TOUCHLESS 3D INTERFACE CONTROLLER FOR THE SEAPERCH ROV

Author:
DALLIN M. BRIGGS
Brigham Young University
Provo, UT 84602
Email: dallinbriggs@gmail.com

Abstract

This research demonstrates a new method for operators to control the SeaPerch remotely operated vehicle (ROV). Engineers and researchers are constantly seeking ways to improve the manner in which humans interface with robots. This is particularly important when robots become more complicated to operate and the user's attention becomes spread over several simultaneous tasks. This touchless 3D interface allows the user to control the robot simply by moving his/her hand in the desired direction for the robot.

Introduction

As robots become more capable of doing multiple tasks at one time, their human counterparts must also be able to manage and control more simultaneous tasks. Currently behind almost every robot is a human user that is somehow interfacing with the robot's controls to manipulate the robot into completing a desired task. For a simple robot, the controls are often a couple of switches, buttons, or levers. These controls are designed to make it easy for the user to accurately control the robot. However, as more functions are added to the robot, more controls are often added as well. Soon the once simple controller becomes very complicated for the user to remember which switch controls each function of the robot and errors can be easily introduced into the system through operator error. Although the SeaPerch is a very simple robot, the goal for this research is to demonstrate the ability of new methods for robotics controllers to reduce the work load of the operator and to reduce the total amount of operator error. A similar study to reduce the operator error through the use of natural hand motions was conducted by the Tokyo Metropolitan Institute of Technology [4]. The idea of using hand motion to control the SeaPerch ROV stemmed from this research.

This study shows that using intuitive and relatively natural hand movements can be a more effective way to manipulate a robot's movements. By creating a touchless 3D controller, the operator can control the robot in the desired direction by moving his/her hand in the same direction. For example, if the operator wants the SeaPerch to descend, he would simply move his hand in a downward direction. If the operator wants the SeaPerch to turn right, he would simply move his hand to the right and so forth. This particular 3D controller senses the operator's hand motion through capacitive sensing. A method for using capacitance to read hand position will be presented in this paper.

Although the methods for creating a device that measures hand movement in this research are certainly not the most accurate, it can still allow for basic control over the SeaPerch ROV. As funding permits, more accurate sensors may be purchased to make the 3D controller more accurate and expand the capabilities of it.

Methods

In order to control the SeaPerch robot in a three dimensional space with hand motion, we need a way to measure the position of the hand in a 3D coordinate system. Whenever we think of a 3D coordinate system in mathematics, we usually picture three coordinate planes. If we are using the x, y, and z axis in a Cartesian coordinate system, we can picture the xy, xz, and yz planes. Measuring the distance from these three planes can give us the exact location of any point in 3D space. Using this same principle, we can find the position of the user's hand through measuring the distance it is from three plates that are positioned to form the shape of a half-cube as shown in Figure 1. To measure the distance from each plate, we will use what is called capacitive sensing.

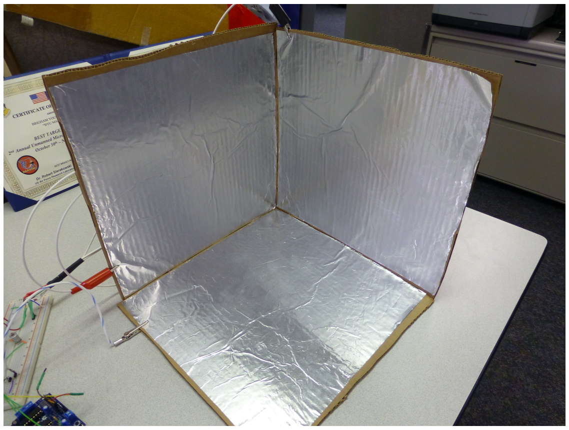

Figure 1: *Pictures are the three capacitive plates representing the xy, xz, and yz planes used to reference hand position.*

Capacitive Sensing Basics

To understand capacitive sensing, it is good to review some of the principles of capacitors. The simplest form of a capacitor mainly has two electrically charged plates that are separated by air or some other dielectric material [3]. Energy is stored in the capacitor because the two plates are insulated from one another, but the negative and positive charges are attracted to each other through the dielectric. Recall, however, that when a capacitor's voltage is changed, such as in an AC circuit, there is a time constant that causes a delay in the change in voltage. This time constant is given by

$$\tau = RC \tag{1}$$

where τ is the time constant, R is the resistance of the circuit and C is the capacitance. In order to increase the time constant, we must either increase R or C. Capacitive sensing utilizes the change in capacitance to measure the distance between a user's hand and a charged plate.

