A Flexible and Innovative Platform for Autonomous Mobile Robots

Jessica Howe

January 10, 2003

1 Introduction

In building a new system for control and morphological design of autonomous mobile robots one must be certain that the system they are presenting is flexible, stable, easy to use, and meets the needs of the people that will be using that system. Over the past year, people at MIT have been designing a new platform for autonomous mobile robots that is designed to meet all of these goals. I hope to prove through the course of research proposed herein that these goals are indeed met.

Our system is made whole by two distinct components: a programming language called CREAL and a hardware platform that it may run upon. The programming language caters specifically to behavior based autonomous robots and there is great flexibility offered in the types and numbers of sensors and actuators that may be used with this system. I plan to test the limits of the hardware offered and expand the control system to facilitate the use of multiple control architectures in a user friendly way. In particular I will be creating an abstraction framework so that there is a simple way to implement various architectures within our platform.

2 Background

There are many decisions to be made when students, researchers, and hobbiests build autonomous mobile robots. One may build the entire thing from scratch, designing the control system and building the body themselves. Another option would be to buy a pre-built pre-programmed robot that requires

only minimal user intervention to 'build'. There are, of course advantages and disadvantages to each of these options. In building the whole thing yourself you have total flexibility in design and get very familiar with the hardware that is in use, but it takes much longer to implement and could leave the builder with a not-quite-working machine if they don't know exactly what they're doing. A pre-built machine will take almost no time to set up and is almost guaranteed to work, but the user is limited to the parts that the manufacturer provided.

Most robot builders fall somewhere in between those two extremes, using either a pre-built platform that requires only control programming or using pre-built control hardware and parts but designing both the body of the robot and the control system model themselves.

I will briefly describe the common choices that are made in control systems and body design, as well as the types of pre-built platforms that are currently available.

Note that in this paper the phrase "robot" will refer strictly to autonomous mobile robots, not remotely operated or pre-programmed robots which are commonly found in factories. Additionally I will be limiting my discussion to robots that are small enough to lift and excluding things such as autonomous vehicles.

2.1 Control Architectures

An autonomous mobile robot must sense its immediate environment and act upon those senses to interact with and manipulate its surroundings. The situatedness of a robot within its environment implies both that the machine may immediately sense its surroundings, and that its actions have immediate consequences as to how the sensing of its environment changes. It is key for an autonomous machine to be able to react quickly and appropriately to these changing situations.

There are a handful of popular design architectures that are used to control the 'brains' of these robots. These techniques may be implemented in various sorts of programming and hardware domains, but the central design structure remains the same. I will briefly review the architectures and methods that are commonly used to control autonomous mobile robots.

- **Subsumption**

Subsumption architecture is a control system that is based upon embodiment, situatedness and direct sensing of the environment and immediate action in response [5]. Various sensory inputs will trigger response signals which are prioritized through a system of inhibition and suppression. Based on the current input signals and the signals actively propogating through the system, appropriate response signals are generated. State may be held in subsumption modules through the looping of signals through the system. This architecture has been used with many types of robot morphologies, including walking 'insects' [3], map-constructing wheeled robots [13], and wheeled vacuum cleaners [9].

This architecture may be implemented in many languages, but is best implemented in a language specifically designed for it, such as Behavior Language [4]. Languages such as this can make use of real-time processing of signals. Programming this architecture in higher level procedural languages which favor variables, function calls and require state to be kept, on the other hand, will lead to delays and improper reactions due to the simple fact that these languages require a large overhead to run on any system.

- **Neural Nets**

  Neural nets are used by many roboticists to employ a similar sense-and-react model. Like subsumption, neural nets take sensory inputs and use these to produce immediate reactions within the environment. The main difference between these two architectures is that rather than giving preference to one signal via suppression or inhibition, as in subsumption, neural nets take a combination of all input signals, with some weighted differently than others, to produce the final output action. There is also no state within a neural net architecture, for the input signals are immediately used to calculate the output signals. As in subsumption it is difficult to configure a neural net to produce a particular desired output action, and it gets more difficult with more complicated desired output.

- **Other Architectures**

  In the 1980's Ronald Arkin developed Autonomous Robot Architecture (AuRA), which is based on both a hierarchical planner and a

schema theory based reactive controller [1]. This type of architecture has been implemented in robots that perform various tasks, including exploration and object retrieval, and multi-agent foraging with communication. This is not a strictly reactive system, such as subsumption and neural nets, so I will not go into detail in comparison to our platform.

