DOI: 10.2195/lj_Proc_author1_en_yyyymm_01
URN: urn:nbn:de:0009-14-xxxxxx

Time and resource utilization of a voxel-based simulation for indoor multicopters

Zeit- und Ressourcennutzung einer voxelbasierten Simulationsumgebung für indoor Multikopter

***Hendrik Kumpe*[1]**
***Benjamin Küster*[1]**
***Malte Stonis*[1]**
***Ludger Overmeyer*[2]**

[1]IPH – Institut für Integrierte Produktion Hannover gGmbH
[2]Leibniz University Hanover, Institute of Transport and Automation Technology

Constantly increasing production volumes and new challenges in production environments with the same amount of space are forcing manufacturing companies to deal with the planning of production layouts. The problem often is a non-existent or outdated production layout plan. Autonomous multicopters can help by simplifying layout capture. That's why a voxel-based simulation is investigated to develop and train path planning algorithms with and without artificial intelligence. First, the temporal behavior and the resource utilization of the simulation is investigated. Then, the time factor of simulation is compared to real time and what advantages companies and developers have when using it.

[Keywords: UAS, digital twin, simulation, voxel-based, layout design]

Ständig steigende Produktionszahlen und neue Herausforderungen in der Produktionsumgebung bei gleichbleibenden Platzverhältnissen zwingen produzierende Unternehmen, sich mit der Planung von Produktionslayouts auseinanderzusetzen. Das Problem dabei ist oft ein nicht vorhandener oder veralteter Produktionslayoutplan. Autonome Multikopter können dabei helfen und die Layoutaufnahme deutlich vereinfachen. Daher wird eine voxelbasierte Simulation untersucht, um Explorationsalgorithmen mit und ohne Künstliche Intelligenz entwickeln und trainieren zu können. Zunächst wird untersucht, wie sich die Simulation zeitlich verhält. Anschließend wird auf die Ressourcennutzung eingegangen und dargelegt, wie groß der Simulationszeitfaktor im Gegensatz zur Echtzeit ist und welchen Vorteil Unternehmen und Entwickler bei einer Nutzung haben.

[Schlüsselwörter: UAS, digitaler Zwilling, Simulation, voxelbasiert, Layoutplanung]

1 INTRODUCTION

The use of autonomous multicopters is becoming increasingly interesting due to the wide range of possible applications in manufacturing companies. For example, automated inventory or layout planning according to the guideline VDI5200 [1-4] can be significantly simplified by using an autonomous multicopter. Figure 1 shows an example of an autonomous multicopter. The one shown was built for the layout recording use case.

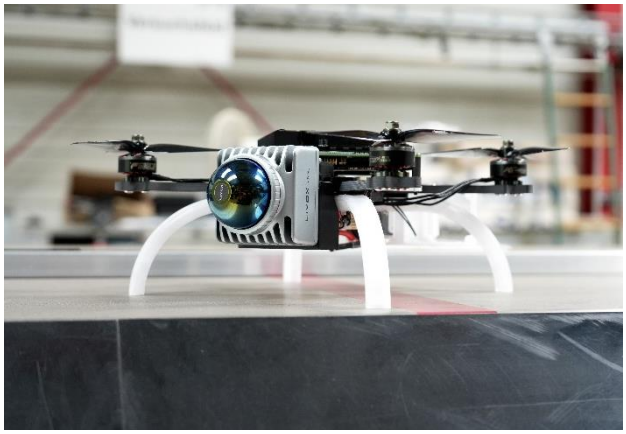

Figure 1. Example for an autonomous multicopter

Because of its enormous impact, the article focuses on the path planning of autonomous multicopters for indoor layout recording. This is a critical quality issue for autonomous exploration in unknown spaces [5]. An approved way to create path planning algorithms and train artificial intelligence is to run simulations [6]. There are two major challenges with current solutions. First, they are computationally intensive because they simulate the entire multicopter, which takes up a lot of computing resources. Second, the simulation environments used are unrealistic because there are straight edges and corners that are easier to use than in reality.

DOI: 10.2195/lj_Proc_author1_en_yyyymm_01
URN: urn:nbn:de:0009-14-xxxxxx

2 SIMULATION

The goal is a simulation software for the development and training of path planning solutions based on approaches with and without artificial intelligence. It can be assumed that all exploration solutions are based on a voxel abstraction. In this context, voxel-based means that the world is represented in cubes, in a 3D grid structure, similar to 2D pixels for images [7]. This type of representation is shown in Figure 2.It represents a small research production hall with an open hall gate in a voxel resolution of 10 cm.

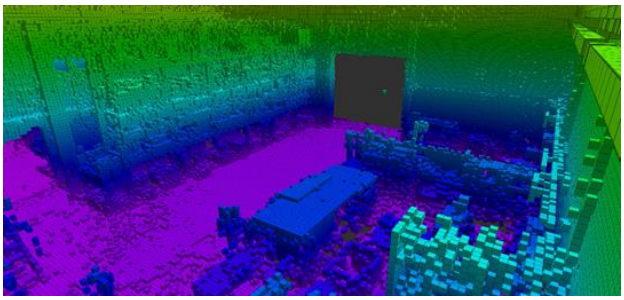

Figure 2. voxel-based environment recorded from production research hall

Based on the assumption that all path planning solutions are voxel-based, many functions and sensors of a fully autonomous multicopter can be neglected. This has a big advantage. The simulation can be used much faster on less powerful computers. On the one hand, this opens up the possibility for a larger group of people to work on the topic of exploration. And on the other hand, significantly faster throughput times. Artificial intelligence training, for example, can be significantly accelerated and even easily parallelized.

