2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Automatic tool for Gazebo world construction: from a grayscale image to a 3D solid model

Bulat Abbyasov[1], Roman Lavrenov[1], Aufar Zakiev[1], Konstantin Yakovlev[2], Mikhail Svinin[3] and Evgeni Magid[1]

Abstract— **Robot simulators provide an easy way for evaluation of new concepts and algorithms in a simulated physical environment reducing development time and cost. Therefore it is convenient to have a tool that quickly creates a 3D landscape from an arbitrary 2D image or 2D laser range finder data. This paper presents a new tool that automatically constructs such landscapes for Gazebo simulator. The tool converts a grayscale image into a 3D Collada format model, which could be directly imported into Gazebo. We run three different simultaneous localization and mapping (SLAM) algorithms within three varying complexity environments that were constructed with our tool. A real-time factor (RTF) was used as an efficiency benchmark. Successfully completed SLAM missions with acceptable RTF levels demonstrated the efficiency of the tool. The source code is available for free academic use.**

I. INTRODUCTION

Robot simulations nowadays provide significant support in testing new algorithms for robotic systems within a broad variety of tasks. One of these tasks is a mobile robot navigation and its most interesting and important part, which is an autonomous navigation [1]. In the navigation terms, a map is a simplified representation of an environment that is built from captured by a robot sensory data, e.g., with a help of on-board cameras or a laser range finder (LRF), and further applied for navigation purposes. During a mapping procedure, a robot moves around an environment and creates a map. However, an effective use of a map for navigation is questionable if the robot doesn't locate itself within this map at every moment. Thus, a correctly constructed map improves localization accuracy, while proper localization process ensures accurate positioning of a robot within the map.

New navigation methods and approaches should be carefully modelled and tested before integrating them into a real robot software. For modelling and testing, we use Robot Operating System (ROS) framework [2] and Gazebo simulator. Testing a virtual robot in Gazebo is an efficient, inexpensive and independent on real hardware availability way to verify new concepts and methods correctness and applicability with existing robots [3].

A navigation algorithm validation requires simulated landscapes and objects that provide a high realism level, a

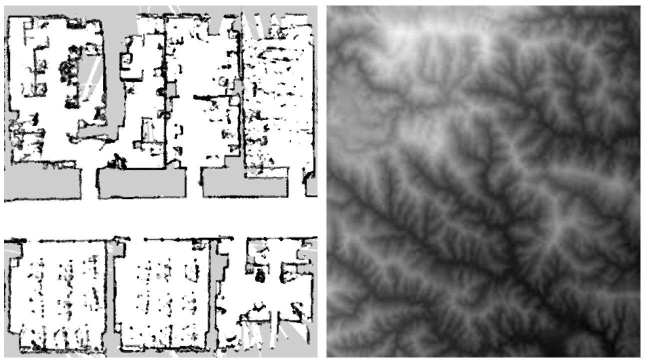

(a) (b)

Fig. 1: (a) An occupancy map that was built from 2D Hokuyo LRF sensory data at Laboratory of Intelligent Robotic Systems. The map encodes occupancy data where white pixels represent free cells, black pixels are occupied cells and gray pixels are not yet explored; (b) a grayscale image, which represents uneven mountain landscape

good approximation of real-world scenarios and are easy to construct. Such requirements are fulfilled by employing real data for map construction, e.g., embedding LRF data (Fig. 1(a)) or a grayscale map of a terrain (Fig. 1(b)) into an occupancy grid map [4], [5] within Gazebo environment. The basic idea of an occupancy grid is to represent a map as a field of random variables at an evenly spaced grid. Each cell is a random variable that models an occupancy with 0 value for an occupied cell, 1 for a free cell and 0.5 for an unknown cell (Fig. 1(a)) [6].

Our research team experience in modelling various robots and algorithm testing in Gazebo environment demonstrated that significant time and efforts are spent for creating testing environments [7]. Therefore, a tool that quickly converts 2D input data into a 3D Gazebo world would have a high value. Our new convenient tool generates a Collada format model, which is supported by Gazebo framework and thus a resulting map could be directly imported into Gazebo as a mesh and used for various purposes. To validate the quality of Gazebo worlds, which were created with the tool, we ran several SLAM algorithms with Husky mobile robot in these worlds and compared their performance using a real-time factor (RTF) as an efficiency benchmark. RTF shows a ratio of calculation time within a simulation (simulation time) to execution time (real-time) for a particular task. For example, if it takes 4 seconds to compute 1 second

[1] Authors are with Laboratory of Intelligent Robotic Systems (LIRS), Higher Institute for Information Technology and Intelligent Systems, Kazan Federal University, 35 Kremlyovskaya street, Kazan, 420008, Russian Federation. The corresponding author is Roman Lavrenov `lavrenov.roman@gmail.com`
[2] Moscow Institute of Physics and Technology, Moscow, Russia
[3] Ritsumeikan University, Noji Higashi 1-1-1, Kusatsu, 525-8577, Japan

