Arena 3.5: Advancing Social Navigation by Generating Collaborative and Highly Dynamic Environments at Scale

Volodymyir Shcherbyna[1,*], Linh Kästner[1,2,*], Huajian Zeng[3],
Maximilian Ho-Kyoung Schreff[1], Halid Osmaev[1], Nam Truong Tran[1],
Diego Diaz[1], Jan Golebiowski[1], Harold Soh[2]

[*]Equal Contribution

[1]Technical University Berlin (TUB), Germany

[2]National University of Singapore (NUS), Singapore

[3]Technical University Munich (TUM), Germany

***Abstract*—Building upon our previous contributions, this paper introduces Arena 3.5, an extension of Arena-Bench [1], Arena 1.0 [2], Arena 2.0 [3] and Arena 3.0 [4]. It is an iterative extension to Arena 3.0, which additionally includes automated generation of social environments using text prompts and generative models. We significantly enhance the realism of human behavior simulation by incorporating a diverse array of new social force models and interaction patterns, encompassing both human-human and human-robot dynamics. The platform provides a comprehensive set of new task modes, designed for extensive benchmarking and testing and is capable of generating realistic and human-centric environments dynamically, catering to a broad spectrum of social navigation scenarios. In addition, the platform's functionalities have been abstracted across three widely used simulators, each tailored for specific training and testing purposes. The platform's efficacy has been validated through an extensive benchmark and user evaluations of the platform by a global community of researchers and students, which noted the substantial improvement compared to previous versions and expressed interests to utilize the platform for future research and development. Arena 3.5 is openly available at https://github.com/Arena-Rosnav.**

I. Introduction

As the integration of human-robot collaboration becomes increasingly essential in fields such as healthcare, logistics, and delivery, the necessity for robots to navigate through dynamic and human-centric environments is paramount. The realm of social navigation, where robots maneuver and interact in human-populated settings, is rapidly gaining attention. Critical factors in navigation include not just safety, but also operational smoothness, user stress, acceptance, interaction, and maintaining efficiency. Recent years have seen strides in social robotics [5],[6],[7] with numerous research works proposing platforms and approaches for social navigation and benchmarking.

However, many existing platforms are constrained to specific planning approaches, either learning-based or traditional, and exhibit limited extensibility [6]. Simulations often feature overly simplified human behavior models, leading to a widened simulation-to-reality (sim2real) gap. Furthermore, the focus on single simulators can be restrictive due to inherent limitations of individual simulators. Testing capabilities are also often constrained, with little environmental randomization or planner diversity. In our prior work Arena 1.0 [2], we tackled the challenge of deploying both learning-based and traditional planners within a unified simulation environment. We expanded this approach to include the deployment of 2D trained deep reinforcement learning (DRL) agents in Gazebo simulators, enhancing the realism of robotic kinematics and reducing the sim2real gap in Arena 2.0 [3]. Additionally, we integrated an array of planners and an evaluation pipeline with Arena Bench [1].

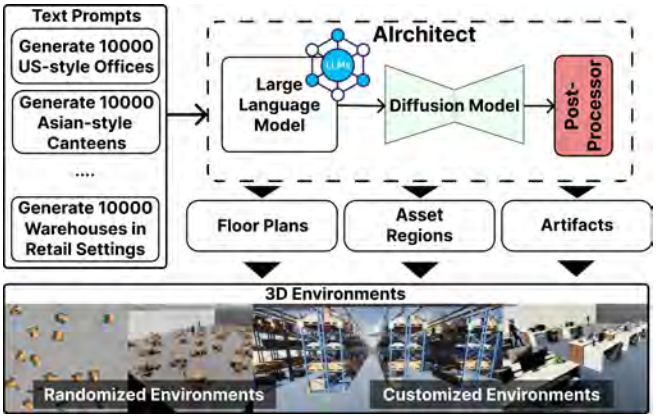

Fig. 1: Extended world and map generation tool using large language models (LLMs) and generative models. The core idea is to generate realistic environments using text prompts. As a result, the user can incorporate cultural, regional, or architectural differences into the generation process using the knowledge of the LLM. This allows for the creation of diverse simulation environments for training and testing purposes, which is especially useful for social navigation problems as regional differences can be modeled more accurately.