Another major advantage is that the simulation world can be a recorded real world. This approach offers the benefit that artificial intelligence and algorithms learn directly in a very realistic world, without clear straight walls, corners, etc. It is also possible for the user to simply record worlds, which has the advantage of saving time in the preparatory work and at the same time adapting the training or test world to the later operational environment – production area. The recording can be made, for example, using the autonomous multicopter, which will later be equipped with the path planning algorithm. In addition, there are many published sample environments that can be used.

Figure 3 shows the functional diagram of the simulation. It shows the standard input and output parameters for an exploration algorithm of an autonomous multicopter with a defined field of view (FOV). Also shown are the internal branches from program start to execution of three central processing unit (CPU) threads. The function of the graphics processing unit (GPU) with the video random access memory (VRAM) is also shown. The voxel map in the use case is implemented using the Octomap repository due to the different general conditions for this article. It is also possible to choose a different approach for other purposes. The approach is applicable to all voxel-based methods.

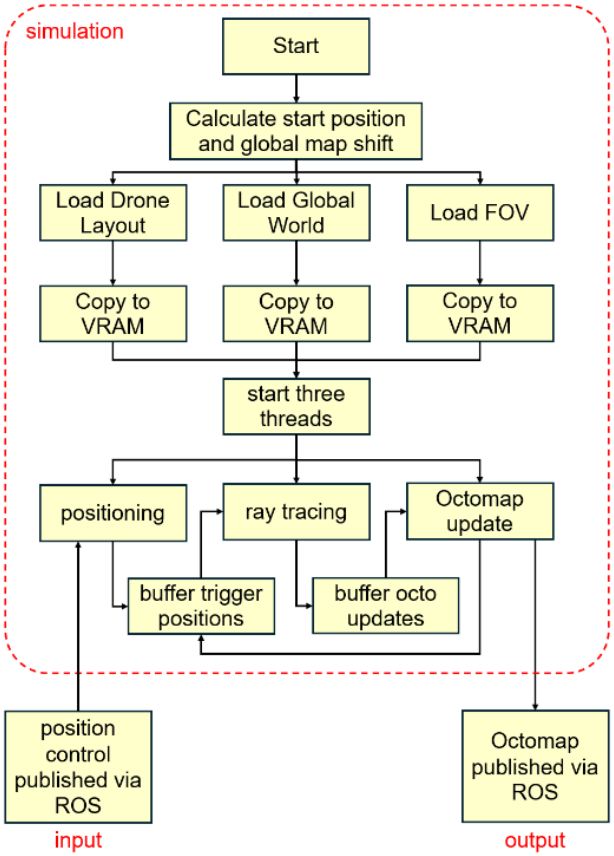

Figure 3. functional diagram of voxel-based simulation

3 STUDIES ON PROGRAM STRUCTURE

In order to examine the simulation, it is necessary to observe the program structure and perform tests. Three worlds and paths of different sizes were used for this article. A small (S), a medium (M), and a large (L) world were used to examine the capabilities as well as the advantages and disadvantages. Worlds S and L correspond to production research halls. World L is a worst case scenario with a completely open space without obstacles. The paths are exact steps through the worlds, which are exactly the same for each test iteration. The FOV is modeled on a Livox Mid360. The replacement with any other laser scanner with a defined FOV is possible with little effort.

DOI: 10.2195/lj_Proc_author1_en_yyyymm_01
URN: urn:nbn:de:0009-14-xxxxxx

Table 1. test world and path characteristics

World	S	M	L
Voxel size [cm]	20	10	5
Steps of the path	771	1,759	1,779
Number of explored voxels	294,275	3,961,478	40,148,882
Number of voxels Multicopter	64	343	2,197
Number of voxels FOV	127,729	127,729	227,003
Real dimensions [m]	18×13×8	18×30 ×8	25×25×8

The tests presented in this article were conducted on a laptop with a 12th-generation Intel i7-12800H x 20 processor, 32GB of random access memory (RAM), and an Nvidia GeForce RTX 3080 Ti laptop graphics card. Tests were also conducted on other laptops for general validity. The results presented in this article were recorded using only the above computer architecture for purposes of comparability.

3.1 DATATYPE FOR GLOBAL MAP

An important investigation concerns the type of data storage for the global simulation map. One general condition is that there is a static world size where voxels are fixed in a grid and can assume four different states. The states are the combinations of occupied/free and explored/not explored. There are several approaches for this top-level storage. One is a structure in which the voxels are stored individually with coordinates and properties in a list. This has the advantage that only as much memory is used as is actually needed. The disadvantage is that you have to iterate to find them. Not necessarily through all elements, but at least through overlayers if they exist, see also Octomap [8]. Another approach is to store the data in an array or a list, where the index is also the coordinate of the grid structure. The disadvantage of this approach is that non-rectangular worlds may require much more memory than necessary. On the other hand, it has the advantage that cells can be accessed directly and multiple cells can be modified in parallel. For the production environment, tests have shown that the memory requirements of the respective implementations differ by only about 5-10% depending on the input world. Further testing also showed that the implementations based on the Octomap approach are difficult to parallelize. In contrast, parallelizing memory accesses to individual cells of an array or a list can be implemented very well without creating unwanted states.