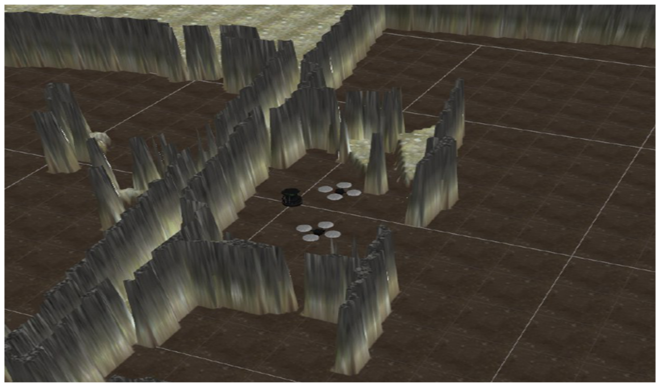


Fig. 2: Successfully imported heightmap that was verified with two quadrotors and Turtlebot robot

of execution in a simulation, RTF equals 0.25. The larger is RTF value, the more efficient is a simulation. Among a large number of SLAM techniques that exist nowadays [8], we selected HectorSLAM [9], Gmapping SLAM [10] and Google Cartographer [11] algorithms that are most widely used for mobile ground robots navigation in uneven terrains.

The main contribution of this paper is LIRS World Construction Tool (LIRS-WCT) - a new efficient tool for creating a realistic 3D virtual environment of Gazebo world from an arbitrary 2D image or 2D LRF data. A practical usability of worlds, which are created by LIRS-WCT, was evaluated by running the three selected SLAM algorithms with Husky robot and different complexity maps. Moreover, the new tool performance was compared to the only open source Gazebo world construction tool [12], [13] and it demonstrated a significantly better performance in terms of RTF.

II. RELATED WORK

To the best of our knowledge at the moment the only open source project that allows converting grayscale images into Gazebo-compliant 3D map is a tool by Lavrenov and Zakiev [12], [13]. It provides an automatic image processing before importing it into Gazebo simulator as a heightmap and allows for noise filtering, original image cropping and resizing, and automatic converting of images to grayscale. The tool receives an image as an input (Fig. 1 demonstrates input examples) and generates a SDF-world file, which describes objects and environment in XML format [14]. Figure 2 shows the world, which is stored as a SDF-element heightmap, with Turtlebot robot [15] and a pair of Hector quadrotor UAVs [16].

This tool is based on a heightmap technique method. Our experience using heightmaps demonstrated that Gazebo performance is very slow when heightmaps are used with a robot model being spawned in a world and a real-time factor (RTF) of the simulation decreases rapidly to zero. Tests were performed with Intel(R) Core(TM) i5-3210M 2.50GHz PC with DDR3 6 GB RAM and AMD Radeon HD 7500M/7600M Series 2 GB graphic card. We believe that low performance is caused by the default physics engine of Gazebo (ODE), which has to compute complex collision objects. Moreover, increasing a number of objects with the heightmap technique crashes the simulator as RTF rapidly converges to zero. An additional restriction of the original tool [12], [13] is imposed on input image size. To use a grayscale image as Gazebo heightmap, both height and width of the image must be equal to $2^N + 1$ pixels, where N is a natural number. Thus, only square maps could be created.

III. SOLID MODEL APPROACH

In this paper we propose to improve the original approach with a solid model strategy. Instead of computing complex collision objects, which are created from a heightmap, Gazebo's physics engine operates with a single monolith ready-to-use terrain model. Using an integral solid model of environment and objects is preferable because it simplifies calculations that are performed by Gazebo physical engine, including heavy collision computations. This section describes a new tool that converts a grayscale image into a 3D model in Collada format. Several new features were added to the original tool, including texture application, height adjustment, dimensions scaling and X-axis rotation features. The tool automatically performs three sequential steps, which are described below: a grayscale image conversion into a 3D model, applying a texture to the 3D model, an automatic world file generation. Finally, we validated resulting worlds by operating different models of ground mobile robots within the created worlds.

A. Grayscale image conversion to a 3D model

This stage is responsible for converting a grayscale input image into a 3D model of $.dae$ format. It also allows for scaling of the input image dimensions. We employed source code of HMSTL tool [17] and LIBTRIX library [18], and used Application programming interface (API), which is provided by ASSIMP library [19].

Our tool converts a grayscale image into an STL triangle mesh in two steps:

1. Transformation of a grayscale image into a heightmap. A heightmap is a 2D array, where elements store height information about each point of an area. This stored information is interpreted as a "height from the ground level" of a surface and sometimes is visualized as a grayscale image intensity, where black color represents a minimum height and a white color represents a maximum height. A standard RGB 8-bit image can only show 256 values of gray and hence only 256 different heights.

