# Bringing reality to evolution of modular robots: bio-inspired techniques for building a simulation environment in the SYMBRION project

Martin Saska and Vojtěch Vonásek and Miroslav Kulich and Daniel Fišer and Tomáš Krajník and Libor Přeučil

***Abstract*— Two neural network (NN) based techniques for building a simulation environment, which is employed for evolution of modular robots in the Symbrion project, are presented in this paper. The methods are able to build models of real world with variable accuracy and amount of stored information depending upon the performed tasks and evolutionary processes. In this paper, the presented algorithms are verified via experiments in scenarios designed to demonstrate the process of autonomous creation of complex artificial organisms. Performance of the methods is compared in experiments with real data and their employability in modular robotics is discussed. Beside these, the entire process of environment data acquisition and pre-processing during the real evolutionary experiments in the Symbrion project will be briefly described. Finally, a sketch of integration of gained models of environment into a simulator, which enables to fasten the evolution of a real complex modular robot, will be introduced.**

## I. Introduction

A valuable representation of environment is an important part of modular systems aiming to employ evolutionary principles for creating a complex robot from simple autonomous modules. One can find three main aspects where such a model of the workspace would be crucial in modular robotics. First of all, the environment itself affects the purpose of the evolution. In most of the modular systems [1], robots are able to form complex structures to be able to overcome obstacles in the operating environment which would be impossible for single robots (see Fig. 1 for a motivation). Secondly, the single robots, but also the complex organism itself, need a model of the environment to be able to aggregate to more complex structures to solve their tasks to navigate to docking stations etc. The third purpose of the environment modelling in modular robotics arises mainly in the Symbrion project [2] where a concept of an intensive evolution via simulations has been proposed and investigated. It was shown that it is inefficient and possibly dangerous (for the robots) with existing technologies to leave robots to do autonomous reconfigurability blindly. Rather a combination of numerous runs of evolutionary algorithms in simulators and only a few hardware trials of perspective structures found in simulated environment is preferred.

Such a close connection of simulations and a real evolution of robotic organisms brings additional demands on the process of simulation environment building. These required features will be verified and discussed in the experimental part of this paper. The first requirement is precision of the gained model. Only simulations realised with models matching the reality are meaningful for trials with real robots. Here, one should mention that important aspects of the model precision are not limited to shape and size of objects in the robots' workspace only but also additional properties important for the simulation of movement and friction of robots on objects' surfaces need to be considered. Another important aspect is the possibility to set the resolution of gained models which may affect the precision. For an initial growing of the modular organisms via the evolutionary process a spare model with lower amount of information could be sufficient, which speed-up the evolution. Contrariwise, an evolution of robot's behaviour to properly interact with environment requires a more precise model. Therefore, relations between the complexity of models, their quality and computational efficiency of simulations are discussed in the experimental section of this paper.

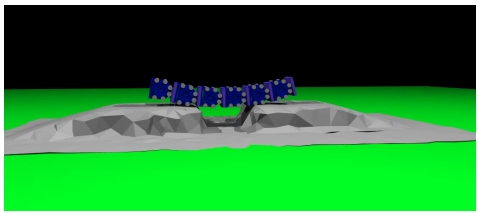


Fig. 1. A modular robot overcoming a pit between two ramps.

Many projects have appeared in last two decades in the field of modular robotics. They have addressed problems like mechanical design, motion planning, self-reconfiguration and morphology evolution of multi robot organisms [1]. Beside the construction of mechanical robots, software simulators have been often developed. They can be used prior to hardware experiments, e.g., for evolving a 3D morphology of an organism. Several projects developed their own simulators, like Molecubes [3] or Symbrion/Replicator [4], other use commercial simulators, e.g., Webots [5]. The simulators usually provide a simple arena with none or a small number of elementary obstacles. In Symbrion project, the robots are designed to survive in more complicated environments with holes and more complex obstacles. To be able to verify the results of simulations with real robots, the physical properties of robots and arena in the simulator have to be close to parameters of real robots and environment. For more

Martin Saska, Vojtěch Vonásek, Miroslav Kulich, Daniel Fišer, Tomáš Krajník and Libor Přeučil are with Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague. {saska,vonasek,kulich,tkrajnik,preucil}@labe.felk.cvut.cz, danfis@danfis.cz

This work is supported by European Union under the grant Symbrion-Enlarged no. 216342. and by MŠMT under the grant COLOS no. LH11053.

complex environments it is important to have an automatic tool for creating the simulation environment. In this study, we describe how 3D reconstruction techniques can be used for this task.