3.2 RAY SIMULATION

The simulation of rays is another important quality aspect of the simulation software, for which there are also many approaches. The simplest is to calculate the rays based on the given FOV and query the relevant voxels accordingly. For example, using the “Fast Voxel Traversal Algorithm for Ray Tracing” method from Amanatides and Woo [9]. The disadvantage of this is due to the fact that it takes a long time to compute the rays over and over again, and that voxels close to the simulated multicopter are hit repeatedly with each ray.

For example, the FOV of the Livox MID 360 requires 227,003 rays in the abstraction (5 cm voxel size) for the simulation, which corresponds to a query frequency of the closer voxels of up to 50,000 times [10]. This time could be saved with another method. For example, a kind of tree structure in which each ray in the form of an outer voxel is an outer branch. This reduces the number of queries per voxel to a minimum, but not to one. The disadvantage of this method is that it cannot be parallelized very well. Tests have shown that a single ray trace with parallelization on the CPU or GPU can be significantly faster than a tree structure with little parallelization.

3.3 PARALLELIZATION

Since the ray tracing algorithm is partially dependent on parallelization, this must also be examined. The available options are no parallelization, CPU and GPU parallelization. The advantage of no or light parallelization is that less memory-intensive methods can be used. The advantage of CPU parallelization is that you can work directly in RAM and there are no unnecessary delays due to data transfer, as is the case with GPU parallelization.

As shown in Figure 4, the parallelization of the GPU with a simple ray simulation has a clear time advantage for each individual case in the tests carried out. The figure shows the comparison between the three variants in the three different worlds with a similar program structure without updating and publishing the Octomap. For the comparison, 20 threads on the CPU and 200 blocks and 200 threads on the GPU were used. The figure shows the advantageous use of the GPU with a computing time of less than 2 ms, while the CPU is well over 14 ms to 300 ms in both single- and multithreading. Furthermore, an initial tendency can be observed as to how the L worlds behave in comparison to the S and M worlds.

DOI: 10.2195/lj_Proc_author1_en_yyyymm_01
URN: urn:nbn:de:0009-14-xxxxxx

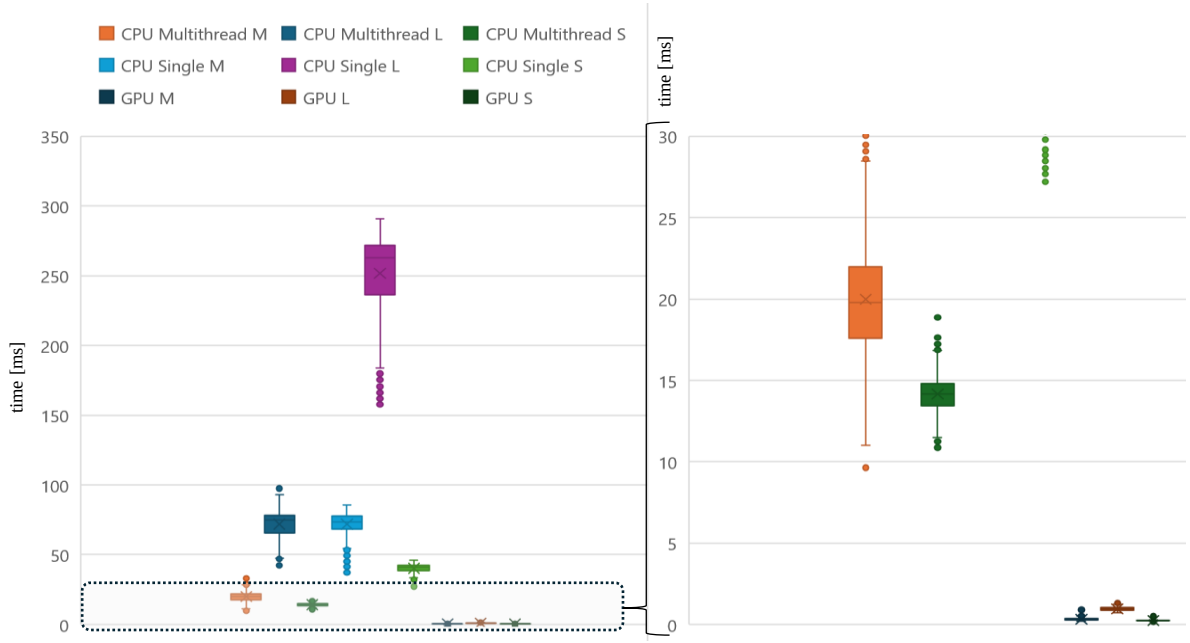

Figure 4. CPU/GPU and single/multithread comparison

3.4 OCTOMAP

In order to continue the investigation and identify potential for improvement, a time analysis of the individual steps is shown in Figure 5. It can be seen which process requires how much time in which step. It is visibly problematic that updating the Octomap as an output in connection with publishing via an ROS topic sometimes takes over 20 times as long compared to the other functions. The reason for this is that the Octomap cannot be updated in parallel.

In addition, the complete Octomap is published each time. This can be improved by, for example, using a different voxel-based map transport technique that only publishes updates. A limitation of this work is the use of Octomap, which means that further implementations need to be investigated in terms of time to speed up the simulation with Octomap.

