Evaluation of Gesture-Based Controls for Robotic Systems

Lauren Frazier

May 1, 2012

Contents

List of Tables

List of Figures

Acknowledgements

These are acknowledgements.

Abstract

This thesis aims to test the effectiveness/ease of use of smartphone gesture-based robotic control systems vs. traditional control systems. Robotic control systems are becoming more common, especially in the military. Arm and hand gestures are typical human forms of communication, so applying that to a robotic control system can yield a more intuitive system. With military applications, there are lives at stake, so having the most efficient, intuitive control system can make a large difference in the success of a mission and the safety of the soldiers involved. "Interactions and Training with Unmanned Systems and the Nintendo Wiimote" (Varcholik, Barber, and Nicholson) describes a gesture based control system that uses the Nintendo Wiimote to determine arm/hand gestures and control a robot. I propose to create a gesture based system using a smartphone and conduct an experiment similar to Varcholik, Barber, and Nicholson, collecting survey data from the participants on the the effectiveness and ease of use of each system.

Chapter 1

Introduction

This thesis aims to test both the perceived and actual effectiveness and ease of use of smartphone gesture-based robotic control systems vs. traditional control systems.

The need for human-robot interfaces is increasing rapidly. The military has already begun using unmanned vehicles in several different arenas (air, ground, water). In order to develop the most efficient human-robot interface, we turned to a traditional form of human-human communication, arm and hand gestures. Arm and hand gestures are typical human forms of communication, so applying that to a robotic control system can yield a more intuitive system. With military applications, there are lives at stake, so having the most efficient, intuitive control system can make a large difference in critical moments and improve the safety of those involved. A more intuitive system will also reduce the training time and expenses for the operators of the vehicle.

"Interactions and Training with Unmanned Systems and the Nintendo Wiimote" [3] describes a gesture based control system that uses the Nintendo Wiimote to determine arm/hand gestures and control a robot. They then conducted a study where subjects used Wiimote gesture system and a more standard system and filled out a survey to indicate how effective the Wiimote system was as compared to the standard control system for the robot.

I proposed a gesture based system using a smartphone and conducted a human factors experiment similar to Varcholik, Barber, and Nicholson. The system uses a Samsung Galaxy S II phone for the gesture-based input, a tilt-based controller, and a touch screen D-Pad controller, and a Microsoft XBOX 360 controller for the more traditional input. Subjects used each of the four controls in a random order to guide an iRobot Roomba vacuum cleaner through a short time trial course. The raw data (video and observations during the time trials) was analyzed with metrics like the time required to

complete the course, number of times the subject went outside the course boundaries, number of times the subject acknowledged making a mistake, etc. After the experiment is complete, subjects also filled out a survey, and both sets of data were used to determine which control scheme is more effective and intuitive.

This thesis introduces an inexpensive gesture based control system that uses commercial, off the shelf hardware (an Android smartphone and a Roomba vacuum cleaner). It also provides data showing both subjects' objective performance and their perceived experience with each controller.

The rest of this thesis is organized as follows. In the next chapter, I present a taxonomy of related work. Chapter 3 provides implementation details for the different controls. Chapter 4 describes the design of the human factors experiment, including choice of hardware, track design, experimental procedure, and the questionnaire. In Chapter 5, I analyze and discuss the results of the experiment. Chapter 6 provides the conclusion and a brief summary.

Chapter 2

Related Work

While this paper has most of its origin in [3], there is a large body of work about robotic control systems that led up to this point.

Chapter 3

Implementation of Control Systems

In this chapter, I will describe the implementation of the four control systems used in the experiment. First, I describe the "standard" XBOX 360 controller setup, then the application used for the D-Pad and tilt controllers, and finally the gesture-based control application.

The robot in question is an iRobot Roomba 560 vacuum cleaner. None of the four control schemes require making modifications to the Roomba, and the vacuum/brushes are turned off during the experiment. It is controlled from a standard wireless Microsoft XBOX 360 controller and a Samsung Galaxy S II Android phone.

