Department of Computer Science

Robot Navigation & Control

by Daniel James & Dominic Orchard

Supervised by Nasir Rajpoot

Introduction

Our research, carried out during summer 2007, explores the many facets and problems involved with robot control, navigation and the development of natural reactive behaviour in robots. This research was carried out using a two-wheel differential drive robot designed and built in 2005 by technicians in the DCS (Department of Computer Science) at the University of Warwick that has been largely unused until now due to an incomplete and untested control system. At the outset of the project is was clear that a large amount of initial work was needed to improve and complete the sensory and control systems for the robot before further experimentation and research could be carried out. This task took a considerable amount of time, but has been vital for bringing the robot into a usuable state for further research.

Developing Control Systems

Side Sensors

The robot is equipped with four outward facing ultrasonic sensors on each of the four sides; these are mounted roughly half way up each side. Each sensor is controlled locally by a microprocessor that controls the firing of the sensor and calculates the distance based on the time taken between firing the ultrasonic pulse and receiving an echo signal. Each of these is connected via a bus to a central master controller which controls the local controllers, collects all results and interfaces with the computational unit of the robot, the host computer.

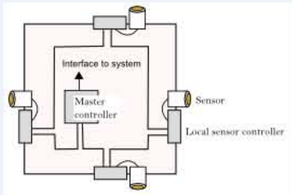


Figure 1: Ultrasonic sensor arrangement

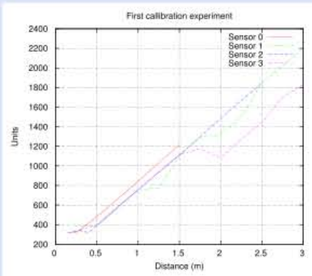


Figure 2: Graph of initial experiments with sensors

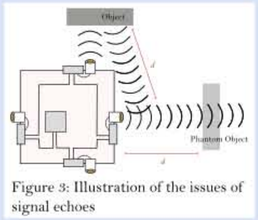


Figure 3: Illustration of the issues of signal echoes

The original implementation of the sensor controller system had several problems that caused inconsistent and error prone results as can be seen in Figure 2. The original master controller software fired a sensor every 2 milliseconds upon request from the computer. However this system suffered from inconsistent readings and the detection of "phantom" objects.

For example: Sensor A fires on one side and 2ms later the next sensor, Sensor B, is fired on another side. The wave from Sensor B doesn't reflect off any nearby objects, however the wave from Sensor A reflects off an object near to it, reflecting some of its wave onto Sensor B. Sensor B reads this as an echo from an object near itself, and returns a false positive reading.

At minimum sensitivity the ultrasonic sensors can detect objects at a distance of roughly 5.6m (11.2m in total journey time), this equates to roughly 33ms, hence the original timing gap of 2ms was significantly defficient. The new controller implementation leaves 50ms gaps and has hardware and software timeouts to detect up to just over 5m accurately and consistently over each sensor with very low error in repeat readings (see Figure 4). The 50ms gap also affords time for handling data requests from the host computer without slowing down the collection of results. The master controller's main mode of operation continually collects data and builds a queue of the last 5 measurments, hence measurements from the last second are recorded at the hardware level and can be used to calculate velocities of approaching objects.

