference (ECC)*. IEEE, 2015, pp. 3352–3357.

[36] X. Xiao, B. Liu, G. Warnell, J. Fink, and P. Stone, "Appld: Adaptive planner parameter learning from demonstration," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4541–4547, 2020.

[37] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[38] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6252–6259.

[39] C. Rösmann, "Time-optimal nonlinear model predictive control," Ph.D. dissertation, Dissertation, Technische Universität Dortmund, 2019.

[40] P. T. Singamaneni, A. Favier, and R. Alami, "Human-aware navigation planner for diverse human-robot ineraction contexts," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[41] Z. Xie and P. Dames, "Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles," *IEEE Transactions on Robotics*, pp. 1–20, 2023.

[42] K. Li, Y. Xu, J. Wang, and M. Q.-H. Meng, "Sarl: Deep reinforcement learning based human-aware navigation for mobile robot in indoor environments," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 688–694.

[43] L. Kästner, C. Marx, and J. Lambrecht, "Deep-reinforcement-learning-based semantic navigation of mobile robots in dynamic environments," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1110–1115.