3.1 XBOX 360 Controller

The XBOX 360 controls for the Roomba are written in Python (see Appendix). The program uses a library called RoombaSCI to communicate with the Roomba via Bluetooth. RoombaSCI is a Python wrapper for iRobot's Serial Command Interface Specification, which allows commands to be sent to the Roomba through its serial port. RoombaSCI abstracts tasks like setting up the connection and sending bytes to specific motors into commands like "forward", "stop", and "spin left" [**?**]. It also allows the programmer to set the speed of the Roomba within the range allowed by the hardware (-500 - 500 mm/s). [1]

A RooTooth bluetooth-serial adapter was used to communicate with the computer wirelessly. The Pygame library is also used to properly capture the input from the controller.

The end result is a controller that can be used to move forward/backward

and rotate to the left or right in place. The right trigger sends the "forward" command, the left trigger sends "backward", and the left analog stick pressed to the left or right sends a "left" or "right" command. The robot continues moving forward/backward/left/right as long as the trigger or analog stick is held, and will stop when it is released, or another button is pressed. When using the XBOX 360 controller, the robot can only move at the maximum speed, and it can only perform one of forward/backward or rotation at any given time (forward motion must stop in order to turn).

xbox photo

3.2 Cellbots Application

The D-Pad and the tilt controls were downloaded to the phone as part of a single app called Cellbots. Cellbots is an open source Android application, available for free online or in the Android Market. The app comes with several control schemes, including the D-Pad controls and the tilt controls. It also included voice controls and an on-screen Atari-style joystick which were not used in the experiments but could be utilized in future research.

screenshots of cellbots

The D-Pad controls require the user to press one of four on-screen buttons, arranged in a cross. Like the XBOX 360 controller, the robot moves forward/backward and rotates in place when the appropriate button is held down, stopping when the button is released. Also like the XBOX 360 controller, the D-Pad only allows one speed and one type of movement at a time.

The tilt controls use the phone's accelerometers to control the robot. The user holds the phone horizontally (screen facing up) and holds down an on-screen button. While the button is held down, tilting the phone forward (rotation around its x-axis) causes the robot to begin to drive forward. The phone can be rolled right or left (rotating around its y-axis) to steer. The tilt controls differ from the other controls in two ways: first, the speed can vary. The more the phone is tilted forward, the faster the robot moves, up to the maximum speed of 500 mm/s. The second difference is that the tilt controls allow forward/backward motion and turning at the same time. For example, if the phone is tilted forward and rolled left, the robot will perform a left turn, rather than rotating in place.

phone axis diagram

3.3 Gesture-Based Control Application

The gesture-based controls are implemented on top of the existing Cellbots application. Cellbots is an open source application, so the gesture controls

were added as a fifth control type. To recognize the gestures using accelerometers only, I turned to another open source Android application called GestureTrainer. GestureTrainer is an implementation of the concepts outlined in [2] for gesture recognition using accelerometer data.

screenshot of gesture-trainer and cell-bots

Chapter 4

Human Factors Experiment Design

General description and justification for choices in the experiment.

4.1 Platform

4.2 Track Design

4.3 Questionnaire Design

4.4 Experiment Procedure

Chapter 5

Results

In this chapter, I will discuss the results of the experiments.

5.1 Time Trials

5.2 Questionnaire

5.3 Analysis and Hypothesis

Chapter 6

Conclusion

This is a conclusion.

Bibliography

[1] iRobot Corporation. irobot roomba serial command interface (sci) specification, 2005.

[2] K. Jacobson. Python library for the roomba sci, 2008.

[3] Robert Nesselrath. Takg - ein toolkit zur automatischen klassifikation von gesten. Master's thesis, University of Saarland, 2008.

[4] Paul Varcholik, Daniel Barber, and Denise Nicholson. *Interactions and Training with Unmanned Systems and the Nintendo Wiimote*, volume 2008, pages 1–9. NTSA.