Symbiotic Sensor Networks in Complex Underwater Terrains: a Simulation Framework

Vadim Gerasimov, Gerry Healy, Mikhail Prokopenko, Peter Wang, and Astrid Zeman

CSIRO Information and Communication Technology Centre
Locked bag 17, North Ryde, NSW 1670, Australia
mikhail.prokopenko@csiro.au

Abstract. This paper presents a new multi-agent physics-based simulation framework (DISCOVERY), supporting experiments with self-organizing underwater sensor and actuator networks. DISCOVERY models mobile autonomous underwater vehicles, distributed sensor and actuator nodes, as well as multi-agent data-to-decision integration. The simulator is a real-time system using a discrete action model, fractal-based terrain modelling, with 3D visualization and an evaluation mode, allowing to compute various objective functions and metrics. The quantitative measures of multi-agent dynamics can be used as a feedback for evolving the agent behaviors. An evaluation of a simple simulated scenario with a heterogeneous team is also described.

1 Introduction

This paper presents a software simulation system for self-organizing underwater sensor and actuator networks. The simulation system provides a test-bed for co-evolution of distributed and mobile sensors and actuators, and required communication topologies. The broad aim is to develop *symbiotic* sensor/actuator networks which include agents recognizing and forming relationships of mutual benefit across various types: e.g., network nodes may assist navigation of submersible robots, while being powered by the robots.

The underwater sensor/actuator networks are intended to protect critical marine infrastructure and water resources. Examples of such safety-critical structures include offshore oil platforms, deep-ocean well heads, tankers, dams, bridges, pipelines, etc. Typical protection and response tasks comprise tracking oil spills to their sources, source identification and diagnostics (e.g., measurement of oil slick thickness), and actions such as burning, skimming, and dispersing. Offshore hydrocarbon exploration by sensor arrays for the identification of petroleum systems is another potential domain of interest.

Simulation of the underwater sensor/actuator networks should account for possible off-shore deployment on demand or in advance, as autonomous devices (nodes or submersible robots) with compound-specific chemical sensing, propulsion and actuation, acoustic and optical communication, and multi-agent self-organizing teamwork capabilities. An important scenario considered in this study is the deployment of a heterogeneous team with some primary agents (leaders) having different or more advanced sensors, and secondary agents with more powerful actuators following the leader(s) as a backup team until there is a need for distributed actuation. A prey-and-predator variant of this scenario is a search and containment task with the primary agent being the target

pursued by the secondary agents. In either case, the employed agents have to deal with a problem that changes concurrently with the problem-solving processes, and cooperate in solving tasks which are distributed over space (3D) and time, across complex underwater terrains. The development of self-organizing strategies, when an incremental loss of a portion of the network leads to an incremental loss in quality, rather than a catastrophic failure, is our main focus.

The following Section describes the simulation system created by the CSIRO DISCOVERY[1] project, developed to study symbiotic behavior in underwater self-organizing sensor and actuator networks. It is followed by preliminary experimental results (Section 3) and conclusions.

2 Simulation Platform

2.1 Architecture

The main requirements of the physics-based simulation of distributed multi-agent systems include simulation of mobile autonomous underwater vehicles, as well as distributed sensor and actuator nodes, and multi-agent *data-to-decision* (D2D) integration, ranging from data validation by individual agents to decision integration and action coordination by self-organizing sub-networks and networks of agents. An important part of the D2D integration is quantitative measures of multi-agent dynamics [13, 14, 6, 17] which can be used as a feedback for evolving the agent behaviors across multiple runs. Such measures can use either full information on agents' states and their interconnections or work with partial information, obtained locally: *localizable* measures [16]. Of course, localizable measures can be embedded in the agents themselves and be accessible to local "hierarchs" (e.g., cluster-heads [10, 15]), controlling agent behaviors during run-time via an adaptive feedback.