However, in environments centered around human activity, robots must undergo rigorous testing for human-robot interaction. In these settings, realistic human simulation is crucial, which makes it necessary to not only model

This research / project is supported by A*STAR under its National Robotics Programme (NRP) (Award M23NBK0053). This research is supported in part by the National Research Foundation (NRF), Singapore and DSO National Laboratories under the AI Singapore Program (Award Number: AISG2-RP-2020-017).

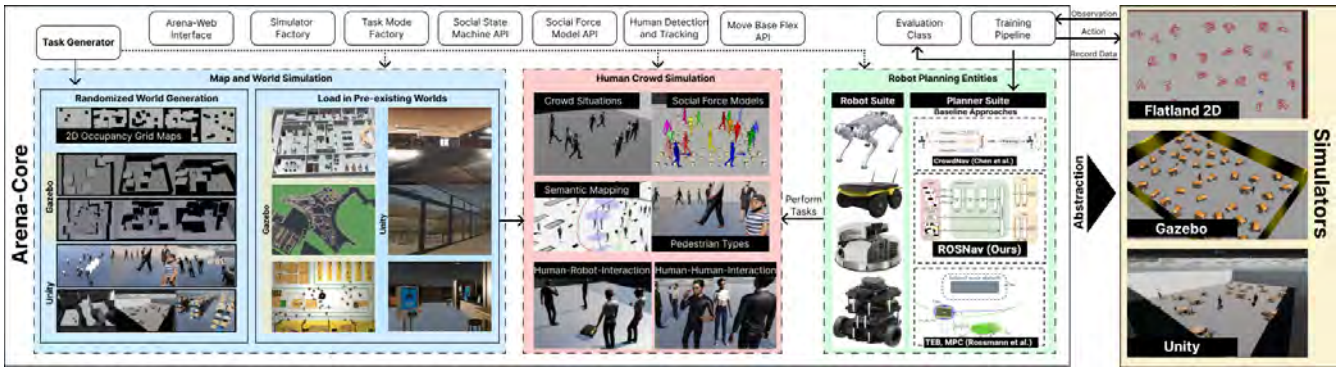

Fig. 2: System architecture and modules of Arena 3.0. At its core, the platform consists of map and world generator algorithms, realistic pedestrian simulation, and comprehensive robot and navigation algorithm suites. The core architecture is fully abstracted from the simulators, allowing for cross-simulator scenario compatibility, with specific functionalities like RGB-D data available in Unity or Gazebo, and LiDAR applicable in all three simulators. Additionally, the platform features a training pipeline and an extended version of the MBF navigation framework to support the development and refinement of navigation approaches. Supplementary modules, such as the evaluation class and the web-based companion app, provide tools for data analysis and manual scenario creation. Using the provided API endpoints, the user can extend the platform with new planners (Move Base Flex API), task modes (task factory), social force models (SFM API), or simulators (simulator-factory). Further, computer vision modules for pedestrian detection and tracking are integrated to provide pedestrian data for further use.

interactions among humans but also between humans and robots. These scenarios present new challenges and opportunities for robots to adapt their behavior and decision-making processes.

With these considerations, we developed the third iteration of Arena to provide a comprehensive platform focusing on advancing social navigation with Arena 3.0 [4], which compromises an extensive software stack containing multiple modules and simulation environments focusing on social navigation. There, we incorporated state-of-the-art social force models and social interaction patterns and enhanced the task generator to include tasks tailored for specific situations like blockages or emergencies, but also dynamically and randomly generated human-centric environments such as canteens, offices, or industrial halls, offering a vast array of training and testing environments. In Arena 3.5, we extend these scene generation capabilities by including a large language model and diffusion models to generate limitless indoor environments at scale. Using this combination, the user can incorporate cultural and architectural differences between regions into the simulation to enhance realism. These functionalities are abstracted in the core module and are compatible on three simulators: Flatland, Gazebo, and Unity.

