Interactions and Training with Unmanned Systems and the Nintendo Wiimote

Paul Varcholik, Daniel Barber, Denise Nicholson
Institute of Simulation & Training
University of Central Florida
Orlando, Florida
pvarchol@ist.ucf.edu, dbarber@ist.ucf.edu, dnichols@ist.ucf.edu

ABSTRACT

As unmanned systems continue to evolve and their presence becomes more prolific, new methods are needed for training people to interact with these systems. Likewise, new interfaces must be developed to take advantage of the increasing capabilities of these platforms. However, the complexity of such interfaces must not grow in parallel with advancements in unmanned systems technology.

A common form of human communication is through the use of arm and hand gestures. Applying gesture-based communication methods to human-to-robot communication may increase the interface capabilities, resulting in less complex, natural and intuitive interfaces. In the context of military operations, hand and arm gestures (such as those listed in the Army Field Manual on Visual Signals, FM 21-60) may be used to communicate tactical information and instructions to robotic team members. We believe that a gesture-based interface provides a natural method for controlling unmanned systems and reduces training time and training costs for military personnel by reusing standard gestures.

The research presented explores these hypotheses through interactions with unmanned systems using computer-mediated gesture recognition. The methodology employs the Nintendo Wii Remote Controller (Wiimote) to retrieve and classify one- and two-handed gestures that are mapped to an unmanned system command set. To ensure interoperability across multiple types of unmanned systems, our technology uses the Joint Architecture for Unmanned Systems (JAUS); an emerging standard that provides a hardware and software independent communication framework.

In this paper, a system is presented that uses inexpensive, commercial off-the-shelf (COTS) technology for gesture input to control multiple types of unmanned systems. A detailed discussion of the technology is provided with a focus on operator usability and training. Finally, to explore the efficacy of the interface, a usability study is presented where participants perform a series of tasks to control an unmanned system using arm and hand gestures.

ABOUT THE AUTHORS

Paul Varcholik is a doctoral candidate in the Modeling & Simulation program at the University of Central Florida. Paul has been a software engineer since 1994, and was most recently a lead software engineer at Electronic Arts where he worked on video game titles including Madden NFL Football, NASCAR, and Superman Returns. He is a graduate research assistant at the Institute for Simulation and Training for the Applied Cognition and Training in Immersive Virtual Environments (ACTIVE) lab. He also teaches classes, on video game development, at Seminole Community College and the Florida Interactive Entertainment Academy in Orlando, Florida.

Daniel Barber is a research associate for the Applied Cognition and Training in Immersive Virtual Environments (ACTIVE) lab at the Institute for Simulation and Training (IST) at the University of Central Florida. Daniel's research focus in robotics and intelligent systems includes machine learning, human-agent collaboration, control systems, path-planning, computer vision, communications, and environment modeling. He has several years experience working in the Robotics Laboratory at UCF designing autonomous ground vehicles. Current efforts within the ACTIVE lab include an open source implementation of the Joint Architecture for Unmanned Systems (JAUS) called JAUS++, and the development of unmanned system simulators for Live Virtual and Constructive (LVC) robotic environments for training.

Denise Nicholson's research focus on human systems modeling, simulation and training includes virtual reality, human–agent collaboration, and adaptive human systems technologies for Department of Defense applications. She joined the University of Central Florida in 2005 with over 18 years of government service ranging from bench-level research at the Air Force Research Lab to leadership as the Deputy Director for Science and Technology at the U.S. Navy's NAVAIR Training Systems Division.

Interactions and Training with Unmanned Systems and the Nintendo Wiimote

Paul Varcholik, Daniel Barber, Denise Nicholson PhD
Institute of Simulation & Training
University of Central Florida
Orlando, Florida
pvarchol@ist.ucf.edu, dbarber@ist.ucf.edu, dnichols@ist.ucf.edu

INTRODUCTION

Military forces are being increasingly supplemented by unmanned systems of various types, including: UAVs (Unmanned Aerial Vehicles), UUVs (Unmanned Underwater Vehicles), USVs (Unmanned Surface Vehicles) and UGVs (Unmanned Ground Vehicles). Though these vehicles provide different capabilities and levels of autonomy, they all have the common requirement of some form of human interaction.

