Control and Simulation of Robotic Swarms in Heterogeneous Environments

Ted Shaneyfelt, *Student Member, IEEE*, Matthew Joordens, *Member IEEE*, Kranthimanoj Nagothu, *Student Member*, John Prevost, *Student Member*, Anjan Kumar, S.S., Mirsaeid Ghazi, Mo Jamshidi, *Fellow, IEEE*
Autonomous Control Engineering (ACE) Center and ECE Department
The University of Texas, San Antonio, TX, USA
Email: {ted.shaneyfelt, matthew.joordens,kranthimanoj.nagothu}@utsa.edu,moj@wacong.org

***Abstract*—Simulation provides a low cost method of initial testing of control for robotic swarms. The expansion of robotic swarms to heterogeneous environments drives the need to model cooperative operation in those environments. The Autonomous Control Engineering center at The University of Texas at San Antonio is investigating methods of simulation techniques and simulation environments. This paper presents results from adapting simulation tools for diverse environments.**

Keywords— System of Systems, Simulation, Mobile robots.

I. INTRODUCTION

THE focus of ongoing research at the Autonomous Control Engineering (ACE) center at the University of Texas at San Antonio (UTSA) is on developing technology for Systems of Systems in the area of swarm robotics, working cooperatively in heterogeneous environments. A specific example is harvesting and transporting deep sea resources[1] as illustrated in Fig. 1. Attention has been paid to related applications and both simulation and development of swarm robots [2][3]. This paper overviews experience in facing the challenges that arise, including evaluation of simulation platforms, adaptation of homogeneous environment simulation platforms for alternate environments, physical experiments in control and communication through various medium[4][5], efficient mechanisms for object avoidance [6][7] and target switching, and adaptive modeling in mathematic control[8]. Control is discussed in section II, and simulation in section III.

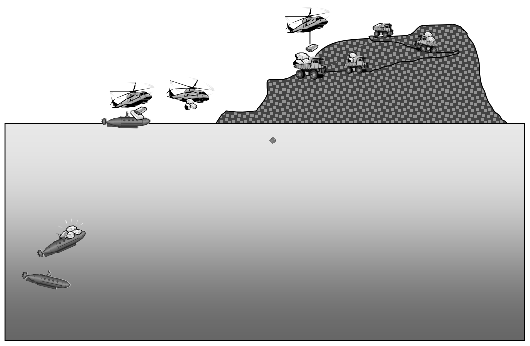
Fig. 1 Autonomous Harvesting of Manganese Nodules

II. CONTROL

A. Mathematical Control

There has been a strong focus on new System of Systems (SoS) concepts and strategies. The concept of SoS arises from the need to more effectively implement and analyze large, complex, independent, heterogeneous systems working cooperatively. The need for mathematical modeling among a group of heterogeneous systems in order to realize a common objective is essential to design a System of Systems.[8]

The concept of SoS arises from the need to more effectively implement and analyze large, complex, independent, heterogeneous systems working cooperatively. The SoS paradigm presents a new school of thought in Systems Engineering. The driving force behind the desire to view these systems as a System of Systems is to achieve higher capabilities and performance than would be possible with a traditional stand-alone system. The SoS concept presents a high-level viewpoint and allows understanding of the interactions between each of the independent systems. However, this concept is still at its developing stages [9][10][11].

Each system in a System of Systems should be modelled to be controlled. There are generally two methods of modelling, physical modelling and mathematical modelling. In the classical modelling each system is modelled regardless of its cooperativeness with other systems. In fact one could design a system based on a model, but there is not a requirement for considering the its influence on other systems. The problem even gets worse in case of designing an optimal of adaptive controllers, which tend to be highly sensitive to the environmental influences on the system, such as noise.

The problem of finding optimal control laws for constrained linear systems without disturbances or model uncertainty is relatively well understood. However, existing results, which allow for the explicit incorporation of disturbances or model mismatch in the optimal control problem, are either too conservative or computationally intractable.

B. Adaptive Modeling

Once a physical model of the environment is set up, an adaptive mathematical model may be used to control of the robotic systems.

The following is the mathematical background of Adaptive Modelling (AM). One can implement the same algorithm in all systems within a System of Systems.

