2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

A Survey of Commercial & Open Source Unmanned Vehicle Simulators

Jeff Craighead, Robin Murphy, Jenny Burke, and Brian Goldiez

Abstract— **This report presents a survey of computer based simulators for unmanned vehicles. The simulators examined cover a wide spectrum of vehicles including unmanned aerial vehicles, both full scale and micro size; unmanned surface and subsurface vehicles; and unmanned ground vehicles. The majority of simulators use simple numerical simulation and simplistic visualization using custom OpenGL code. An emerging trend is to used modified commercial game engines for physical simulation and visualization. The game engines that are commercially available today are capable of physical simulations providing basic physical properties and interactions between objects. Newer and/or specialized engines such as the flight simulator X-Plane or Ageia PhysX and Havok physics engines, are capable of simulating more complex physical interactions between objects. Researchers in need of a simulator have a choice of using game engines or available open source and commercially available simulators, allowing resources to be focused on research instead of building a new simulator. We conclude that it is no longer necessary to build a new simulator from scratch.**

I Introduction

Computer simulations, and their extension into video games, of unmanned systems are an emerging topic. There are at least three motivations for robot simulators. One is the role of simulators in adoption of new technology, another is their potential for low cost training, and finally their utility in research. The range of robot computer simulations is economically and technically diverse. This report surveys 14 widely available, computer based robot simulators and ranks the options available to a researcher or training expert.

Computer simulations and video games may support the transfer of robotics to new application domains. Robots such as those used for urban search & rescue, bomb disposal, surveillance, and military purposes can be important tools in responding to critical incidents. However, many law enforcement and rescue agencies are unwilling to spend a large portion of their limited budget on a tool without direct experience. Physical simulations, such as emergency response exercises, are expensive, rare, and not conducive to insertion and evaluation of new technology. Computer, or virtual, simulation and games are mechanisms to provide the community of robot operators and potential robot operators with inexpensive, enjoyable robot and sensor training environment. Thus these are attractive mechanisms for allowing a community to experiment with new technology.

In addition to allowing the user community to evaluate new technologies, computer simulation and video games are pedagogically proven techniques for training. Recent studies have shown that game-based learning have the potential to improve the transfer of skills over classroom-based activities.[1], [2], [3], [4], [5] In addition, gaming technology is increasingly being used for non-entertainment purposes. For instance, UTSAF [1] uses the Unreal game engine to provide a relatively high fidelity visualization for the US Army's OneSAF battlefield simulator with a very small monetary investment when compared to the cost of a custom simulator of equivalent capability.[6], [7], [8]

The remainder of this report is organized as follows. Section II defines the evaluation criteria used in our survey. Section III identifies simulators that are commercially available or are based on commercially available products and Section IV examines open source simulators. Section V gives some concluding remarks. Note that Section IIand Section III provide only cursory descriptions of each simulator due to space considerations.

II Evaluation Criteria

In order to evaluate simulators from the perspective of a robotics researcher, criteria for utility must be established. This report builds on prior work by Alexander[1] and identifies four criteria that can be used to judge the

Jeff Craighead, Robin Murphy, and Jenny Burke are with the Institute for Safety, Security and Rescue Technology at the University of South Florida. 4202 E. Fowler Avenue, Tampa, FL, USA. craighea@cse.usf.edu, murphy@cse.usf.edu, jlburke4@cse.usf.edu. Brian Goldiez is the Deputy Director of the Institute for Simulation & Training at the University of Central Florida. bgoldiez@ist.ucf.edu.
This work was sponsored, in part, by the US Army Research Laboratory under Cooperative Agreement W911NF-06-2-0041. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ARL or the US Government. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

[1]UTSAF - UnrealTournament Semi-Automated Forces

quality of any virtual robot simulator. A simulator that has high marks for each criterion will have broad appeal because of its robust capabilities for both testing control algorithms in an environment with high physical fidelity and studying human robot interaction in an environment with high functional fidelity. Each simulator was reviewed based on published specifications for each of the four criteria. Simulators that did not indicate support for specific features were assumed to not include them. The simulators were rated high, medium, or low for a particular criterion based on the definitions below.