The need for consistent computer control between disparate unmanned systems has led to the development of communication protocols such as STANAG 4586 and JAUS (Joint Architecture for Unmanned Systems). However, no such standard exists for communicating between humans and unmanned systems. Indeed, a variety of interfaces have been developed for command and control, and these have largely been specific to their paired robotic platforms. Moreover, as technology in unmanned systems continues to advance, these human-robot interfaces have grown in complexity. We propose that this trend has an upper bound where the complexity of the interface is such that it is inefficient or prohibitively expensive to teach a person to operate an unmanned system. Additionally, we believe that the pace of advancement in human-robot interaction is lagging behind that of unmanned systems technology, and that this limits the speed at which unmanned systems can evolve.

Following the example of standard communication protocols for control of unmanned systems, this paper explores a common mechanism for human-robot interaction. More specifically, we investigate the application of intuitive forms of human-to-human communication for human-robot interaction. The focus of this investigation is on hand and arm gestures mapped to a set of robotic commands. The goal of this effort is to improve operator usability while reducing training cost across a variety of unmanned systems.

RELATED WORK

Unmanned System Interoperability

Designing a human-machine interface for mobile robot control is a complex task. Interfaces must be able to account for different levels of autonomy in unmanned systems, offer real-time feedback, and allow operators to accomplish mission tasks in an efficient manner. As more capabilities are added and different types of unmanned systems are combined, these interfaces must adapt. However, these changes can result in operator control units that are difficult to learn and use.

Multiple implementations of robotic control interfaces have been designed and evaluated (Trouvain, 2002), (Chen, 2008), (Space and Naval Warfare Systems Center, 2008), (Barber, 2008). The Tactical Control Unit developed by Army Research Laboratory's Robotics Collaborative Technology Alliance (RCTA), is an example of an operator control station for teleoperation and autonomous navigation of multiple unmanned systems. Although it is an extremely capable application, it does require significant training time before an operator is capable of performing mission relevant tasking. Additionally, some of these interfaces will only work with specific types or models of unmanned systems due to non-standard communication protocols. The Multi-robot Operator Control Unit (MOCU) developed by the Space and Naval Warfare Systems Center (2008), attempts to reconcile this issue with a design that can incorporate multiple robotic message protocols – a design which is in line with the efforts presented here.

In the recently published *Unmanned Systems Roadmap: 2007-2032* (DoD, 2007), the US Department of Defense (DoD) details the future of robotic military equipment and how to proceed in the development and procurement of unmanned systems. This document identifies critical DoD needs, with one being a focus on the increase in standardization and interoperability of unmanned systems to better facilitate the broad range of

DoD users, and improved methods for training operators. One of the standards promoted by this roadmap is the Joint Architecture for Unmanned Systems (JAUS).

JAUS is an international standard developed by the JAUS Working Group (2008) and maintained by the Society of Automotive Engineers (SAE) Aerospace council, providing a high level interface domain for unmanned vehicles. It is a component-based message passing architecture which specifies standard fields and formats for communication among unmanned systems. JAUS has the benefit of being independent of any hardware or software system, and is applicable to all unmanned system types (air, ground, surface, underwater) making it ideal for inclusion within human-robot interaction applications that must work across multiple platforms.

