Voxels: A Lightweight Simulation for Mobile Robotics

Maxim Simon, Zdeněk Rozsypálek, Tomáš Krajník

Virtually simulated environments are often used to test software or train neural networks for robotics. Most robotics simulators were created to support development, and therefore focus on realistic simulation of sensors and dynamics. However, machine learning demands high simulation speed, as training of a successful policy requires to simulate extended periods of time. We present Voxels, an open-source robotic simulator that favours speed over physical or visual realism, and rapid, rather than complex, generation of the environment.

When a new method for autonomous mobile robotics is being developed, its transition from idea to implementation on a robotic platform is difficult to execute incrementally and effectively. Proper training and validation require repetitive testing, evaluation, and debugging in relevant environments and conditions. Such testing demands extensive logistical support, exacerbated by hardware attrition. Furthermore, ground-truthing and benchmarking the robot's behavior requires external infrastructure. Even with all logistics support available it is still problematic to identify the root cause of the robot behaving in an unexpected manner.

Implementing and validating any method first in a simulation alleviates most of these problems and eliminates hardware and logistical issues. Modern frameworks, such as ROS(2), make the process easier by interfacing the method's inputs and outputs with a suitable simulator. Virtual environments can be generated to fit the needs of a test, adjusted at will, and fetched at a moment's notice. A simulator offers ground truth for free, while allowing to set sensor fidelity. This, combined with tailored environments, boosts the development efficiency.

Contemporary simulators were designed primarily to allow iterative manual development of robotic methods. However, as most nowadays methods are developed through machine learning, the number of simulation trials is much higher than in manual development. This creates a strong demand for simulation efficiency [1]. The demand is particularly high in the case of reinforcement learning, where training a robotic method can require time equivalent to weeks, months, or even years of real time. Furthermore, machine learning requires a high diversity of input data [2], creating a need for easily reconfigurable or even procedurally-generated simulation environments. These factors can be more crucial than simulation fidelity [3].

Engines like Unity [4], Unreal [5], or OAISYS [6], and robotic simulators such as Gazebo [7] offer high-fidelity sensor simulation but lack the efficiency required for rapid training of machine learning models. Simulators suited for reinforcement learning, like AI Habitat [8], Robothor [9] or iGibson [10], [11], are tailored for small-scale indoor environments.

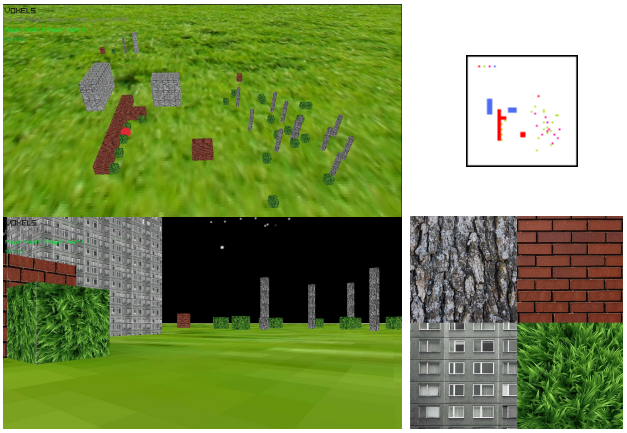

Fig. 1. **Top left:** View from a 3rd player view. The red cube is the simulated robot. **Bottom left:** View from the simulated robot's front-facing camera. **Top right:** Input 64 x 64 pixel image, based on which the environment was generated. Red pixels are bricks, blue are buildings, pink are trees, and green are bushes. **Bottom right:** Texture atlas—source for images that are pasted on voxels in the simulation.
Source images for textures are courtesy of: wikimedia: tree, grass (for ground); unsplash: bricks; pngtree: bush; pinimg: building.

We built a new simulator 'Voxels', which favours speed and variety over visual or physical realism. The simulation world is made up of voxels, which are procedurally generated from a 2D input image by mapping pixels' colours and positions to types and positions of voxel clusters, see Figure 1. As the main user input to the simulator is a 2D image describing the spatial layout of the simulated world, the user can quickly generate a sufficiently diverse set of environments, thereby supporting the robustness of machine-learned methods. All voxels in one cluster have the same dimensions and visual appearance. The texture they are covered with is picked from a texture atlas, see Figure 1, which can be expanded. The ground and the far background are rendered as a large pasted texture. Movement of the robot is planar and simulated kinematically. It is rendered as a single red cube and is controllable entirely through ROS(2) topics, from which it receives inputs and to which it outputs sensor data. Additional control is possible through arrow keys. Further keys allow teleporting the robot to a given position and switching between 1st and 3rd person views.

