Point to Location Framework

ERIK REED, BENOIT BERNADEL, KRYSTAL YING, and SARALEE KUNLONG
Carnegie Mellon University

We propose the Point to Location Framework, a novel method of controlling devices with orientation being crucial to their functionality. This framework allows a user to control a directional device by pointing to a desired location, commanding the device to pivot to face the location in its reference frame.

Categories and Subject Descriptors: H5.2 [**User Interfaces**]: Input devices and strategies—*Mobile*

General Terms: Java, Sunspot, Directional Devices, Frame of Reference

■

1. INTRODUCTION

If we look at devices we use today – cellphones, laptops, and even printers, we are very interested in communicating with the world with wireless devices without mobility restrictions. People would also like the ability for remote control – the ability to free their hands for other work or the ability to control something without necessarily being right next to it. Unfortunately, some of these remote control systems are not intuitive or have poor performance or lag if they are not in the same frame of reference as the device they wish to control. Typically, the user has to manually direct a device to a point at a specific location and verify the direction by reading output from the device, or in the case of a camera, check to see if the image has been captured

Our goal was to produce a more intuitive and easy control framework for directional devices. Here we use the term directional device to denote a device where its orientation is useful to control in its functionality. Examples of directional devices include cameras, lasers, lights, drills, antennas, and distance meters. Our proposed framework gives the ability for a directional device to point to a location specified by a user.
Examples of applications include:

—Easily directing a light on a mount while performing work in a dark environment (construction workers, miners)

—An individual controlling a camera on a tripod when they are alone, or when the camera is mounted out of reach

—Directing a drill to a position for machining purposes

—Controlling a spotlight without having to be behind it, or even nearby

2. RELATED WORK

We have investigated into what has already been done in this area and we found several research that share similar ideas of what we're trying to achieve.

In previous research by Sun Labs, the SunSpot is used as a controller of servos that were connected to a camera. Two Sunspots were used, one SunSpot as a base station and as is a controller which communicated through radio signals. The camera was mounted on the servos, and the servos moved by input range values given in the Servo control program. The method used for the Servo control program in this research was setting the absolute minimum and maximum value of the Servo's location (Servo's range). The two main functions of this implementation are Pan and Tilt; the servo pans when it moves through the entire servos range and tilts when it inclines forward. In our implementation, the absolute range values of the Servo is not required, we use geometric distance calculation to indicate the location the servo would move to, thus allowing the servo to move to the exact location the user points to instead of panning the entire servo range. This gives the user more control of any devices mounted to the servo.

Research by CMU students [Cheng et al. 2010] explored the idea of training robots to behave like human by using Motion Tracking algorithm to translate human movements into motor control contexts, such as motor ID, voltage and time. This employed a Web camera, allowing the user to control a robotic arm by human arm movement. In our implementation, we did not use computer vision techniques or motion tracking algorithm. The method we usde to control the mount does not need a web camera or require the user to be in a specific position to control the mount and can be integrated with different kinds of devices, not only robotic arms. The difference of human body from different human operators is also not a factor in our method thus eliminating the accuracy challenge from mapping human body joins to sections on the robot arm. Also, by using our method, the user does not have to worry about having the right movements or watching the device by moving it. The user simply has to perform two steps: point to the device, and point to the location, then the device can move by itself to the desired location.

After a fair amount of research, there is no such implementation, at least at an industry level, of such a device that allows a user to direct a device at a location without the use of any complex computer vision algorithms. By not relying on complex algorithms and allowing the device to move without regard to a certain point in a simple and intuitive way, we believe our project can be more applicable and more suitable to real-life practices.

3. METHODS

The ability for a directional device to calculate a new location relies on geometric and trigonometric relationships between obtained sensor readings. Using a distance sensor and a compass angle, a side-angle-side triangle can be created to solve for the desired angle of an external device (Figure 1). Figure 2 demonstrates how the law of cosines and law of sines is applied to solving for the unknown angle.

4. PARTS AND IMPLEMENTATION

4.1 Overview

In creating the Point to Location device, we used two Java Sunspots, arduino-like microcontrollers allowing for a high level programming interface in Java with simple implementations for the gathering of accelerometer readings, temperature, and with the ability to push current to pins and attachable breadboards, among other things. One Sunspot is fixed to a platform to pivot and rotate using servos while the other is handheld. We adopt the nomencla-

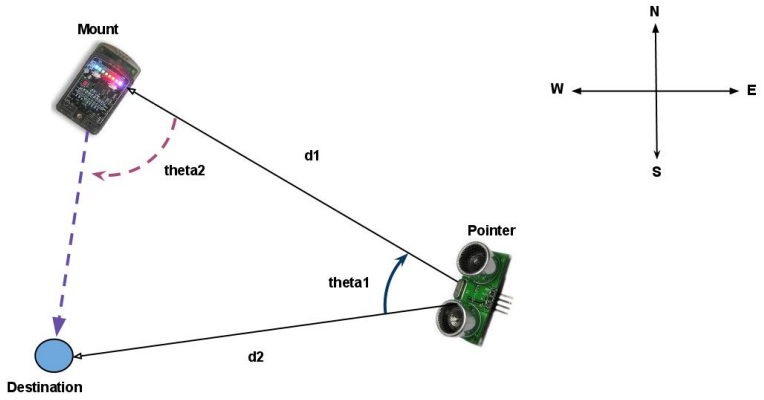

Fig. 1. The triangle created by using two distance measurements and an angle. The solution to the side-angle-side triangle is computed via law of cosines and law of sines.