Human-Robot Interaction

Over the past few decades many different methods for human-robot interaction have been explored – that go beyond traditional keyboard/mouse/joystick style control – looking for more natural and intuitive forms of communication. These explorations have included efforts in spoken, visual, tactile, and gestural communication, and hybrids of each approach. Waldherr (2000) presented a human-robot interaction system that utilizes gestures detected through vision processing. Our gestural control system differs from Waldherr's work, first in that we recognize gestures through a 3D input device rather than vision, but also in that our recognition system is hosted on an intermediary computing platform, and not on the robot itself. While this offloads the gesture recognition tasks from the unmanned system (which likely has limited computing resources) it introduces the problem of identifying which robot you are communicating with. We work around this issue by allowing communication with only one robot at a time, and by requiring the user to specify which robot he/she is communicating with through a graphical user interface (GUI). Arguably, this issue exists for any unmanned system that self-hosts the recognition system. For instance, if two or more robots are facing a user, how might that user indicate which unmanned system to issue commands to? Moreover, if the communication is solely dependent on a visual connection, how do we communicate to a robot that's turned away from us or where the visual connection is otherwise occluded?

Rogalla (2002) presented work on controlling robots through gesture and speech recognition. Speech can augment or disambiguate the information being conveyed to the unmanned system. Speech systems can be a very effective form of human-robot communication, but breakdown in noisy environments, and likewise suffer from uniquely isolating the intended communication recipient. Though our research was not focused on the task of discovering available unmanned vehicles, and isolating communication, this is an interesting topic of research, and one which we believe can be successfully addressed through a hybrid of human-robot interaction techniques.

Guo (2008) presented work controlling a robotic dog (the Sony AIBO) using the Nintendo Wiimote. In this work, the authors use the Wiimote's accelerometer values to move and steer the robot and to place the robot's arms in various poses. This work showed that the Wiimote interface outperformed a traditional HRI keyboard control scheme.

The remaining sections of this paper present our efforts in combining the advancements in human-robot interaction with the communications standards that enable unmanned system interoperability.

METHODOLOGY

Robotic Platform

To take advantage of the capabilities exposed through JAUS, an unmanned ground vehicle called Gamblore (Figure 1) was used within the efforts described in this paper. Designed for the Intelligent Ground Vehicle Competition sponsored by the Association for Unmanned Vehicle Systems International, Gamblore provides a differential drive platform with sensors needed for autonomous operations including: LIDAR, GPS, Digital Compass, Speedometer, and Cameras. An 802.11g access point allows other computers to connect to the vehicle wirelessly.

The software design approach used for Gamblore pulls directly from the component services described by the JAUS Reference Architecture, breaking down the system into a series of modules, each providing a specific capability for other JAUS compliant components to use. A C++ implementation, called JAUS++, developed within the ACTIVE Laboratory (2008) was used as the basis for creating JAUS compliant components for Gamblore which include: Primitive Driver, Global Pose Sensor, Velocity State Sensor, Visual Sensor, Global Vector Driver, and Global Waypoint Navigator. The primary services used

within the usability study described later in this document, were the Primitive Driver, Global Pose, Velocity State, and Visual Sensors. These services give the operator open-loop control of the drive system, position, attitude, velocity, and camera data.

Figure 1. Gamblore – Unmanned Ground Vehicle

Vehicle Control Software

We communicate to the JAUS++ library through a software system designed for generic human-robot interaction. The graphical user interface (GUI) for this software is presented in Figure 2. Importantly, this GUI, and the underlying JAUS communication mechanism, was not developed for the robot (Gamblore) used in the usability experiment. In fact, this software was developed entirely against a simulated set of JAUS services and was not connected to a physical robot prior to the usability study. The importance of this fact is that the software system is capable of querying and controlling any JAUS-equipped unmanned vehicle that provides services including: control, global pose, speed, and streaming video. Moreover, these services are independently accessible, allowing communication to a vehicle that may contain only some, or all, of the supported services. The status of the service (connected or disconnected) is displayed in the area marked *Services* in the bottom-center portion of the GUI. As a service is connected or disconnected the corresponding status indicator changes from green to red.

When the user has line of sight to the vehicle, reference to the GUI might be infrequent – and indeed this was observed during our usability study; however, when teleoperating the vehicle, the instrumentation exposed by the GUI is a necessity.

