This is an accepted version of the article: Vonásek, V., Kulich, M., Fišer, D., Krajník, T., Saska, M., Petrík, V., and Přeučil, L. (2012). **Techniques for Modeling Simulation Environments for Modular Robotics**. *Proccedings of International Conference on Mathematical Modelling*, which has been published in a final form at https://doi.org/10.3182/20120215-3-AT-3016.00037.

Automated system for building 3D simulation environment for modular robotics

Vojtěch Vonásek * **Miroslav Kulich** * **Tomáš Krajník** *
Martin Saska * **Daniel Fišer** * **Vladimír Petrík** * **Libor Přeučil** *

* *Czech Technical University in Prague, Faculty of electrical engineering, Department of cybernetics, Technicka 2, Prague 6, 166 27*
{vonasek,kulich,tkrajnik,saska,fiser,preucil}@labe.felk.cvut.cz,
petrikvl3@fel.cvut.cz

Abstract:
In modular robotics, complex structures can be formed from basic modules to solve tasks which would be difficult for single robots. The development of techniques for adaptation and evolution of multi-robot organisms is the subject of Symbrion project **?**. In the project, the bio-inspired evolutionary algorithms are massively simulated prior to run them on real hardware. It is crucial to evolve behaviors of the robots in simulation, that is close to a real world. Hence, accurate and efficient representation of an environment in the simulation is needed. The robots learn simple motion primitives or complex movement patterns during many runs of the evolution. The learned skills will be then used during the experiments with real hardware. In this paper, we present methods for building 3D model of a real arena using a laser rangefinder. The resulting 3D models consist of triangles and it can be constructed in various level of details. We will show, how the models influence the speed of the simulation.

Keywords: modular robotics, simulation, 3D model reconstruction

1. 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. In most of the modular systems Yim et al. (2007), robots are able to form complex structures to be able to overcome complex obstacles in the operating environment which would be impossible for single robots. 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. In Symbrion project, a concept of an intensive evolution via simulations has been proposed. 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. 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 can 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.

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 Yim et al. (2007). 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 Zykov et al. (2008) or Symbrion/Replicator Lutz Winkler and Heinz Wörn (2010), other use commercial simulators, e.g. Webots. 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 environments with holes and complex obstacles. To be able to verify the results of the simulations with real robots, the simulated robots and environment have to be close to parameters of the real robots and environment. For more 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.

[1] This work was supported by Ministry of Education of Czech Republic under projects No. 7E08006, No. 7E11017 and by the EU FP7 projects No. ICT-216240 and No. ICT-216342

Methods for surface reconstruction can be roughly divided into two categories Rêgo et al. (2010). Static methods, such as Hoppe et al. (1992); Šára and Bajcsy (1998); Amenta et al. (1998); Duan and Qin (2003), 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 Koutnik et al. (2006), DalleMole and Araujo (2008). These approaches approximate the input point cloud by adaptation of an initial mesh Qin et al. (1998) 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.

This paper is structured as follows. The core of the paper is section 2 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 2.1. The algorithms for 3D object reconstruction are described in detail in subsection 2.2. The experimental results and comparison of the utilized algorithms are presented in section 3.

2. APPROACHES FOR BUILDING A SIMULATION ENVIRONMENT

2.1 Data collection and pre-processing

The environment is scanned using two perpendicular SICK laser rangefinders mounted in a differential drive G2Bot robotic platform Chudoba et al. (2006). 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 (the robotic platform is depicted on Fig. 1). 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 Blaer and Alien (2006). In the presented approach, the collected data (see Fig. 2 for example) has been stored and processed off-line. Again, the method could be extended for on-line 3D object reconstruction and

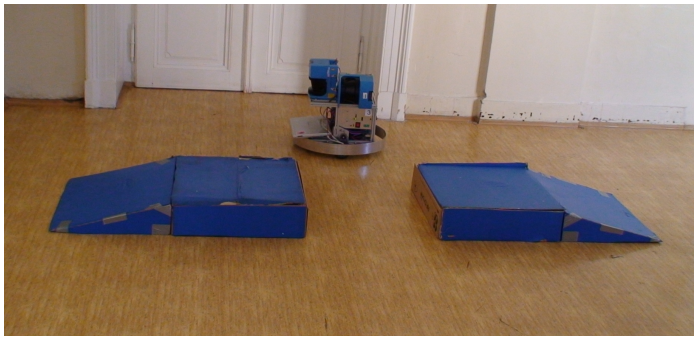


Fig. 1. The mobile robot equipped with two SICK laser rangefinders scanning the environment.