Capacitance of two electrically charged plates is given by the equation

$$C = \epsilon_r \epsilon_0 A/d \tag{2}$$

where ϵ_r is dielectric constant (for air $\epsilon_r \approx 1$) and ϵ_0 is the electric constant ($\epsilon_0 \approx 8.854x10^{-12}Fm^{-1}$). Since the area, A, of our plates won't be changing, the only way to change the capacitance is to change the distance, d, between the plates. In the case of our 3D touchless controller, one plate is a charged piece of aluminum foil and the other 'plate' is the user's hand. Using equations (1) and (2), we can derive that

$$\tau = R\epsilon_r \epsilon_0 A/d \tag{3}$$

which shows that the time constant is inversely proportional to the distance between the charged plate and the user's hand. Therefore, if the user brings his hand closer to the plate, the time it takes for the circuit to change states will be longer. If the user distances his hand from the plate, the time will decrease.

Using this principle, we can use an Arduino microcontroller to alternate the state of the voltage on the plate and then measure the time it takes for the change to be received back to the Arduino. This is done by connecting the plate to a send pin on the Arduino that is changing states and then connecting a receive pin that receives the changed state [1]. The Arduino then computes the time it takes for that changed state to return to the receive pin and then returns an arbitrary value. For this paper, an Arduino Uno microcontroller was used for this calculation.

Testing

To use the this arbitrary time value to control the motors on the SeaPerch, we establish three states of control for each motor: forward, stop, and reverse. Each state will accept a range of time values that are being returned from the Arduino to initiate that state. Because the time values are arbitrary and differ with each different user, humidity, and other external variables, the program written on the Arduino must be able to determine the range of the incoming values and calculate the median value each time the program is executed. To do this, a calibration sequence is used. The calibration only requires the user to move his hand from the outside-upper corner to the inside-lower corner of the box. This will provide a minimum and maximum value for the time data and a mean can be calculated from these values.

To test this method, the Arduino is connected to only the bottom plate of the box. If this method is successful, the Arduino will be connected to all three plates to receive a 3D position of the user's hand. The Arduino program is written to output the values of the time since the program began along with the current, minimum, and maximum values of the received time and the median value and its upper and lower bounds. With the program in this state, the values for each of these measurements are able to be recorded from the Arduino serial monitor and plotted for further analysis.

To obtain a set of data that would verify that the Arduino is indeed recording the maximum and minimum and calculating the mean, a user moved his hand towards the bottom plate starting from 12" away. Once the user was approximately 1" away from the plate, he moved his hand back to 12" away and repeated the process. This provided results that showed the Arduino storing the appropriate values and returning them in real-time. To verify that these values were capable of controlling the motors on the SeaPerch, a small servo motor was attached to a simple circuit that was attached to an Arduino motor-shield. This setup can be seen in Figure 2.

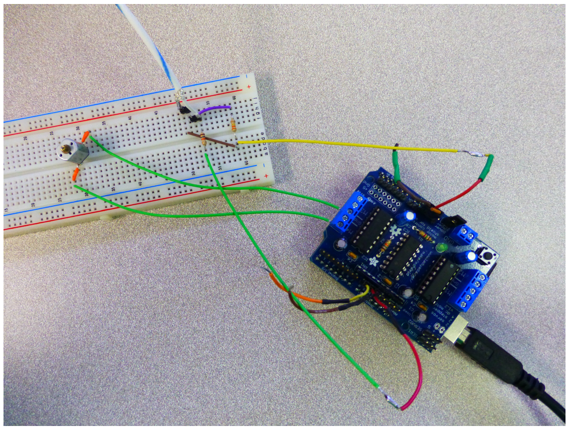

Figure 2: *This shows the temporary setup used to test controlling the motor with change in hand position.*

Results

When the Arduino is first initiated and the user has her hand 12" or more from the plate, the time values being received are at a minimum. The Arduino records the minimum value and stores it. As the user moves her hand towards each plate and stops just before touching the

