IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 5, Ver. IV (Sep. - Oct. 2016), PP 94-101
www.iosrjournals.org

Prototyping and Simulation of Robot Group Intelligence using Kohonen Networks

Zhijun Wang, Reza Mirdamadi, and Qing Wang
Department of Computer Science, Mathematics, and Engineering
Shepherd University, Shepherdstown, WV25443, USA

Abstract:*Intelligent agents such as robots can form ad hoc networks and replace human being in many dangerous scenarios such as a complicated disaster relief site. This project prototypes and builds a computer simulator to simulate robot kinetics,unsupervised learning using Kohonen networks, as well as group intelligence when an ad hoc network is formed.Each robot is modeled using an object with a simple set of attributes and methods that define its internal states and possible actions it may take under certain circumstances. As the result, simple, reliable, and affordable robots can be deployed to form the network. The simulator simulates a group of robots as an unsupervised learning unit and tests the learning results under scenarios with different complexities. The simulation results show that a group of robots could demonstrate highly collaborative behavior on a complex terrain. This study could potentiallyprovide a software simulation platform for testing individual and group capability of robotsbefore the design process and manufacturing of robots. Therefore, results of the project have the potential to reduce the cost and improve the efficiency of robot design and building.*

Keywords:*Robot Simulation,Mobile Ad Hoc Networks, Swarm Intelligence, Unsupervised Learning, Self-Organizing Maps, Kohonen Networks.*

I. Introduction

Industries have benefited drastically from the robotic work force, such as assisted surgical robots in the medical industry and assembly line robots in car factories. Robots have taken over the duties of dangerous and tedious jobs from workers, resulting in greater manufacturing precision and higher productivity. The discipline of robotics involves researches in many related fields such as dynamics, engineering, machine learning, computer vision, human-computer interaction, and neural networks.

A robot is composed of a set of hardware components, such as motors, controllers, sensors, processing units, as well as software components that control its motion and reactions. One important aspect is robot motion. Many kinds of robots have to walk or run on some surfaces to finish tasks. Effective design of robot hardware and software components is critical for robot motion. Due to the complexity of the problem, models are used to focus on major factors as the first-degree approximation. Minor aspects can be added and addressed gradually. A group of robots can collaborate and form an ad hoc network [1-3]. The network can use machine learning techniques, learn from the environment, and act as a unit. This kind of networks has many potential applications, such as a wide-ranged disaster relief site, a large battle field, and the sea bottom terrain exploration. Each robot is equipped with a GPS system besides a computer system, which is fairly powerful these days, even for the economic ones. Sensors that detect temperature, humidity, air qualify, level of debris, elevation, and trace of lives can be equipped depending on specific application requirements.

There are many commercially available robots that can be adapted to suit many scenarios. For example, the Baxter robot from Rethink Robotics comes with the direct programming access via a standard ROS API interface [4]. It comes with a Linux development workstation with sample programs for researchers to learn and use.The Baxter is an affordable humanoid robot platform with two seven-axis arms, integrated cameras, sonar, torque sensors, and accelerometers. It is entirely safe to operate around humans without the need for safety cages or other guarding equipment. With its precision and power Baxtor offers, it can serve as a good candidate with a few minor modifications and additions.

A software robot model starts with basic components using the hierarchical modeling method. While current robots have various types and shapes, this project focuses on human-like robots, which means, the model starts with basic components such as upper and lower arms, upper and lower legs, a head, and a torso. Fig. 1 showsa simple computer graphics model of the robot.