The main contributions of this work are the following:

- Integration of human behavior models, enhancing modules with human and human-robot interaction forces.
- Development of an extended task generation toolkit, enabling users to create and design specific worlds, scenarios, and tasks for training and testing. This includes the dynamic and random generation of socially-centered environments such as canteens, warehouses, or offices.
- Abstraction of core functionalities across three widely used simulators: Flatland2D for efficient 2D training, Gazebo for realistic robot kinematics testing, and Unity for photorealistic training scenarios.
- Provision of comprehensive APIs, facilitating the straightforward extension of new modules, which includes social force models, social state machines, and intermediate planner modules.
- Enhancement of the robot and planning suite. The planning framework has been advanced to MBF [8], which offers improved navigation performance and is extended by an intermediate planner concept for greater customization of user-developed planners.
- Integration of LLMs and diffusion models to generate diverse indoor environments using text prompts. Therefore, we provide a user interface in which the user can input a textprompt and download the resulting simulation file.

II. Overview of Arena 3.5

Arena 3.5 consists of several key modules, which include pedestrian simulation, world and task generation, a training pipeline, and the integration of planner and robot suites. It features specific API endpoints for the integration of self-developed modules and supplementary tools like the evaluation class or a web-based front end for manual scenario creation.

A. System Design

Figure 2 outlines the modules of Arena 3.0, which introduced a core module, denoted as Arena-core, that abstracts functionalities from three simulators, enabling most functions to run concurrently across these simulators (subject to limitations like RGB-D being exclusive to Unity and Gazebo). While

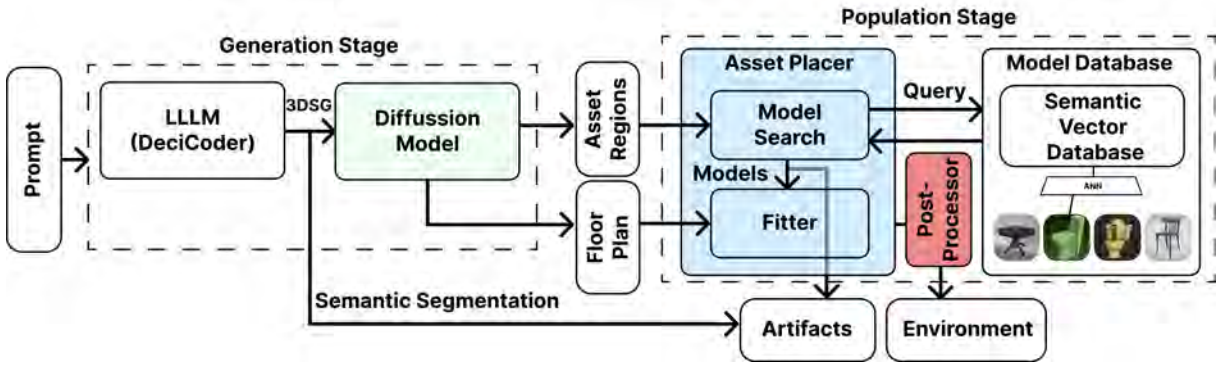

Fig. 3: Data flow of the *Generation Stage* and the *Population Stage*. The Generation Stage combines multiple SotA technologies to process text inputs into a floor plan image and room asset locations. 3D scene graphs are used as an intermediate data structure to divide the problem into a text transformation task solvable by an LLM, and a graph transformation task solvable by a spatial GNN.The *Population Stage* populates the floor plan's asset zones with 3D models by employing the *Asset Placer*. A pre-built semantic vector *Model Database* is queried for a related model, which is arranged into the zone by a *Fitter* algorithm. After a final post-processing step, the end result is a finished environment consisting of 3D walls and models.