Suppose i is the number of actuators in the system that implement the control law. Also let n be the order of the system, j be the number of independent state velocity variables and X and X_a be the state vector and vector of the independent state velocity variables of the system respectively. These variables uniquely describe the dynamic of a system through AM. X and X_a can be written as:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{j-1} \\ x_j \\ \vdots \\ x_n \end{bmatrix} \quad , \quad X_a = \begin{bmatrix} x_j \\ \vdots \\ x_n \end{bmatrix} \tag{1}$$

X_a is an auxiliary matrix to obtain the adaptive model of the system. The adaptive modelling of the system is obtained by iteration of (2):

$$\begin{aligned} u = [\alpha_1 \cdots \alpha_i] \times V + \sum_{n=1}^{m} \left([\alpha_{i+j_n+1} \cdots \alpha_{i+j_n}] X_a^n \right) \\ + [\alpha_{i+j_m+1} \cdots \alpha_{i+2j_n}] \times sign(X_a) \end{aligned} \tag{2}$$

where α_y, $y = 1 \ldots (i + 2j_m)$, are unknown real coefficients and $j_0 = 0$. m is assigned by user. V is the control effort applied by the actuators of the system. Accuracy of the model will increase by increasing m. The more the value of m, the more instructions should be implemented by the microprocessor, the more time will be wasted by adaptation and consequently microprocessor will lack time to calculate the control vector. Therefore user should find a compromise between the accuracy of the system and the time will be taken to reach the adaptive model. (2) is a very general formula and somehow cumbersome to implement. In order to implement (2), we divide it into three sub-formulas. Each special case has been considered as an experiment.

To solve (2) for α_y one should perform 3 types of experiments. By performing the 3 experiments explained below and compare the results with the mathematical model of the system, an error vector will result. We use the error vector to make corrections in a set of experiments.

The procedure of the experiments is as follows:

Experiment 1:

The first experiment is to eliminate the X_a term in (2). In this case (2) will be reduced to (3):

$$u = [\alpha_1 \cdots \alpha_i] \times V^{\pm} \pm [\alpha_{i+j_m+1} \cdots \alpha_{i+2j_n}] \tag{3}$$

in which V is the control effort applied by the actuators of the system. In many cases in the field of robotics the movement of the interfaces between actuators and the other parts of the system is in both directions, counter-clockwise and clockwise or in positive and negative directions. These directional movements are modelled by V^+ and V^-.

Experiment 2:

The second experiment is to obtain the terminal velocity of the state variables. In this case (2) will be reduced to (4):

$$\begin{aligned} u = [\alpha_1 \cdots \alpha_i] \times V + [\alpha_{i+1} \cdots \alpha_{i+j}] X_a \\ + [\alpha_{i+j_m+1} \cdots \alpha_{i+2j_n}] \times sign(X_a) \end{aligned} \tag{4}$$

Practically there is a slight change in (2) by eliminating the higher order terms (e.g. $m = 1$) to obtain (4).

Experiment 3:

The last experiment is a periodic excitation. In this case (2) will be reduced to form of (5):

$$u = [\alpha_1 \cdots \alpha_i] \times V + \sum_{n=1}^{m} \left([\alpha_{i+j_n+1} \cdots \alpha_{i+j_n}] X_a^n \right) \tag{5}$$

By increasing the frequency of periodic excitation and also m, the adaptive model will be more accurate.

By implementing experiences 1, 2 and 3 for different control efforts and frequencies, a Fundamental System of Solutions will be obtained, $A\alpha = B$. One can apply a least squares approximation in form of (6):

$$\alpha = \left(A^T A\right)^{-1} A^T B \tag{6}$$

The user decides the degree of tolerance for the error vector.

One can simply compare the error vector with a reference vector in a conditional statement and decides to repeat the experiments or not. If the results are satisfactory, by A and B matrices, the α vector will be yielded by (6).

By this algorithm an adaptive model of the system will be yielded and can be used in implementing the control law.

One should consider that by increasing the m index in (2), the accuracy of the model be increased; however by doing so, it will take more time for the microprocessor to perform instructions and it may lead to a lack of time to perform the digital controller instructions.