Methods for surface reconstruction can be roughly divided into two categories [6]. Static methods, such as [7], [8], [9], [10], are based on geometric techniques. As they connect a set of input points, the size of the resulting mesh corresponds to the number of the input points. Post-processing is thus needed to get results with desired resolution. Learning methods and self-organizing structures as their special case belong to dynamic methods [11], [12]. These approaches approximate the input point cloud by adaptation of an initial mesh [13] by modification of geometry, size or/and topology of the initial mesh.

One of the targets of the Replicator/Symbrion projects is a long-term robot autonomy in a changing world. Since the environment model must keep up with the current state of reality, a mechanism for model refinement and update must be considered. Using bio-inspired dynamic methods for surface reconstruction, the proposed model can be updated with sensory data acquired by the robots, which operate within the modeled environment. The robots can not only provide new range measurements, which contribute to precision and completeness of the generated meshes, but also might add information which cannot be perceived by range sensors at all. For example a robot, which is equipped by a camera, can add texture or color information to the mesh or recognize, localize and add known objects (e.g., power plugs) to the map. Taking into account all aforementioned requirements, we have selected two appropriate methods of 3D modelling which will be adjusted and verified for the purpose of modular robotics here.

This paper is structured as follows. The core of the paper is section II describing utilized approaches of environment scanning and modelling. Particularly, a robotic platform adjusted for 3D scanning of workspace of modular robots and a procedure of data collection, followed by a specification of methods of post-processing for data errors and noise reduction, is described in subsection II-A. The algorithms of 3D object reconstruction are described in detail in subsection II-B with focus on possibility of setting of different resolution of gained models which is important due to the required computational time of the simulation. The usage of obtained models in physical simulations of evolution of modular robots and approaches solving interactions of the model of environment with the modular organisms are described in subsection II-C.1. Experimental results and comparison of utilized algorithms are presented in section III. Finally a discussion of applicability of presented approaches for simulation of evolution of modular robots is included in section IV.

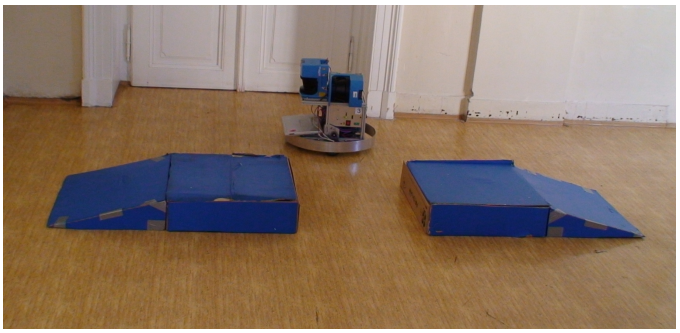


Fig. 2. The mobile robot equipped with two SICK laser range finders scanning the environment.

## II. Approaches for building a simulation environment

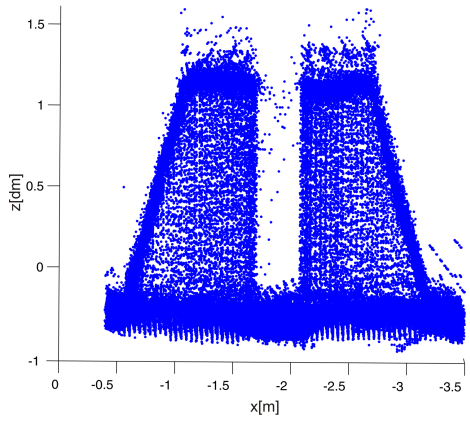
### A. Data collection and pre-processing

The first step of the effort of bringing reality to the evolution of modular robots lies in the data collecting. We have utilized a differential drive G2Bot robotic platform [14]. The robot has been equipped with two perpendicularly mounted SICK laser range finders (see Fig. 2 for details on the measurement configuration). While the first horizontal laser together with odometry is employed for the robot localization, the second vertical laser pointed to the robot side is employed for the 3D environment scanning. Although the robot is localized using the on-board sensors, we plan to use the Ubisense localization system [15] for the final experiments.