1.) *Physical Fidelity* - According to Alexander "Fidelity in this context can be described as the extent to which the virtual environment emulates the real world." and that "Physical fidelity is defined as the degree to which the physical simulation looks, sounds, and feels like the operational environment". While a computer game cannot simulate the feel of the operational environment, it can simulate the visual and audible portions of that environment. Alexander argues that learning can be enhanced by a higher fidelity simulation. In the context of robot operation, where the operator is typically removed from the operating environment, a high fidelity visual and audio simulation will be similar to what is encountered when operating a real robot. We look at the rendering and audio capabilities of each simulator to identify its physical fidelity ranking. A simulator with high physical fidelity is able to render the environment with high resolution textures, shaders, lighting, reflection, and bump mapping. Models with a large polygon count (>2000) with major parts modeled in with geometry, not textures should be used. Animation of vehicles should be visually correct. A high physical fidelity simulator should also include vehicle and environmental sounds with intensities based on proximity. A medium physical fidelity simulator can render the environment in 3D, however no requirements are placed on object detail. Additionally some vehicle sounds should be present. A low physical fidelity simulator uses 2D rendering and no sound is required.

2.) *Functional Fidelity* - Alexander defines functional fidelity as "the degree to which the simulation acts like the operational equipment in reacting to the tasks executed by the trainee". Functional fidelity should be the primary goal of the game or simulation since the trainee will come to expect the real equipment to behave in a similar manner. We interpret this to mean physical behavior of the equipment and look at the physics simulation capabilities of a simulator to identify functional fidelity. High functional fidelity is defined as the simulation of most of the forces acting on a vehicle and its actuators including gravity, drag, and accelerations from motors and collisions on specific elements of the vehicle. An example is a simulator that simulates the torque applied to each motor connected wheel, then calculates the appropriate vehicle acceleration based on the rotation of each wheel. These calculations should take surface properties into account to simulate wheel slippage. Medium functional fidelity simulators include simulations of forces on the vehicle as a whole instead of on individual elements. A low functional fidelity simulator does not simulate forces applied to the vehicle but only velocities or absolute position.

3.) *Ease of Development* - Ease of development is defined by several questions, those questions are: How easily can an environment be created to conduct a training exercise within the simulator? How easily can the simulator be modified to simulate new equipment? Is documentation available with support from the author? What languages can be used to modify the simulator? A simulator that provides developer documentation, supports the importing of objects from 3D modeling packages, and can be programmed in several languages will receive a high rating. Simulators that only provide some of these will be rated medium and those that provide none are rated low.

4.) *Cost* - For a simulator to be useful it must not be time consuming to install or run and accessible in terms of initial monetary cost for both the developer and end user. Simulators that are free and include an installer are rated low, simulators that are free but are difficult to install or simulators that are cheap and easy to install receive a medium rating. Simulators that are expensive (>$200) receive a high regardless of ease of installation.

III Games & Commercially Available Simulators

There are a wide variety of robot and aircraft simulators that are commercially available as well as games that can be modified to simulate a robot in an environment. The available robot simulators have a very limited scope in terms of the operating environment. While some allow small custom environments to be created, others provide no environment except for a holodeck like grid. The positive side of the commercial robot simulators is that they usually allow the creation of any shape and size robot. The aircraft simulators such as X-Plane[9] and Microsoft's FlightSimulator[10] provide expansive environments with real terrain data. X-Plane even allows simulation of orbital and Mars environments. Games such as Unreal and FarCry, the two most popular for academic modification, allow the

creation of any type of environment but are slightly limited in terms of vehicle creation.