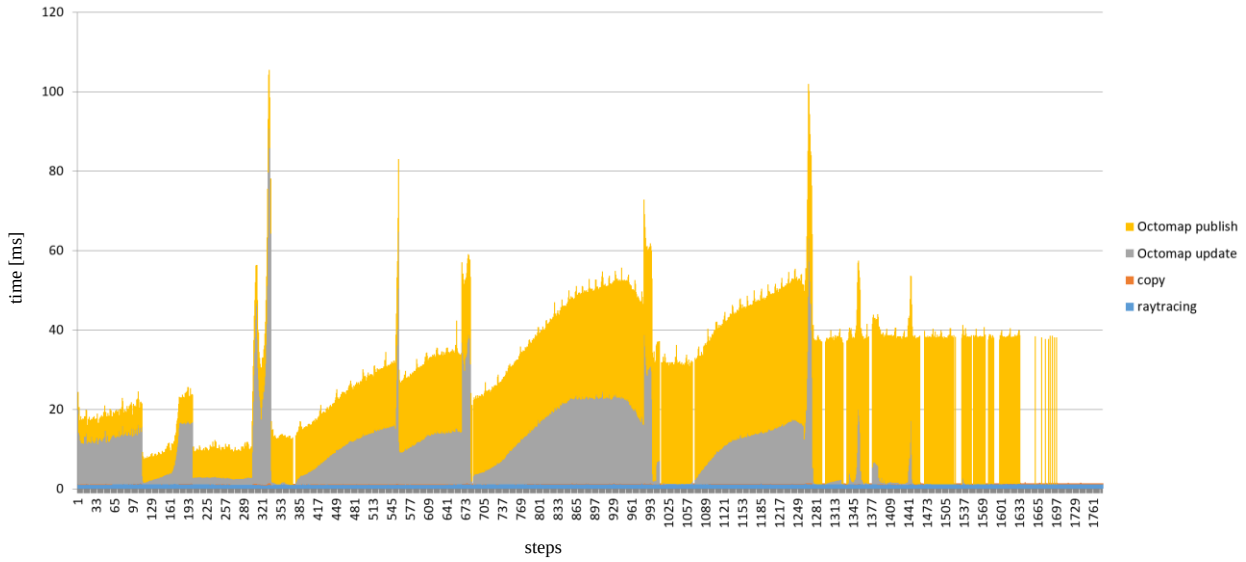

Figure 5. time allocation after first implementation

DOI: 10.2195/lj_Proc_author1_en_yyyymm_01
URN: urn:nbn:de:0009-14-xxxxxx

One approach to investigate is the parallel execution of the Octomap update. As before, raytracing is triggered on the GPU at each step and the result is copied to RAM. The idea, instead of updating the Octomap in singlethreaded mode, the values are written to a buffer in multithreaded mode. A parallel CPU thread takes these values from the buffer and updates the voxel map. This process speeds up step processing. But the processing must always wait until the buffer is empty enough. Tests have shown that a good buffer size is two million elements. The reason this is thought to be an advantage is that areas are often repeated. When this happens, the buffer fills up on a flight through or to an unknown area and empties itself without delay on the return flight. This assumption has been tested and proven in various experiments in worlds S and M.

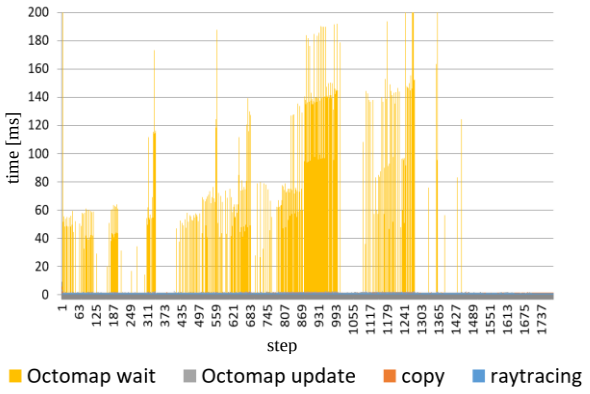

Figure 6. time allocation Octomap parallel

But there is also a disadvantage, which can be seen in Figure 6. In world L, for example, this approach means that more time is needed in total to complete the specified path. The reason for this is the high rate of new voxels per step. It can also be seen that the total time of individual steps with this parallel method for world L takes significantly longer than the steps in Figure 5. The figure also shows that the idea of the concept works and in the end no more time is spent waiting for the parallel Octomap thread.

The method of updating the Octomap directly without a parallel buffer is even approx. 1.38 times faster for world L overall. This is primarily due to the rate at which the map is published. Each time the map is published, the Octomap is prevented from being updated. Which means that with the direct update method the map is only published every 152 ms on average, whereas with the parallel method the map is published every 139 ms. Depending on the application, this behavior can be a compensate for the overall time disadvantage.

Ultimately, a decision has to be made as to which of the two methods is more suitable, depending on the application scenario. For the production and layout recording use case, which is the subject of this article, the buffer and parallel working variant is preferable, as tests with the S and M simulation worlds have shown.

Another way to reduce the time required for Octomap processing is to use the “lazy_evaluation”, “prune” and “updateInnerOccupancy” options. These various functions can be found in the Octomap wiki [8]. The table 4 in the appendix shows the use of the functions. “lazy_eval” and “updateInnerOccupancy” in combination and separately “prune” with “lazy_eval” set to default false and without “updateInnerOccupancy”. The total time required by the simulation for one run is shown. The tests, as shown in Table 4 (appendix), do not show any positive effect on the total times by using the functions.