C. *Obstacle Avoidance*

Autonomous vehicle navigation in an unknown environment is an important issue in robot motion planning research. Human and other animals use their visual and other sensory perceptions for their motion. But this is a complex task for a mobile robot, especially when navigation in an unknown environment. The robot can sense the environment with the sensors mounted on it like camera, touch-sensor, sonar, and laser range finder. Many authors have proposed various schemes for mobile robot navigation and tried to solve different navigation problems. [12] [13] [14] [15] [16] [17] [18]

Local and global frame of coordinate systems are used for robotic vehicle navigation. Local frame model is needed to sense obstacles, surrounding environment and available free path to navigate and global frame model is required to know the actual localization of vehicle along with its path deviation from the target. These data are essential to make proper movement decision for avoiding obstacle and tracking the target. For proper understanding of the navigation environment we have considered the physical limits of the robot, limits of the sonar reading, sonar beam opening angle α and also the sonar placement angle γ.

D. *Forward Safe Path (FSP)*

In the previous works, researchers [16] [17] have used envelope of increasing radius around the mobile robot to define level of safe zone. Those choices of safe envelopes require expert knowledge or those are heuristically assigned and also those are not optimal. Moreover, having a semicircular envelope around the front path will cause oscillation in the mobile robot movement. If the obstacle is within the nearest envelope but outside the physical limit of robot, then also obstacle will be sensed as a “Near” and according to the logic definition of obstacle avoidance the mobile robot will turn unnecessarily. We have proposed a new definition of this envelope and named it as a “Forward Safe Path” (FSP) condition. If any obstacle comes within this safe path then only avoidance logic will start working otherwise the mobile robot will continue its motion in its forward direction.

E. *Target Switching Approach (TSA)*

path planning of the vehicle is a local path planning problem [6]. When the robot is racking a target, the initial location (x_{R0}, y_{R0}) and target location (T_{Rx}, T_{Ry}) are known to vehicle. (In a three-dimensional navigation, such as with submersible or airborne vehicles, the model is extended with the additional coordinate). If the vehicle makes a turn to avoid an obstacle then it will be deflected from the target directional path. The vehicle has to seek the presence of the target from its new position and makes necessary turn after avoiding the obstacle to resume its target directed motion. Moreover the turning direction (whether left or right) is also an important issue. In this present work, we have proposed a scheme to find instant target directional turning angle and direction. We also have proposed another scheme to avoid obstacle and resume target directional motion based on switching the target direction. These schemes are also capable of removing the dead-cycle problem of the vehicle. In this scheme whenever vehicle gets an opening towards the target from the deflected position it makes a turn towards the target and follows the target directional path and then the future avoidance action will be governed by FSP criteria. Let the robot is in the target direction and to avoid an obstacle it takes a right turn. Then the target will be in its left side. Now, in the successive movements and obstacle avoidance the left turn will have the highest priority. So, the robot will seek opening at its left side and left sided sonar will have highest priorities. If there are no openings in the left side then it will consider the FSP condition to move further in its current forward path. If the path is blocked by an obstacle then according to the FSP turning, it will take right turn to avoid the obstacle. Even if the vehicle makes successive right turns to avoid obstacles, detected from the FSP condition after the primary shifting from the last target direction, still the target will be assumed at the left side of it until it crosses that last target directional line. So, in this TSA last target directional position is remembered and stored as coordinates (x_{RT} , y_{RT}). The concept is same if the vehicle takes an initial left turn to avoid the obstacle from its target directional path and then the right turning requirement will be the primary goal.

When the vehicle is deflected from its target directional path, this TSA model will play the major role to avoid obstacle. If the vehicle is at the left side of the last target directional line, then the task is to find opening at the right side of the vehicle and right sided sensors are given highest priorities to decide on

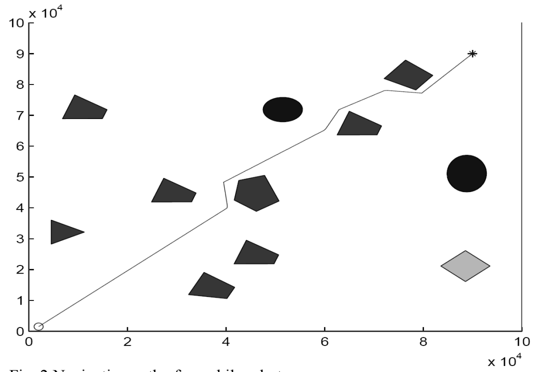

Fig. 2 Navigation path of a mobile robot

the obstacle avoidance and target seeking. If the vehicle is at the right side of the last target directional line, then the task is to find opening at the left side of the vehicle and left sided sensors are given highest priorities to decide on the obstacle avoidance and target seeking.