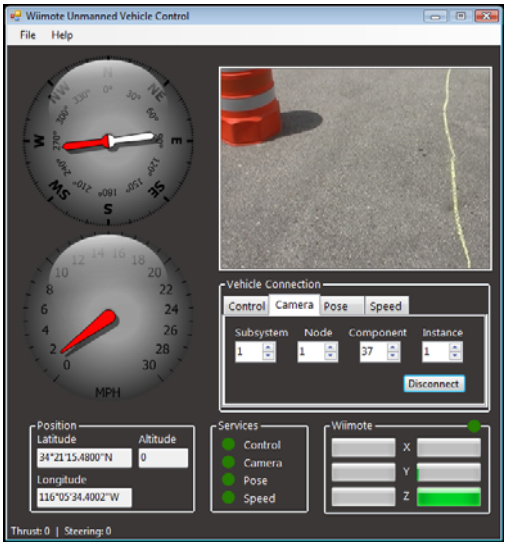

Figure 2. Graphics User Interface for Unmanned Vehicle Control

Control Input Device

For driving the vehicle, we needed an intuitive input device; one which required little training, or naturally mirrored existing control paradigms. We chose the accelerometer-based Nintendo Wii Remote Controller (Wiimote) represented in Figure 3, an inexpensive, commercial-off-the-shelf (COTS) input device which allows for a variety of input schemes. The Wiimote communicates wirelessly using the BlueTooth communication protocol and open-source libraries allow access to the Wiimote from a PC.

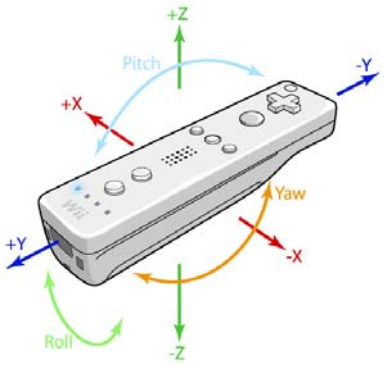

Figure 3. The Nintendo Wiimote

For this effort we implemented the following control systems:

- Traditional Joystick Control
- Accelerometer/Motion Steering-Wheel Style Control
- Hand and Arm Gestural Control

For joystick style control, the user holds the Wiimote horizontally in both hands, with the directional pad (D-pad) facing the user and to the left. To move the unmanned vehicle forward, the user presses the upwards-pointing D-pad button; and likewise for turning the vehicle left, right, and moving in reverse. With this control scheme, the inputs are digital in nature – meaning that the user doesn't have precise control of the values of the thrust or steering values – when the user presses the up button the vehicle moves at set (though configurable) value. The user can thrust and steer simultaneously, by depressing the D-pad buttons along the diagonal. For example, pressing upper left to move the vehicle forward while turning to the left. This control style is representative of traditional vehicle control through joysticks.

To employ the accelerometer (motion-control) control scheme, the user again holds the Wiimote horizontally in both hands, in the fashion of a steering wheel. While depressing the button labeled *2* (rightmost button) the user tilts the Wiimote to the left and right to steer the vehicle in the respective direction. To move the vehicle forward or backwards, the user rotates the controller away or towards himself. Thrust is mapped to the z-axis of the Wiimote, while steering is mapped to the x-axis. These are decimal values, ranging from -4.0 to 4.0 and are visualized in the progress-bars located in the lower right-hand section of the GUI. With this input system, the user has fine-grained control over the movement of the vehicle. Tilting or rotating the Wiimote slightly will produce small steering or thrust values, and vice-versa. This control style is analogous to driving with a steering wheel. In fact, the Wiimote has a miniature steering wheel accessory to complete the analogy.