Alexander, et al. in "From Gaming to Training: A Review of Studies on Fidelity, Immersion, Presence, and Buy-in and Their Effects on Transfer in PC-Based Simulations and Games"[1] argue that commercial games and game engine based simulations have the potential to provide an environment that is as high-fidelity as is technically possible. Nielsen and Goodrich in "Comparing the Usefulness of Video and Map Information in Navigation Tasks"[11] used the Unreal2 game engine with the USARSim modification to examine how video and maps affect human interaction with a robot while navigating the robot through an environment. Stephen Hughes and Michael Lewis in "Robitic Camera Control for Remote Exploration"[12] use the Unreal2 game engine with USARSim to study the effects of camera placement on the human control of robot mounted cameras.

Table I presents a listing of the available simulators and provides a subjective rating of capabilities in terms of physical fidelity, functional fidelity, ease of use, and cost.

USARSim - The open source urban search & rescue robot simulator USARSim, based on the Unreal2 engine, is primarily aimed at ground vehicles. The engine best supports bipedal and wheeled robots, however it is possible to add support for other robot types. It should be noted that the Karma physics engine used in the Unreal2 engine provides only basic simulation of forces on specific objects within the environment. Robots and environment objects are created in 3rd party modeling applications. Worlds can be created using an included utility. Robots can be programmed using UnrealScript or controlled over a network connection using USARSim's UDP control protocol. USARSim is used for the RoboCup Rescue competition's simulation league. The Unreal2 game engine is one of the dominant commercial simulator platform for robotics simulation for unmanned ground vehicles. It has been used to simulate robots, train army recruits and fire fighters, as well as conduct studies on search and rescue tasks.[13], [14], [11], [15], [7]

X-Plane - X-Plane[9] is a commercially available flight simulator developed by Laminar Research. X-Plane has received FAA certification as a training simulator when used with certain hardware configurations because of its high fidelity simulation of flight model and visualization. X-Plane uses blade element analysis to drive it's flight model. Included with the package are the simulator, global scenery generated using data from NASA's terrain mapping radar missions, an airfoil designer, and a plane maker application. X-Plane has been used for testing and pilot training for the Carter Copter and Space Ship One experimental vehicles. We have successfully used X-Plane to test a micro UAV controller developed in MATLAB.

Microsoft Flight Simulator - The Microsoft Flight Simulator[10] provides detailed visuals for the aircraft and environment. It uses a less accurate lookup table driven flight model for aircraft simulation, however it has been used by the US Navy as a training aide for pilots. Microsoft provides an SDK as a download for Flight Simulator which provides access to simulator data via a network, weather, terrain, scenery, instrumentation, and aircraft creation. Aircraft must be created in a 3rd party 3D modeling application such as GMax or Lightwave.

Webots - Webots PRO[16], [17], [18] is a ground robot simulator that uses the open source Open Dynamics Engine[19] for it's physics simulations and an extended VRML97 based environment. Webots provides several small built in robots such as the Khepera, Pioneer2, and Aibo as well as the means to import custom robots from 3rd party modeling applications using the VRML97 format. World size is defined by the user and can be as large as needed. Webots PRO supports various sensor types such as camera, range finder, GPS, light sensors, etc; as well as effectors like grippers, limbs, and wheels. WebotsPRO can compile controllers created within the simulator to work on real robots given that the hardware is supported by the compiler. Hohl in demonstrates the remote control of and controller transfer to an Aibo robot through Webots.

Simbad - Simbad[20] is an open source Java based 3D robot visualization environment. It does not support any physics calculations, only simple collision detection with objects placed in a flat world. The goal of this simulator is to provide a simple environment to test robot controllers and AI algorithms, as such the support for high fidelity visualization is not present. The standard sensors are sonar, camera, light, and bump sensors. Robots are represented as simple geometric primitives. As an open source project the ability to add new sensors is present in this simulator. Simbad will run on any operating system with a Java client with the Java3D library.

eyeWyre - eyeWyre Studio[21] is a development environment and simulator for BASIC Stamp 2 microcontroller based robots. The eyeWyre provides physics simulation in small environments. The environments and robots are limited to those provided with the package, making it nearly useless for a research environment.