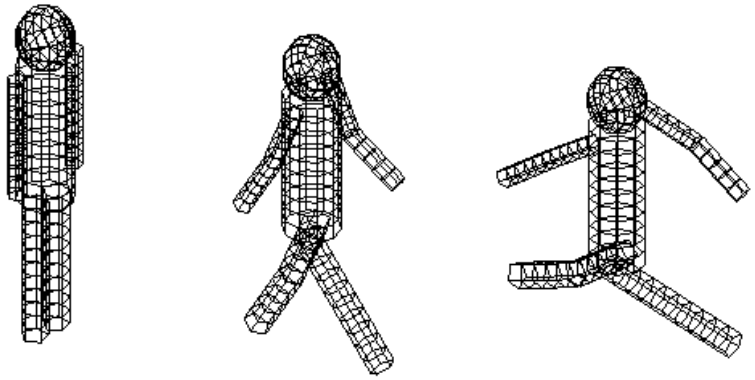

Figure 1.Computer Graphics Model of Robot and Robot Motion

The modeling process starts with the above described model and adds kinetic dependencies of robotic components so they can be simulated in a software environment. This will help the design and building the hardware portion of the robot and provide an efficient way to conduct simulation for robot motion and dependencies among components. A second-generation graphics model includes more controls and environmental factors such as terrains, textures, and lights. A virtual floor is added to serve as a ground for the robots and objects. An appropriate number of polygons are used for every 3D object to balance the efficiency and visual effect of the implementation program.User interaction and control features are designed and implemented so a robot designer can use the simulator to try different options before the physical building the robot parts. In addition, the software simulation classes may be reused or adapted by the robot as the built-in software components.

The rest of paper is organized as follows. Section II discusses related neural networks implemented in the simulation; Section III explains the prototyping and modeling methods of robots and robot ad hoc networks; Section IV covers simulation design and implementation techniques; Sections V demonstrates simulation results with analysis; Section VI concludes the paper and discusses possible future work.

II. Related Neural Networks

Neural network techniques and machine learning algorithms can be incorporatedinto robots to help them learn how to learn to walk properly and adapt to certain surfaces, as well as to acquire possible tasks and set priorities for them. Other aspects of artificial intelligence such as pattern recognition algorithmscan be added as well. This project serves as a starting point of a sophisticated simulation platform of robotic motion, learning, pattern recognition, and other robotic features that improve collaboration of robots as a team for high productivity in certain environments.

Machine learning algorithms are the major artificial intelligence components that are simulated in this project. There are two major kinds of machine learning algorithms - supervised learning and unsupervised learning. Supervised learning algorithms define desired outputs or patterns, use training data to train neural networks, and apply the trained networks in the real-life scenario. However, in an unknown domain such as a disaster site or the sea bottom, the terrain and tasks are unknown so it is impractical to use supervised learning. In this case robots have to explore the terrain, detect patterns and tasks, and come up with topographic maps by themselves. Unsupervised learning, in which the network learns without predefined patterns and forms their ownclassifications of features of the unknown domain.

Quite a few unsupervised learning algorithms are fairly good candidates. One of them is based on competitive learning,in which the neurons compete for possible activation. The winning neuron activates; the neighboring neurons also adapts to some extents based on a statistical distribution, either a Gaussian or Poisson one. During the learning process, the neurons are forced to organize themselves according to environments and the network is thus called a Self-Organizing Map (SOM) [5-7]. In an ad hoc network, each neuron can be an intelligent agent such as a robot.

The goal of an SOM in this study is to transform a high-dimensional data pattern into a two dimensional discrete map so it is easy to handle by the network. The transformation is performed in an orderly manner to obtain a topographic map. In this project, a specific kind of SOMscalled Kohonen Network is considered. Kohonen networks have a feed-forward structure with onetwo-dimensional discreet computational layer. Each neuron is connected to all the source nodes in the input layer. Fig.2 shows the structure of a typical Kohonen network.