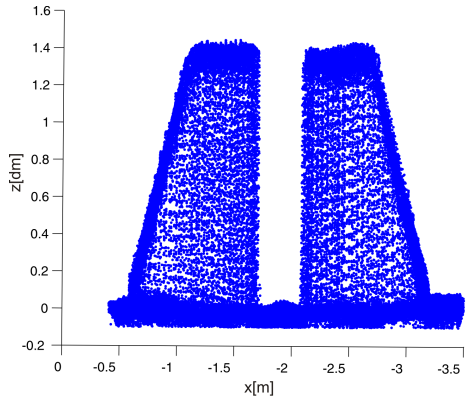
During the data collection, the robot has to be driven in the environment in such a way that the vertically mounted sensor scans obstacles completely. Since the coverage path planning is not the aim of this paper, we have teleoperated the robot through the environment during experiments but one can find plenty of algorithms for autonomous control of robots [16]. In the presented approach, the collected data (see Fig. 3(a) for example) has been stored and processed off-line. Again, the method could be extended for on-line 3D object reconstruction and environment mapping using a state-of-the-art method, but this would exceed the scope of this paper.

Several different kind of errors and uncertainties have been identified in the obtained datasets, which requires a pre-processing of data before utilization of methods for 3D object reconstruction. First of all, shaking of the robot during the scanning need to be compensated. Here, we have taken advantage of the 180 degree view of the SICK sensor which ensures that the straight floor of the scanned area is always included in each scan. Based on the declination of the line-segment representing the floor in the scan, the data are transformed to correct the unwanted deviation of sensor heading. Afterwards, the noise caused by imprecise sensors[1]

[1]Resolution of the employed SICK device is 7mm (in distance of 4m) and the robot is localized with an error less than 1cm and 0.5 degree during the whole experimental run.




(a) Raw point-data obtained with the vertically mounted SICK.




(b) Smoothed point-data with the listed pre-processing methods.

Fig. 3. Dataset obtained during the scanning of the two ramps with the pit.

is suppressed by a moving average of 5 neighbouring data samples in each laser scan.

The most significant error occurs due to the so called *mixed pixel problem* [17]. This problem is caused by the integration of light from multiple sources by a single pixel, particularly around the edges of objects, resulting in erroneous range measurements. Fortunately these measurements can be filtered out using a thresholding of a 3D histogram which corresponds to number of measurements belonging to a cell in a 3D grid. Such a segregation is possible thanks to continuous scanning in both horizontal and vertical directions where the wrong points are isolated in the space. The obtained pre-processed dataset (see Fig. 3(b)) is utilized as an input for the algorithms of 3D object reconstruction described in the following section.

### *B. 3D object reconstruction*

Due to necessity of numerous runs of evolutionary steps in the simulator, a crucial feature of algorithms for environment modelling is the possibility to build models of the real world with different accuracy and amount of stored information. Density of information describing objects in simulations significantly affects the computational complexity of each evolutionary step. Similarly as in nature, some tasks and learning processes require only a rough approximation of the environment and sometimes details of the selected objects in the neighbourhood are needed. See Fig. 4 and Fig. 5 for examples of obtained models with different resolution.

Finally, the concept of Symbrion robots being able to form complex 3D structures requires to form a 3D model of their workspace. In Symbrion Grand Challenge [18], which should demonstrate the ability of self-reconfigurability of modular robots, the organisms are not limited to a planar scene. Two methods for 3D reconstruction are described in this paper:

*1) GSRM-based method:* Standard self-organizing maps typically represent an input point cloud by a graph containing a set of vertices connected with edges, i.e., they don't explicitly find faces. This is a crucial issue, as these approaches can not guarantee geometric consistence of produced results and therefore their use in simulators is either problematic or impossible. Rêgo et al. [6] introduced Growing self-reconstructions maps (GSRM) that reconstruct surface in the form of a triangular mesh where faces of the mesh form a two-manifold (i.e., every point on the surface must have a neighborhood homeomorphic to a disk). The method is based on growing neural gas (GNG) [19] where the number of neurons and topology of the network change during the self-organization process. Competitive Hebbian learning used in GNG is extended in GSRM so that faces are created together with new points and edges. Moreover, removal of edges and incident faces is proposed in [6] in order to keep the two-manifold property during the whole learning process. After the learning process finishes, positions of the neurons are learned but the topology is still incomplete.