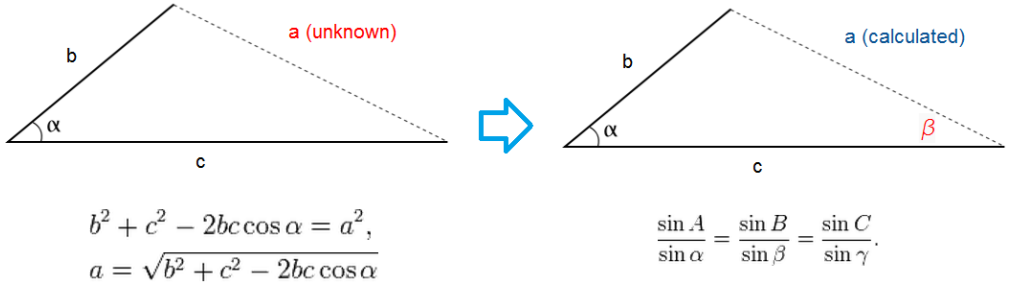

Fig. 2. The triangle created by using two distance measurements and an angle. The solution to the side-angle-side triangle is computed via law of cosines and law of sines.

ture of referring to the fixed Sunspot as the Mount, which rotates and pivots to turn to face as desired location, and the controlling, pointing Sunspot as the Pointer. The Pointer acts as the handheld device responsible for relaying data to the Mount, which performs computations to direct servos and turn to the location specified by the Pointer.
In addition to the Sunspots, the following parts were used:

—SainIC Arduino Ultrasonic Module HC-SR04 Distance Sensor
—(2x) GY-26 Digital Compass Sensor Module
—HS-81MG Servo
—HS-311 Servo

The Pointer collects and transmits data necessary for the Mount to determine the location to pivot to in its reference frame. The pointer was fixed with a SainIC Arduino Ultrasonic Module HC-SR04 Distance Sensor, which acquires distances to objects within a 10 meter range using emitted infrared signals. Additionally, a GY-26 Digital Compass Sensor Module was attached to collect the angle coordinates used in the Mounts calculations. These two sensors used the Sunspot's 5 volt power supply, allowing the Pointer to be mobile. Both the Mount and Pointer employed the accelerometer functionality of the Sunspots, allowing the tilts along each axis of the devices to be captured without an external sensor.

The Mount was also fixed with a compass module, which was used to determine its current angle and for calculating how to move to a desired angle. To allow for two degrees of freedom, panning and tilting, two servos were attached, along with a base platform to hold the devices in place. The HS-81MG is larger than the HS-311 and was used for panning, while the HS-311 was used for tilting. The HS-81MG held the HS-311, Sunspot, and compass module in place.

4.2 Pointer

The Pointer sends data to the Mount via IEEE 802.15.4 transmission when a switch on the Sunspot is pressed. This data consists of three 64-bit floating point numbers and an integer; the three floating points contain the angle, tilt, and distance sensor readings, while the integer specifies the command intended by the pointer. We specify

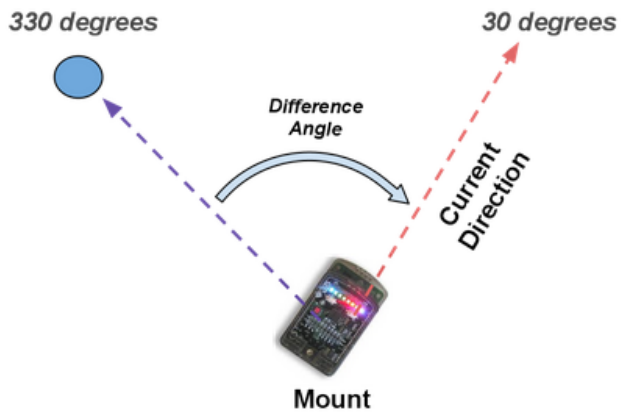

Fig. 3. The calculated difference angle.

two commands: the first notifies the Mount of initial angle, tilt, and distance values without any computation, while the second triggers the Mount to perform the new angle computation based on the parameters of the first command and move pivot to a new angle. This allowed for flexibility in future implementations; the distance to the Mount can be synchronized by using the first command, and assuming the Pointers location is stationary, the second command could direct the Mount to new positions without requiring two distance and two angle measurements.