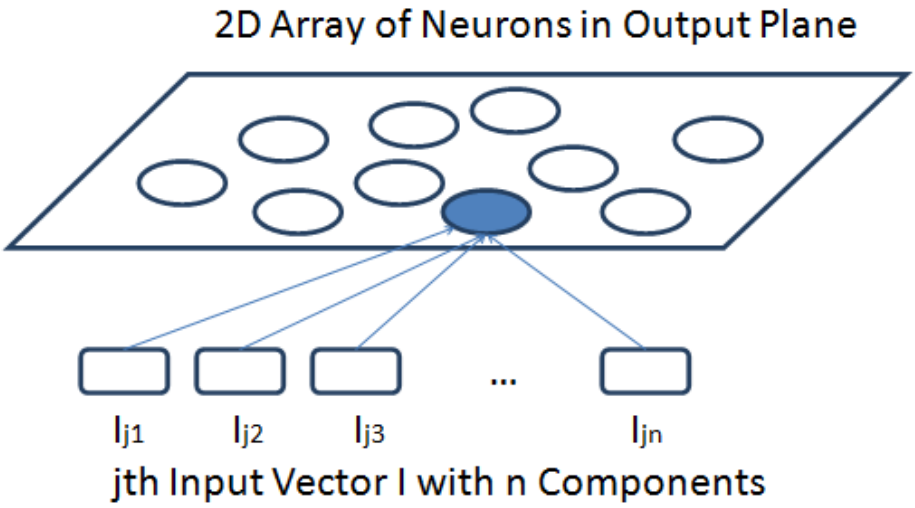

Figure 2. Structure of a Typical Kohonen Network with a High-Dimensional Input Space and a Two-Dimensional Output Space

The goal of a Kohonen network is to map the high-dimensional terrain data to a low-dimensional spatially discrete topographic map. The Kohonen network algorithm defines a non-linear transformation and has the following procedures [8, 9].

1. Use vectors to represent weight components for neurons in the network.
2. Initialize the vectors using random numbers in [0, 1].
3. Sample a random input vector from the input space.
4. Find the output neuron which is closest to the sampled input vector.
5. Update weights using a delta learning rule with a predefined learning rate.

If the weights stop changing, the process converges and network is trained.
If the predefined number of iterations passes, report non-convergence.
Else repeat procedures 3 to 5.

In the above algorithm a few components are critical. One of them is the evaluation method used in competition.An evaluation function has to be designed to rank the neurons in terms of the input vector and to find the winning neuron. Assume the vectors have n dimensions, an n-dimensional Euclidean distance between the input vector and the neuron weight vector can be used as the evaluation function as shown in (1).

$$f_j(\vec{I}) = \sum_{d=1}^{n}(I_d - w_{jd})^2 \qquad (1)$$

$\vec{I}$is the input vector. w_{jd} is the *dth* component of the *jth* neuron. A statistical distribution, usually the Gaussian distribution is used to select neighbor neurons to update. Therefore, the winning neuron and its neighbors update their weights to adapt to the input vector so they are forced to order themselves to adapt the input space. The learning rule is defined by (2).

$$\Delta w_{jd} = c(t)g(t)(I_d - w_{jd}) \qquad (2)$$

$g(t)$is the normal distribution function centered at the winning neuron.*c(t)* is the learning rate function that is defined by (3)

$$c(t) = c_0 e^{-\frac{t}{t_o}} \qquad (3)$$

$c_0$andt_0 are constants. $c_0$is in the range of [0, 1] while t_0 depends on the convergence time. For the convergence process, one of three scenarios could happen like other neural network training processes.

1. The weights stop changing and the training process converges with a genuine representation of the input space.
2. The weights stop changing but the trained network has defects that do not reflect certain or all features of the input space.
3. The maximum number of iterations passes but convergence is not reached. The training fails. This case happens rarely in a Kohonen network with appropriate setups.

In the one-dimensional case, a topological ordering, which corresponds to a Directed Acyclic Graph (DAG), can be used to represent the order of tasks. The featured topographic map generated by a Kohonen network is topologically ordered in the two-dimensional plane; in which the spatial location of a neuron in the output space corresponds to a feature of the input space.The topological ordering of neurons is caused by the weight update equation (2) that forces the weights of the winning neuron and related neighbors to move toward the input vector. The weight updates result in the reordering of neurons according to input patterns. Therefore, the two-dimensional output pattern reflects the features of the high-dimensional input terrain or space but with a much simpler representation which is able to be utilized easily by the network. Both the input domain and the output network are usually not intuitively obvious to human being; however, neurons or robots are able to use the information to identify terrain features, find optimal routes, and prioritize tasks to be performed.