Topology learning is therefore performed by presenting all input data are to fixed neurons once again. This leads to creation of almost all necessary edges but some faces remain non-triangular and thus triangulation of these faces is done in final post-processing. The stopping criteria for the learning process is the desired number of neurons.

Due to insufficient description of post-processing in [6] we developed another post-processing method. During the post-processing, all nodes in the mesh are visited and checked if there are exactly two edges emanating from node that have each exactly one incident face. A new face is then created between those two edges and edge connecting the opposite nodes which is created if not already present in the mesh. These steps are repeated until new faces can be created. This approach patches almost all holes that remained in the mesh after geometry and topology learning performed by GSRM. The resulting triangular mesh was then used in the experiments.

*2) GG - based method:* Growing grid (GG) is another self-organizing feature map that adapts according to Hebbian rule [20]. Similarly to GSRM, GG is a growing structure, i.e., the number of neurons changes (increases) during learning. On the other hand (and in contrast to GSRM), GG has a fixed structure, which has a form of a rectangular grid. The

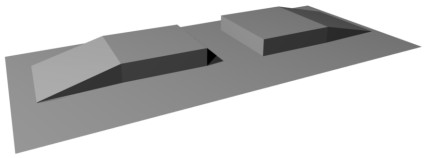


Fig. 6. Simple "hand-made" 3D model.

process of adaptation has two distinct phases: a growth phase and a fine-tuning phase.

The growth phase starts with 2x2 structure forming a rectangle. A random sample from the training set is generated and the nearest neuron together with its topological neighborhood is adapted to it. If the number of generated samples reaches a defined constant (derived from the actual network size), a whole new row or column is added into the grid. In order to determine a correct position for insertion, a local counter of each neuron is maintained storing the number of samples for which the neuron was a winner. A new column/row is inserted between the neuron with the highest counter and its direct neighbor having also the highest counter. After the insertion, local counters of all neurons are reset. The adaptation process continues until the desired network size is achieved.

After the growth phase finishes, the size of the network as well as raw estimate of neurons positions is determined. The purpose of the fine-tuning phase is then to tune neurons positions more precisely. The process is similar to the one in the growth phase. In each step, a random signal is generated and the winning neuron together with its neighborhood is adapted. The number of steps is determined with respect to the network size.

### C. Simulation

To speed up the evolution of robots, a simulator can be used. Nowadays, several different simulators, suited for different types of multi-robot systems, exits. For swarm robotics, where the multi-robot teams move independently, sensor inputs are simulated more precisely than a physical motion of the robots, as discussed in details in [21]. Also, the realistic models of the robot environment is crucial in this type of simulations. Contrariwise, for evolving a motion of multi-robot organisms (already joined swarm robots into a complex body), the sensor simulation is not so important, however, an accurate simulation of motion of such complex organisms is required. In this case, a physic engine needs to be used to simulate the motion of the organism.

In the presented project, where the simple swarm robots are subsequently integrated to the complex organism, also the employed models of the environment and the simulator itself have to be adapted as the multi-robot system converts from the swarm into a complex body and back from the complex robot into the simple units. The simulation needs to be simplified to provide only such a set of functions, which is important for studding of current behaviours of the system.

*1) Robot3D simulator:* The Robot3D simulator provide simulation of both robotic swarms and organisms [4]. The simulator is based on Delta3D framework which allows both physical and sensor simulations. To model the 3D environment, the gained 3D models constructed by the above described methods can be loaded in to simulator.