Microsoft Robotics Studio - Microsoft's Robotics

Studio[22] provides a networkable, service-based-architecture framework for developing real robots. The package includes a simulation runtime that is similar to a game engine in terms of physics and visualization capabilities. Robotics Studio uses the Ageia PhysX physics engine, which is one of the highest fidelity engines available to date. At this point Robotics Studio is a beta release and is incomplete in terms of features. Currently users are limited to constructing robots with the built in sensors and effectors. The system requirements are relatively narrow; Robotics Studio will only run on Windows XP or Vista and requires Visual Studio 2005 to run. Robots must be programmed in C# or VisualBasic.NET.

MATLAB - MATLAB[23] is a numerical simulation environment that supports visualization via a Virtual Reality toolkit. Toolboxes are available that provide quick access for building various robot controllers based on fuzzy logic, neural networks, and genetic algorithms. MATLAB can communicate via network with other simulation environments that may provide better physics simulation and visualization for rapid controller prototyping. The MATLAB, Simulink, and the VR Toolbox set runs on Windows, Mac, and Linux and is available from Mathworks for $1200. Additional toolboxes are $200 each. MATLAB has been primarily used to simulate unmanned systems using first order dynamics for evaluation of coordination and control algorithms for multi-robot teams. MATLAB has been successfully used for UGV[24], [25]; UAV[25], [26], [27]; USV[28], [29]; and UUV[30], [31], [32], [33] simulations.

Unity - The Unity[34] engine is a Macintosh based commercial game engine and development environment. While Unity is a blank slate as far as included content, it uses the Ageia PhysX engine for physics simulation. Unity provides both Javascript and C# APIs as the primary means of controlling simulation objects. The pro version includes the ability to directly access OpenGL, create C++ plugins to add features to the engine, and the ability to compile for Windows.

IV Open Source Simulators

MissionLab - "Behavior-Based Formation Control for Multirobot Teams"[35], Balch and Arkin, presents the use of reactive behaviors for formation control in simulation and on real robots using two architectures, AuRA and the UGV Demo II architecture. The current version of MissionLab uses a distributed architecture, allowing various pieces of the simulation to run on different machines. This also allows the same interface to be used to control real robots. Visualization is provided by both 2D and low fidelity 3D OpenGL displays. The current version of MissionLab supports all examined vehicle types: UGV, UAV, UUV, and USV. The 3D display can render terrain generated from a height map with along with a low poly model of robotic vehicles. The 2D displays can render maps and image overlays as well as custom graphs. MissionLab does not appear to support any physics simulation which would be necessary for vehicle simulation. Extensive documentation is provided, but C/C++ is the only language supported.

Player/Stage/Gazebo - The Player/Stage/Gazebo project[36] is an open source project that provides a 2D and 3D environment for robot testing. Stage and Gazebo are networkable simulation environments, Player defines an interface for robots and sensors to communicate with Stage and Gazebo. Stage is a simple 2D environment that provides basic collision detection and range sensor modeling. Gazebo is a 3D environment that brings the basic simulations of Stage into the 3rd dimension. Gazebo provides a camera sensor as well as the ability to use complex objects in the environment. Gazebo presents a simple low fidelity OpenGL based visualization of the environment. While Stage does not support physics simulation, Gazebo can make use of the ODE physics engine.

SimRobot - SimRobot[37], described in "SimRobot - A General Physical Robot Simulator and its Application in RoboCup"[38], Laue, et al., is a physics based robot simulator with a 3D OpenGL based display. SimRobot uses the Open Dynamics Engine[19] for physics calculations which gives it an edge over many custom simulators. The use of a custom OpenGL visualization environment however could be improved on by using a preexisting rendering engine like OpenSceneGraph[39]. Robots and environments are specified using XML by specifying part types and positions. Several sensor types are supported, including cameras, range sensors, touch sensors, and actuator state. SimRobot was used by the German team for the 2005 RoboCup competition, however it is not limited to RoboCup robots or environments. The paper shows an office environment simulated in the SimRobot simulator.