III. Prototyping and Modeling

3.1 Technical Aspects of Robot Prototyping and Modeling

This project involves both design and implementation of the simulator. Efficiently designing the artificial intelligence system involves a comprehensive research on current resources and technologies. The working and efficient simulation engine is designed and implemented using C++ and the cross-platform 3D graphics library OpenGL, which is rich in user interactive features via the GLUT windowing library. The open source version of freeglut is used for implementation.

The class Robot models common attributes and actions of robots in the network. Attributes include robot dimension, mobility, a linked list of arrays and a hash table as the storage mechanism, sensor types, and transmission range. Actions a robot can perform include transmitting message, processing received message, translating on the terrain, updating the shared neural network, and synchronizing with the network. The robot motion modeling starts with two fundamental motion types – translation and rotation. Translation moves a robot component by certain amounts along x, y, and z axes in a Cartesian coordinate system. Rotation rotates a component along an axis in space, which is calculated using Euler angles. Since each component is connected to other components, the freedom of motion can be limited. The robot network is modeled by an aggregation of robot instances.

3.2 Mobile Ad Hoc Network Modeling

In a mobile ad hoc network, each node in the network serves as a router that discovers routes, maintains routes, and forwards packets. During the past few years, mobile ad hoc networks have received intensive attention because of their widespread applications. Examples include indoor networking among handhelds and notebooks for a conference, and outdoor networking in a military operation as well as the temporary network set up for a rescue operation after a major disaster. The common characteristic among those applications is that a fixed network infrastructure is not readily available [2, 3]. The mobility of robots is modeled using the random walk model. Robots have speeds in the two orthogonal directions with random values, which are within certain ranges. A fairly good coverage of the terrain can be reached using the random walk model. If the network asks a robot to move due to the learning process, the robot will update its position according to the network requirement.

IV. Simulation Design and Implementation

4.1Design and Implementation of the Motion Model

The hierarchical dependencies among the ten components of a robot are designed and implemented, in which torso is designed as the central component that controls other components. The translation and rotation of the head depends on the torso. The movement of left and right upper arms and legs depend on the torso; while the movement of the left and right lower arms and legs depend on the upper ones.

As of the implementation and programming part, the following features are used. Firstly, a tree structure is used to model the hierarchical structure of the robot components. Secondly, OpenGL states are used to store the states of movement for each component. The push and pop matrix feature is used to implement the dependencies of movements. Programming is done in C++ with OpenGL. C++ is adopted since robots are modeled using a class and instantiated as instances of the class.

4.2 Computer Graphics Technologies

State of art technologies in computer graphics are incorporated in the project to offer more features and flexibilities in the design and implementation process. For example, programmable rendering pipelines are researched for flexibilities and efficiency for the project.The new computer graphics technologies such as the OpenGL Shading Language (GLSL) have the potential to make the simulator more flexible, efficient, and powerful. The project also researches on the possibilities to include them. The simulation program is tested for speed and efficiency under difference circumstances with various textures, lights, and material properties. The

number of polygons used in a scene is an important parameter which affects both the smoothness of the visual effect and the speed of the simulation.