2. Conversion of a heightmap into STL format file. STL file provides a triangular representation of a 3D geometrical surface, where the surface is represented by an ordered number of small triangles. Using LIBTRIX library's converting algorithm, we generate STL file from the heightmap. In addition to the conversion, we pass a Z-scale coefficient, which influences the created model height.

Open Asset Import Library (ASSIMP) is a library that imports and exports various 3D model formats including a scene post-processing in order to generate missing render data. The library loads 40+ 3D file formats into a unified

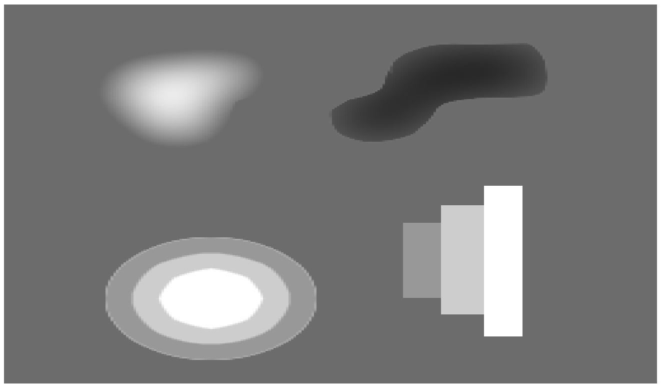


Fig. 3: A prepared for conversion gradient map

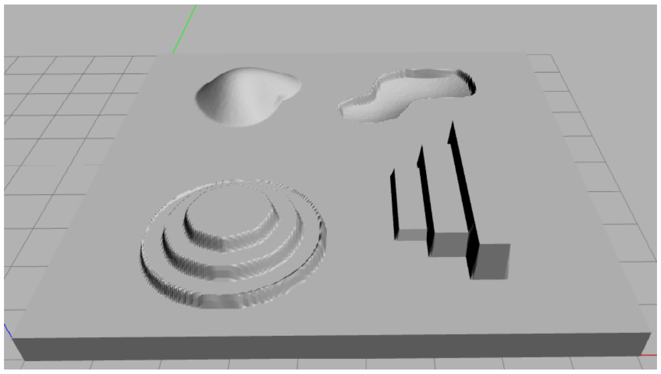


Fig. 4: A converted untextured model in Collada format

data structure, which allows accepting various format files into a single data structure for further processing. In this step, we used the library's C++ API for loading a STL model and exporting it as a Collada file. Likewise, we could adjust a model size or x-axis rotation angle. Figures 3 and 4 demonstrate a transformation of a gradient map into an untextured model of Collada format.

B. Applying a texture

In order to reproduce visual properties of surfaces or objects the tool applies a user-selected texture. A process of imposing an image on a model is performed with UV-mapping [20]. The mapping projects a 2D image onto a 3D model's surface while carrying out compliance between the object surface coordinates (X, Y, Z) and texture coordinates (U, V); U and V take values within [0,1] interval. Since for a 3D world creation we typically use urban images with prevailing orthogonal side orientations, we had selected a cube projection of texture coordinates so that each face corresponds to an entire surface of the texture. The advantages of this method include a small level of distortions and a relatively seamless appearance of a textured model. To generate UV-coordinates, 3D normal vectors for all vertices of the model are constructed and a vertex orientation is verified, which further will be considered for lighting and projection of texture coordinates. The UV-coordinates are normalized and their correspondence with a model vertex is established.

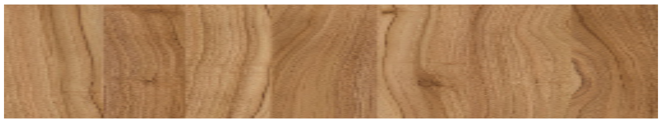


Fig. 5: A texture that was applied to a model in Fig. 4

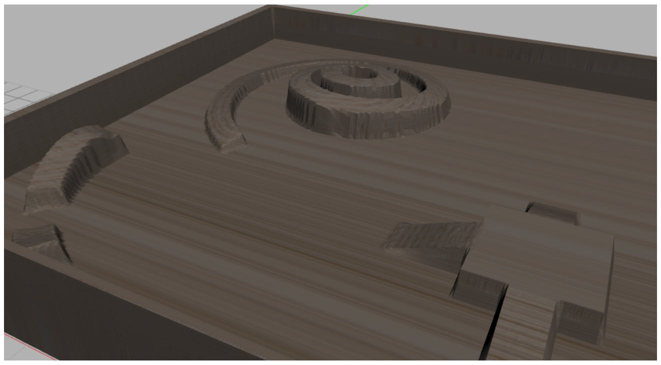


Fig. 6: A textured model

C. Automatic world file generation

Automatic world file generation function takes a generated model and creates a world that is populated with this model. We use SDF-element mesh, which requires a 3D model file (a generated mesh file) for environment construction. Next, this generated world is ready for loading into Gazebo simulator. An example of a Husky robot operating within a generated by our tool world is shown in Fig. 7.