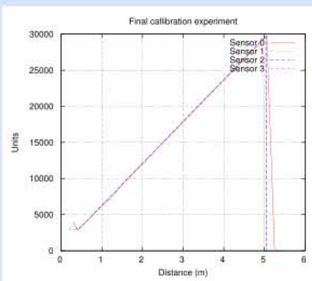


Figure 4: Graph of final experiments with sensors

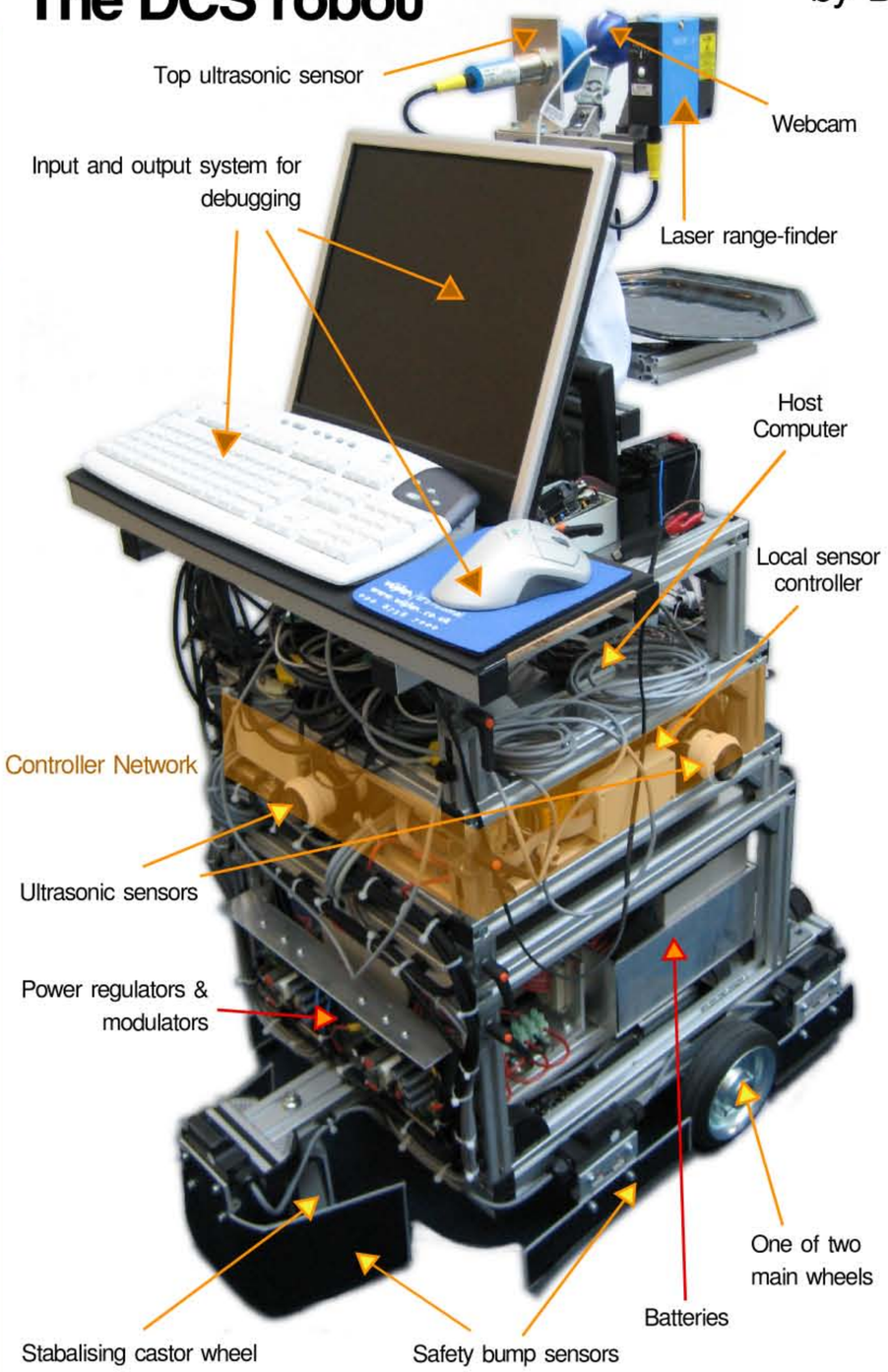
Motors

The locomotion system is comprised of two motors and two positioning controllers. The controllers allow for distance and velocity centric motor operations to be performed with ease and with a high degree of accuracy. Each wheel, motor and controller form an independent triple that allows the host computer to request finite robot movements such as forward, backward and spin left or right; as well as infinite movements in line or arcs which continue until requested to halt.

Top Sensors

The robot is also equipped with two forward facing sensors mounted at the very top of the robot controlled by a single controller; a laser distance sensor and another ultrasonic sensor of higher quality than the side sensors. Both of these sensors produce a continuous analog signal giving the controller the ability to instantaneously sample the data from either sensor whenever the host computer requests it. While the primary purpose of the side sensors is to facilitate collision detection, the top sensors are intended to be used for measuring more precise distances.

The DCS robot



Email: Daniel.James@warwick.ac.uk, D.A.Orchard@warwick.ac.uk

Figure 5 - Representation of an environment in robot's memory

Navigation

Navigation systems developed from virtual environments are being implemented on the robot, giving it the ability to explore an environment and build an internal map of its surroundings. The internal map is then used to navigate the environment efficiently using the A* algorithm to calculate the shortest paths to target destinations.

Waiter

The Waiter program is a reactive program written in C. It has two modes of behaviour "Wandering" and "Serving". When in "Wander" mode the robot has the behaviour of wandering around a room keeping a fair distance from most objects, switching into "Serve" mode if a dynamic object such as a person comes within close proximity. When the robot is executing "Serve" mode it remains in one spot but turns to face its nearest object, hence turning to people standing around it. These people can then retrieve food (cheese or biscuits) from the tray mounted on the front. After 4 cycles of serving, or alternatively if the people surrounding it retract, the robot switches back into "Wander" mode.

This program has been used in several environments with high crowd densities, such as open days and receptions, generally performing well at offering people food and avoiding collision.

Functional Reactive Programming

Functional Reactive Programming is a programming paradigm developed from research of the Yale Haskell Group at Yale University, which is purposed for developing systems that are a both continuous and discrete. Programs are developed using the Haskell functional programming language, in a high level declarative style using the concept of 'arrows' to express stream processing like operations [1]. A robot programmed in a reactive style will operate by reacting to external input from its environment and producing an appropriate change to its output (motors and actuators). There is a generic FRP framework which was complemented with code specific to the DCS robot and its input and output capabilities. Using this pairing, simple reactive programs were used to experiment with this approach to robot control.

Subsumption Architecture

In the Brooks' paper "Intelligence without representation" [2] a system known as the "subsumption architecture" is proposed for emergent behaviour of intelligence in robotics. The subsumption architecture decomposes a system into independent parallel layers receiving input directly from sensors and performing output directly to control systems. Layers are prioritised and interact via mechanisms of suppression and inhibition. Example layers would be 'avoid', 'wander' and 'explore'.

Attempts were made to implement such an architecture using the functional language Haskell. It remains to be seen how expressive this abstraction can be in order to facilitate a subsumption architecture for reactive systems as only simple test programs have currently been written.

1. Brooks, R. A. (1991) , Intelligence without representation, `Artificial Intelligence'

2. Hudak Et. Al. (2002), Arrows, Robots and Functional Reactive Programming, Summer School on Advanced Functional Programming 2002

URSS

THE UNIVERSITY OF WARWICK