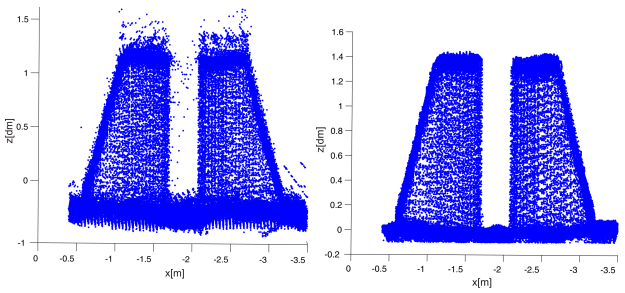


Fig. 2. Raw point data (left). Data with removed noise points (right).

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 [2] 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* Godbaz et al. (2008). 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. 2) is utilized as an input for the algorithms of 3D object reconstruction described in the following section.

[2] 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.

2.2 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.

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, which should demonstrate the ability of self-reconfigurability of modular robots, the organisms are not limited only to a planar scene Kernbach et al. (2010). Three methods for the 3D reconstruction of the model environment are briefly described in the following subsections.

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. Growing self-reconstructions maps (GSRM) Rêgo et al. (2010) 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) Fritzke (1995) 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 Rêgo et al. (2010) 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 to the 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 Rêgo et al. (2010) 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.

GG–based method Growing grid (GG) is another self-organizing feature map that adapts according to Hebbian rule Fritzke (1997). 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 process of adaptation has two distinct phases: a growth phase and a fine-tuning phase.

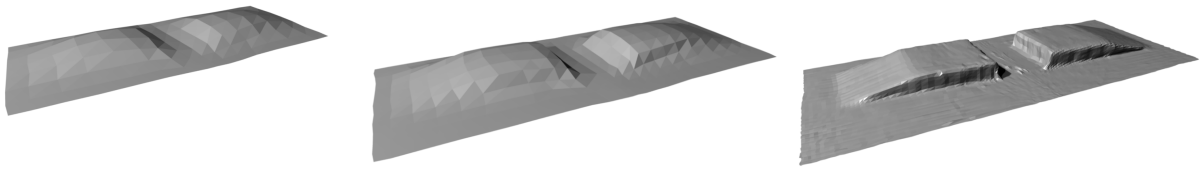
The growth phase starts with 2x2 structure forming a rectangle. A random sample from the training set is generated and the nearest neuron together with its topological neighborhood is adapted to it. If the number of generated samples reaches a defined constant (derived from the actual network size), 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.

Plane fitting method (PF) Although the above described methods can provide detailed 3D model, a simple model can be more suitable in certain situations, e.g. for sensor simulation. Such a model can be constructed by fitting planes to the 3D data. We propose a plane-fitting method, which consists of two phases. In the first phase, the 3D data are divided into several clusters, where each cluster represents a plane. This is done by finding all points inside a specified radius from a randomly picked point. If a plane can be fitted to this points, new cluster is created. This is repeated until there is a predefined number of clusters. This can lead to a situation, where points belonging to the same plane are divided into several clusters(see Fig 5a). Hence, the clusters which represent same planes, have to be merged (Fig. 5b). The result of the first phase is thus a set of planes fitted to the data. In the second phase, the intersections between the planes are found, which allows to define a border of each plane. The border is represented as a polygon lying on the plane (see Fig. 5c). The 3D model is thus represented by a set of 3D polygons. For the purpose of simulation, the resulting 3D model is triangulated (example of the model is depicted on Fig. 5d). The proposed method provides 3D models with significantly less amount of triangles, which is crucial for the performance of the simulation.

2.3 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

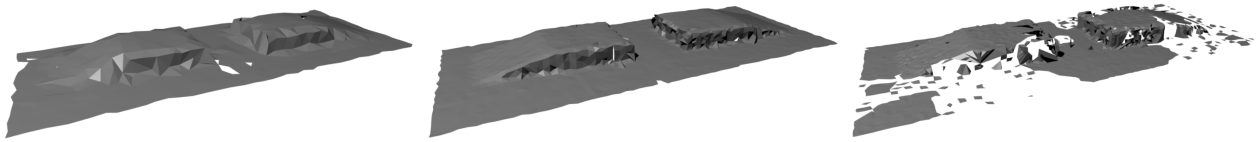


(a) $50 \cdot 10^3$ of steps; the obtained model consists of 210 triangles.

(b) $200 \cdot 10^3$ of steps; the obtained model consists of 576 triangles.

(c) $2 \cdot 10^6$ of steps; the obtained model consists of 13860 triangles.

