Interactive OAISYS: A photorealistic terrain simulation for robotics research

Marcus G. Müller[1,2*], Jaeyoung Lim[2*], Lukas Schmid[2], Hermann Blum[2], Wolfgang Stürzl[1], Abel Gawel[2], Roland Siegwart[2], Rudolph Triebel[1]

***Abstract*— Photorealistic simulation pipelines are crucial for the development of novel robotic methods and modern machine vision approaches. Simulations have been particularly popular for generating labeled synthetic data sets, which otherwise would require vast efforts of manual annotation when using real data. However, these simulators are usually not interactive, and the data generation process cannot be interrupted. Therefore, these simulators are not suitable for evaluating active methods, such as active learning or perception aware path planning, which make decisions based on the observed perception data. In order to address this problem, we propose a modified version of the simulator OAISYS, a photorealistic scene simulator for unstructured outdoor environments. We extended the simulator in order to use it in an interactive way, and implemented a developer-friendly RPC interface so that it is easy for any environment to integrate into the simulator. In this paper, we demonstrate the functionality of the extension on 3D scene reconstruction to show its future research potential and provide an example of the implementation using the middleware ROS. The code is publicly available under `https://github.com/DLR-RM/oaisys`**

I. INTRODUCTION

Simulation frameworks are important tools for the development and evaluation of most robotic methods. Simulations make it is possible to generate controlled conditions to test individual modules and scenarios [1] [2], which would have otherwise be challenging with real experiments. Furthermore, they can be used to simulate scenarios which cannot be easily recreated or are too dangerous to be tested in the real world. Simulators that can produce photorealistic images have drawn attention in recent years mainly due to the success of modern machine vision methods like deep neural networks [3]. However, most of such methods need large amounts of annotated data to train a desired model. Since large-scale manual annotation is prohibitively expensive and error-prone, high-quality synthetic data makes for an attractive alternative.

For machine vision approaches, the data needs to be as photorealistic as possible in order to reduce the so called sim-to-real gap. Since creating such photorealistic data requires a lot of computational resources as well as long rendering times, it is more common to use recorded datasets instead of interactive simulation environments.

Such datasets are usually carefully designed by experts and then rendered with high amount of computational resources. The benefit of datasets is that their results are reproducible and can be reused for multiple experiments. However, static datasets are not suitable for evaluating active methods, such as active perception, active learning, which make decisions interactively based on observed data. For such tasks, it would be more beneficial to have direct access to a simulator, which has parameters that could be interactively changed, such as altering sensor positions to actively choose view points based on previously perceived data.

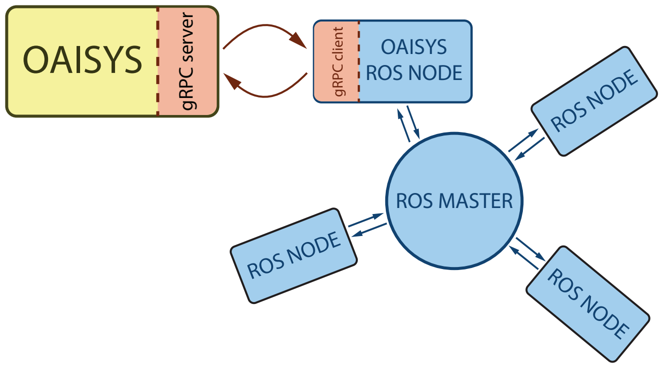

Fig. 1: Conceptual illustration of the iterative version of OAISYS with the middleware, ROS.

In recent years, simulators based on game engines have gained popularity, which can help with such approaches [4] [5] [6]. However, most of these simulators focus on industrial environments or autonomous driving in structured environments. Less simulators are available for unstructured outdoor environments while also producing photorealistic output images.