D. World validation

To evaluate created with our tool worlds, we checked the worlds' performance within collision detection and sensory data processing tasks. The choice of particular robot models was influenced by our desire to evaluate popular in ROS community robots of a different size, with different controllers and types of onboard sensors. Therefore, we used Husky (Fig. 7), Jackal (Fig. 8, top image) and Turtlebot (Fig. 8, bottom image) robots [21]. Collision detection demonstrated a correct interaction of each robot with a landscape while a robot traversed the environments without

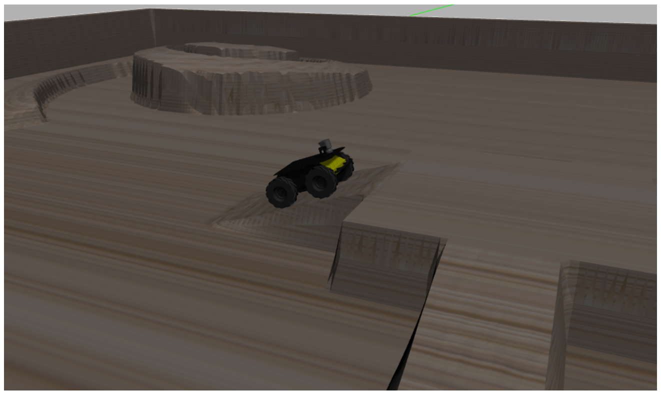


Fig. 7: A generated world with a Husky robot in Gazebo simulator

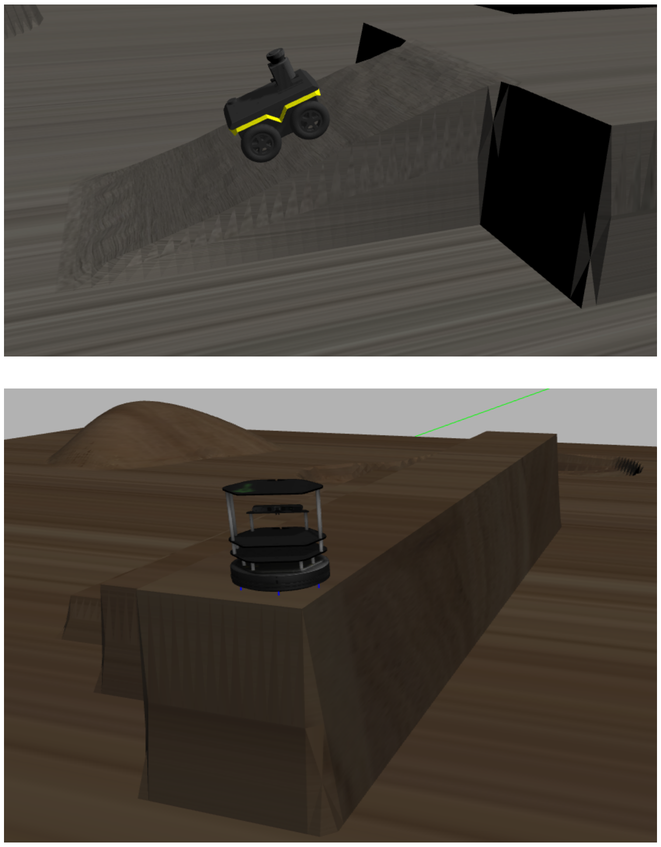


Fig. 8: Collision detection. Jackal robot climbs a 45 degrees slope (top) and a Turtlebot robot on a barrier top (bottom)

going through obstacles or revealing any other failures.

Sensory data processing was verified with SLAM methods using a LRF. A mission of a robot was to traverse a 3D environment that was constructed with our tool (Fig. 9, left) and to build a map using three various SLAM algorithms: GMapping, HectorSLAM, and Cartographer. The algorithms are described in more details in Section IV. All three algorithms successfully constructed a map in a form of an occupancy grid and achieved an average RTF of 0,38 with a Husky, 0,52 with a Turlebot and 0,47 with a Jackal robot. Figure 9 (right) demonstrates the resulting map as a grayscale image in PGM format that was obtained by the Husky robot with HectorSLAM algorithm.

IV. LIRS WORLD CONSTRUCTION TOOL

LIRS World Construction Tool (LIRS-WCT) could be used with an intuitive graphical user interface (Fig. 10) or in a command line mode. The command line mode has mandatory parameters (M), optional parameters (O) and optional parameters with default values (D), which are described in details in Table I.