all simulators were already available and could be selected in the previous version of Arena, only basic functionalities such as loading an empty world, spawning a robot, or loading obstacles were provided and the user had to implement further functionalities for each simulator individually. In contrast, Arena 3.0 introduces the complete abstraction of all functions to completely automate processes making it possible to include new functions across all simulators simultaneously by only extending the core module. This design also enhances comparability of approaches and interoperability, e.g., for algorithms trained in a 2D simulator and later validated in Gazebo. A vital component of this system is the crowd and human simulation, incorporating various advanced social force models, social interactions patterns, and varying human types. Using the improved map and task generators, users can generate diverse environments in socially-centric settings like canteens, offices, or warehouses, with each episode offering variation through our map generators. Pre-built worlds, such as a replica of the National University of Singapore campus or hospitals, are also available for loading. This variety enables users to design highly specific or randomized scenarios for both qualitative and quantitative testing and validation. Furthermore, the robot and planner suite was extended offering a wide range of options. Additional modules, like the extended navigation stack and training pipeline, are provided to develop and fine-tune navigation strategies. Furthermore, we also included computer vision approaches such as YOLO [9] to detect and track humans. Furthermore, the companion Webapp Arena-web [10] has been extended and adapted significantly to ensure that all new functions and modules of Arena 3.0 were compatible with the Webapp. It is thus denoted as Arena-Web-v2. In addition, Arena 3.5 introduces an extension to the world generation module using generative models, which is denoted as Arena-gen. For a more detailed explaination about the core features of Arena 3.0, we refer to our previous work [4]. In the following, the additional world and map generation module using Arena-gen will be explained in more detail.

III. World and Map Generation

Arena 3.5 introduces world and map generation using text prompts and generative models. Figure 3 illustrates the system design of our approach. Figure 4 showcases exemplary worlds generated using our platform. For a demonstration of worlds generated using text prompts, we refere to our supplementary video, which is also made available online.

A. System Design

Our generation process consists of a 2-stage pipeline, a generation stage and a population stage. Generation uses an LLM that transforms natural language prompts into a a machine-readable graph. This graph is used in GNN inference to produce (1) an annotated floor plan image, and (2) *asset* regions placed within the rooms. During populations, the space of the assets is filled with models from a semantic model database. The floor plan, together with all placed models, is transformed into a final 3D environment that can be loaded by a simulator. Subdividing the process in this manner allows us to exploit the full extent of the expressiveness of the generative model during generation and then solving the model-populating sub-problem using more modular semi-classical approaches. The system design is illustrated in 3.

1) Dataset: Our dataset consists of (3DSG, floor plan image) pairs that are used to train our GNN. The samples are based on the *CubiCasa5K* [11] dataset, which contains almost 5000 suitable real-world floor plan scans. The provided CubiCasa5K data processing detects the room and object segmentations using pre-trained CNNs, which we use as a base for our own data processing. We extract the 3DSG by (1) filtering and re-categorizing assets, (2) classifying doorways as either inter-room or external, (3) building a room connectivity matrix from inter-room doorways, (4) assigning