Another possibility is the usage of 3D creation suites like blender [7] and Maya [8]. Game engines, like Unreal Engine [9] and Unity [10], have the advantage that they can render data in real-time. However, 3D creation suites rendering engines tend to produce outputs of higher visual quality. As a result, the sim-to-real gap is usually smaller with such engines compared to game engines.

OAISYS is a simulator which is designed to create many different unstructured outdoor environments [11]. Furthermore, it is designed to create photorealisitc output images, which makes it suitable for modern machine vision tasks. However, it was not possible to use it in an iteractive way.

To overcome this shortcoming, we present in this paper a modification to the simulator OAISYS in order to make it usable in an interactive fashion. Furthermore, we extended the possibility to use it with the popular robotic middleware, ROS. See Fig. 1. To show its functionality and potential opportunities, we demonstrate our extensions with two robotic tasks. A scene reconstruction task where a 3D

[*] Equal contribution
[1] Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Germany
[2] Autonomous Systems Lab, Swiss Federal Institute of Technology (ETH Zürich), Switzerland

scene is reconstructed from a predefined viewpoint set, and a perception aware planning task where the next view is chosen interactively with the simulation.

Therefore, we can summarize our contributions as follows:

- several extensions of OAISYS to enable an iteratable mode for the simulator
- integration of the simulator with the common middleware, ROS
- show different evaluation experiments to demonstrate the potential of the mentioned extension

In the next section, we will first give a brief overview over the OAISYS simulation framework. Afterwards, we will introduce the extensions we made for the simulator. Finally, we will show two evaluation experiments to demonstrate the possibilities of our extension.

II. OAISYS

In this section, we will give a brief overview over the simulator OAISYS [11].

OAISYS is a simulator for unstructured outdoor environments with a focus on planetary environments. The simulator is able to generate automatically different worlds, which can be configured via a configuration file. It builds up on the free and open-source 3D computer graphics software, Blender [7], and therefore, able to achieve photorealistic image results. The basic workflow of the simulator will be explained in the following.

First, OAISYS loads a basic mesh into the simulator, which is deformed by a modifier using random noise as deformation parameters. The resulting object, called the stage, is used as terrain base mesh. Although, the basic mesh could be any object, it will be a plane in most outdoor cases. In order to get local terrain information, the simulator loads terrain textures provided by the user and blends them together. Each texture can have its individual semantic label. Since the blending process is known, the semantic label for each part of the new blended texture is kept. The resulting blended texture is applied as material to the stage, which gives the stage color information as well as roughness and local deformation data. Next, the simulator gives the option to place 3D assets onto the surface of the stage. OAISYS has the option to scatter a large amount of objects due to the usage of blender's particle system.

After the terrain is set up with all its textures and meshes, the light setup can be created as well as the placement of the desired sensors. The movement of the sensors can either be done randomly or via a csv file.

Once the entire scene is in place, the rendering is executed. OAISYS can render a variety of modalities, such as RGB, depth and different semantic maps.

Once all modalities are rendered, the light and sensor setups are updated to take another sample or a new batch is created. When a new batch creation is triggered, all assets are updated to get a new sample of the next environment. This procedure is continued until all samples and batches are rendered.

Although the simulator provides a variety of options by default, it can also be extended with custom modules.

Fig. 2 gives an overview over the OAISYS pipeline. For more detail information about the simulator, we refer to the