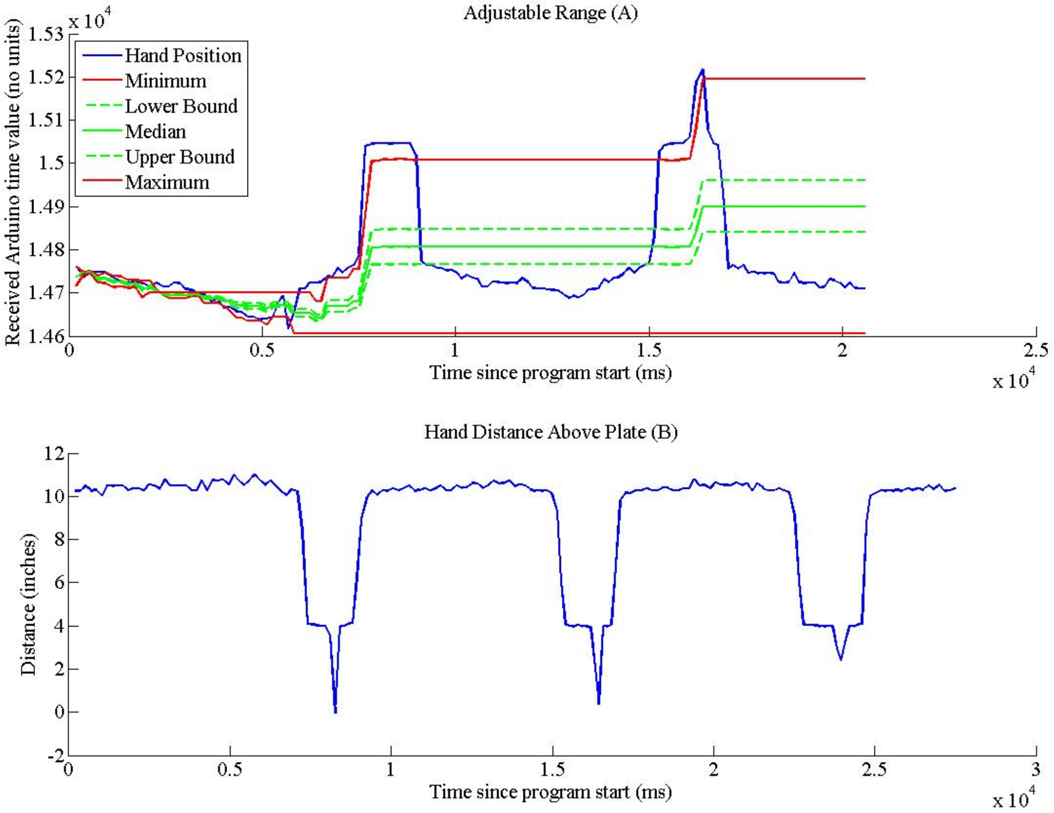

Figure 3: *Graph (A) shows the Arduino automatically adjusting the bounds for motor control to compensate for the user's range of motion. Graph (B) shows the accuracy and repeatability of the Arduino estimating the user's hand position relative to the plate.*

plate, the time values for the plate reach their maximum values. The Arduino records the maximum and minimum values and uses these to calculate the mean. The mean is then given an upper bound and a lower bound that is used for keeping the motors in the off state. In Figure 3(A), test results can be observed in testing the Arduino's ability to automatically adjust for different ranges of motion the user utilizes. As the program is initiated, all values for the maximum, minimum, hand position, and median are relatively the same. The user then moved his hand towards the plate, stopped approximately 6" from the plate, and then removed his hand again. At this point, it can be observed that the maximum, median, upper, and lower bound values have been adjusted to allow for motor control on the SeaPerch. A few seconds later, the graph shows the user moving his hand towards the plate, stopping 1" away, and removing his hand again. The maximum, median, upper, and lower bound values have been adjusted once again to compensate for the user's full range of motion. Having the Arduino do this automatically makes it very easy for the user to calibrate the system as he is using the controller.