FlightGear - FlightGear[40] is a open source simulator that uses by default a blade element analysis, similar to X-Plane. Global scenery is available for FlightGear. Aircraft models must be created in an external 3D modeling application and an XML file describing the various aircraft features must be created by hand. FlightGear has been used for various academic projects. For example, Summers, et al. in [41] used FlightGear to simulate a UAV carrying environmental sensors and Cervin, et al. in [42] used FlightGear to create an interface for a real

UAV. FlightGear is available as a free download under a GPL license. The entire source code is available for modification and is under constant development. The application runs on Windows, Mac, and Linux operating systems.

The Project MAKO Simulator - SubSim - Bräunl, et. al present a simulator developed for an international AUV competition, similar to RoboCup. SubSim[43] is a custom design that uses the Newton Game Dynamics[44] physics engine along with the Physics Abstraction Layer (PAL)[45], the wxWidgets[46] windowing framework, OpenGL for 3D rendering, and tinyXML[47] for vehicle dynamics definition. The plugin based architecture allows users to extend and modify the simulator as needed. The PAL allows several physics engines to be used and higher level sensors and effectors to be modeled. For instance PAL is used to model the DC motors that drive the propulsion system. The simulator supports standard sensor types such as inertial, range, touch, and camera. Visualization is via simple textured OpenGL display. 3D Models for visualization must be in the open source Milkshape[48] 3D format. SubSim is one of the more complete, readily available UUV simulators. It is very extensible and thus can be modified for many UUV simulation tasks.

V Discussion & Conclusions

This report concludes that it is no longer necessary to build a robotic simulator from the ground up. There are many available simulators and game engines that can be used to simulate a robotic vehicle with high physical and functional fidelity. Any modifications or enhancements made to these existing simulators should be released to the community to drive the capabilities of available robot simulators. We can also conclude that while there has been some work using simulators for human robot interaction studies, there has been no work that addresses robot simulation for use as a training tool.

This paper presented a brief review of the significant features of 14 commercially available or open source simulators. Due to space considerations we were unable to include a more thorough evaluation of each simulator in this work, this will be addressed in forthcoming publications. Table I lists the simulators reviewed and identifies how each meets the requirements put forth in Section II. MATLAB is the tool of choice for many simulation applications, however it simply does not provide the real time high fidelity visualization or physical simulation necessary. MATLAB can be used successfully in conjunction with the following three simulation packages for robot controller rapid prototyping. Three available simulators meet the fidelity requirements that have been outlined: the Unity game engine, the X-Plane flight simulator, and the Microsoft Robotics Studio. Each of these provides top of the line physics simulation and 3D visualization in addition to being extensible. Of those only Unity has a high cost, however this should not rule out Unity as a potential engine as it is much easier to use.

References

[1] A. L. Alexander, T. Brunyé, J. Sidman, and S. A. Weil, "From gaming to training: A review of studies on fidelity, immersion, presence, and buy-in and their effects on transfer in pc-based simulations and games," November 2005. [Online]. Available: http://www.darwars.com/downloads/DARWARS%2520Paper%252012205.pdf

[2] M. Prensky, "Digital game-based learning," Book Synopsis, Games2train, pp. 21–21, October 2003.

[3] A. L. Aitkin, "Playing at reality: Exploring the potential of the digital game as a medium for science communicaiton," Ph.D. dissertation, The Australian National University, October 2004.

[4] C. Fabricatore, "Learning and videogames: An unexplored synergy," in *Association for Educational Communicataions and Technology (AECT) 2000 Workshop*, 2000.

[5] C. S. Green and D. Baveller, "Action video game modifies visual selective attention selective attention," *Nature*, no. 423, pp. 534–537, May 2003.