The DISCOVERY multi-agent physics-based simulation platform supports the following components: 1) Simulator; 2) Visualizer; 3) Agent; 4) Metric-Analyzer. A simulation session is carried out in client-server style. The Simulator (server) provides a domain (a virtual environment), simulates all the actions of objects in this domain and controls a scenario according to a set of rules. This is a well-known approach to simulation, used, for example, in the RoboCup Simulation League [9, 2, 1]. The characteristics of the Simulator are specified by a set of server parameters, e.g., the amount of noise added to sensory perceptions and the maximum speed of an agent.

Agents are controlled by autonomous client programs which connect to the server through a specified port. Each client program can control a single agent. All communication between the server and the clients is done via TCP/IP sockets. Using these sockets, client programs send requests to the server to perform an action (e.g. "thrust"). When the server receives such a message it handles the request and updates the environment accordingly. Upon an agent's request, the server also sends sensory information about the agent's neighbourhood to the agent. Clients communicate with each other indirectly, via the server, using messaging protocols which restrict the communication.

The server is a real-time system using a discrete action model, i.e., working with discrete time intervals (cycles) of a specified duration, e.g., 10 ms. During this period

[1] CSIRO: Commonwealth Scientific and Industrial Research Organisation, Australia.
DISCOVERY: Distributed Intelligence, Sensing and Coordination in Variable Environments.

clients can send requests for agent actions to the server. At the end of a cycle the server executes the actions and updates the state of the world. Sending no request during a given cycle means that the agent misses an opportunity to affect its current dynamics. Sensing and acting are asynchronous: clients can send action requests to the server once every cycle, but they receive information on requests. This information is fragmented, limited and degrades with the distance.

Simulator also supports an evaluation mode, allowing to compute objective functions and metrics. In this mode, each agent regularly updates Simulator with a predefined set of agent's internal parameters, enabling calculations on both local and global levels. For example, it is possible to compute entropy of agents' states and characterize diversity of their behaviors. In addition, the evaluation mode includes computation of spatiotemporal distribution of agents, etc. The collected data can be used by MetricAnalyzer off-line, and contribute to genetic algorithms evolving agents between experimental runs.

Visualizer displays the virtual world, being connected to the Simulator via TCP/IP. Although similar to an agent, it has no physical representation in the simulated environment and uses a different set of commands. The Simulator sends information to the Visualizer each cycle or upon request, containing the current state of the world. Visualizer also provides a visual interface to the server in order to specify, start, pause and stop a scenario.

The environment Simulator is the most computationally-intensive part of our software system where the performance is critical. There is a range of libraries available for physical simulations, however, we were not able to identify any library or simulator that would adequately cover a required combination of fluid dynamics and kinematics, as well as the right balance between the precision and performance of the simulation system. While we adopted some design and implementation ideas from the available 3D simulation systems such as ODE, Gazebo/Stage, Juice, Webots, the Simulator was developed based on our own set of routines satisfying own coding requirements and standards. A socket-based multi-agent sever-client communication suite (DBP-MAP: Deep Behaviour Projection Multi-Agent Platform), developed earlier by CSIRO [11, 12], was our starting point in developing communication architecture for the simulation system. The project also builds on the expertise developed within CSIRO Underwater Robotics [5], CSIRO Directed Self-Assembly in Multi-Agent Networks [8], and CSIRO-NASA Ageless Aerospace Vehicles [17, 13, 14] projects. Considering the requirement of cross-platform compatibility, Java3D is a good choice for the development of the combination of the rendering engine and user interface. As the Visualizer relays the data incoming from the Simulator to the native graphics engine without any significant calculations, Java performance penalty is not significant.

2.2 Terrain and Collision Modelling

Terrain of the seabed is modelled using a rectangular mesh. The terrain can be randomly generated or user-defined by creating a Height Map, a gray scale raster image file (pixel, not vector based), in which the RGB value of each pixel is mapped to a corresponding vertex height, thus creating a three dimensional mesh which represents the seabed.