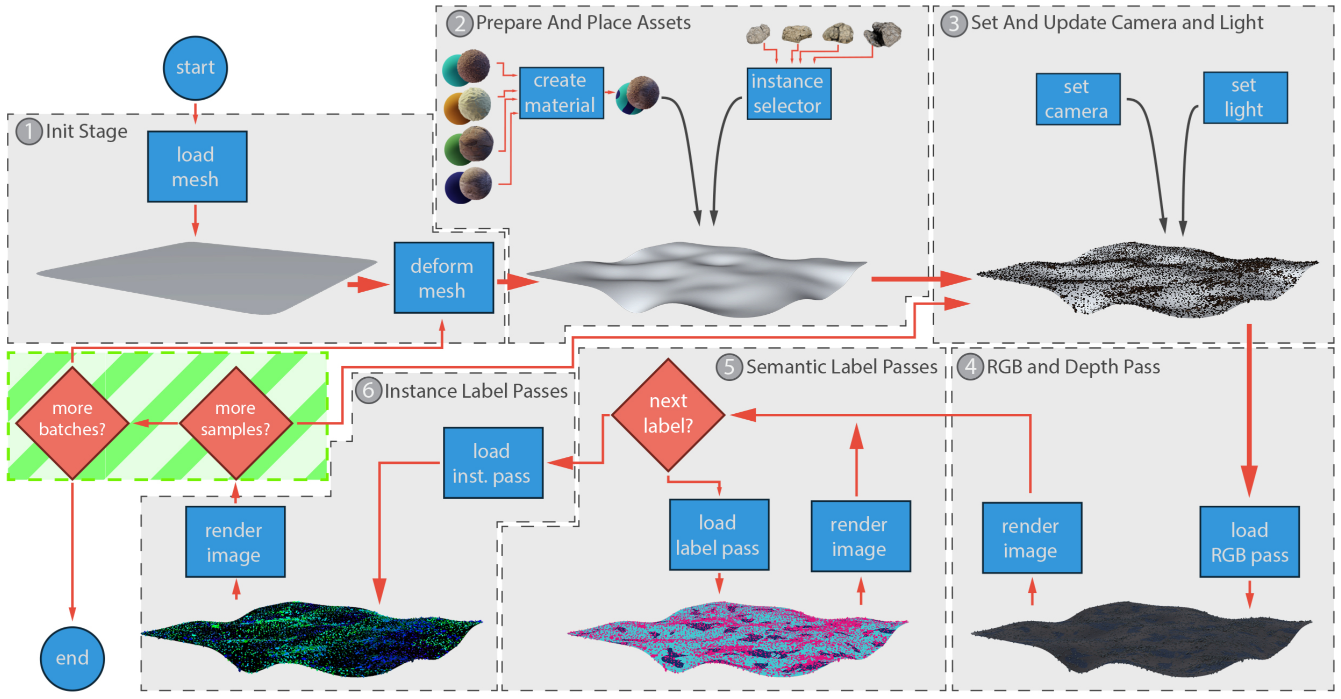

Fig. 2: Basic Flow Diagram of OAISYS. First, the main stage mesh is loaded and deformed 1. Afterwards, the terrain materials are created randomly and mesh assets are picked randomly 2. In 3, the camera and light are set up. In 4, the RGB pass of all assets is loaded and rendered. Afterwards, the semantic labels are loaded and rendered 5. This procedure is repeated on as many semantic levels as desired. In 6 the instance labels are loaded and rendered. Afterwards, either more samples of the sample asset configuration are processed or a new batch ends. The process is completed when the number of desired batches is reached. The green shaded box shows the part which were adapted for the iterative usage.

publication of the simulator [11].

III. Interactive Mode and Further Improvements

In order to use OAISYS for interactive methods, like perception aware path planning or active learning, the core implementation has been extended. In the following, we introduce which improvements were made and why they were important to give OAISYS an interactive option.

A. Interactive Mode

To use the simulator in an interactive mode, it is necessary to send meta data after each sample step to adjust parameters at runtime. The meta data could include the position of the sensor base or the position and strength of a light source. Although the parameters could be provided as a range in which the simulator picks random samples for each batch, it was not possible to change the range during the simulation. As a result, every parameter was predefined previously via a config file. Therefore, it was necessary to extend OAISYS with the option to transfer meta data for each step. In this paper, we focus on transferring poses as meta data. However, the basic principle is not limited to it. Furthermore, we only send meta data for each new requested sample step. The general parameters defined in the config file for the generation of a new batch is not changed at runtime. However, this could be done easily in the future since it is very similar to the adaption done in this paper.