The final input scheme, hand and arm gestural control, utilizes a recognition system to classify a Wiimote motion as a pre-trained gesture. The recognition system uses a machine learning algorithm that transforms the raw stream of XYZ accelerometer values into features used to uniquely identify a gesture. Recently developed by members of the Institute for Simulation & Training at the University of Central Florida, the recognition system uses a linear classifier and 29 features based on work by Rubine (1991) on 2D pen-based hand markings. The gestures classified are 3-Dimensional and thus the feature set has been extended over Rubine's original 2D set. Recent studies (awaiting publication) have shown that this recognition system maintains a 95% classification accuracy with training sets as low as 20 samples per gesture. For our efforts, we chose a very small gesture set – *Move Forward*, *Stop*, *Turn Left*, and *Turn Right* – and trained the recognition system with 20 example gestures for each. The gestures, depicted graphically in Figure 4, are performed by holding the Wiimote vertically (the normal Wiimote position) in the right hand, and pressing and releasing the trigger button (the B-button) to indicate the beginning and end of a gesture. The gesture recognition system is constantly monitoring the state of the Wiimote, and watching for newly performed gestures. When the B-button is released, the recognizer transforms the stream of XYZ accelerometer data into feature space, and attempts to classify the gesture as one that has been previously trained. On a successful classification, the gesture is relayed to the unmanned system as a command. In our scenario, we map the *Move Forward*, *Stop*, *Turn Left*, and *Turn Right* gestures into corresponding vehicles movements. Similar to the joystick style control, these movements are performed at a fixed velocity. But unlike the joystick control system, which requires that the user continuously press a button to provide movement, the gestural control system maintains the command provided by the most recent gesture until a new gesture/command is recognized.

Notably, this recognition system functions on dynamic gestures only – and not on static poses. This is opposite of many vision-based systems which can detect a still hand/arm position, but have difficulty analyzing movement in real-time. This limitation of the Wiimote gesture recognition system is due to the lack of position information provided by the raw accelerometer data. It's not possible to distinguish between static poses with the 3-axis accelerometer values alone.

Figure 4. Selected Gestures and Representations of their Movements

All of the three control styles are immediately accessible – meaning that mode switching is somewhat

of a hidden task. Using the D-pad indicates that you're using the joystick style control, while pressing and releasing the B-button indicates that you're using gestural control. The accelerometer/motion control system requires the most overt mode switching, which is depressing the 2-button. Releasing the 2-button, or not depressing any of the D-pad arrow buttons, stops the vehicle, because no active movement commands are being passed to the vehicle. Likewise, using the D-pad, or the Wiimote's motion control, will immediately revoke any active gestural commands.

Feedback on the current thrust and steering commands is displayed in the lower left-hand corner of the GUI. This feedback is augmented for the gestural control system, where sound files are played to indicate which command was recognized.

Usability Study

An informal usability study was conducted, to explore the efficacy of using gestures for controlling unmanned vehicles. Participants were required to navigate Gamblore through an outdoor obstacle course using each of the three control schemes. After a brief orientation period, the user performed three laps of the course, exercising a different control style with each lap. The only empirical performance metric used was the time required to complete each lap. At the completion of the obstacle course, the participant was asked to complete a qualitative questionnaire. The questionnaire asked the following questions using a five-level Likert scale:

1. Instructions on how to perform gestures were clear
2. Gestures were easy to learn
3. Gestures were easy to remember/recall
4. Gestures were easy to perform
5. The Gestures performed did not cause my hands and/or wrists to become fatigued
6. The Wiimote was easy to use
7. The system recognized the gestures accurately
8. Controlling the vehicle through gestures was easy
9. Controlling the vehicle through the Wiimote's motion tracking was easy
10. Controlling the vehicle through the Wiimote's directional pad was easy

The participants were also asked to rank their preferred control style and to provide any comments on their experience.

DISCUSSION

Control Software and Hardware Performance

Though relatively new, the JAUS++ library proved to be quite robust and communicating to the library from the graphical user interface software was very easily accomplished. As mentioned previously, because JAUS works independently of the hardware system, we were able to develop the control software system completely separate from a live vehicle. The transition from a simulated vehicle to Gamblore was seamless and required only adjusting the upper- and lower-limits for speed and steering values.

Notably, JAUS provides for automatic discovery of available components, and this would allow for a user to easily switch between multiple unmanned systems. However, for the purposes of this effort, we did not expose the functionality through the GUI.

The full hardware/software system was capable of driving the vehicle through the obstacle course using all control styles. The GUI tracked the vehicles position, speed, and attitude, and provided streaming video from Gamblore's cameras. Videos of a number of test runs are available at http://www.bespokesoftware.org/.