- **Programming Language Hack**

  Often beginner roboticists will program their control system with a particular programming language model in mind, rather than a robot architecture in mind. For example, if one were to use C to program their robot, they most likely would create procedures that will create desired actions within the environment based upon some sensory inputs. When checking multiple sensors for the presence of some trigger inputs, one must use only the tools available and loop through to check if any of the trigger input requirements are currently met. This is both time consuming and makes designing systems that are complicated or that must compute quickly very difficult.

It is in the research proposed in this paper that our platform will be made to easily accommodate a few of the formal architectures listed above.

2.2 Morphologies

2.2.1 Locomotion

Mobile robots can take many forms. One defining characteristic is the form of locomotion used. The choices of morphology and function of a robot will greatly influence the type of control system that is appropriate and feasible to implement. The two most popular forms of locomotion used in mobile robots are described below.

- **Wheeled**

  Wheels are often the simplest form of locomotion to implement within a mobile robot. The bare minimum needed to produce motion is a mere one actuated wheel, two if direction control is desired as well. In almost any control platform, signaling an axis to turn at a constant rate is straightforward and not very difficult. Wheels are used in the vast

majority of mobile robots seen today, but in some situations wheels may not be the best choice. In terrain that is rough, uneven, slippery, or particularly nonuniform wheeled robots may find difficulty in successfully traversing. Out of simplicity of design most wheeled robots are often built with one actuator to control forward motion and one actuator to steer. This type of robot is not designed to move in any arbitrary direction from its current location. In many environments the turning radius required by wheeled robots of this sort make their use impractical, while the complexity of other types of wheeled robot that can in fact move in any direction and turn in place make their use impractical as well.

- **Legged**

  Constructing walking legged robots is not as simple as constructing wheeled robots that are able to locomote. There are many issues to be dealt with, including balance, proper gait and complexity, to name a few. On the other hand legged robots can perform many tasks that most common wheeled ones cannot, such as climbing stairs or moving over rough terrain, omnidirectional motion, and the ability to easily right themselves if found upside down.

2.2.2 Environment Manipulations

There are many ways in which a mobile robot may manipulate its surroundings. Manipulations may be done by physically changing the state of the surrounding environment, but also by doing something that will change the perception the robot has on its surroundings. The most straightforward way to change perception of the surrounding environment is to move the robot within it, changing various sensor readings in the process. From the robot's point of view it sees that the environment has changed.

A robot may also take a more pro-active role in changing its perception of the environment: by moving something that was stationary. This may be done in multiple ways, such as using the weight of the robot to push an object around, by using arms or other manipulators that are able to grab, pick up and move objects, or by signaling some other system to react as in the opening of supermarket doors when a person approaches. An example of such a situation when direct environment manipulation is needed is the action of kicking a ball as done by soccer playing robots in RoboCup [10].

2.2.3 Sensing

There are multiple types of sensors that are commonly used in autonomous mobile robots, from bump, light, and IR sensors, to accelerometers and inclinometers. The choice of which sensors to use on a robot is completely up to the designer and depends on what type of task this robot is to perform. But in practically all robots that are built, some sort of sensing is performed. Without this key element there is no form of feedback that is given to the robot and no way for it to interpret its surroundings. The types of sensors that a robot contains does not usually categorize the robot, as does the style of locomotion, but it is key to note that sensors play a crucial role in the performance and abilities of robots.

2.3 Commercial Platforms

There are many commercial platforms available which may be used to build autonomous mobile robots, some more complicated than others. I describe below the most commonly found commercial platforms which the functionality and scope of our platform may be compared.

- **Lego Mindstorm**

  The Lego Mindstorm is an inexpensive platform that is very popular with beginning roboticists [7]. It provides a simple graphical programming interface and facilitates the interaction of sensors, actuators, and control system. The tradeoff for simplicity is a lack of flexibility in the types and number of peripherals that may be used and the types of computation that may be performed. One may adapt many types of peripherals to work with the system but there are limitations on the range of peripherals that the Mindstorm will support.

- **Handy Board / Cricket**

  The Handy Board [11] and Cricket [12] are control systems designed at the MIT Media Lab that allow more flexibility than the Mindstorm but are similar in many ways. The Cricket is a small device that allows only a couple of sensors and actuators to be connected, but is also fairly inexpensive and is very easily programmed. The Handy Board offers more flexibility in the quantity of peripherals that may be used and the computation power available, but is also more expensive than

either the Cricket or the Mindstorm. Nonetheless it is often used by university student who would like more support and flexibility than are offered with the other two mentioned systems, which I might add have children as their main audience.

- **Kephera**

  The Kephera is a small pre-built robot that is often used by researchers [15]. It is a fully functioning programmable mobile robot that has room for attachment of peripherals but contain wheels that are used as its primary source of locomotion. It is flexible in the types of peripherals and computation it supports but its size limits the types of environments it may be used in and the sizes of peripherals that may be attached. It is usually both too expensive and too complex for students to make beginner usage practical.