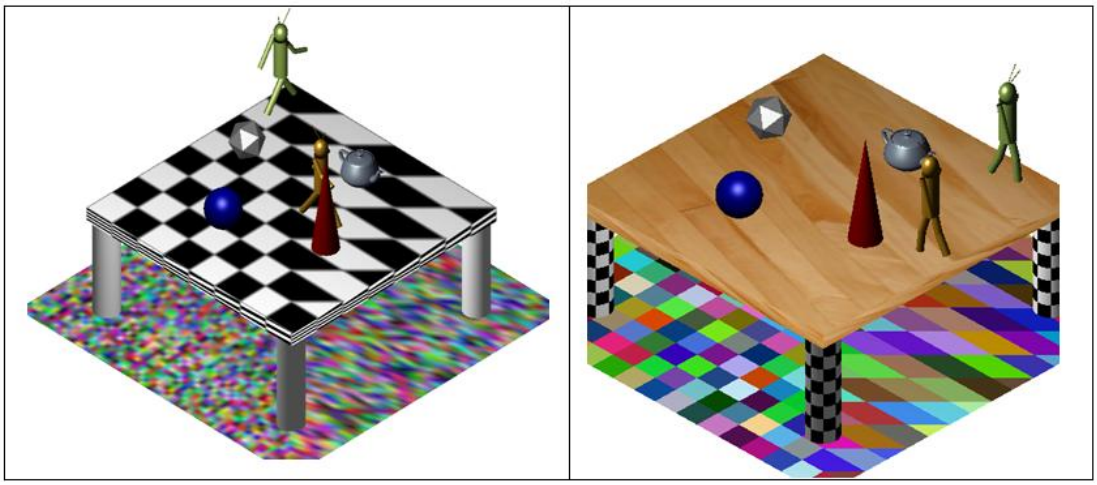

Figure 3. Graphics Modeling of Robots with Objects, Lights, Materials, and Textures

4.3 Artificial Intelligence Components

After the graphic component of the simulation is designed and implemented, the next generation of simulators are be designed and implemented to incorporate neural network and machine learning components. Firstly, a few fundamental artificial intelligence learning algorithms have been experimented such as game playing techniques using the state space framework, error-checking methods, and neural network methods such as Hopfield networks and Bidirectional Associative Networks (BAM). The Kohonen network is chosen in the final version of the simulator.

V. Simulation Results

5.1 Unsupervised Learning

The simulation starts with a simple case in which the terrain is divided into n^2 squares so shades form a pattern with red, green, and blue components, which is three-dimensional. Fig. 4 shows a deployment of 12 robots on the terrain with 2,500 squares. This is a classical application of Kohonen networks. The robots can train themselves to converge to the correct two-dimensional color pattern within 2,000 iterations in 98% of all simulation runs. The percentage increases to 99.3% with 6,000 iterations.

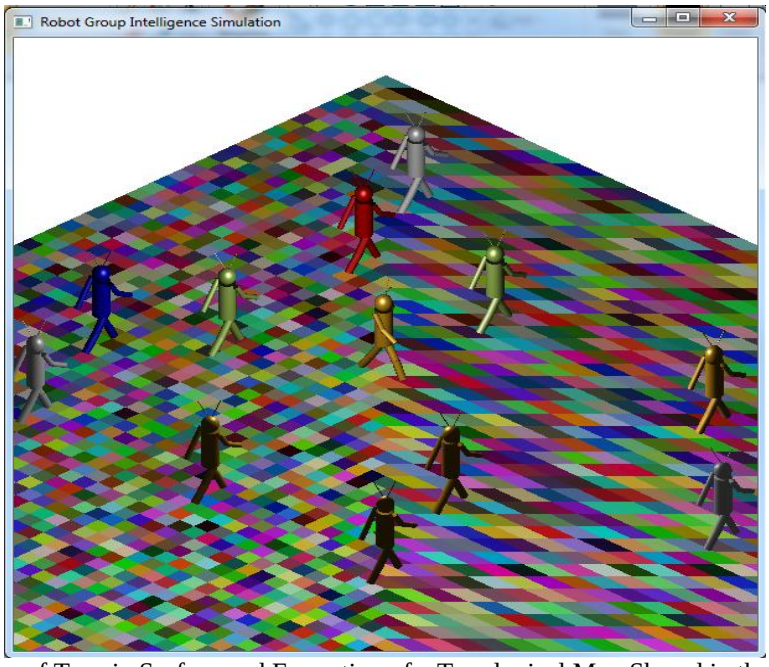

Figure 4. Detection of Terrain Surface and Formation of a Topological Map Shared in the Ad Hoc Network

However, when the terrain surface becomes more complicated, the non-convergence rate for the same number of iterations increases steeply. Firstly, the borders among squares on the terrain are blurred using the linear-interpolation approach. Secondly, three cones that represent tasks on the terrains are added. Fig. 5 shows this scenario. It turns out that in over 60% of all cases, the process cannot converge in 10,000 iterations. To address the issue, the number of robots is considered to be the bottleneck for the performance. We consider increasing the number of robots to form a bigger group, or a swarm, in which individuals collaborate and produce particle swarm intelligence. This team intelligent behavior is to be demonstrated in the next subsection with simulation results.