F. FSP Simulation Results

We have considered the model of PatrolBotTMψto show the simulation results. From the PatrolBotTMψ we get R=267mm, α =150, γ =200, number of sonars=8, sonar limit=5000 mm, S_{lim}=100mm and R_{max}=293.4mm. The navigation workspace is 100m× 100m and Q_s = 1. The simulation results demonstrate the applicability of navigation algorithm in highly unstructured unknown environment. The starting and Target location are denoted by *oψ* and ⇑. Fig. 2 shows a navigation path of a mobile robot. **Error! Reference source not found.** shows another complex situation where the robot is taking the optimal path after it overcomes a dead-zone. While passing through a close obstacle vehicle does not suffer from the oscillation problem and smoothly avoids those obstacles.

III. SIMULATION

The ACE center currently does most simulation on Microsoft Robotics Studio (MSRS), and is planning on expanding to also make use the University of Hawaii at Hilo's Huinalu Supercomputer to port simulations into a parallel environment and increase the scope of the research.

Programming/Simulation Environment

It was determined that the Microsoft Robotics Studio (MSRS) SDK add-in to Visual Studio would be the best programming/hosting environment to create and simulate the robots.

Microsoft created the MSRS to allow researchers, developers and industry to have a robust development platform for creating and simulating robots using industry-standards such as the C# programming language, XML and web-services.

At the core of the MSRS is the Decentralized Software Service (DSS). The DSS allows for a "service-model" approach to the software development. This means that the various parts required to control and run the robots can all be created modularly. The re-use of these code modules expedites the time required to deliver projects. Once created, the modules are registered into the DSSHost program as services that then can be subscribed to by any entity registered with the DSSHost. This allows for easy synchronization and coordination of the robots in real-space or virtual-space.

The entities that represent the individual functional components of the robots are created as modules and registered with the DSSHost. Examples of these modules are the Laser Range Finder, the Sonar Range Finder, the wheel motor drives, articulated joints, the thruster engines and GPS location devices. These modules communicate by utilizing the Concurrency and Coordination Runtime (CCR) using asynchronous messaging. For example, when the Laser Range Finder records an object in the path of the laser, it sends a message to the DSSHost through the CCR. Any entity that is listening to the Laser Range Finder Service gets notified of the message and can respond as appropriate. This asynchronous messaging paradigm allows for very high concurrency of events to be processed and should allow for a multiple robots to be simultaneously engaged.

At the heart of the developers tool-set is the Visual Programming Language (VPL). The VPL allows for a LabView like graphical modeling approach to create the entity behaviors that belong to each entity. The developer uses a palette of objects and behaviors and "drags and drops" them onto the VPL design surface (**Fig. 3**). By connecting the visual objects together and setting the exposed properties correctly, the developer can create a variety of distinct entity behaviors. These behaviors can then be set to trigger upon notification of the appropriate message from the DSSHost.

Underwater Emulation

In order to create a simulation that can emulate an underwater environment, several parameters need to be explored [3].

The first thing to understand is how to create a bounding area that will contain water (our pool). Next, there needs to be a boundary restriction that will not allow the water entities to go out of the area defined as the pool (from the top). Once this is established, exploration of the available environmental physical properties must be explored so the robots can be made neutral, or positively, buoyant. Another property of water that differs from air is viscosity. An object in water does not obey the same rebounding and restitution properties is it would if on land or in the air. Water has a damping effect that acts a friction opposing motion in the direction of force. Finally, it must be determined how to establish motion itself on the underwater entity.

Along with establishing the correct set of mechanics for an underwater environment, the submarine itself must be modeled so an accurate visual representation of the sub can be used in the simulation.