The result of the tests on Octomap behavior shows that, depending on the conditions and the expected world, the option of direct updating or parallel updating should be used. In the use case of layout recording in production, a parallel update and not the use of “lazy_eval”, “updateInnerOccupancy” and “prune” is recommended on the basis of the tests. For deviating use cases, e. g. for scanning a cave or similar, separate tests must be carried out in order to make a qualified statement.

3.5 ANALYSIS OF STEP TIMES AND TOTAL TIMES

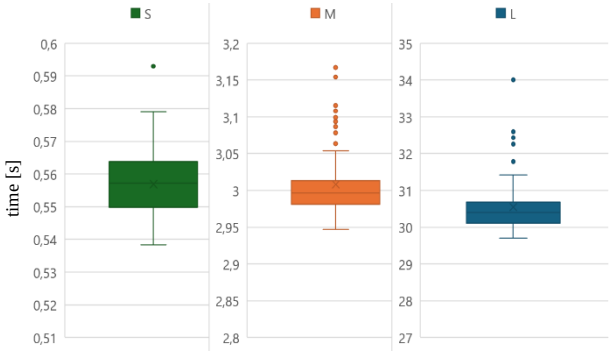

Figure 7. analysis of total times

After examining each program workflow and looking for optimizations, the time results for the three worlds are shown in figure 7 and 8. 100 test runs were performed for each world and the total times are plotted in Figure 7. It can be seen that not all runs take exactly the same amount of time. Shown are the median (solid line inside the box), the interquartile range (outline of the box), the 1.5-fold interquartile range (also called whiskers - the horizontal lines outside the box) and the outliers (points outside).

DOI: 10.2195/lj_Proc_author1_en_yyyymm_01
URN: urn:nbn:de:0009-14-xxxxxx

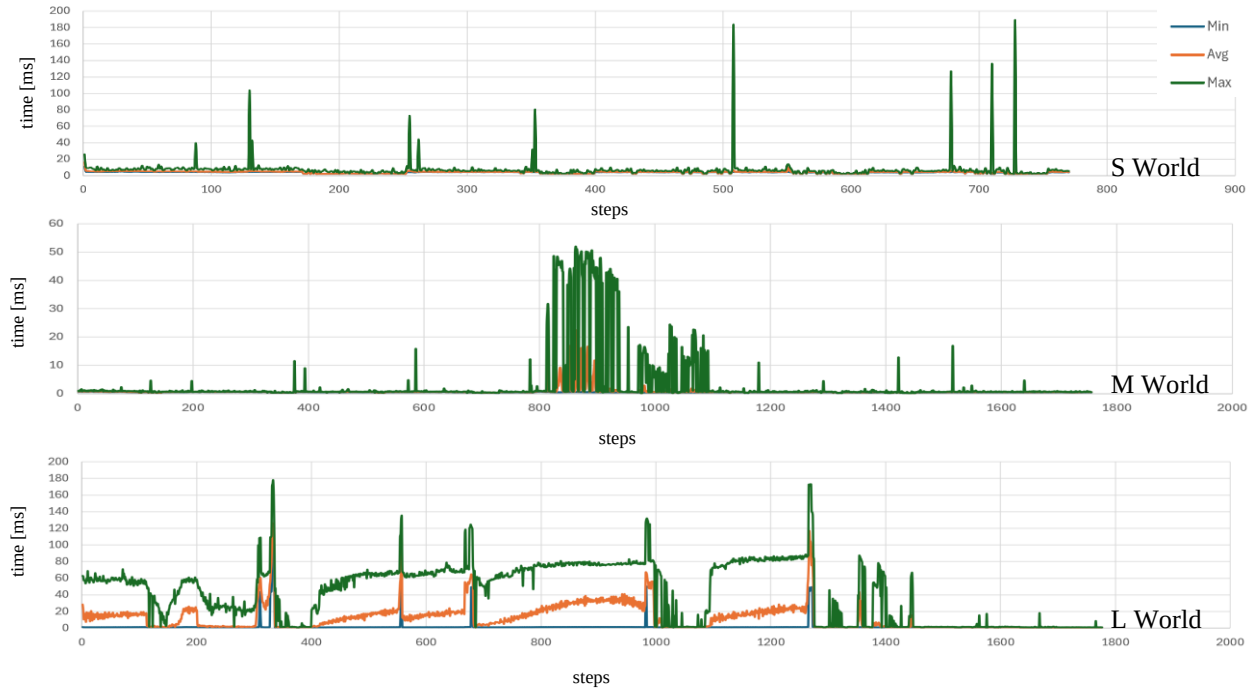

Figure 8. analysis of step times

In addition to the total time, the minimum (blue), average (orange) and maximum step times (green) of the 100 runs of each of the three worlds are shown in Figure 8. Furthermore, the following table shows the average, minimum and maximum step times of each world.

Table 2. step time average, minimum and maximum

World	Average [ms]	Minimum [ms]	Maximum [ms]
S	0.411	0.194	18.912
M	0.724	0.184	52.034
L	12.437	0.802	177.838

The results presented show the maximum speed the simulation can achieve when given the appropriate motion input. With a step average of 0.7 ms, in sum with tolerance 2.1 ms and a voxel size of 10 cm, a simulation time factor of 23.8 can be achieved at a maximum flight speed of 2 m/s in laser scanning mode. At a realistic flight speed of 1 m/s and a step average of 0.7 ms, a factor of 142.9 can be achieved. With this factor, a simulation that would normally take a full week in real time would optimally take only 71 minutes. Realistically, this speed is unlikely to be achieved by any path planning algorithm, but the speed of the simulation still offers significant advantages over conventional simulation solutions.