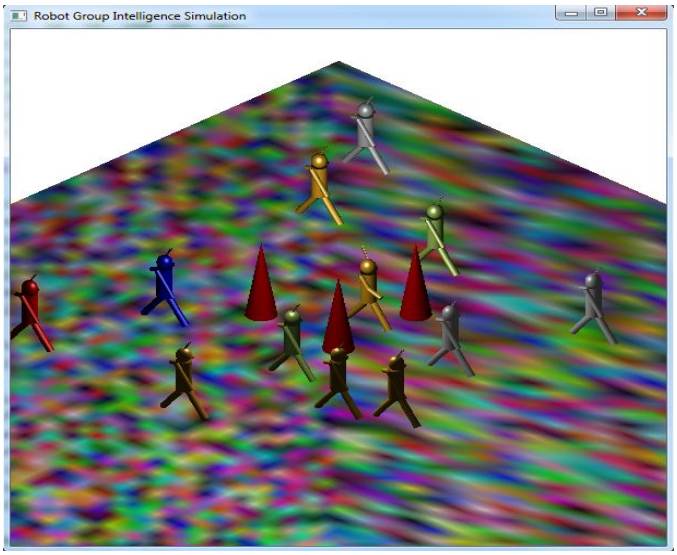

Figure 5. Terrain with Same Resolution but Blurred Border Feature and Task Objects on Terrain

5.2 Group Intelligent Behaviors

Fig. 6 shows the simulation result with 90 robots on a terrain divided into 10,000 squares with blurred borders using the linear-interpolation approach. Three red task cones, which have higher task priority, and two blue spheres, which represent tasks with lower priority, are added to the terrain to challenge the intelligent agent group. In 3,000 iterations the robots are able to form a topographic map that approximately reflects the task distribution on the terrain with more robots surrounding the tasks with the higher priority. The result on a complex terrain is not perfect in most simulation runs. However, most results are fairly good in a statistical sense. Therefore, in real-life application scenarios, a team of robots with an appropriate number for the corresponding terrain and tasks could be deployed so they could form an effective ad hoc network, learn from the terrain and tasks by themselves, and carry out the necessary tasks according to their priorities.

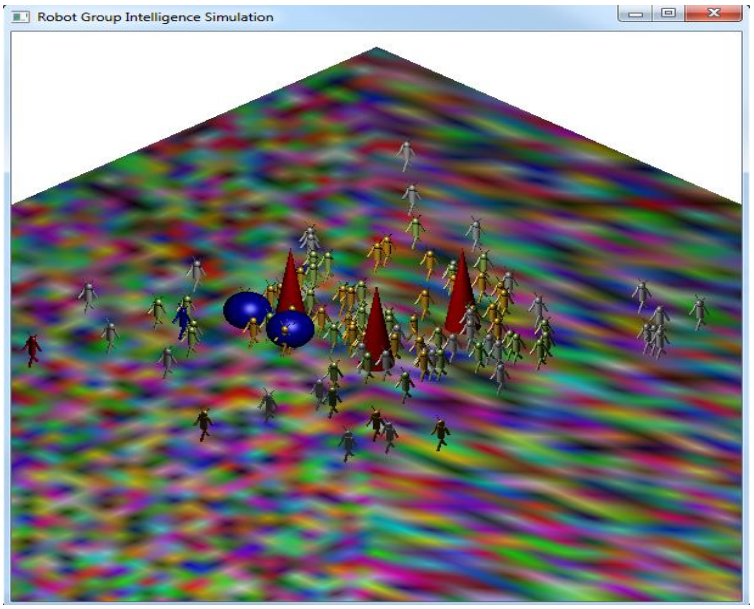

Figure 6. Improved Collective Behavior Demonstrated by Larger Group of Robots

A systematic study has been conducted to study the relationship between the complexity of terrain with prioritized tasks and the convergence rate within a reasonable time frame or number of iterations. Because in reality, robots do not have an infinite amount of time to learn without supervision, although doing so almost guarantees a successful training. Therefore, a tradeoff must be made between the team learning time and the number of robots, which is linearly proportional to the cost to build the network. Fig. 7 shows the simulation scenario with 100,000 blurred squares and more task objects on the terrain. The result shows that a group of 120 robots is able to achieve a similar convergence ratio to the scenario shown in Fig. 6.