3 Control Platform

3.1 CREAL

I will be exploring and expanding the range of functions and capabilities of CREAL, a programming language designed by Rodney Brooks [6]. CREAL, or CREAture Language, is designed to support large numbers of modules executing in parallel and can be run with very little overhead. In particular, this language was written to provide control for autonomous mobile robots. It supports multiple threads of concurrent computation and is capable of controlling all parts of a robot in as close to real time as we can get.

3.2 Hardware

The hardware that will be used to control my robot consists of a group of inter-meshing task-specific boards designed by Edsinger et al [8]. The central control of the 'stack' of boards is a Rabbit 2300 manufactured by Rabbit Semiconductor [14] which will run CREAL. The Rabbit will also be responsible for interacting with the various other task boards to retrieve sensor readings and send actuation commands.

4 Morphology Implementations

I intend on first building multiple control systems on one test robot. These control systems will test everything from the capabilities and limitations of CREAL and the ease of use, to the ability of the stack to support many various types of sensors and actuators. These tests will be the first set of thorough runs on these platforms and will be used both as a formal presentation of the development platform that has been produced and as a beta test to make sure that these products are capable of performing as we expect them to.

I will be using a two-wheeled robot with two passive castors to conduct the range of my experiments. Upon this base I will attach all other components that are to be used. Below I will describe the types of experiments that I will perform and the types of control systems that will be built to test our claims about the platform.

The first stage will be a setup phase in which I configure the system to support all of the various sensors and actuators that will be used in later tests. This will consist of writing drivers for each type of peripheral device so that they may be controlled by a CREAL program running on the stack. I hope to individually cover as many varieties of sensors and actuators as possible and find the strengths and weaknesses of each.

I also plan on testing the limitations that CREAL and the stack place on using many sensors at once. I will try to add many sensors of the same type in such a way that task performance of the robot should improve with the addition of sensors. In theory this idea should not hold after some crucial point in which the limitations of the system are pushed. These limitations could possibly include a hard cap on the number of possible sensors that the system will recognize, a turning point in which the accuracy of sensors is compromised, or a divide after which the computation speed of the Rabbit is slowed down below a critical real-time pace.

Similarly I wish to test the limits of the system in terms of actuation capabilities. I will perform the same two types of tests as with sensors: testing the support capabilities that our system has for multiple types of actuators, and to test the limitations of the system. The limitations that may be reached could include a hard cap in the number of distinct actuators that may be controlled, a critical level where power drawn is too extreme for the system to support, or a level of control needed to monitor all actuators such that real-time capabilities and correctness are compromised. I am only implementing

these tests on one wheeled robot, but tests such as this could show that the actuation requirements needed in a legged robot, both in quantity and strength of actuators, could be fulfilled.

In addition I plan on testing a mix of these two through experiments described later in this paper: to run a system in which there are multiple types of sensing and actuation that run concurrently in order to perform a task. In this respect I hope to test if there are any problems with the way that sensing and acting interact with one another within the system, either by timing differences, issues of accuracy, or bounds that the platform may place upon combined use.

In total these experiments will be done to test the flexibility of this system and to prove or disprove the claims of being truly multipurpose. Whichever limitations exist within the platform I hope to find them and document them. Seeing that this platform is still evolving and developing these tests could lead to a broader and more efficient system that truly will live up to the goals that we place upon it.

5 Creating Architecture Compatibility

Throughout testing I hope to show the flexibility that our system provides in designing multiple types of autonomous mobile robots. Part of this flexibility is the ability to implement particular types of control models that accurately match the needs of the planned system. As noted earlier, there are architectural models that are commonly used in systems today, and in order for our platform to be truly flexible it must be shown that these models can correctly and efficiently be implemented.

In addition, the user must be able to program such a system easily and efficiently. The trouble with using CREAL to program a neural net based control system, for example, is that even though the theoretical design of the system may be straightforward the code required to build it is often not. This same reasoning holds with all of the control architectures described in section 2. I plan on building functionality into our system that will give users a clean interface in which they can easily and correctly specify the details of these various architectural designs and have them integrated into a CREAL program. This entails creating abstraction layers that enable the user to cleanly specify the details of their theoretical design and have it correctly compiled into messy CREAL code.