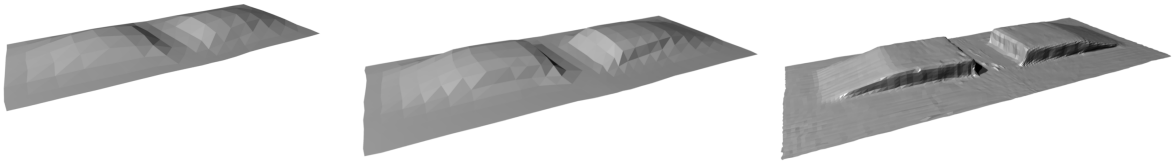
To represent the geometry of robots and arena in the simulator, two approaches can be used: represent the geometry by a set of primitives (e.g., spheres, boxes or cylinders) or using a 3D triangular mesh. The geometric primitives are suitable for modelling simple robots and scenes. Moreover, they allows to easily detect collision between objects. This is very important in physical simulation where the collision detection is frequently used to determine the forces which have to be applied to the objects in order to remove the collision. The 3D triangular mesh representation is suitable for robots and environments with more complex shapes. To represent a robot geometry using 3D meshes, one can create the mesh based on a CAD model of the robot. This ensures that the robot geometry properties will be the same as for real robots. Although an arbitrary shape can be modelled by the 3D mesh, it is more difficult to detect collisions between a general (usually nonconvex) shapes. This can slowdown the performance of the collision detection as well as the performance of physical simulation.

The geometry of robots in the simulator is also used to for simulation of sensors like IR-based distance sensors or laser range finders. These sensors are simulated by computing collision between a ray emitted from the sensor and other objects. The selection of proper 3D models, which will be used in the simulator, is therefore crucial.

## III. Experimental results

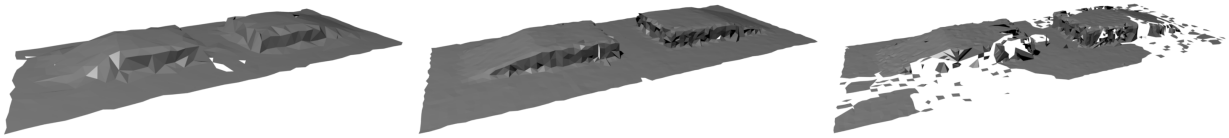
In this section, the meshes obtained by the aforementioned methods are used to measure performance of the simulator. Two metrics are used to evaluate how the meshes influence the simulation : the speed of physical simulation and the speed of sensors simulation.

The influence of various meshes on the performance of the simulator was verified in following sets of experiments. The 3D meshes constructed by GSRM and Growing grid methods have been used. The properties of the meshes are described in Tab. I. The number in the brackets in GSRM methods denotes the desired number of neurons; for GG method, this denotes how many times one point from the input 3D point cloud was used for learning. To compare the performance with various meshes, a simple hand-made 3D model of the arena was created, see Fig. 6. This model contains much less faces than if employing the approaches using real sensor point-datasets.



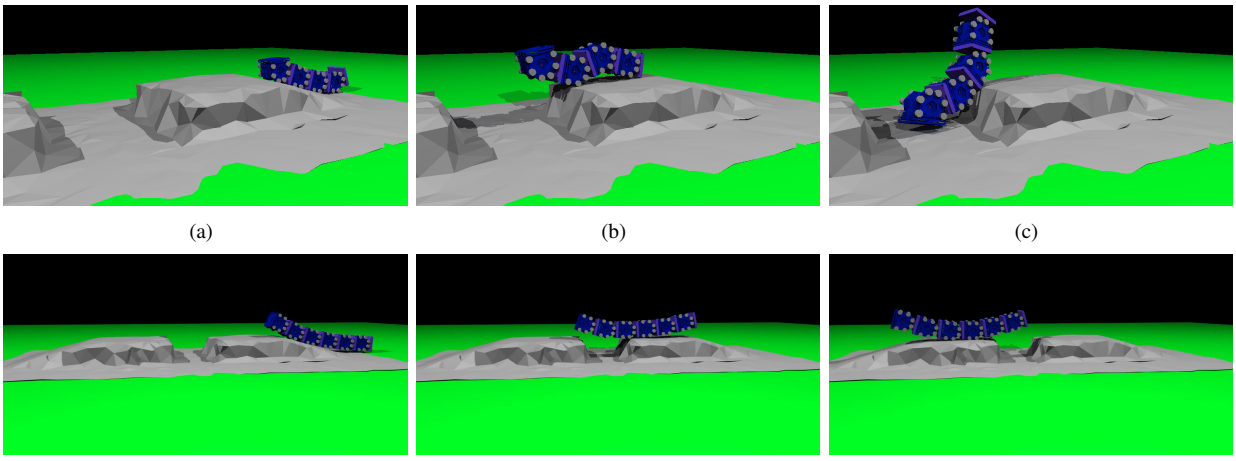
(a) 50k of steps; the obtained model consists of 210 triangles. (b) 200k of steps; the obtained model consists of 576 triangles. (c) 20M of steps; the obtained model consists of 13860 triangles.