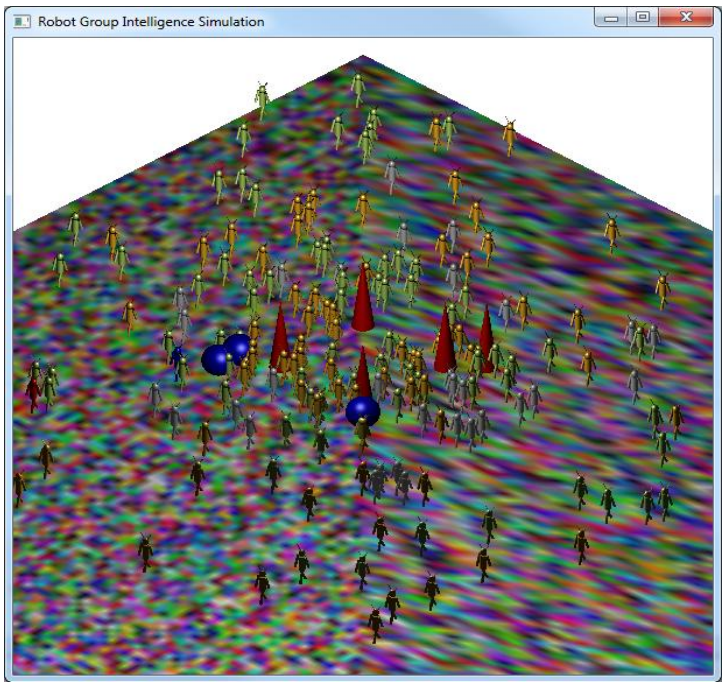

Figure 7. Linear Relationship between Terrain Complexity and Number of Robots

Relationship between the convergence rate and the number of robots was studied. Fig. 8 shows a typical curve between these two variables. The terrain resolution is 10,000 and the time threshold is 2,000 iterations for Fig. 8, which shows the convergence rate as a function of the number of intelligent agents in the network. The convergence rate increases rapidly when the number of intelligent agents is small and reaches a plateau when the number goes beyond a certain limit. For the scenario shown in the above figure, the magic number is around 120. Simulation results under other terrain resolutions reveal a similar trend. Therefore, a simulation can be conducted to decide on an appropriate number of intelligent agents to be deployed, which can maximize the productivity and minimize the total cost.

Fig. 9 indicates that the time it takes to adapt to a terrain is approximately proportional to the complexity of the terrain, which is defined by the number different shaded spots that represent terrain features. In the scenario shown in Fig. 9, the number of robots is 120.

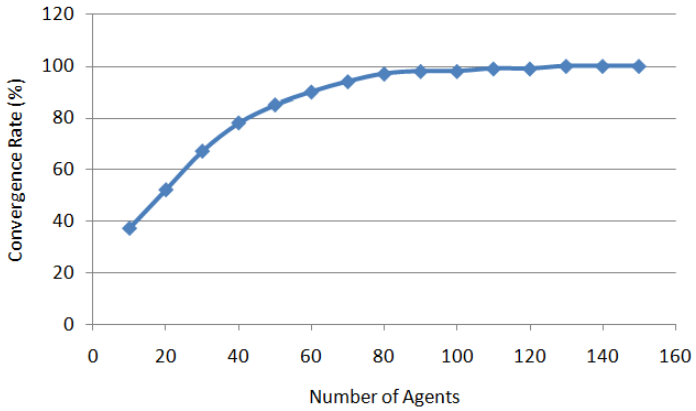

Figure 8. Relationship between Kohonen Network Training Convergence Rate and Number of Intelligent Agents