V. VIRTUAL EXPERIMENTS

A. Experimental configuration

We compared the worlds that were generated by *Gazebo_heightmap_preparation* tool (GHPT) [12] and by

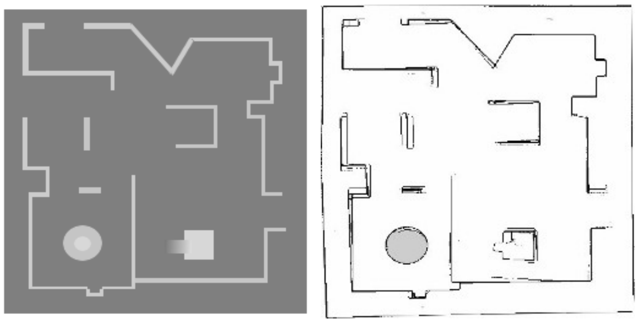
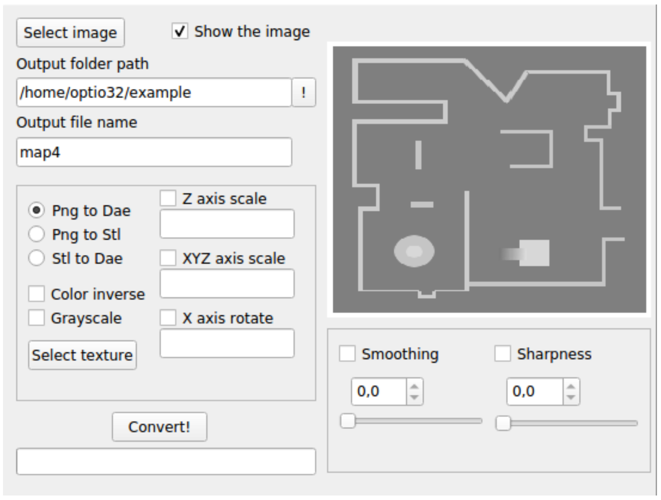


Fig. 9: Top view of the original 3D environment (left) and a sensory based map that was constructed by the Husky robot (right)



[✓] Show the image

Output folder path: /home/optio32/example !

Output file name: map4

[● Png to Dae] ○ Png to Stl ○ Stl to Dae

☐ Z axis scale

☐ XYZ axis scale

☐ X axis rotate

☐ Color inverse

☐ Grayscale

☐ Smoothing 0,0

☐ Sharpness 0,0

Fig. 10: Graphical User Interface mode of LIRS-WCT

Table I. LIRS-WCT input parameters description

Parameter	Type	Description
input image filepath	M	Set an absolute path to a source grayscale image
output model filename	M	Set a filename for a generated mesh model
output folder path	O	Set a folder for output files
color inverse option	O	Using a preliminary image color inverse
png to dae conversion	D	Convert a grayscale png image to dae model
stl to dae conversion	O	Convert stl file model to dae model
png to dae conversion	O	Convert png file to stl model
smoothing	O	Set a smoothing coefficient for image defects removal
sharpness	O	Set a sharpness coefficient for image clarity after using a bluring
z axis scale	O	Set z-axis scale factor (affects output model height)
grayscale option	O	Convert an arbitrary image to grayscale
scale	O	Set xyz-axis scale factor (affects output model dimensions)
x axis rotate	D	Set x-axis rotate angle (default value is -90 degrees)

our new LIRS-WCT tool. Figure 11 demonstrates examples of 3D environments that were generated from the same input by GHPT with a heightmap technique (top image) and by our tool with a monolith-model technique (bottom image).

Effectiveness of the navigation task within three different environments, which were generated by the two approaches, was compared using RTF indicator of SLAM algorithms execution. Real-time and simulation time values were obtained from Gazebo topic *world_stats*. The three different environments have a varying complexity of a terrain: simple, medium, and advanced (Fig. 12). Six virtual environments were generated, applying GHPT and LIRS-WCT tools for the