[6] J. Manojlovich, P. Prasithsangaree, S. Hughes, J. Chen, and M. Lewis, "Utsaf: A multi-agent-based framework for supporting military-based distributed interactive simulations in 3d virtual environments," in *Proceedings of the 2003 Winter Simulation Conference*, 2003.

[7] P. Prasithsangaree, J. Manojlovich, S. Hughes, and M. Lewis, "Utsaf: A multi-agent-based software bridge for interoperability between distributed military and commercial gaming simulation," *Simulation*, vol. 80, no. 12, pp. 647–657, 2004.

[8] J. Manojlovich, P. Prasithsangaree, S. Hughes, J. Chen, and M. Lewis, "Utsaf: Getting the best of consumer graphics into military simulations," in *Proceedings of the 47th Annual Meeting of the Human Factors and Ergonomics Society*, October 2003.

[9] A. Meyer, "X-plane." [Online]. Available: http://www.x-plane.com

[10] Microsoft, "Microsoft flightsimulator." [Online]. Available: http://www.microsoft.com/games/flightsimulator/

[11] C. W. Nielsen and M. A. Goodrich, "Comparing the usefulness of video and map information in navigation tasks," in *Proceeding of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, 2006, pp. 95–101.

[12] S. Hughes and M. Lewis, "Robotic camera control for remote exploration," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2004, pp. 511–517.

[13] J. Schell, "Hazmat: Hotzone." [Online]. Available: http://www.etc.cmu.edu/projects/hazmat_2005/

[14] I.-C. M. Mike Schneider, Kathleen M. Carley, "Detailed comparison of america's army game and unit of action experiments," Center for Computational Analysis of Social and Organizational Systems, Institute for Software Research International, Carneggie Mellon University, Tech. Rep. CMU-ISRI-05-139, 2005.

[15] J. Wang, M. Lewis, and J. Gennari, "Usar: A game based simulation for teleoperation," in *Proceedings of the 47th Annual Meeting of the Human Factors and Ergonomics Society*, October 2003.

[16] "Webots." [Online]. Available: http://www.cyberbotics.com/products/webots/

[17] O. Michel, "Cyberbotics ltd - webotstm: Professional mobile robot simulation," in *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, 2004, pp. 39–42.

Simulator	Physical Fidelity	Functional Fidelity	Ease of Development	Cost
USARSim	High	Medium	Medium	Medium
X-Plane	High	High	Medium	Medium
FlightGear	High	Medium	Medium	Low
MS Flight Simulator	High	Medium	Medium	Medium
Webots	Medium	Medium	Medium	Medium
Simbad	Medium	Low	Medium	Low
Player/Stage/Gazebo	Medium	Low	High	Medium
eyeWyre	Medium	Low	Medium	Medium
MS Robotics Studio	High	High	Medium	Low
MATLAB & Simulink	Low	Medium	Medium	High
MissionLab	Medium	Low	Medium	Medium
SimRobot	Medium	Low	Medium	Low
SubSim	Medium	Low	Medium	Medium
Unity	High	High	High	High

TABLE I

A COMPARISON OF AVAILABLE MOBILE ROBOT SIMULATORS. RATINGS ARE SUBJECTIVE IN TERMS OF BEST TECHNOLOGY AVAILABLE TO DATE.

[18] L. Hohl, R. Tellez, O. Michel, and A. Ijspeert, "Aibo and webots: Simulation, wireless remote control and controller transfer," *Robotics and Autonomous Systems*, vol. 54, no. 6, p. 472, June 2006.

[19] "Open dynamics engine." [Online]. Available: http://www.ode.org/

[20] "Simbad 3d robot simulator." [Online]. Available: http://simbad.sourceforge.net/

[21] "eyewyre studio." [Online]. Available: http://www.eyewyre.com/

[22] "Microsoft robotics studio." [Online]. Available: http://msdn.microsoft.com/robotics/

[23] "Matlab." [Online]. Available: http://www.mathworks.com

[24] W. Wijesoma, P. P. Khaw, and E. K, "Control and navigation of an outdoor agv using fuzzy reasoning," in *International Conference on Intelligent Transportation Systems*, October 1999, pp. 544–549.