We encountered an issue related to wireless communication to the vehicle's 802.11g access point. Wi-Fi interference would occasionally disrupt our connection, placing the vehicle in a standby state and preventing the transmission of new user commands. Losing a Wi-Fi connection never resulted in any danger to human participants or the vehicle due to a backup RF communication system that could override the higher-level computer control. The takeaway from this problem is the need for a reliable wireless connection between the robot and the controlling computer platform.

Additionally, the decision to use drive control that was open-loop via the Primitive Driver service resulted in vehicle drift. Small differences in motor speed and terrain caused Gamblore to drift to the right when driving distances greater than 4-5 feet. Also, if the vehicle went off of the designated course, and into heavy grass, it did not always have enough power to drive out. We had intentionally limited the upper bound for speed and steering to less than half of the total available power. We chose this because of the small scale of the course being traversed and hesitance to allow inexperience users to operate at full speed. We

believe that these issues affected performance and perception of control styles during the usability experiment. Specifically, these problems were difficult to overcome in a gesture-based control system where users are not continuously sending new input as in the case of accelerometer or joystick style control. The issues related to open-loop control and drift can be resolved in future studies by switching to a closed-loop driver service such as the Local Vector Driver or Global Vector Driver components described within the JAUS Reference Architecture.

Usability Study Results

Results from the usability study showed an average of 84, 104, and 123 seconds for a lap around the course using accelerometers, D-pad, and gestural control systems respectively. Across all participants the preferred control styles were (from most preferred to least preferred): accelerometer control, D-pad/joystick control, and then gestural control. For the remaining qualitative questions, the average responses indicated a general acceptance of all three styles of operation, but that there was difficulty in gesture recognition with a few participants.

These results lead to a number of possible conclusions. First, the task performed – driving a vehicle through an obstacle course – is a fairly low-level task, requiring almost continuous intervention from the user. The robot was not functioning in even a semi-autonomous mode, and was under full control of the human operator. Moreover, the task required the lower-level commands of: go forward, stop, turn left or turn right; as opposed to a higher set of commands such as "follow me" or "go to this destination". With such a low-level task, it's not surprising that the study participants would prefer a control scheme that provided the highest control fidelity, as with the accelerometer control system. Recall, that with this input scheme, as opposed to D-pad or gestural control, users could control movement speeds incrementally, as a function of tilt and rotation provided along the X and Z Wiimote axes. Additionally, the accelerometer control system required constant input from the Wiimote for the vehicle to move; with the gestural command scheme, once a gestural command was provided, the robot would continue with that command until another was given. As previously mentioned, vehicle drift required constant course corrections from the user; which paired well with the continuous input coming from the accelerometer and D-pad control systems. However, the gestural control system works better when performing commands less frequently. In this study, participants had to perform gestures one after the other to correct course errors – and if the user had difficulty performing a gesture, this would hinder performance.

Notably, the gesture recognition system was trained with a fairly low number of examples, and those came from only two individuals (neither who participated in the study). The implication, is that, for the users who had difficulty performing the gestures, the recognition accuracy could have been improved had those users contributed training examples to the machine learning algorithm.

Continuing, if gestural control had been paired with a Vector Driver component, which maintains a speed and heading (automatically correcting for vehicle drift), we believe performance and preference for the gestural input scheme would have improved.

Implications to Training

As mentioned previously, all participants were provided a brief orientation session before navigating the vehicle through the obstacle course. It was observed that participants were able to learn all three control schemes within only a few minutes. This includes the set of four gestures needed for gestural control. Notably, none of the participants had ever driven Gamblore before and were unfamiliar with its driving style. Additionally the average times and performance were comparable to that of the authors/developers of the system. A possible conclusion that can be made from these observations is that the ease of use and intuitive nature provided by the Wiimote input device had an impact on how quickly users could learn the task. Moreover, gestural communication did not present a learning cost greater than that of joystick or accelerometer style control. A formal study is required to validate this conclusion.