The two commands were sent to the Mount depending on the Sunspot switch pressed, triggered via Java event handlers. To retrieve angle measurements from the GY-26 Compass Module, a packet was sent to the compass using the Sunspots Inter-Integrated Circuit (I2C) interface. The response packet contained 8 bytes, 2 of which contained an integer corresponding to the compass angle times 10, a value between 0 and 3600. Both the Pointer and Mount used the average of 5 compass readings before performing a transmission or any calculations. Additionally, the compasses were calibrated before use by sending a calibration packet to each compass and slowly rotating for approximately two minutes.

In retrieving the distance, the Sunspot demoboard was used to push current through IO pins D0 and D1, triggering a response from the SainIC Ultrasonic Distance Module. A pulse was sent through D1, causing the distance module to emit and echo. The time elapsed in microseconds until the echo is received was returned via IO pin 0. Dividing this value by 148 yields the distance traveled one way in inches. As with the compass module, distance measurements were averaged over 5 sensor readings before any transmissions or calculations.

4.3 Mount

The Mount controls two servos using an additional 4V power supply and the Sunspots high current H0/H1 pins. The servos are controlled using a Sunspot Servo library, allowing a value between 0 and 1 to be used to direct the servos to a position within their operating range. When powered on, the Mount acts idle, listening every .25 seconds for a transmission from the Pointer. If the command to turn to a location is received, the formulas described in Methods are used to calculate the new angle for the Mount. This angle will be referred to as the goal angle. Additionally, a arcsin lookup table is used in the computations due to restrictions of the Java ME math libraries.

The two servos, controlling tilt and horizontal movement, are controlled separately and use separate angle values. The tilt calculations use the Sunspot accelerometer tilt values as angles, while the horizontal movement uses compass sensor readings. To direct the servo to navigate to the goal angle, the difference angle between the current angle and the goal angle is calculated (Figure 3). This angle has multiple correct values, so the smallest is calculated. With the angle difference obtained, the servo begins moving in the direction of the goal angle in several iterations, halting when it current compass readings are within a tolerance of 1 degree from the goal angle.

5. HOW TO USE

To use the Point to Location Framework, a directional device was fixed to the Mount. Provided there is line of sight to the Mount and the range of the distance meter are not exceeded, the user can point the Pointer at the Mount and press switch 1. This causes a visible notification on the Pointer that the sensor readings were successfully retrieved, as well as on the Mount to notify that the transmission was received. Pointing the Pointer to another location and pressing switch 2 commands the Mount to perform angle calculations and turn to face the location specified by the Pointer. There will be a visual notification of success from both the Mount and Pointer, or a flashing red light indicating an error from the Mount. This implies the servo motors are dysfunctional, or the Mount is attempting to turn to a location outside of its servo motor range.

6. RESULTS

In order to evaluate the accuracy of our system, we took measurements on the accuracy of each of our components. The distance sensor was stated to have a range of up to 10 meters. We took 46 trials with the distance sensor. We placed the distance sensor 37 inches away from a specific location using a tape measure. We then com-

pared the distance calculated by the distance sensor. In this ideal case of the item being measured is against a wall rather than in free space, the average percent error was 4.4%. However, the percent error in free space could get as high as 63.4% error in magnitude.

We observed, however, that the majority of error was a result of faulty compass measurements. To test accuracy, the compass was aligned along a grid and rotated 90 degrees. The desired difference of 90 between these measurements is used to determine error, which is $90-$ the recorded value. The compass was rotated a full circle back to the original alignment and this process was repeated 11 times. The measured difference varied depending on which angles were being measured; in the case North to East, the mean error was 1.21 which a standard deviation of 2.20. This error would likely be acceptable overall. East to South had an increased mean error of 7.21 with a standard deviation of 2.33, while South to West had the worst accuracy with a mean error of 20 and standard deviation of 2.00. The mean error of up to 20 in magnitude substantially affected the calculated angles of the Mount. As a consequence, whether or not the Mount correctly went to a desired location was hit or miss.

7. FUTURE WORK

Sensor precision caused significant error in accuracy during our experiments. In order to create a reliable product, these issues must be addressed or minimized. This could be accomplished by building an error table to attempt to extrapolate error or by anticipating interference. Using different types of sensors to find an optimal combination of accuracy, size, and weight would be ideal. Different means of reducing interference from magnetic fields could also be helpful, such as placing the compasses farther from metal.

Additionally, the servos used for our experiments are limited to 180 degrees of motion. This only allows for the Mount to point at locations in a semisphere. The distance meter is also limited by range, resulting in a constant distance after this range has been exceed. The consequent, induced error is proportional to the distance more than the range capacity. A side effect of using an ultrasonic distance sensor is that the echo generated can easily miss the intended target, especially if the target is small and there are walls in the distance. Using a different form of distance meter, such as one not involving timing an echo for distance estimation, would alleviate the error when target locations are small.

REFERENCES

CHENG, H.-T., SUN, Z., AND ZHANG, P. 2010. Imirok: Real-time imitative robotic arm control for home robot application. *Carnegie Mellon University Research Showcase*.

SUN. Project sun spot remote control camera. *Sun Labs Research*. http://www.sunspotworld.com/docs/AppNotes/SunSPOTCamera.pdf.