3.6 Resource performance

It is also essential to consider the resource consumption when examining the times. Table 3 shows the maximum memory resource utilization of the simulation of the three test worlds over 100 tests.

Table 3. analysis of RAM and VRAM usage

World	RAM usage [MiB]	VRAM usage [MiB]
S	153	208
M	324	230
L	845	364

The table shows that only a small amount of memory is required per simulation. For world L, this is 2.1 % with a 364 MB memory usage of the VRAM. On the RAM it is also approx. 2.1 % with a utilization of 845 MiB.

DOI: 10.2195/lj_Proc_author1_en_yyyymm_01
URN: urn:nbn:de:0009-14-xxxxxx

Figure 9 and 10 show additional resource parameters. The GPU utilization of the simulation in % (1), the GPU utilization of the total system in % (2), the power applied to the GPU in W (3), and the CPU utilization of the simulation in % (4) is shown. The values shown are compiled from 100 tests in world L. The simulation ran from the 8th second of the recording and ended 10 seconds before the end of the recording, so that the idle state can be observed in comparison to operation. For the CPU, 100 % is full utilization of one of the 20 threads. For the GPU, 100 % means that all streaming multiprocessors are fully utilized.

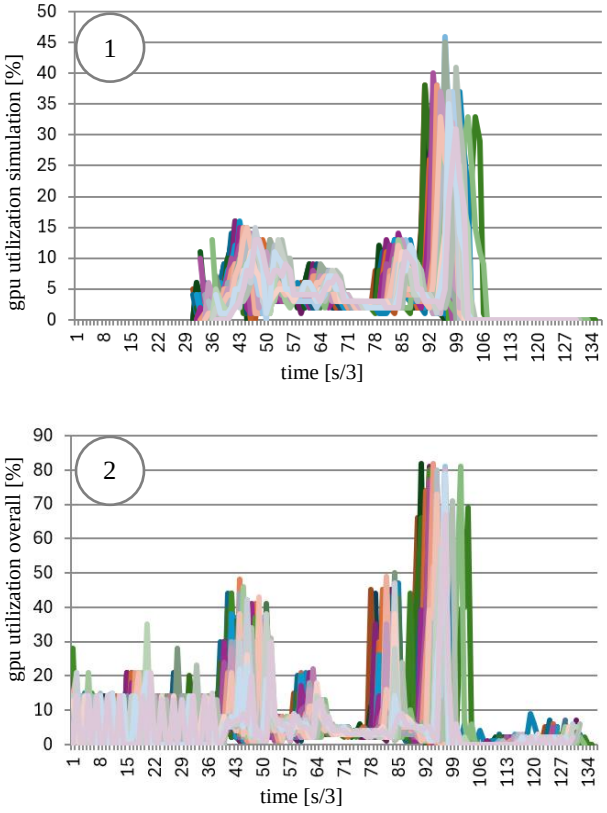

Figure 9. analysis of gpu utilization

Figure 9 and 10 shows that the GPU resource utilization of the simulation does not exceed 50 %, but the total GPU utilization briefly exceeds 80 % at maximum. This means that when running multiple simulations in parallel, care must be taken to ensure that all GPU performance peaks do not coincide. Otherwise, the simulation will slow down due to unavailable GPU resources. Tests have shown that running 4 simulations of world L in parallel with a simultaneous start does not cause any noticeable loss of overall performance. The figure also shows a healthy utilization of up to 150 % of CPU threads and a positive trend in GPU power consumption.

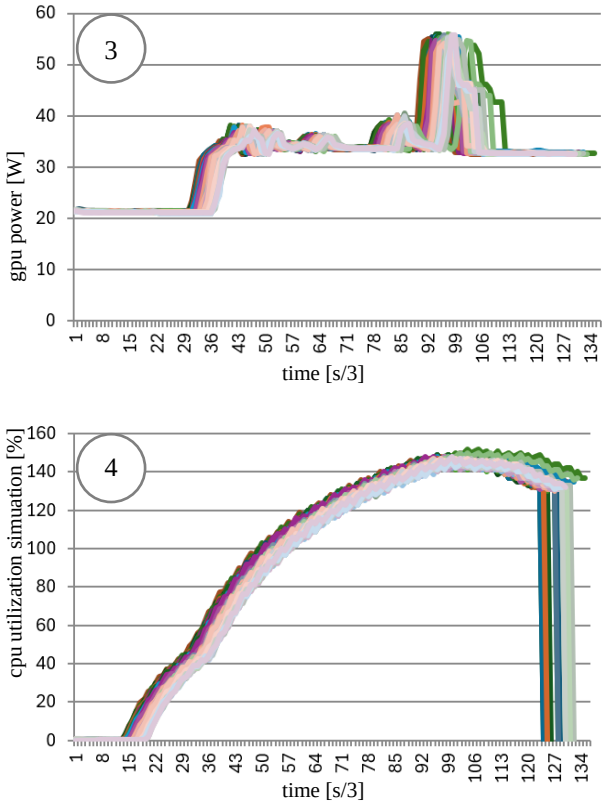

Figure 10. analysis of gpu power and cpu utilization

4 BEHAVIOR OF SIMULATION

4.1 RECORDING OF A VOXEL WORLD

There are several ways to record voxel worlds. The easiest way is to take a multicopter with laser scanner and fly it manual through the own production environment. Another approach would be for the user to use the pool of published indoor laser scans and use them as a simulation world. It is also possible to set up a simulation with e. g. AirSim and scan it to get a voxel world. But note, that this destroys the advantage of using real worlds.