In order to make OAISYS interactive, we modified the parts of the simulator pipeline which is shaded green in Fig. 2 Instead of looping over the number of requested samples, the simulator stops at every sample and waits until the stepping function is called again. Instead of just executing the stepping function of each module once the function is called, we also parse meta data. The meta data is in the format of a dictionary. The dictionary contains a key entry for each main modules of OAISYS. As value, it contains the meta data which is passed to the corresponding module. If data is transferred to the sensor and the render module, then the dictionary contains the keys *sensor* and *render*. Currently, OAISYS features four main modules: *sensor*, *environment*, *asset*, and *render*. If no meta data is supposed to be transferred, the meta data parameter is set to *None*.

A remote process control (RPC) interface based on gRPC [12] is used to interact with OAISYS. By using a RPC framework, the simulator can interact with an application of any environment or language as long as it has a RPC client. The RPC server is running an instance of OAISYS and can call the interactive functions of it. Via the client software, one can send commands to the server, which are executed by OAISYS. The client can request actions, such as creating a new batch or sample and render a scene. Furthermore, it can also receive the results of a rendering of a sample. Since the creation of a new environment and the rendering of a sample can take some time, it was necessary to include the functionality to check the status of the particular action.

B. OAISYS-ROS Bridge

In order to use the interactive mode of OAISYS in a convenient way, we provide an example implementation with the Robot Operating System, ROS [13], a widely used middleware in the robotic community. In order to do that, we implemented a ROS node, which is wrapping the functionalities of the gRPC client. While this paper uses ROS as an example, the RPC interface is not specific to ROS and it is very easy to integrate a new client to interact with OAISYS. This client is performing all interactions with the corresponding RPC server as explained in the previous section. Furthermore, the ROS node takes care of taking the information gathered from OAISYS and transforms it to the equivalent ROS topic. To send messages to the simulator, the ROS node is reacting on ROS requests. In the moment, the ROS nodes pipes the rgb images and depth images into ROS topics. The node can be extended for further custom topics, which shall be relayed to the ROS node. Fig. 1 illustrates the architecture of OAISYS and the middleware interface.

C. Reuse of Stage

OAISYS creates a new random stage element every time a new batch is created. While this is one of the strengths of the simulator, there are several occasions where one might want to reuse a stage and not modify it at all, such as when some pre-processing is applied to the stage or when someone wants to use a custom stage. Another case is if one wants to run an experiment again with a different methods, but the same environment. Therefore, we modified the basic StageCreation Class provided by the vanilla OAISYS version and extended it with the feature to reuse a stage. In order to reuse only the stage and not any objects scattered on the surface, the object has to be cleared.

IV. Experiments

In order to demonstrate the functionality of our extensions we evaluate them with two different experiments. First a task of scene reconstruction and a second task for active view planing. Both are important tasks in robotics and active fields of research.

A. Experimental Setup

A terrain environment resembling an exoplanet was used to generate scenes. The scene mainly consists of an uneven rough surface and randomly placed rocks.

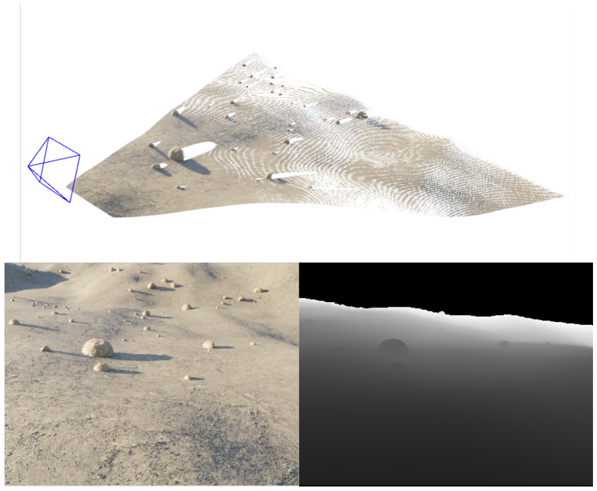
Fig. 3: Simulated RGB-D camera images(Bottom) and pointcloud visualization(Top)