Fig. 3. 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. 4. GSRM-based method applied for a scene reconstruction with different number of learning steps.

a b

c d

Fig. 5. Initial clusters found in the data. Here, the clusters are visualized by rectangles (a). The result of cluster merging (b). Found borders of each plane (c). The resulting 3D model (d).

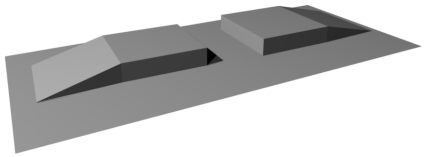


Fig. 6. Simple "hand-made" 3D model

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 ero (2000). 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 Symbrion project, the Robot3D simulator was developed Lutz Winkler and Heinz Wörn (2010). The Robot3D simulator provide simulation of both robotic swarms and organisms. The simulator is based on Delta3D framework which allows both physical and sensor simulations. For the physical simulation, the Open Dynamics Engine (ODE) is used.

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 allow to easily detect collision between objects. This is very important in physical simulation where the collision detection is frequently used. 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 geometric shapes are also used for simulation of sensors like IR-based distance sensors or laser rangefinders. 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.

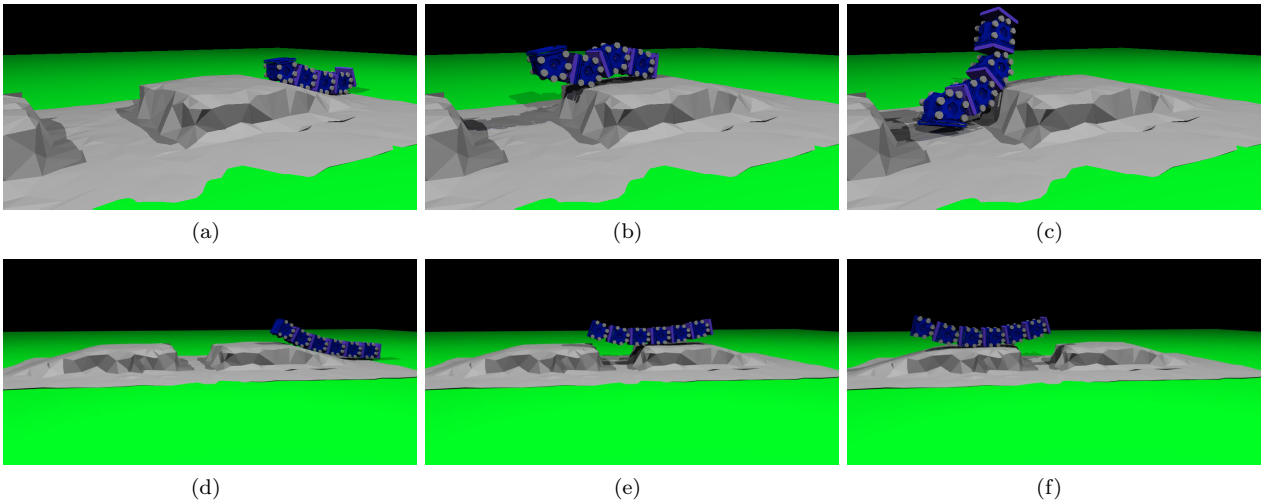


Fig. 7. An insufficiently evolved robot for overcoming the pit (a)-(c). A sufficiently evolved robot overcoming the pit (d)-(f).

3. EXPERIMENTAL RESULTS

In this section, the reconstructed 3D models are used in two simulated scenarios to verify how they influence the speed of the simulation. For the physical simulation, the ODE physical engine has been used and for simulating sensors, methods from Delta3D framework have been employed. For the experiments, these parameters were used: max nodes in GSRM method was set to 2200 and number of iterations in the GG method was set to $2 \cdot 10^6$. The properties of the resulting 3D meshes are described in Tab. 1. We can see, that both GSRM and GG-based method provide models with high amount of data compared to the plane fitting method. A simple hand-made 3D model of the arena was also created, see Fig. 6. The model created by PF method consists of less number of triangles. The reason is, that the PF method did not find two vertical walls inside the pit.

Table 1. Properties of the 3D meshes used in the experiments.

Name	**Triangles**	**Computation time [s]**
GSRM	4175	20
GG-based	13860	270
PF	35	71
Simple	39	—

3.1 Snake organism outstepping a pit

In the first scenario, a snake made of several Scout robots Kernbach et al. (2010) is supposed to outstep a pit. The snake moves using the Scout's wheels with constant velocity. To enable the snake to go up the descent, the main hinges of the robots are controlled to form U-shaped snake (Fig. 7(f)). During each trial, where the snake outstepped the pit, 3500 measurements of collision detection and physical simulation were made. The average times from 100 trials are described in Tab. 2.