In our terrain model vertices are equidistant along the x and z axis, while the height, or y, coordinate is either derived from a height map, or generated with frac-

tals, or produced as a combination of both these methods. The fractal technique used is the Diamond-Square algorithm [7], which is a form of mid-point displacement using a square base. Starting with the outside corner vertices, the height of the mid-point vertices is generated by averaging the surrounding vertices and adding a random displacement, first in diamond and then square step. The algorithm was modified to allow the seeding of the grid with values from a height map, thus resulting in a user defined terrain, smoothed and modified by noise. There are three methods of loading a terrain into the Simulator: a user-defined height map, a smoothed height map and a randomly generated terrain.

A user-defined height map can be of any width and height in pixels and is stretched to a fixed size in the Simulator. The RGB values are scaled by a factor of 0.2, with a value of 240 indicating sea level. For example, a value of 0 will be 48 meters below sea level, while a value of 255 will be 3 meters above sea level.

A smoothed height map must have a width and height of the form $2x + 1$, where x is an integer, i.e., height maps with odd-numbered widths and heights would be valid. A smoothed height map is handled similarly to a user-defined height map, except that the number of vertices used in the terrain mesh is expanded. The heights of these additional points are then generated using the Diamond-Square algorithm [7]. Figure 1 shows a resulting terrain. Finally, a random terrain can be generated, by taking four random vertices which are used as input into the Diamond-Square algorithm.

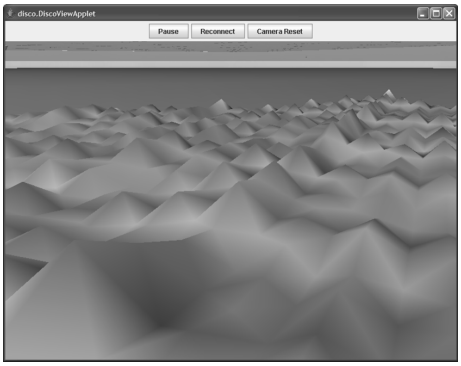

Fig. 1. An example underwater terrain obtained with a smoothed height map.

Terrain collision detection is implemented using Coldet, an Open Source (LGPL) Collision Detection library [3]. The library provides a number of collision detection techniques: intersecting polygons, spheres and rays. We model the terrain as a set of triangles (polygons), and predict collisions between agents and the terrain using a sphere centered on the agent's centre of mass.

When a collision is detected, the physical movement of the agent is altered. Collisions can be modelled as either elastic, where kinetic energy is preserved or non-elastic, where some of the kinetic energy of the colliding objects, is transformed into another form during the collision. In DISCOVERY collisions are modelled as elastic, although this can be modified to provide more realistic behaviour in the future.

The agent's velocity is then calculated at the impact point, considering the reduction in time taken to reach it. The agent's velocity is reflected against the surface plane of the terrain polygon where the impact took place.

2.3 Physics-based Simulation

A physically realistic dynamic modelling of volume of water, both soluble and insoluble contaminants present in water (crude oil, salts, etc.), insoluble contaminants on water surface (crude oil), and underwater robots can be achieved only as a balance between physical accuracy and computational performance. We selected an appropriate adjustable scale of the model, including the simulation grid step, the time step, and an option of selectively disabling certain simulation features to speed up some experiments: e.g., a purely underwater simulation can be done without water surface calculations. If an experiment does not require precise water dynamics, the system can assign constant parameters, such as current direction and rotation, to the water volume grid. At the moment, insoluble liquid contaminants moving in the water are simulated as solid objects affected by buoyancy, gravity, and currents in a normal force-momentum-position cycle. The water surface and surface-bound contaminants, such as crude oil slicks, can be either linked to the water grid or simulated separately (the method currently used in DISCOVERY) from the body of water as weight-spring meshes (updated in a normal force-momentum-position cycle of the physical simulation).

2.4 Simulator-Agent Sensor Protocol

In principle, every environment variable maintained by the Simulator may be perceivable by an agent. One of the DISCOVERY objectives is to design, select and verify a correct set of relevant sensors suitable for a multi-agent task, and couple them with available actuators. In the initial setup, an agent has a number of sensors: e.g., a chemical sensor, a temperature sensor, a pressure sensor, a conductance sensor, a flow sensor, an internal battery sensor, an accelerometer, a compass, a collision sensor, a sonar sensor and a communication (acoustic and optical) sensor. The Simulator provides sensory data to agents upon request.