Fig. 4. GG-based method applied for a scene reconstruction with different number of learning steps.



(a) 800 steps; the obtained model consists of 1011 triangles. (b) 2200 steps; the obtained model consists of 4175 triangles. (c) 5000 steps; the obtained model consists of 5895 triangles.

Fig. 5. GSRM-based method applied for a scene reconstruction with different number of learning steps.



(a) (b) (c)

(d) (e) (f)

Fig. 7. An insufficiently evolved robot for overcoming the pit (a)-(c). A sufficiently evolved robot overcoming the pit (d)-(f).

| Name | Method | Triangles | Computation time [s] |
|---|---|---|---|
| GSRM | GSRM (2200) | 4175 | 20 |
| GG-based | Growing grid ($20 \cdot 10^6$) | 13860 | 270 |
| Simple | hand-made | 39 | N/A |

TABLE I

PROPERTIES OF SELECTED 3D MESHES, WHICH ARE USED IN EXPERIMENTS. NUMBER OF TRIANGLES FORMING THE MODEL AND COMPUTATIONAL TIME REQUIRED FOR ITS CREATING.

### A. *Snake organism outstepping a pit*

In the first scenario, a snake made of several Scout robots [18] is supposed to outstep a pit. The snake moves using wheels of the Scout robots whose velocity is constant. To enable the snake to go up the descent, the main hinges of the robots are controlled to form a U shaped snake (Fig. 7(f)). The average times needed for computing the physical simulation step and collision detection computed from 100 runs are described in Tab. II. In each run where the snake out-steps the pit, a 3500 measurements of collision detection and physical simulation were made.

The speed of the physical simulation step is not influenced by the used mesh because the kinematics of the snake was constant during outstepping of the pit. The number of forces and constraints for integration of forces are the same in each step which does not influence the computation time. The meshes constructed by the presented methods slightly influence the speed of the collision detection. However, this is caused by the roughness of the meshes rather than by the number of triangles in the meshes. The triangles in the

| Mesh name | Physical step [$\mu s$] | | Collision detection [$\mu s$] | |
|---|---|---|---|---|
| | mean | dev | mean | dev |
| GG-based | 147 | 73 | 825 | 280 |
| GSRM | 150 | 75 | 830 | 283 |
| Simple | 152 | 77 | 846 | 285 |

TABLE II

SIMULATION AND COLLISION DETECTION TIME FOR SNAKE ORGANISM OUTSTEPPING THE PIT.

| Mesh name | mean [ms] | dev [ms] |
|---|---|---|
| GG-based | 362 | 134 |
| GSRM | 119 | 44 |
| Simple | 5.45 | 1.31 |

TABLE III

PERFORMANCE OF A LASER RANGE FINDER SIMULATION. REQUIRED TIME FOR SIMULATION OF ONE SCAN OF LASER.

meshes are represented by a hierarchical bounding box tree which allows to quickly determine possible collisions, thus the number of triangles does not influence the computation time significantly.

### B. Simulation of sensors

Beside the physical simulation of modular robots, the Robot3D provides sensor simulation. The speed of the sensors simulation is determined by laser range finder sensors simulation which is computed by detecting collision between laser rays and objects. To determine the distance from sensors to objects, a ray is emitted from the sensors and a collision between this ray and objects is detected. Unlike in the previous case, the triangular mesh is not organized in a tree structure, and therefore, it increases the computation time needed to compute nearest collision of a ray. The speed of this process is influenced by the number and type of geometric primitives of the objects and angular resolution of the laser rangefinder. The influence of various meshes to the speed of range finder simulation is summarized in Tab. III. It can be seen, that the computation of laser beams on large meshes is more time consuming than on a simple mesh. The presented experiments show, that while the larger meshes can be used in physical simulations, they are not suitable for sensor simulations as implemented in the Symbrion project.

To speed up the sensor simulation, the used 3D meshes should have less number of triangles. Another approach is to use a mixed 3D model, which consists of meshes and other geometric primitives (planes, spheres). To detect the geometric primitives in the 3D meshes, the approach presented in [22] can be used.