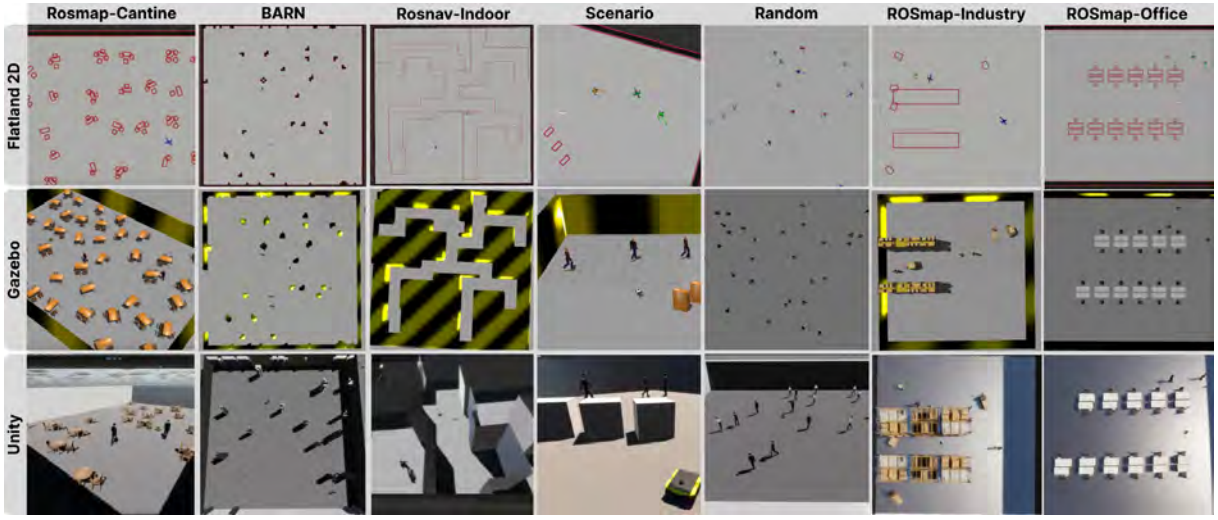

Fig. 4: Different map and world generator modes on three simulators. The map and world generation is abstracted from the simulators making the creation of worlds and scenarios unified across all simulators. This provides the ability to train and test the same agents on different simulators leveraging the strength of each simulator.

the remaining assets to the sub-graph of the containing room. The result is a hierarchical graph in our 3DSG format paired with a cleaned floor plan image for direct insertion of our dataset. We calculated a series of graph metrics and annotation statistics to verify the diversity of our produced dataset.

2) LLM: The purpose of our LLM is to transform a user input describing any indoor environment into a valid 3DSG yaml file. For our narrow use case, fine-tuning an existing pre-trained LLM implementation is the most direct approach to achieve both, a high understanding of user input and a conforming model output. Code completion LLMs are a popular use case that is the most suited for our objective, leading us to choose *DeciCoder* [12] based on the *StarCoder* [13] dataset. Fine tuning was performed on a hand-annotated dataset verbally describing scene graphs.

3) GNN: The core module of the generation stage is the GNN that creates the final floor plan and asset regions from an intermediate 3DSG. We base our GNN on the state-of-the-art *HouseDiffusion* [14] model, which generates room/dorway segmentations from *RPlan* room-connectivity graphs. It uses graph transformers and multi-head attention for input-constrained continuous denoising in tandem with entropy-oriented discrete denoising, resulting in significant improvements over previous models in realism, diversity, and input conformity. The solution offered by HouseDiffusion roughly corresponds to solving the generation problem for the 3DSG upper half. We extend the architecture in multiple ways to solve our wider problem.

4) Asset Placement: The main addition is the placement of assets defined in a room node sub-graph within the room vertex boundaries. We realize this as a parallel continuous denoising process to the existing room denoising. Processing is performed on each room and associated sub-graph individually. Doorways are treated as other assets for the new attention masks, but are not subjected to the asset denoising themselves. Assets are initialized as random quadrilaterals and undergo the same discrete denoising process as rooms. Corresponding continuous constraints are (1) appropriate size for asset type, (2) full containment within the room, (3) no overlap with other assets.

B. Population Stage

Once the generation stage is completed, the generated entities are spawned and orgnized during the population stage to create the 3D simulation environment. We use a model database as a natural language queriable vector database containing references to models and associated meta data.We build the database by recursively parsing a directory structure containing models and annotation, then generating (1) a Unity Asset Bundle, and (2) a queriable vector database. Build and Query functionalities are exposed programatically through a Python module API. Additionally, we provide a unified command line interface (CLI) that is wrapped and integrated into ROS [15][16] with *rosrun* scripts.