Actuators include thrusts, fins, a sonar, and communication devices (acoustic and optical). As a result of experiments, new sensors and actuators may be added and some of the listed sensors and actuators may be suppressed. Thrust allows the agent to accelerate both positively and negatively in a particular direction. A specified force provides acceleration in the direction of the thrust actuator. Three thrust actuators need to be constructed for movement within the (x, y, z) plane. Fin allows a moving agent to change its current direction. Each agent has a particular buoyancy value which allows it to float towards the surface of the water when there is no downward thrust.

3 Preliminary Experiments

The main difficulty in tracking and identification of an underwater source of contamination is that the insoluble contaminants (e.g., oil bubbles) can be sensed mostly only locally and may rise to the surface quite a distance away, being shifted by currents, winds, etc. The problem is complicated by non-trivial dynamics of underwater plumes and bubbles, in particular tracking of their gradients, as well as complexities of the underwater terrain, and may require heterogeneous teams of agents with distributed sensing and actuation.

In the context of offshore hydrocarbon exploration, identification of petroleum systems presents an additional challenge. It is well-known that hydrocarbon can be entrapped in sub-terrain reservoirs formed in non-porous rock. Exploring features of interest in such complex environments by a sensor/actuator network is likely to require

heterogeneous teams: not only to spread across a large area, but also to combine different sensing and actuation modalities (from sonars to magnetic and seismic surveys).

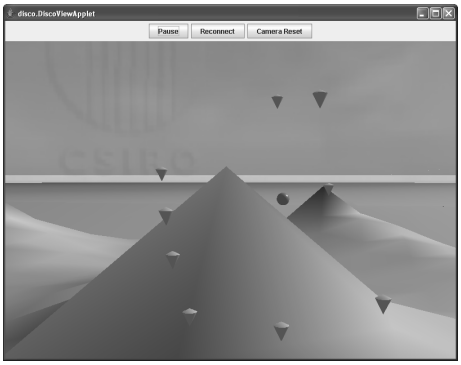

Fig. 2. An emergent spherical surface around the Explorer (a small sphere in the center), with several Supporter agents (conically shaped).

To illustrate the simulator capabilities, we designed a scenario with a heterogeneous team (one Explorer and ten Supporter agents). The Explorer is a task-oriented agent which has a goal of finding a proxy to a feature of interest (e.g, shallowest point in the local terrain, a chemical anomaly sensed by a combination of chemical sensors, etc.). While moving in the environment, it also emits an omnidirectional signal which can be used by Supporter agents to detect its direction, if there is an unobstructed line of signal between the Explorer and the receiving agent. The Supporter agents have the same actuators as the Explorer, with addition of sensors which are able to detect the signal emitted by the Explorer. The Supporters follow two simple rules: (i) move along the direction in which the Explorer is sensed until a specified range is reached within a tolerance limit; (ii) when the distance to the Explorer is less than the specified range, move away from the Explorer. If the signal from the Explorer is not received, a Supporter agent treats the last direction as valid, and may use extra-thrust in following this direction ("extra-thrust behaviour"). We also experimented with the third behaviour: (iii) within the tolerance limit of the specified range, maintain it by moving orthogonally to the direction of the Explorer. The rules (i)-(iii) lead to emergence of a spherical surface around the Explorer, on which the Supporter agents randomly move in order to maintain the range (Figure 2). There is a difference between two kinds of emergence: pattern formation and intrinsic emergence, distinguished by Crutchfield [4]: a) pattern formation refers to an external observer who is able to recognize how unexpected features (patterns) 'emerge' or 'self-organize' during a process (e.g., spiral waves in oscillating chemical reactions) — these patterns may not have specific meaning within the system, but obtain a special meaning to the observer when detected; b) intrinsic emergence refers to the emergent features which are important within the system because they confer additional functionality to the system itself, like supporting global coordination and computation (e.g., the emergence of coordinated behaviour in a flock of birds allows efficient global information processing through local interaction, which benefits individual agents). To verify whether the emergence of a spherical surface is intrinsic and contributes to the quality of the supporting task, we run a number of experiments.