To establish a ground plane that can be used for water, the HeightFieldEntity class was used. This class allows for a ground to be established by defining a matrix of heights, each separated by a defined column area.

After empirically ruling out various other attempts[3], it

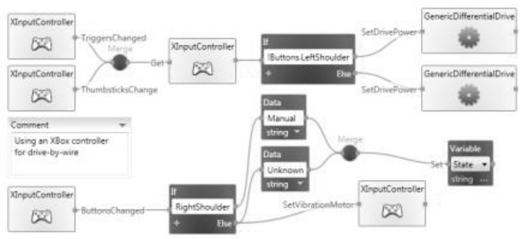

Fig. 3 Microsoft Visual Programming Language

was discovered that a terrain mesh could be created that allowed for a variety of heights; all based on the color values in a bitmap image. A bitmap was created as n pixels by m pixels. This corresponds to a terrain n meters x m meters in the simulation environment. Next, the bitmap receives color values corresponding to the desired relative heights of the terrain. In our case, a square of white was created in the center of a solid black image. The white area represented our pool; the black area represented the ground level. On runtime, a nice depression appeared in a level terrain. The robots were then initialized inside the pool. The upper boundary was then created and initialized on the ground level, effectively creating a vertical translation boundary for our robots.

In water, it was desired to have the sub maintain positive buoyancy, or at least be neutral. After much time and effort, a public property named IgnoreGravity was discovered that seemed to work. Setting this property to true allowed the sub to be positioned midway between the floor and the upper boundary. When the simulation was run, the sub stayed where it was initialized. Gravity had no effect on the sub, effectively acting as if the sub was maintaining neutral buoyancy.

The sub was then given an initial velocity by creating a vector of the desired meters per second as a 3D vector. When the simulation was run, the sub slowly moved along the appropriate path at the pre-set velocity.

The only thing remaining to establish a true underwater simulation was to somehow change the properties of the medium the sub was traveling through to emulate the viscosity present in water. There was no entity representing “air”, but since the effect could be modeled as friction along a vector opposing the motion of the vehicle some form of damping property may be good enough. Two such properties were discovered, LinearDamping and AngularDamping. These two properties are accessible at runtime by putting the simulation in “edit” mode and manually setting the values. Unfortunately, there is seemingly no way of accessing the proper object layer at design time, making it impossible to initialize the environment with these values. When the properties are set at runtime, the sub reacts to being bumped in the predicted manner. It moves away from the applied force, but quickly loses momentum and comes to a stop.

The code used to create a single sub was now essentially complete. In order for the swarm concept to be implemented, the code needed to be modified to allow for any number of sub objects to be instantiated in the simulator. This was accomplished by factoring the instance name, initial position vector and initial velocity vector. This allowed the method to be called with these as parameters, thereby allowing for it to be called once for each instance of a sub desired. Fig. 3 shows two subs running in the simulated underwater environment.

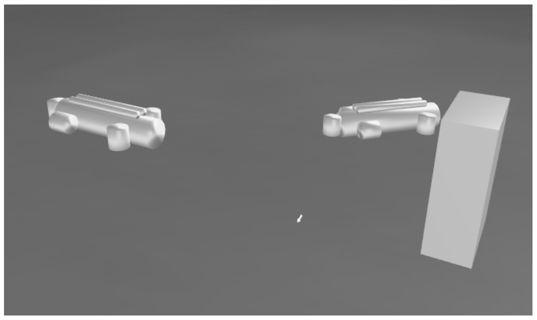
Fig. 4 Two submarines in the simulator.

ACKNOWLEDGMENT

Special thanks to Dr. Sevki Erdogan and the University of Hawaii at Hilo for offering the use of the Huinalu supercomputer for simulations, and helpful suggestions for this paper.

REFERENCES

[1] Ted Shaneyfelt, Sevki Erdogan, Azim Maredia, Gnanadeep Vemuri, Bhargavaram Pachala, Dong Yue, Sohel Karovalia, Ming-Zhu Lu, Shi-Zhong Yang, and Chenyu Gao, “Towards Net-Centric System of Systems Robotics in Air, Sea and Land” , Third International Conference on System of Systems Engineering, June 2008, Monterrey, California, USA.