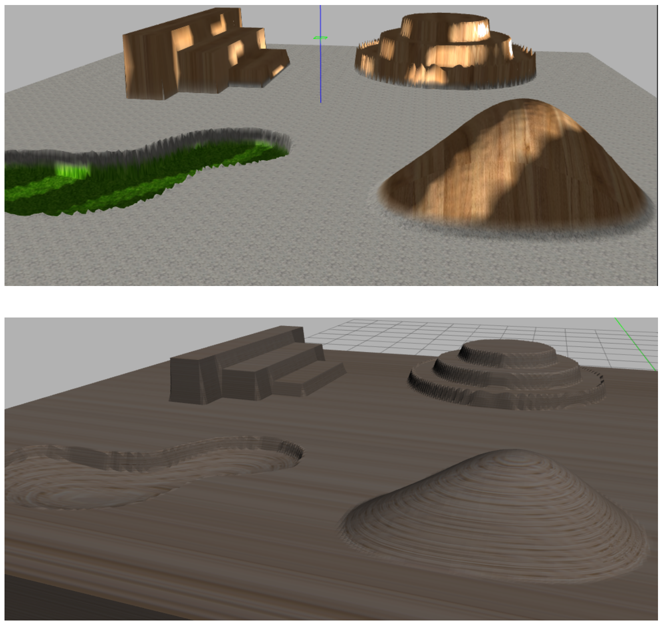


Fig. 11: A 3D landscape example. Gazebo engine uses heightmap technique to create a 3D terrain (top). A monolith-model is constructed by LIRS-WCT and loaded by Gazebo engine (bottom)

three input terrain images. For our simulation experiments in Gazebo, we used Husky robot in a navigation task within the six worlds. Husky is a fully supported in ROS/Gazebo unmanned ground vehicle that was designed for outdoor environments.

SLAM algorithms that were selected as a benchmark are ROS GMapping [10], ROS HectorSLAM [9] and ROS Cartographer algorithms [11]. GMapping package provides a laser-based SLAM (with ROS node $slam\_gmapping$), which employs laser scans and odometry to build a 2D occupancy grid map and to update it as a robot moves through the environment. HectorSLAM (using $Hector\_mapping$) is an approach that could be used without odometry and with robots that exhibit roll/pitch motion (of a sensor, a robot or both); it leverages a high update rate of modern LIDAR systems and provides 2D pose estimates at a scan rate of the sensors, being sufficiently accurate for many real-world scenarios. Finally, Google Cartographer (using ROS Cartographer algorithm) provides a real-time SLAM in 2D and 3D across multiple platforms and sensor configurations.

B. Results

All three SLAM algorithms with a Husky robot failed when they were tested in the worlds that were created with GHPT while demonstrating a similar behavior: after SLAM execution start, Gazebo demonstrated RTF of around 0,97, which exponentially went down to 0.09, 0.05 and 0.011 for a simple, medium and advanced map respectively. With such RTF values execution of a SLAM algorithm becomes infeasible (due to robot's low speed). Our multiple navigation experiments empirically demonstrated, that the minimum

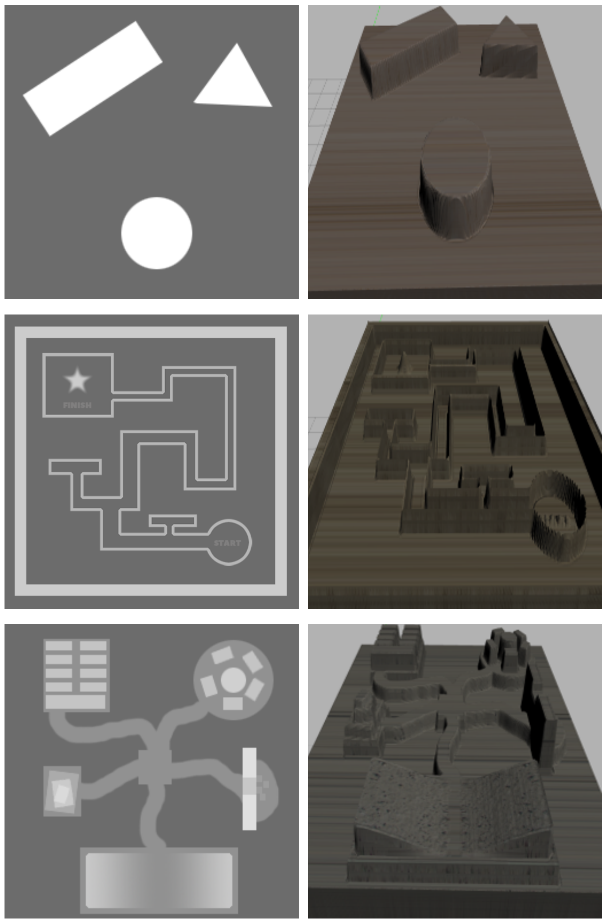
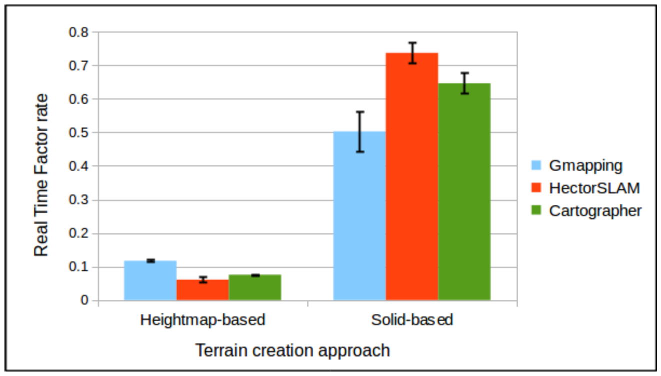