Since OAISYS is capable of rendering complete information of the scene such as appearance, depth and semantics, it is possible to simulate various sensor modalities. OAISYS also supports simulating multiple cameras simultaneously, enabling simulation of realistic camera setups such as stereo camera pairs. Given that the true scene information is passed, different noise models of the sensors can be easily implemented. For the experiments in this paper, a RGB-D camera was simulated by combining the RGB and depth renderings.

A sensor module was implemented which sends a viewpoint pose to OAISYS, waits for the rendering to finish and publishes the RGB and depth renderings in a RGB-D sensor data format. A pinhole camera model was used to generate pointclouds from the depth renderings. Fig. 3 shows an example of the RGB and depth image size of an image size of 640×480 on the bottom, and the resulting point cloud on the top. The pointcloud and vehicle odometry information is then passed to the mapping and planning packages to simulate data coming from a real robotics system.

Using the simulated RGB-D sensor, a scene reconstruction task and an active view planning use case was considered to demonstrate the simulator running interactively for a given robotics tasks.

The experiments were run on a desktop PC, equipped with an Intel Core i7-10700 CPU and a Nvidia Quadro P2200 Graphical Processing Unit. The simulations were run on Ubuntu 18.04 and a client implementation of ROS Melodic was used to interact with OAISYS. While the Average rendering times of each scene was 39.61 ± 2.25 seconds on this setup, the rendering times can vary depending on the configuration and hardware setup.

B. Scene Reconstruction

For the scene reconstruction task, a set of predefined view points were used to demonstrate the scene reconstruction capabilities from scene renderings from OAISYS. This represents a mapping task that is commonly done with aerial vehicles. Voxblox [14], a truncated signed distance field (TSDF) based volumetric mapping framework was used as a map representation since it has been used in many robotic tasks, such as exploration and mapping.

Downward facing view points were generated from a grid pattern covering the terrain to simulate an aerial mapping survey [15]. Fig. 4 shows the reconstructed mesh generated from 30 viewpoints with a voxel resolution of 10 cm in Voxblox. Due to the photorealism of OAISYS, high resolution textures are visible in the reconstructed surface as well as shadows generated from lighting conditions.

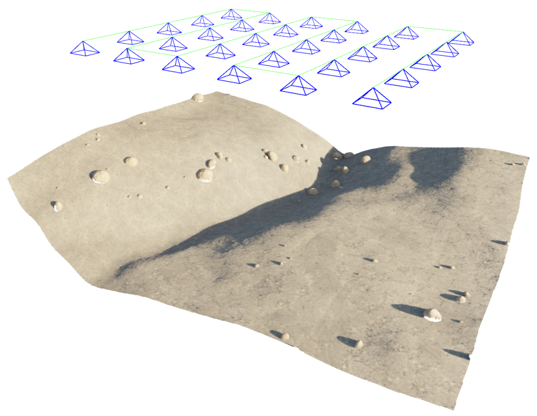
Fig. 4: Visualization of scene reconstruction results using Voxblox [14]

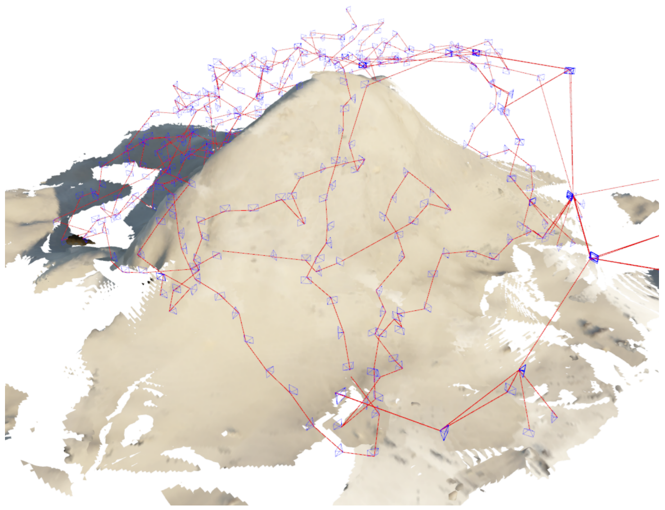
Fig. 5: Reconstruction results of the active mapping use case mapping a steep hill