[25] H. J. Kim, R. Vidal, D. H. Shim, O. Shakernia, and S. Sastry, "A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles," in *Proceedings of the 40th IEEE Conference on Decision and Control*, vol. 1, December 2001, pp. 634–639.

[26] S. Rasmussen, J. W. Mitchell, C. Schulz, C. Schumacher, and P. Chandler, "A multiple uav simulation for researchers," in *AIAA Modeling and Simulaiton Technologies Conference and Exhibit*, August 2003.

[27] S. Rasmussen and P. Chandler, "Multiuav: A multiple uav simulation for investigation of cooperative control," in *Proceedings of the 2002 Winter Simulation Conference*, 2002, pp. 869–877.

[28] T. S. VanZwieten, "Dynamic simulation and control of an autonomous surface vehicle," Master's thesis, Florida Atlantic University, December 2003.

[29] A. Leonessa and T. S. VanZwieten, "Neural network model reference adaptive control of a surface vessel," in *Proceedings of the 43rd Conference on Decision and Control*, 2004, pp. 662–667.

[30] T. Prestero, "Verification of a six-degree of freedom simulation model for the remus autonomous underwater vehicle," Master's thesis, Massachusetts Institute of Technology and Woods Hole Oceanographic Institution, September 2001.

[31] ——, "Development of a six-degree of freedom simulation model for the remus autonomous underwater vehicle," in *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, November 2001, pp. 450–455.

[32] E. Omerdic and G. Roberts, "Thruster fault accommodation for underwater vehicles," in *Proceedings of the st IFAC Workshop on Guidance and Control of Underwater Vehicles*, April 2003.

[33] P. Baccou and B. Jouvencel, "Simulation results, post-processing experimentation and comparisons results for navigation, homing and multiple vehicles operations with a new positioning method using transponder," in *Proceedings of the 2003 International Conference on Intelligent Robots and Systems*, vol. 1, October 2003, pp. 811–817.

[34] "Unity." [Online]. Available: http://www.unity3d.com

[35] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, December 1998.

[36] "Player/stage/gazebo." [Online]. Available: http://sourceforge.net/projects/playerstage

[37] "Simrobot." [Online]. Available: http://www.informatik.uni-bremen.de/simrobot/

[38] T. Laue, K. Spiess, and T. Röfer, "Simrobot - a general physical robot simulator and its application in robocup.pdf," in *Proceedings of RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence, 2006, pp. 173 – 183. [Online]. Available: http://www.tzi.de/kogrob/papers/rc06-sim.pdf

[39] "Openscenegraph." [Online]. Available: http://www.openscenegraph.org/

[40] "Flightgear." [Online]. Available: http://www.flightgear.org

[41] P. Summers, D. Barnes, and A. Shaw, "Determination of planetary meteorology from aerobot flight sensors," in *Proceedings of the 7th ESA Workshop on Advanced Space Technologies for Robotics and Automation*, November 2002.

[42] B. Cervin, C. Mills, and B. C. Wünsche, "A 3d interface for an unmanned aerial vehicle," in *Proceedings of Image and Vision Computing New Zealand*, 2004, pp. 249–253.

[43] "Subsim." [Online]. Available: http://robotics.ee.uwa.edu.au/auv/subsim.html

[44] "Newton game dynamics engine." [Online]. Available: http://www.newtondynamics.com/

[45] "Physics abstraction layer." [Online]. Available: http://www.adrianboeing.com/pal/index.html

[46] "wxwidgets." [Online]. Available: http://www.wxwidgets.org/

[47] "Tinyxml." [Online]. Available: http://sourceforge.net/projects/tinyxml

[48] "Milkshape 3d." [Online]. Available: http://www.swissquake.ch/chumbalum-soft/index.html