The Metric-Analyzer automates the scenario by setting simulation parameters, repeatedly running the Simulator and agents for each experiment, logging the relevant

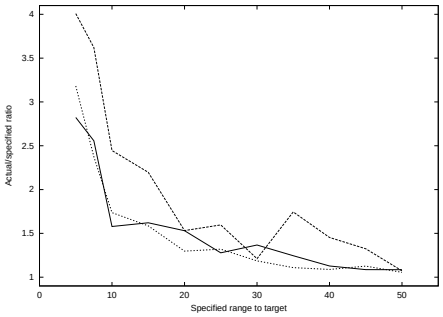

Fig. 3. The actual/specified ratio: sphere-tracing behavior (solid line), extra-thrust behaviour (dashed line), simple behavior with rules (i) and (ii) only (dotted line).

information (velocities, distances, etc.), and collating the results (the overall experiment runs for several days). In the Explorer-Supporters scenario it is important to have the Supporter agents maintaining the specified range over time, i.e. the actual achieved range should be as close as possible to the specified one. The actual/specified ratio, averaged over the team of the Supporter agents and over time after a certain initial interval, is plotted in Figure 3 against different specified ranges. The results of multiple experiments deployed in the same terrain indicate the difficulty of maintaining close ranges (within $[5; 10]$ meters), for a team of Supporter agents. The Supporter agent closest to the Explorer always attains the specified range (the ratio is close to 1.0). It is worth pointing out that it is not the same agent but rather the one which is the closest at a given time, i.e. there is always at least one Supporter agent directly observing the Explorer. We observed that a) the extra-thrust behavior does not attain better quality, and b) the emergence of a spherical surface is not intrinsic: it does not contribute to the quality of the supporting task, measured in terms of the actual/specified ratio. A spherical surface may, however, be beneficial if an equidistant spatial spread is important.

4 Conclusions

This paper presented a new multi-agent physics-based simulation framework (DISCOVERY), supporting experiments with self-organizing underwater sensor and actuator networks. The simulation system provides a test-bed for co-evolution of distributed and mobile sensors and actuators, and required communication topologies in challenging scenarios. In order to illustrate its capabilities, we briefly described a simple simulated scenario with a heterogeneous team (the Explorer-Supporters scenario), and its evaluation by Metric-Analyzer. Some of the tasks for future experiments include: identification of hazards to safety-critical structures; response to contamination of water supplies such as oil spills; perimeter formation for absorption barriers and traffic exclusion zones; collection, storage, transportation and analysis of contaminated items/evidence. The DISCOVERY Simulator is intended to be a flexible tool for simulation, quantitative analysis and design of complex underwater sensor and actuator networks, with varying degrees of autonomy, decentralized control, and data-to-decision integration.

References

1. de Boer, R. and Kok, J.R. The Incremental Development of a Synthetic Multi-Agent System: The UvA Trilearn 2001 Robotic Soccer Simulation Team. Master's Thesis, University of Amsterdam, The Netherlands, 2002.