IV. Conclusion

In this paper, we introduced Arena 3.5, an iterative improvement to our previous version Arena-Bench and Arena 1.0, Arena 2.0, and Arena 3.0. In this version, we extended the existing map and world generator with the capability to generate diverse simulation worlds from user text prompts. Therefore, our approach consists of an LLM to interpret and process the text prompt and provide it to a difussion model to generate meaningful simulation worlds. As a result, the user can now add a diverse range of cultural and architectural differences into the text prompt to generate a large variety of simulations for training and testing purposes. The platform also integrates realistic pedestrian movements and social

force modeling, including human-human and human-robot interactions.

ACKNOWLEDGEMENTS

This research / project is supported by A*STAR under its National Robotics Programme (NRP) (Award M23NBK0053). This research is supported in part by the National Research Foundation (NRF), Singapore and DSO National Laboratories under the AI Singapore Program (Award Number: AISG2-RP-2020-017).

REFERENCES

[1] 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.

[2] L. Kästner, T. Buiyan, L. Jiao, T. A. Le, X. Zhao, Z. Shen, and J. Lambrecht, "Arena-rosnav: Towards deployment of deep-reinforcement-learning-based obstacle avoidance into conventional autonomous navigation systems," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6456–6463.

[3] L. Kästner, R. Carstens, H. Zeng, J. Kmiecik, T. Bhuiyan, N. Khorsandhi, V. Shcherbyna, and J. Lambrecht, "Arena-rosnav 2.0: A development and benchmarking platform for robot navigation in highly dynamic environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 11 257–11 264.

[4] L. Kästner, V. Shcherbyna, H. Zeng, T. A. Le, M. H.-K. Schreff, H. Osmaev, N. T. Tran, D. Diaz, J. Golebiowski, H. Soh, and J. Lambrecht, "Arena 3.0: Advancing social navigation in collaborative and highly dynamic environments," Robotics Science and Systems 2024.

[5] 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.

[6] A. Francis, C. Pérez-d'Arpino, C. Li, F. Xia, A. Alahi, R. Alami, A. Bera, A. Biswas, J. Biswas, R. Chandra *et al.*, "Principles and guidelines for evaluating social robot navigation algorithms," *arXiv preprint arXiv:2306.16740*, 2023.

[7] 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.

[8] "Move Base Flex," https://github.com/magazino/move_base_flex, accessed: 2024-01-22.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[10] L. Kästner, R. Carstens, L. Nahrwold, C. Liebig, V. Shcherbyna, S. Lee, and J. Lambrecht, "Demonstrating arena-web: A web-based development and benchmarking platform for autonomous navigation approaches," *Robotics: Science and Systems (RSS)*, 2023.

[11] A. Kalervo, J. Ylioinas, M. Häikiö, A. Karhu, and J. Kannala, "Cubicasa5k: A dataset and an improved multi-task model for floorplan image analysis," *CoRR*, vol. abs/1904.01920, 2019. [Online]. Available: http://arxiv.org/abs/1904.01920

[12] DeciAI Research Team, "Decicoder," 2023. [Online]. Available: https://huggingface.co/deci/decicoder-1b

[13] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, Q. Liu, E. Zheltonozhskii, T. Y. Zhuo, T. Wang, O. Dehaene, M. Davaadorj, J. Lamy-Poirier, J. Monteiro, O. Shliazhko, N. Gontier, N. Meade, A. Zebaze, M.-H. Yee, L. K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov, Z. Wang, R. Murthy, J. Stillerman, S. S. Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, N. Fahmy, U. Bhattacharyya, W. Yu, S. Singh, S. Luccioni, P. Villegas, M. Kunakov, F. Zhdanov, M. Romero, T. Lee, N. Timor, J. Ding, C. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, J. Robinson, C. J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C. M. Ferrandis, S. Hughes, T. Wolf, A. Guha, L. von Werra, and H. de Vries, "Starcoder: may the source be with you!" 2023.

[14] M. A. Shabani, S. Hosseini, and Y. Furukawa, "Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising," 2022.

[15] 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.

[16] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074