The recordings in your own production areas are carried out quickly. The recordings for worlds S and M took 3:20 minutes and 5:30 minutes. An example recording in a 3,500 m² production hall with a very precise and slow laser scan flight took 27:40 minutes. It should be noted that the time per m² is not linear, identical and easily transferable. It is dependent on the environmental conditions and the production layout.

Octomap, FAST_LIO and a Mid360 were used for the recordings. But it is not enough to consider the pure recording time. It is also important to consider the rework time. This is <30 minutes for the recorded worlds, assuming they are complete.

DOI: 10.2195/lj_Proc_author1_en_yyyymm_01
URN: urn:nbn:de:0009-14-xxxxxx

4.2 SIZE OF VOXEL

A crucial point of investigation is the voxel size in which the worlds are to be mapped. This is because the number of voxels increases quadratically with half the voxel size. One voxel becomes eight at half size and 64 at quarter size. Changing the voxel size also has the effect that the flight speed of the multicopter can be adjusted in the simulation. The reason for this is the constant time for the step calculation and one step is always one voxel size. Increasing the voxel size can also increase the maximum speed of the multicopter. In lieu of directly increasing the multicopter's speed, it is also possible to adjust the simulation time factor.

Therefore, it is necessary to choose the largest possible voxel size in order to save maximum resources and computing time. This is because every increase in voxel size provides benefits to the simulation and path planning algorithms, as well as to the live flight and onboard-processing.

4.3 SIZE OF VOXEL VS MULTICOPTER

Accordingly, the research question is how large can the voxel be compared to the multicopter size plus the minimum safety distance for the production layout recording scenario? To answer this question, it is important to calculate the safety distance correctly. Note that the safety distance can be different for vertical and horizontal. This is important because turbulence caused by the rotors or suck in effects can cause a crash if the multicopter flies too close to obstacles [11-14]. Defining the scenario is important. Because it is necessary to define the minimum size of the openings through which the multicopter will fly. Depending on the answer to this question, the voxel size can be increased or decreased.

To answer the question, it must be determined how wide or high the minimum fly-through width or height may be (d_{min}) and how wide the maximum diameter of the used multicopter is (d_{copter_max}).

The following values are assumed for the scenario considered in the article:

$d_{min} = 0.8\,m$

$d_{copter\_max} = 0.4\,m$ - Width of the multicopter plus a safety distance of 50 mm

A method for calculating the maximum voxel width has been developed and proposed as follows.

$$l_{voxel} = \frac{d_{min} - d_{\text{copter }\_max}}{2} = \frac{0.8\,m\ -\ 0.4\,m}{2} = 20\,cm$$

In the calculation for the scenario shown, the maximum width or height that can be flown through is twice the height of the multicopter. This is only for this scenario and does not necessarily have to be twice the height.

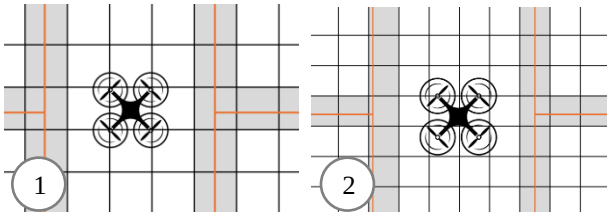

Figure 11. voxel size analysis with 20 cm (1) and 15 cm (2)

Figure 11 shows two scenarios to confirm the calculation. The orange lines represent walls that are 80 cm apart. The gray voxels are occupied voxels and the white voxels are free voxels. The multicopter has a diameter of 35 cm (1) and 45 cm (2) with a safety margin of 50 mm each. It can be seen that the voxel size of 20 cm (1) and 15 cm (2) leaves exactly the right corridor of voxels. This confirms the formulated calculation for the maximum voxel width.

5 CONCLUSION AND OUTLOOK

A voxel-based simulation environment for training and development of exploration solutions with and without artificial intelligence offers enormous possibilities. This has been demonstrated in this article through experiments and studies on speed and resource consumption. For multicopters with a diameter of 0.4 m, a voxel size of 20 cm is required, under the condition of a minimum flight width or height of 0.8 m. A research hall can be completely explored within a few seconds or even less than a second, provided that the exploration algorithm can maintain this speed. The voxel-based approach allows more users to produce results faster and can pave the way for a major advantage in layout capture and factory planning.

The next step is to investigate the advantages of using voxel-based real worlds in simulation and the degree of parallelization that can be achieved even when fully controlled by path planning algorithms. In addition, the use of Octomap can be questioned and, if necessary, a better or alternative solution can be found or developed.

6 FUNDING

This project, AIMS5.0, is funded by the Chips Joint Undertaking and its members, including additional funding from the German Federal Ministry of Education and Research (BMBF) under grant agreement no. 101112089.

LITERATURE

[1] VDI 5200 Part 1 - Factory planning - Planning procedures; 2011.

[2] VDI 5200 Part 2 - Factory planning - Morphological model of the factory for the target definition in the factory planning; 2016.

[3] VDI 5200 Part 3 - Factory planning - Model for the design of global production networks; 2016.