As the simulator does not take into account robot dynamics, and the robot body is a cube, it is not suitable for the development of control algorithms. It is intended for testing or training autonomous high-level navigation strategies, with an emphasis on speed and capability to quickly generate

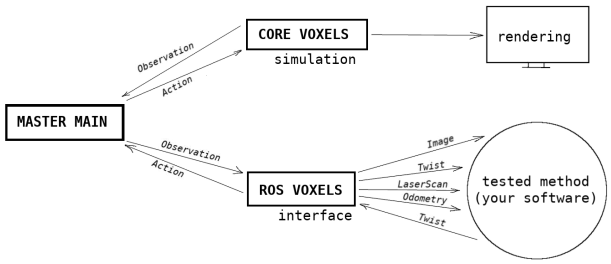

Fig. 2. A diagram illustrating the architecture of Voxels simulation. *Core Voxels* calculates everything regarding the simulation environment. *ROS Voxels* acts as an interface between the simulation and the tested software. *Master Main* acts as a bridge between *Core Voxels* and *ROS Voxels*.

scenarios and environments.

Voxels is implemented in C++ with a ROS(2) interface. For rendering and additional features, it relies on the open-source raylib library [12], which is implemented in C. The architecture of our code is such that allows adding interfaces to be used with the same underlying simulator. As seen in Figure 2, the codebase is divided into 3 parts: *Core Voxels*, *ROS Voxels* and *Master Main*.

Core Voxels handles the logic and rendering of the simulation environment in its entirety. It processes an input *Action* struct and returns an output *Observation* struct. **ROS Voxels** publishes *Observation* data to ROS(2) topics and builds *Action* from subscribed data. **Master Main** acts as a bridge between *Core Voxels* and *ROS Voxels*. It launches 2 threads in which each *Core* and *ROS* run independently of each other, so that iterating the simulation step and ROS(2) spin do not block each other.

Currently implemented sensors are: odometry, colour camera, and lidar. Each is published on its own ROS(2) topic. The robot in the simulation can be controlled through a ROS(2) topic by velocity commands. Additionally, a ROS(2) tf2 transform is available.

Input, struct *Action*, contains only velocity commands. Output, struct *Observation*, currently contains velocity commands, position and orientation, front camera image, and lidar 2D scan.

For future work, the separation of the simulation's codebase allows us to create a reinforcement learning interface, which will act in place of the ROS(2) interface, with making minimum changes to the underlying simulation. The interface will include synchronization of the simulation steps with the learning steps, and the ability to run multiple simulation instances in parallel to reduce training time.

We have built a voxel-style simulation for the development of high-level navigation strategies. The entire testbed environment in which the robot moves is composed of voxels of varying visual textures. From our research into existing simulations, we have concluded that there lacks a low-fidelity, easy-to-use software aimed at testing navigation and decision-making algorithms in all indoor, outdoor, structured, and unstructured environments. Our framework is focused on simulating environments and conditions for testing, with favouring variety and speed over visual or physical realism.

The simulation is used for the development of a visual teach and repeat method [13]. At AMD Ryzen 7 PRO with 8 cores and Radeon RX Vega 7 integrated graphics without the use of any GPU, the simulation can generate up to a 120 camera frames per second of a simulated world containing: a sky dome of 500 units (simulation meters) in radius, 500 x 500 units of grass plane as the ground and 64 x 64 units of procedurally-generated environment, as displayed in Figure 1. Extended documentation of Voxels is provided at our GitHub [14].

REFERENCES

[1] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters, "Robot learning from randomized simulations: A review," *Frontiers in Robotics and AI*, vol. 9, p. 799893, 2022.

[2] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[3] J. Truong, M. Rudolph, N. H. Yokoyama, S. Chernova, D. Batra, and A. Rai, "Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation," in *Proceedings of The 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 859–870.

[4] T. Musil, M. Petrlık, and M. Saska, "Hardnav-simulator for benchmarking robust navigation and place recognition in large, confusing and highly dynamic environments," 2023.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[6] M. G. Müller, J. Lim, L. Schmid, H. Blum, W. Stürzl, A. Gawel, R. Siegwart, and R. Triebel, "Interactive oaisys: A photorealistic terrain simulation for robotics research," in *ICRA 2022 Workshop on Releasing Robots into the Wild: Simulations, Benchmarks, and Deployment*, 2022.

[7] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[8] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A platform for embodied ai research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[9] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford *et al.*, "Robothor: An open simulation-to-real embodied ai platform," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3164–3174.

[10] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D'Arpino, S. Buch, S. Srivastava, L. Tchapmi *et al.*, "igibson 1.0: A simulation environment for interactive tasks in large realistic scenes," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7520–7527.

[11] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. Vainio, C. Gokmen, G. Dharan, T. Jain *et al.*, "igibson 2.0: Object-centric simulation for robot learning of everyday household tasks," 2021.

[12] raysan5. (2026) Raylib website. [Online]. Available: https://www.raylib.com/

[13] Z. Rozsypálek, T. Rouček, T. Vintr, and T. Krajník, "Multidimensional particle filter for long-term visual teach and repeat in changing environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 1951–1958, 2023.

[14] M. Simon and Z. Rozsypalek. (2026) Voxels. [Online]. Available: https://github.com/maximsimon/voxels