2. Butler, M., Prokopenko, M., Howard, T. Flexible Synchronisation within RoboCup Environment: a Comparative Analysis. In P. Stone, T. R. Balch, G. K. Kraetzschmar (eds.) *RoboCup 2000: Robot Soccer World Cup IV*, LNCS, Vol. 2019, 119–128, Springer, 2001.
3. Coldet: an Open Source Collision Detection library. http://www.photoneffect.com/coldet/
4. Crutchfield J. The Calculi of Emergence: Computation, Dynamics, and Induction. *Physica D*, 75, 11–54, 1994.
5. Dunbabin, M., Roberts, J., Usher, K., Winstanley, G., Corke, P. A Hybrid AUV Design for Shallow Water Reef Navigation. IEEE International Conference on Robotics and Automation, 2117–2122, Barcelona, Spain, 2005.
6. Foreman, M., Prokopenko, M., Wang, P. Phase Transitions in Self-organising Sensor Networks. In Banzhaf, W., Christaller, T., Dittrich, P., Kim, J.T. Ziegler, J. (eds.) *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life*, LNCS, Vol. 2801, Springer, 781–791, 2003.
7. Fournier, A., Fussell, D., Carpenter, L. Computer Rendering of Stochastic Models. *Communications of the ACM*, 25(6), 371–384, 1982.
8. Gerasimov, V., Guo, Y., James, G. C., and Poulton, G. T. Physically Realistic Self-assembly Simulation System. In A. Abraham, C. Grosan and V. Ramos (eds.), *Stigmergic Optimization, Studies in Computational Intelligence*, 117–130, Springer, 2006.
9. Kitano, H., Tambe, M., Stone, P., Veloso, M., Coradeschi, S., Osawa, E., Matsubara, H., Noda, I. and Asada, M. The RoboCup Synthetic Agent Challenge. In Proceedings of the 15th International Joint Conference on Artificial Intelligence, 1997.
10. Mahendra, P., Prokopenko, M., Wang, P., Price, D.C. Towards Adaptive Clustering in Self-monitoring Multi-Agent Networks. In R. Khosla, R. J. Howlett, L. C. Jain (eds.) *Knowledge-Based Intelligent Information and Engineering Systems, 9th International Conference, KES 2005, Melbourne, Australia, Proceedings, Part II*, LNCS, Vol. 3682, 796–805, 2005.
11. Prokopenko, M., Wang, P., Howard, T. Cyberoos'2001: 'Deep Behaviour Projection' Agent Architecture. In A. Birk, S, Coradeschi, S, Tadokoro (eds.) *RoboCup 2001: Robot Soccer World Cup V*, LNCS, Vol. 2377, 507–510, Springer, 2002.
12. Prokopenko, M., Wang, P. Relating the Entropy of Joint Beliefs to Multi-Agent Coordination. In G. A. Kaminka, P. U. Lima, Raúl Rojas (eds.) *RoboCup 2002: Robot Soccer World Cup VI*, LNCS, Vol. 2752, 367–374, Springer, 2003.
13. Prokopenko, M., Wang, P., Price, D.C., Valencia, P., Foreman, M., Farmer, A.J. Self-organising Hierarchies in Sensor and Communication Networks. *Artificial Life*, Special issue on Dynamic Hierarchies, Vol. 11(4), 407–426, 2005.
14. Prokopenko, M., Wang, P., Foreman, M., Valencia, P., Price, D., Poulton, G. On connectivity of reconfigurable impact networks in ageless aerospace vehicles. *Journal of Robotics and Autonomous Systems*, Vol. 53, 36–58, 2005.
15. Prokopenko, M., Mahendra, P., Wang, P. On Convergence of Dynamic Cluster Formation in Multi-Agent Networks. In M. S. Capcarrère, A. A. Freitas, P. J. Bentley, C. G. Johnson, J. Timmis (eds.) *Advances in Artificial Life, 8th European Conference, ECAL 2005, Canterbury, UK, September 5-9, 2005, Proceedings*, LNCS, Vol. 3630, 884–894, 2005.
16. Prokopenko, M., Wang, P., Price, D. Complexity Metrics for Self-monitoring Impact Sensing Networks, Proceedings of 2005 NASA/DoD Conference on Evolvable Hardware (EH-05), Washington D.C., USA, 2005.
17. Prokopenko, M., Poulton, G. T., Price, D. C., Wang, P., Valencia, P., Hoschke, N., Farmer, A. J., Hedley, M., Lewis, C., and Scott, D. A. Self-organising impact sensing networks in robust aerospace vehicles. In Fulcher, J. (ed.) *Advances in Applied Artificial Intelligence*, 186-223, Idea Group Inc., 2006.