Additionally, the system presented in this paper allows an identical input control scheme to be used across multiple unmanned systems including those of different types. This results in a reduction of training time and cost through the reuse of the human-robot control system. Indeed, a user need only learn the nuances and domain of the platform being controlled rather than a wholly new control interface.

CONCLUSION

In this paper, we presented a system that uses commercial-off-the-shelf (COTS) technology for gesture input to allow control of multiple types of

unmanned systems. The Nintendo Wiimote provided an input device for implementing three control schemes including: traditional joystick control, accelerometer-based control, and gestural control. A machine learning algorithm was employed to classify dynamic gestures composed of Wiimote XYZ accelerometer values. A graphical user interface incorporated the Wiimote and displayed vehicle telemetry and control feedback. This software system interacted with an unmanned vehicle, called Gamblore, using JAUS. JAUS is a high level domain interface used for communication with unmanned systems that is independent of hardware or software implementation. This combination of tools resulted in a system capable of operating any type of unmanned vehicle.

An informal usability study was conducted to compare the performance and preference of the three control schemes implemented with the Wiimote. This study showed higher performance and user preference for accelerometer based control of the unmanned vehicle over the gestural and traditional joystick styles. However, we suspect this has more to do with the task performed within the study rather than a rejection of gesture based control. Indeed, users were able to successfully complete the obstacle course with relatively small differences in performance across the control schemes. This study implies that training time and cost can be reduced using an intuitive control scheme and even more so through software reuse enabled by interoperability standards such as JAUS.

Future work should expand on the idea of using gestural command for high-level tasks. This might include the creation of a gestural vocabulary for human-robot interactions in a military domain. Such a vocabulary can leverage existing arm and hand gestures already in use by military personnel. Applications of high-level tasks where gestural communication could apply are traffic control points, UAV takeoff and landing, and mixed-initiative (Barber, 2008) teams. We believe this work will establish an intuitive control mechanism for interactions with multiple types of unmanned systems.

ACKNOWLEDGEMENTS

The authors of this paper would like to acknowledge the efforts Larry Davis and Kamran Siddiqui in support of this project.

This work is supported, in part by the Office of Naval Research and also 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.

REFERENCES

US Department of Defense (2007). Unmanned Systems Roadmap: 2007-2032. Retrieved February 5, 2008 from http://www.acq.osd.mil/usd/Unmanned System Roadmap.2007-2032.pdf

JAUS Working Group (2008). Joint Architecture for Unmanned Systems Reference Architecture (Version 3.3). Retrieved April 22, 2008, from http://www.jauswg.org

ACTIVE Laboratory (2008). JAUS++. Retrieved June 20, 2008 from http://sourceforge.net/projects/active-ist/

Rubine, D. (1991). Specifying gestures by example. Computer Graphics, 25(4), 329-337.

Trouvain, B. & Wolf, H.L. (2002). Evaluation of Multi-Robot Control and Monitoring. *Proceedings of the 2002 IEEE Int. Workshop on Robot and Human Interactive Communication*, pages 111-116.

Space and Naval Warfare Systems Center (2008). Multi-robot Operator Control Unit. Retrived May 2, 2008, from http://www.nosc.mil/robots /resources/mocu/mocu.html

Chen, Jessie Y.C. & Barnes, Michael J. (2008). Robotics Operator Performance in a Military Multi-Tasking Environment. *Proceeding of the 3rd ACM/IEEE International Conference on Human Robot Interaction*, pages 279-286.

Barber, D., Leontyev, S., Sun, B., Davis, L., Nicholson, D., & Chen, J Y.C. (2008). The Mixed-Initiative Experimental Testbed for Human Robot Interactions. *The 2008 International Symposium on Collaborative Technologies and Systems (CTS 2008)*, pages 483-489.

Guo, Chen & Sharlin, Ehud. (2008). Exploring the Use of Tangible User Interfaces for Human-Robot Interaction: A Comparative Study. *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI '08)*, pages 121-130.