C. Active View Planning

In order to demonstrate the simulation running interactively with an active planner, we choose the task of exploring an unknown terrain. The goal is to incrementally add the most informative viewpoints by finding the next best view to explore unobserved space within the target volume.

A front facing RGB-D camera attached on an aerial vehicle was considered. Viewpoints were generated sequentially by an active view planner from [16], where most informative view is chosen as the next best view. The planner samples randomly generated viewpoints in the observed free space of the map and evaluates the untility of the viewpoints using a information score based on number of unknown voxels in as well as the surface quality for the surface.

Fig. 5 shows the viewpoints and resulting reconstruction during an active exploration task around a steep hill in the middle of the scene. One unique capability of using a photorealistic rendering pipeline, such as OAISYS, is that selected viewpoints close to the surface can be rendered without compromising image quality. This is because OAISYS is able to render high resolution textures on the surface.

V. Conclusion

In this paper, we introduced an extension of the simulator OAISYS in order to provide an interactive functionality. To make it more accessible for the robotic community, we also provided a ROS interface. We demonstrated the functionality of our work with two experiments. However, the number of potential use cases are greater than that.

Future work might just concentrate on using the meta data for more modules than just the sensor module. Also, it would be beneficial to adopt the gRPC and ROS interfaces in order to provide more message types. Since OAISYS including the described extensions is open source, more functionalities can also be implemented for the interactive version also by other researchers in the community.

ACKNOWLEDGMENT

This work was supported by the Helmholtz Association, project ARCHES (`www.arches-projekt.de/en/`, contract number ZT-0033).

REFERENCES

[1] W. Boerdijk, M. G. Müller, M. Durner, M. Sundermeyer, W. Friedl, A. Gawel, W. Stürzl, Z.-C. Marton, R. Siegwart, and R. Triebel, "Rock instance segmentation from synthetic images for planetary exploration missions," in *Advances in Space Robotics and Back to Earth (IROS WS)*, 2021. [Online]. Available: https://elib.dlr.de/144626/

[2] M. G. Müller, S. Stoneman, I. von Bargen, F. Steidle, and W. Stürzl, "Efficient terrain following for a micro aerial vehicle with ultra-wide stereo cameras," in *2020 IEEE Aerospace Conference*, 2020.

[3] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam, "Blenderproc," *arXiv preprint arXiv:1911.01911*, 2019.

[4] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2018.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *1st Annual Conference on Robot Learning*, 2017.

[6] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conf. on Robot Learning*, 2020.

[7] B. O. Community, *Blender - a 3D modelling and rendering package*, 2018.

[8] Autodesk, INC., "Maya." [Online]. Available: https:/autodesk.com/maya

[9] Epic Games, "Unreal Engine," https://www.unrealengine.com.

[10] Unity, "Unity Engine," https://unity.com.

[11] M. G. Müller, M. Durner, A. Gawel, W. Stürzl, R. Triebel, and R. Siegwart, "A Photorealistic Terrain Simulation Pipeline for Unstructured Outdoor Environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021.

[12] gRPC Authors, "grpc." [Online]. Available: https://https://grpc.io

[13] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[14] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1366–1373.

[15] R. Bähnemann, N. Lawrance, J. J. Chung, M. Pantic, R. Siegwart, and J. Nieto, "Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem," in *Field and Service Robotics*. Springer, 2021, pp. 277–290.

[16] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, 2020.