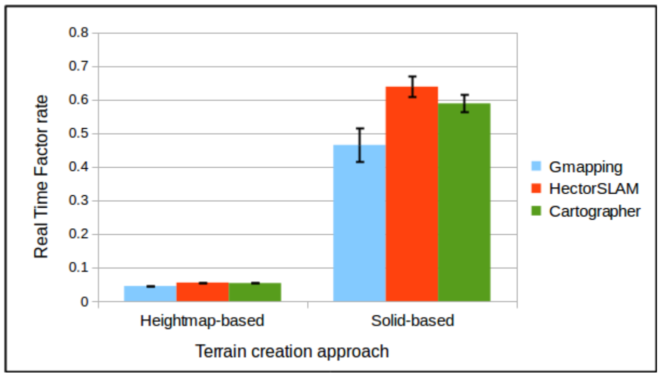
Fig. 12: Three input 2D source images (left column) and three corresponding 3D Gazebo environments, which were created by LIRS-WCT (right column). The map types have a varying difficulty: simple (top), medium (middle), and advanced (bottom)

value of RTF for a comfortable execution of a navigation algorithms should be at least 0.3. Experimentally we found out that this problem is caused by Gazebo physical engine onboard sensor emulation. A number of complex calculations is proportional to a world size and its complexity since a virtual sensory beam interacts with all surrounding objects.

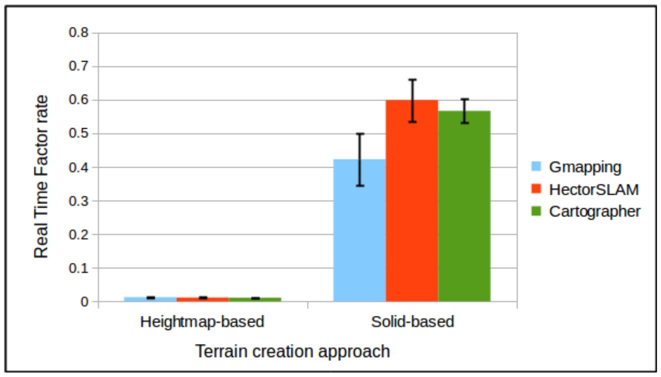
Within monolith-model-based environments that were created with LIRS-WCT, all three algorithms demonstrated an acceptable behavior. RTF took values between 0,42 to 0,73 depending on the complexity of local patches of the environments and was slightly higher in a simple environment. Figure 13 demonstrates SLAM algorithms performance efficiency in heightmap-model-based and solid-model-based environments of a simple (a), medium (b) and advanced (c) complexity types, which correspond to the environments in Fig. 12. According to the performance tables II, III, IV, in all three cases, Hector SLAM over-performed its counterparts in time efficiency, while GMapping demonstrated the worst performance in RTF. Successfully completed SLAM-missions with an acceptable level of RTF demonstrated the



(a)



(b)



(c)

Fig. 13: SLAM algorithms execution efficiency with a Husky robot in heightmap-based (left) and slid-based (right) worlds for a simple (a), medium (b) and advanced (c) terrain complexity type. For each SLAM algorithm an averaged Real-Time Factor (RTF) over time is shown along with the standard deviation. RTF rate reflects the simulation performance.

Table II. Performance comparison of world construction approaches using a simple complexity map

	Terrain construction approach	
SLAM method	Heightmap-based	Solid-based
GMapping	0.11	0.51
HectorSLAM	0.06	0.73
Cartographer	0.08	0.65

efficiency of our LIRS-WCT tool.

Table III. Performance comparison of world construction approaches using a medium complexity map

	Terrain construction approach	
SLAM method	Heightmap-based	Solid-based
GMapping	0.04	0.46
HectorSLAM	0.06	0.64
Cartographer	0.05	0.59

Table IV. Performance comparison of world construction approaches using an advanced complexity map

	Terrain construction approach	
SLAM method	Heightmap-based	Solid-based
GMapping	0.010	0.42
HectorSLAM	0.012	0.60
Cartographer	0.011	0.57

VI. CONCLUSIONS

This paper presented LIRS World Construction Tool (LIRS-WCT) - a new tool for automatic generation of Gazebo worlds from a 2D image or 2D LRF data. The tool converts a grayscale image into a 3D Collada format model, which is supported by Gazebo framework. The resulting world could be directly imported into Gazebo as a mesh and used for various purposes. Constructed with LIRS-WCT environments of three different complexity levels were validated with GMapping, HectorSLAM and Google Cartographer algorithms that were run on a Husky robot within Gazebo simulator. All three algorithms demonstrated an acceptable real-time factor (RTF) between 0,42 to 0,73 depending on the complexity of local patches of the environments. In all testing environments, Hector SLAM over-performed its counterparts in time efficiency, while GMapping demonstrated the worst performance in RTF. Successfully completed SLAM-missions with acceptable RTF level demonstrated the efficiency of our world construction tool. LIRS-WCT tool was compared with $Gazebo\_heightmap\_preparation$ tool [12] and showed significantly better performance. LIRS-WCT is available for free academic use at Gitlab account of our Laboratory of Intelligent Robotic Systems (LIRS)[1].