The results from performing the test discussed in the methods section demonstrated that the Arduino was able to perform all the required functions. Having the user move her hand all the way towards the bottom plate, stopping just before she touches the plate, removing her hand and repeating the procedure showed the repeatability of sensing her hand in the same location multiple times. The results from this test are shown in Figure 3(B). This graph shows that although the Arduino is fairly accurate in predicting the distance the user's hand is from the plate, it cannot detect a hand further than 10" from the plate. As the user moves her hand further than 10" from the plate, noise in the signal from the surrounding environment becomes dominant and the hand position can no longer be observed. However, it can also be seen from this same graph that the Arduino can fairly accurately and consistently predict the user's hand position when it is between 2" to 10" above the plate. When the user moves her hand closer than 2" to the plate, a spike can be seen that appears to make the hand position look like it is touching the plate. This is due to the Arduino not being on the exact same ground circuit as the user, and static electricity affecting the charge on the plate. This spike, however, does not seem to affect the ability of the user to control the motors on the SeaPerch because it does not change the state of motor control.

In practicality, despite there being noise in the circuit that affects the time values of the Arduino, it was found that using this setup provided fairly good results for controlling the motors on the SeaPerch ROV. Because the way the Arduino microcontroller processes loops and stores information, some of the noise is removed from the computations. This allowed the user to move her hand in the desired direction of the ROV and the motors would initiate and the ROV would follow that direction. This operated fairly consistently. However, it was found that after 20 minutes of use, the Arduino would have to be reset. Resetting the Arduino restores the time values and allows the user to recalibrate the system. However, this setup still made for a very inexpensive 3D touchless controller for the SeaPerch ROV.

Conclusions

In the effort to reduce the operation load on the user, this 3D touchless controller offers a more intuitive method of control for the SeaPerch ROV. Instead of pressing switches and buttons to direct the robot, the user only has to move his hand in the desired direction. Approaching robotic controls from this view opens up more opportunities for research in improving the control interfaces humans currently use. Having an operator use 3D motion to control robot systems also allows for potentially more complex operations.

Capacitive sensing is a usable way for controlling a robot with 3D motion. However, in more serious applications, capacitive sensing would not provide accurate enough results for control. This is because capacitive sensing is mostly good for close proximity sensing [5]. Trying to sense objects further away becomes difficult due to noise from the surrounding environment. In the case of the SeaPerch ROV, capacitive sensing works well because there is room for error in the operation. In other cases, as much operational error may not be acceptable. Other systems do exist that provide more accurate results. Such systems typically use a number of cameras viewing objects from different angles to predict motion in 3D space. For the SeaPerch program, however, this capacitive sensing 3D interface is a great experiment in robot controls and demonstrates a new method for controlling a ROV in low risk environments.

Acknowledgments

I would like to thank Kyle McDonald for his research in capacitive sensing and his Arduino

source code for sensing hand position. I would also like to thank Stephen Carlson for his explanations of how the Arduino is computing the time signal.

References

[1] P. Badger. (2013, Apr.) Capacitive sensing library. [Online]. Available: http://playground.arduino.cc//Main/CapacitiveSensor?from=Main.CapSense

[2] K. McDonald. (2008, July) Diy 3d controller. [Online]. Available: http://www.instructables.com/id/DIY-3D-Controller/

[3] G. Rizzoni and T. T. Hartley, *Principles and applications of electrical engineering.* McGraw Hill, 2000, vol. 3.

[4] E. Sato, A. Nakajima, and T. Yamaguchi, "Nonverbal interface for user-friendly manipulation based on natural motion," in *Computational Intelligence in Robotics and Automation, 2005. CIRA 2005. Proceedings. 2005 IEEE International Symposium on.* IEEE, 2005, pp. 499–504.

[5] R. G. Walmsley, M. A. Hopcroft, P. G. Hartwell, G. Corrigan, and D. Milligan, "Three-phase capacitive position sensing," in *Sensors, 2010 IEEE.* IEEE, 2010, pp. 2658–2661.

Appendix

Inexpensive materials were used for this project to match the nature of the SeaPerch Organization's goals. This allows young, independent students to purchase the required materials through public school budgets or their own. Below is a list of the required materials to build the controller. Instructions on how to build the controller can be found at the instructables.com website [2].

- 12" X 12" cardboard (3)
- 10 $K\Omega$ resistors (3)
- 220 $K\Omega$ resistors(3)
- Alligator clips (3)
- Arduino Uno (1)
- Adafruit Motor shield (1)
- Shielded cable (6')
- Aluminum Foil
- Masking Tape
- Desktop or grounded laptop computer