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. Hence, the physical engine has to solve same set of equations (with different initial conditions) in each step. The meshes constructed by the presented methods slightly influence the speed of the collision detection. The triangles in the meshes are represented by a hierarchical bounding box tree which allows to quickly determine possible collisions. Hence, the number of triangles in the mesh does not influence the computation time significantly.

Table 2. Simulation and collision detection time for snake organism outstepping the pit.

Mesh name	**Physical step** [μs]		**Collision detection** [μs]	
	mean	dev	mean	dev
GG	907	242	221	83
GSRM	910	232	222	82
PF	908	244	223	85
Simple	901	253	218	86

3.2 Simulation of sensors

The robotic modules developed in the Symbrion project are equipped with several sensors like camera, laser rangefinders or IR proximity sensors. The simulation of the sensors is thus crucial. In the second scenario, the speed of laser rangefinder simulation is studied. The sensor is simulated by computing collisions between the rays emanating from the sensor and 3D models. The speed of this process is influenced by the number and type of geometric primitives use to represent the simulated objects and angular resolution of the laser rangefinder. The number of rays per one scan was set to 360. The influence of various meshes to the speed of rangefinder simulation is summarized in Tab. 3. The table shows times needed to simulate one scan of the laser rangefinder. 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. 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

Table 3. Performance of a laser rangefinder simulation.

	Sensor simulation	
Mesh name	**mean [ms]**	**dev [ms]**
GG-based	333	125
GSRM	115	43
PF	2.5	0.5
Simple	3.5	0.8

the 3D meshes, the approach presented in Attene et al. (2006) can be used.

4. CONCLUSION

We have presented an approach for automatic building of simulation environment from a laser data for evolution of modular robots. Three 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 the number of triangles in the meshes does not influence the speed of physical simulations. On the other hand, the simulation of laser rangefinder is significantly influenced by the size of the 3D models. It is thus preferred to build the 3D models with less number of triangles. We have presented the fitting-plane approach, which creates the 3D model by fitting several planes to the input 3D points. The sensors simulation on this model was significantly faster, than the sensor simulation with other 3D models.

REFERENCES

(2000). *Evolutionary robotics*. MIT Press.

Amenta, N., Bern, M., and Kamvysselis, M. (1998). A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98. ACM.

Attene, M., Falcidieno, B., and Spagnuolo, M. (2006). Hierarchical mesh segmentation based on fitting primitives. *Vis. Comput.*, 22.

Blaer, P. and Alien, P. (2006). View planning for automated site modeling. In *Proceedings on Robotics and Automation, 2006. ICRA 2006*, 2621 –2626.

Chudoba, J., Mázl, R., and Přeučil, L. (2006). A Control System for Multi-Robotic Communities. In *ETFA 2006 Proceedings*, 827–832. IEEE, Piscataway.

DalleMole, V. and Araujo, A. (2008). The growing self-organizing surface map: Improvements. In *Neural Networks, 2008. SBRN '08. 10th Brazilian Symposium on*, 183 –188.

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

Fritzke, B. (1995). A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems 7*, 7, 625–632.

Fritzke, B. (1997). Some competitive learning methods. Technical report, Ruhr-Universitat Bochum.

Godbaz, J., Cree, M., and Dorrington, A. (2008). Mixed pixel return separation for a full-field ranger. In *Image and Vision Computing*.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.*, 26, 71–78.

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

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

Levi, P. and Kernbach, S. (eds.) (2010). *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*. Springer-Verlag.

Lutz Winkler and Heinz Wörn (2010). Modular Robot Simulation. *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, 133–165. Available at https://launchpad.net/robot3d.

Qin, H., Mandal, C., and Vemuri, B. (1998). Dynamic catmull-clark subdivision surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 4(3), 215 –229. doi:10.1109/2945.722296.

Rêgo, R.L.M.E., Araujo, A., and de Lima Neto, F. (2010). Growing self-reconstruction maps. *Neural Networks, IEEE Transactions on*, 21(2), 211 –223.

Šára, R. and Bajcsy, R. (1998). *Fish-scales: Representing fuzzy manifolds*, 811–817. Narosa Publishing House.

Yim, M., Shen, W.M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., and Chirikjian, G. (2007). Modular self-reconfigurable robot systems [grand challenges of robotics]. *Robotics Automation Magazine, IEEE*, 14(1), 43 –52.

Zykov, V., Williams, P., Lassabe, N., and Lipson, H. (2008). Molecubes extended: Diversifying capabilities of open-source modular robotics.