VII. ACKNOWLEDGMENT

This work was supported by the Russian Foundation for Basic Research (RFBR), project ID 19-58-70002. This work was partially supported by the research grant of Kazan Federal University. The fifth author acknowledges the support of the Japan Science and Technology Agency, the JST Strategic International Collaborative Research Program, Project No. 18065977.

REFERENCES

[1] Y. Liu, Y. Zhong, X. Chen, P. Wang, H. Lu, J. Xiao, and H. Zhang, "The design of a fully autonomous robot system for urban search and rescue," in *International Conference of Information and Automation (ICIA)*. IEEE, 2016, pp. 1206–1211.

[1]LIRS-WCT – Laboratory of Intelligent Robotic Systems World Construction Tool, GitLab, https://gitlab.com/LIRS_Projects/LIRS-WCT

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

[3] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, “Simulation environment for mobile robots testing using ros and gazebo,” in *20th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2016, pp. 96–101.

[4] S. Thrun, “Learning occupancy grid maps with forward sensor models,” *Autonomous robots*, vol. 15, no. 2, pp. 111–127, 2003.

[5] D. Meyer-Delius, M. Beinhofer, and W. Burgard, “Occupancy grid models for robot mapping in changing environments.” in *AAAI*, 2012, pp. 2024–2030.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[7] E. Magid, A. Pashkin, N. Simakov, B. Abbyasov, J. Suthakorn, M. Svinin, and F. Matsuno, “Artificial intelligence based framework for robotic search and rescue operations conducted jointly by international teams,” in *Proceedings of 14th International Conference on Electromechanics and Robotics “Zavalishin’s Readings”*. Springer, 2020, pp. 15–26.

[8] J. M. Santos, D. Portugal, and R. P. Rocha, “An evaluation of 2d slam techniques available in robot operating system,” in *International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2013, pp. 1–6.

[9] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, “A flexible and scalable slam system with full 3d motion estimation,” in *International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2011, pp. 155–160.

[10] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with rao-blackwellized particle filters,” *Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[11] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2d lidar slam,” in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1271–1278.

[12] R. Lavrenov and A. Zakiev, “Tool for 3d gazebo map construction from arbitrary images and laser scans,” in *10th International Conference on Developments in eSystems Engineering (DeSE)*. IEEE, 2017, pp. 256–261.

[13] R. Lavrenov, A. Zakiev, and E. Magid, “Automatic mapping and filtering tool: From a sensor-based occupancy grid to a 3d gazebo octomap,” in *International Conference on Mechanical, System and Control Engineering (ICMSC)*. IEEE, 2017, pp. 190–195.

[14] W. Yao, W. Dai, J. Xiao, H. Lu, and Z. Zheng, “A simulation system based on ROS and Gazebo for Robocup middle size league,” in *Iinternational Conference On Robotics And Biomimetics (ROBIO)*. IEEE, 2015, pp. 54–59.

[15] A. Pajaziti, “Slam–map building and navigation via ros,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 2, no. 4, pp. 71–75, 2014.

[16] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, “Comprehensive simulation of quadrotor uavs using ros and gazebo,” in *International Conference On Simulation, Modeling, And Programming For Autonomous Robots*. Springer, 2012, pp. 400–411.

[17] J. Devona. A simple program to convert a grayscale heightmap image to an stl triangle mesh. [Online]. Available: https://github.com/anoved/hmstl

[18] ——. A rudimentary c library for generating stl files from triangle lists and vice versa. [Online]. Available: https://github.com/anoved/libtrix

[19] T. Schulze, A. Gessler, K. Kulling, D. Nadlinger, J. Klein, M. Sibly, and M. Gubisch, “Open asset import library (assimp),” *Computer Software, URL: https://github. com/assimp/assimp*, 2012.

[20] T. Igarashi and D. Cosgrove, “Adaptive unwrapping for interactive texture painting,” in *Proceedings of the 2001 symposium on Interactive 3D graphics*. ACM, 2001, pp. 209–216.

[21] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS: a practical introduction to the Robot Operating System.* ” O’Reilly Media, Inc.”, 2015.