## IV. CONCLUSION

We have presented an approach for automatic building of simulation environment from a laser data for evolution of modular robots. Two methods for 3D reconstruction have been described. The 3D meshes obtained by these methods differ mainly in number of triangles and consequently in computational time required for model building.

The performance of the physical and sensor simulations with various meshes have been investigated. It has been shown that for the physical simulation the number of triangles in meshes does not influence the speed of physical simulations. Nevertheless, it has been identified (via an empirical observation of various simulations) that the growing-grid method produces smoother results and it is more suitable to build the simulation environment. Contrariwise, the GSRM method is faster, and therefore, it is suitable for varying environment or if a frequent update of models is required.

## REFERENCES


[1] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *Robotics Automation Magazine, IEEE*, vol. 14, no. 1, pp. 43 –52, march 2007.

[2] http://www.symbrion.eu, *[cit. 2011.7.19]*, Symbrion project web-page.

[3] V. Zykov, A. Chan, and H. Lipson, "Molecubes: An open-source modular robotics kit."

[4] Lutz Winkler and Heinz Wörn, "Modular Robot Simulation," *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, pp. 133–165, 2010, available at https://launchpad.net/robot3d.

[5] Webots, "http://www.cyberbotics.com/."

[6] R. L. M. E. Rêgo, A. Araujo, and F. de Lima Neto, "Growing self-reconstruction maps," *Neural Networks, IEEE Transactions on*, vol. 21, no. 2, pp. 211 –223, February 2010.

[7] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *SIGGRAPH Comput. Graph.*, vol. 26, pp. 71–78, July 1992.

[8] R. Šára and R. Bajcsy, *Fish-scales: Representing fuzzy manifolds*. Narosa Publishing House, 1998, p. 811817.

[9] N. Amenta, M. Bern, and M. Kamvysselis, "A new voronoi-based surface reconstruction algorithm," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '98. New York, NY, USA: ACM, 1998, pp. 415–421.

[10] Y. Duan and H. Qin, "2.5d active contour for surface reconstruction," in *Proceedings of the Vision, Modeling, and Visualization Conference 2003 (VMV 2003)*. Aka GmbH, 2003, pp. 431–439.

[11] J. Koutnik, R. Mazl, and M. Kulich, "Building of 3D environment models for mobile robotics using self-organization," in *Parallel Problem Solving from Nature - PPSN-IX. Heidelberg*. Springer, 2006, pp. 721–730.

[12] V. DalleMole and A. Araujo, "The growing self-organizing surface map: Improvements," in *Neural Networks, 2008. SBRN '08. 10th Brazilian Symposium on*, oct. 2008, pp. 183 –188.

[13] H. Qin, C. Mandal, and B. Vemuri, "Dynamic catmull-clark subdivision surfaces," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 4, no. 3, pp. 215 –229, jul-sep 1998.

[14] J. Chudoba, R. Mázl, and L. Přeučil, "A Control System for Multi-Robotic Communities," in *ETFA 2006 Proceedings*. Piscataway: IEEE, 2006, pp. 827–832.

[15] www.ubisense.net, "Real-time location systems."

[16] P. K. Allen, M. K. Reed, and I. Stamos, "View planning for site modeling."

[17] J. Godbaz, M. Cree, and A. Dorrington, "Mixed pixel return separation for a full-field ranger," in *Image and Vision Computing*, 2008.

[18] S. Kernbach, O. Scholz, K. Harada, S. Popesku, J. Liedke, H. Raja, W. Liu, F. Caparrelli, J. Jemai, J. Havlik, and et al., *Multi-Robot Organisms: Stage Of The Art*. IEEE Press, 2010, pp. 1–10.

[19] B. Fritzke, "A growing neural gas network learns topologies," *Advances in Neural Information Processing Systems 7*, vol. 7, pp. 625–632, 1995.

[20] ——, "Some competitive learning methods," Ruhr-Universitat Bochum, Tech. Rep., 1997.

[21] *Evolutionary robotics*. MIT Press, 2000.

[22] M. Attene, B. Falcidieno, and M. Spagnuolo, "Hierarchical mesh segmentation based on fitting primitives," *Vis. Comput.*, vol. 22, March 2006. [Online]. Available: http://portal.acm.org/citation.cfm?id=1127195.1127199