[4] VDI 5200 Part 4- Factory planning - Enhanced economic evaluation within factory planning; 2016.

[5] L. M. González de Santos, E. Frías Nores, J. Martínez Sánchez, and H. González Jorge, 'Indoor path-planning algorithm for UAV-based contact inspection', Sensors (Basel), vol. 21, no. 2, p. 642, Jan. 2021.

[6] A. Seel, F. Kreutzjans, B. Küster, M. Stonis, and L. Overmeyer, 'Dueling Double Deep Q-Network for indoor exploration in factory environments with an unmanned aircraft system', in 2023 22nd International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 2023.

[7] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, 'OctoMap: an efficient probabilistic 3D mapping framework based on octrees', Auton. Robots, vol. 34, no. 3, pp. 189–206, Apr. 2013.

[8] K. M. Wurm and A. Hornung, 'OctoMap'. [Online]. Available: https://octomap.github.io/. [Accessed: 30-Jul-2024].

[9] Amanatides, John & Woo, Andrew. (1987). A Fast Voxel Traversal Algorithm for Ray Tracing. Proceedings of EuroGraphics. 87.

[10] 'Livox Mid-360'. [Online]. Available: https://www.livoxtech.com/de/mid-360. [Accessed: 30-Jul-2024].

[11] N. S. Jamaluddin, A. Celik, K. Baskaran, D. Rezgui, and M. Azarpeyvand, 'Experimental analysis of a propeller noise in turbulent flow', Phys. Fluids (1994), vol. 35, no. 7, Jul. 2023.

[12] H. Choayb, 'Proposition of a propeller shape: A numerical study of its performance', Arab. J. Sci. Eng., vol. 49, no. 2, pp. 2119–2142, Feb. 2024.

[13] R. Dahlstrom, 'Flying safe: How to operate drones near buildings and other structures', Default, 12-Nov-2021. [Online]. Available: https://connect.comptia.org/blog/flying-safe-how-to-operate-drones-near-buildings-and-other-structures. [Accessed: 30-Jul-2024].

[14] Z. Zhang, C. Xie, W. Wang, and C. An, 'An experimental and numerical evaluation of the aerodynamic performance of a UAV propeller considering pitch motion', Drones, vol. 7, no. 7, p. 447, Jul. 2023.

Hendrik Kumpe, M. Sc (*1997) studied electrical engineering and information technology at Bielefeld University of Applied Sciences and Arts and at the Leibniz University Hannover. Since June 2022, he has been working at IPH – Institut für Integrierte Produktion Hannover gGmbH as a project engineer in the field of production automation.

Address: IPH – Institut für Integrierte Produktion Hannover gGmbH, Hollerithallee 6, 30419 Hannover, Germany, Phone: +49 511 27976-224, Fax: +49 511 27976-888, E-Mail: kumpe@iph-hannover.de

Dr.-Ing. Benjamin Küster (*1988) studied industrial engineering at the Leibniz University Hannover. From November 2014 to August 2017, he worked as a project engineer at the IPH – Institut für Integrierte Produktion Hannover gGmbH in the department of production automation. In 2020, he received his doctorate with a thesis on "Automated quality assessment of 8D reports by methods of computational linguistics". Since September 2017 Benjamin Küster is leader of the department of production automation.

Dr.-Ing. Malte Stonis (*1979) studied mechanical engineering at the Leibniz University Hannover with a focus on vehicle systems and biomedical engineering. He has been working at IPH – Institut für Integrierte Produktion Hannover gGmbH since 2006, initially as a project engineer in the field of process technology and from 2008 as head of department. In 2011, he received his doctorate with a thesis on "Multidirectional forging of flat aluminium parts". Since September 2016 Malte Stonis is coordinating managing director of the IPH.

Prof. Dr.-Ing. Ludger Overmeyer (*1964) studied electrical engineering at the University of Hanover between 1984 and 1991. In 1996 he finished his doctorate in mechanical engineering at the University of Hanover. From 1997 to 2001 he worked as project manager, division manager and head of research and development at Mühlbauer AG in Roding. Since 2001 Ludger Overmeyer is Professor of Transport and Automation Technology Institute of Leibniz University Hannover.

Address: Institute of Transport and Automation Technology, Leibniz University Hannover, An der Universität 2, 30823 Garbsen, Germany Phone: +49 511 762 2503, Fax: +49 511 762-4007, E-Mail: ludger.overmeyer@ita.uni-hannover.de

DOI: 10.2195/lj_Proc_author1_en_yyyymm_01
URN: urn:nbn:de:0009-14-xxxxxx

7 APPENDIX

Table 4. Evaluation of benefit when using Octomap function

Test	World	total time [s]
lazy_eval=false no updateInnerOccupancy	S	1.5
	M	32.7
	L	84.1
lazy_eval=true no updateInnerOccupancy	S	2.5
	M	102.1
	L	843.3
lazy_eval=false updateInnerOccupancy	S	2.3
	M	62.4
	L	157.7
lazy_eval=true updateInnerOccupancy	S	3.8
	M	189.0
	L	1,617.2
never execute prune	S	1.6
	M	34.0
	L	90.4
execute prune each update	S	2.0
	M	50.1
	L	124.4
execute prune after each 1000th update	S	1.6
	M	33.8
	L	90.4
execute prune after each 100 ms	S	1.6
	M	33.8
	L	89.9
execute prune after each 1000 ms	S	1.6
	M	33.5
	L	90.2