[2] M. Joordens, "Design of a low cost Underwater Robotic Research Platform," in p*aper submitted to IEEE SoSE Confernec,* Monterey, CA, USA, 2008, p. 6.

[3] J. Prevost, M. Joordens and Mo Jamshidi, “Simulation of Underwater Robots using MS Robot Studio©” in p*aper submitted to IEEE SoSE Confernec,* Monterey, CA, USA, 2008, p. 5.

[4] Ted Shaneyfelt, Matthew A. Joordens, Kranthimanoj Nagothu and Mo Jamshidi "RF Communication between Surface Underwater Robotic Swarms", submitted to World Automation Congress 2008, Sep-Oct 2008, Waikoloa, Hawaii, USA.

[5] Kranthimanoj Nagothu, Matthew Joordens, Mo Jamshidi, “Communications for underwater Robotics Research platforms,” in *proc IEEE Systems Conference*, Montreal, Canada, 2008,p. 6.

[6] Anjan Kumar Ray, Laxmidhar Behera and Mo Jamshidi, “Sonar Based Autonomous Ground Vehicle (AGV) Navigation”, Third International Conference on System of Systems Engineering, June 2008, Monterrey, California, USA.

[7] Patrick Benavidez, Kranthimanoj Nagothu, Anjan Kumar Ray, Ted Shaneyfelt, Shrinath Kota, Mo Jamshidi "Multi-domain Swarm Communication System", Third International Conference on System of Systems Engineering, June 2008, Monterrey, California, USA.

[8] S.S. Mirsaeid Ghazi, Mo Jamshidi, “Adaptive Modeling: A Statistical Approach in Designing a Mathematical Model-Based Controller”, Third International Conference on System of Systems Engineering, June 2008, Monterrey, California, USA.

[9] H. Azarnoosh, B. Horan, P. Sridhar, A. Madni and M. Jamshidi, “Towards optimization of a real-world Robotic-Sensor System of Systems,” *Proc, 2006 World Automation Congress* (Robotics Track ISORA), Budapest, Hungary, July, 2006 (See Volume 19, TSI Press, ISBN 1-889335-33-9, San Antonio, TX, 2006), pp. 223-230

[10] . F. Sahin, M. JAMSHIDI and P. Sridhar, "A Discrete Event XML based Simulation Framework for System of Systems Architectures ", Proc. IEEE System of Systems Engineering Conference, San Antonio, April 16-18, 2007.

[11] S. Mirsaeid Ghazi, A. Jalali, “Low frequencies optimal control of an INVERTED PENDULUM”, *IEEE ICELICE,* Hammamet, Tunisia, December 2006.

[12] P. Carinena, Carlos V. R.and A. Otero, and Alberto J. B.and S. Barro. Landmark detection in mobile robotics using fuzzy temporal rules. IEEE Trans. on Fuzzy Systems, 12(4):423–435, August 2004.

[13] W. L. Xu and S. K. Tso. Sensor-based fuzzy reactive navigation of a mobile robot through local target switching. IEEE Trans. on SMC, 29(3):451–459, August 1999.

[14] Shih-Jie Chang Tzuu-Hseng s. Li and Wei Tong. Fuzzy target tracking control of autonomous mobile robots by using infrared sensors. IEEE Trans. on Fuzzy Systems, 12(4):491–501, August 2004.

[15] Faraz Kunwar and Beno Benhabib. Rendezvousguidance trajectory planning for robotic dynamic obstacle avoidance and interception. IEEE Trans. on SMC, 36(6), December 2006.

[16] H. R. Beom and H. S. Cho. A sensor-based obstacle avoidance controller for a mobile robot using fuzzy logic and neural network. pages 1470–1475. IEEE, 1992.

[17] Anmin Zhu and Simon X. Yang. Neurofuzzy based approach to mobile robot navigation in unknown environments. IEEE Trans. on SMC, 37(4):610–621, July 2007.

[18] Kenneth a. M.and Rajni V. Patel Jing Ren. Modified newton’s method applie to potential fieldbased navigation for mobile robots. IEEE Trans. on Robotics, 22(2):384–391, April 2006.