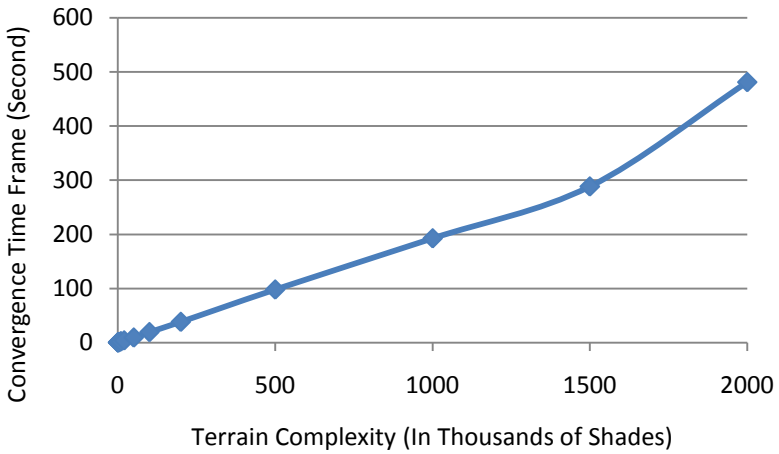

Figure 9. Relationship between Simulation Time Frame and Terrain Complexity

5.3 Simulation Efficiency

Based on a systematic series of tests, it is found out that the single most important factor that affects the speed and efficiency of the simulation is the number of polygons that needs to be rendered in each scene, especially for a dynamic scenario, under which it is not viable to maintain a large amount of static pixel data in the frame buffer. The whole scene needs to be rendered for every frame during the animation process, which costs most of the system resources such as CPU cycles, memory, and GPU cycles as well as the consumption of the onboard memory. A focus is given to the learning process, which results in less than perfect visual effects.

VI. Conclusion and Future Work

A robot motion and learning simulation program is designed and implemented to research on the modeling of robot and robot group behavior using unsupervised machine learning techniques. Computer graphics features such as hierarchical modeling,lights, material, texture mapping, the programmable graphics pipeline features, as well as artificial intelligence components are considered and incorporated in the simulator. The simulation results reveal the relationship among terrain complexity, the machine learning time frame, and the number of robots that form the ad hoc network. Possible future work includes finding better methods to run the artificial intelligence components in order to reduce the CPU, memory, and GPU usage, and implementing the simulation program under the new GPU technologies to speed up the simulation. Newer computer graphics technologies and more artificial intelligence features can be incorporated in the system in a future version as well. The simulation system can also evolve into a platform for robot design and testing under other platforms.

References

[1]. P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, Routing with Guaranteed Delivery in Ad Hoc Wireless Networks, *Wireless Networks, (7)* 2001, 609-616.
[2]. B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, SPAN: An Energy-efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. *Proc. MobiCom*, July 2001, 85-96.
[3]. I. Kang and R. Poovendran, On the Lifetime Extension of Energy-efficient Multihop Broadcast Networks, *Proc. IJCNN*, May 2002, 365-370.
[4]. http://www.rethinkrobotics.com.
[5]. K. Teuvo, Self-Organized Formation of Topologically Correct Feature Maps, *Biological Cybernetics. 43 (1)*, 1982, 59–69.
[6]. T. Heskes, Energy Functions for Self-Organizing Maps, *Kohonen Maps*, Elsevier, 1999.
[7]. A. N. Gorban, A. Zinovyev, Principal manifolds and graphs in practice: from molecular biology to dynamical systems, *International Journal of Neural Systems, 20(3)*, 2010, 219–232.
[8]. A. Ciampi, Y. Lechevallier, Clustering large, multi-level data sets: An approach based on Kohonen self-organizing maps, *PKDD 2000, vol. 1910*, 2000, 353-358.
[9]. Y. Liu and R.H. Weisberg, A review of self-organizing map applications in meteorology and oceanography. *Self-Organizing Maps-Applications and Novel Algorithm Design*, 2011, 253-272.