This abstraction will be produced through a pairing of a specified format in which the user will provide the data, and macros that will take this formatted data and convert them into logically equivalent design models within CREAL. This may be tricky. There are issues that must be dealt with in regards to the specifics of each architecture. For example, is it reasonable to assume that the user will know exact numbers that will be sufficient to specify a neural net that will perform the desired task? Or will forms of net learning such as back propagation need to be put in place in order for the designed structure to accurately perform as desired? Similarly, how will users know the exact values of constants that will be needed within a subsumption architecture, such as timing and threshold values, in order to satisfy a particular goal? Is it valid to just provide this service and expect the user to find the correct values themselves through trial and error? These are questions I hope to have solid answers for.

6 Task Performance Tests

In order to test these modules I will simply try to design systems by hand that will perform a task and compare this performance to what is output by a system of the type described above that takes formatted data and macro expands it into correct CREAL code.

In each case I will first theoretically model how the architecture could solve the given task. I will then adapt this design to be directly performed in CREAL. In the adaptation I will note the level of difficulty that is required to encode by hand such a model in CREAL and the ease of use in doing so from a user standpoint. I will also note any advantages that are seen in using CREAL to directly implement this design and any difficulties that arise in the process.

The next stage of experiments will involve performing the same adaptation from theoretical model to CREAL code, but using the newly designed abstraction tools rather than writing all of the code by hand. These experiments will test the ability of CREAL to embrace various architectures and still produce a desired output. I will also be noting the difficulty and pain induced in designing and creating a functioning system through this method as opposed to by hand.

The tasks that will be performed in these experiments will all require multiple sensor and actuator types and will be implemented upon the same

wheeled chassis that is to be used for the morphology experiments. I outline below three experiments that are to be performed, both in desired task and the peripherals that will be needed to accomplish such a goal. These tasks will each be performed by three separate control systems: one written by hand in CREAL, one using subsumption, and one using neural nets.

- **Exploration and Warning System**

  An example of a functioning subsumption based robot is given in [2] which uses three subsumption layers to explore and navigate through an area. These three layers, from simplest to more complex, include the Avoid-Objects Layer, the Explore Layer, and the Back-out-of-Tight Situations Layer. I will use this framework and subsumption design to implement an exploratory robot that successfully avoids objects and can free itself from unpromising locations. Additionally, my robot will include a layer which will control an audio warning system that responds to high levels of heat. We can think of this as a mobile fire detector.

  This same task will be performed using Neural Nets and written directly in CREAL by hand, but will be performing the same task.

  This robot will be equipped with IR proximity sensors, bump sensors, heat sensors and an alarm. It may also be equipped with a system that is capable of transmitting a warning to a central command center to notify of such an alarm.

- **Object Finding Arm**

  A task that is often desired in robots is the ability for an arm to successfully find and manipulate an object in its surrounding environment. I plan on creating an arm that will attach to our mobile base that will be able to detect when an object is positioned directly under it. The arm will have a 'hand' on the end of it that is capable of recognizing when the desired object is in position and closing upon the object. It will have the ability of knowing when the object was successfully seized and lifted, and if unsuccessful will repeat its quest.

  This robot will be based on the same wheeled chassis but will not be very mobile. It will move the base to search with the sensors on the arm but only within a small range of motion. The sensors on the arm will most likely be light sensors that will notice the desired object based

on its color with respect to the surrounding environment. There will be an LED and a photo detector mounted on the hand to detect when an object is within grasping range, and actuators that will allow the hand to close upon the object.

- **Maze Navigation**

  The third task to be undertaken will be the creation of design systems that will allow a robot to successfully navigate through a maze. Rather than attempting to build a system that will be able to navigate through all mazes, they will be trained on and attempting to all solve the same maze. The maze will be of grid dimensions so that it will be possible to keep state that tells of its assumed location.

  The robot will be equipped with proximity and bump sensors to allow wall following within the fixed-width halls of the maze, and will also allow detection of corners and turns that could be made. There will also be a system put in place to allow the robot to recognize when it is at the end of the maze. This will either be through its assumed location and the known location of the end of the maze, or a marker such as an IR beacon or a light source that the robot will be able to detect upon arrival at the goal location.

7 Schedule

- [current - January 31] **Phase 1: Stack Completion**

  Much of this work has already been completed, but more time must be spent on building and checking that the hardware stack is performing properly. I will write drivers so that all of the sensors and actuators that will be used in the following experiments will be properly controlled through CREAL. A prototype robot will be used to test these various sensors and actuators.

- [February 1 - February 28] **Phase 2: Hand Write Performance Tests in CREAL**

  In this phase I will hand write CREAL code to perform each of the three outlined tasks. This will both get me familiar with the intricacies of CREAL and will help me see the problems that I may face when performing these tasks in alternate architectures.

- [March 1 - April 30] **Phase 3: Modify CREAL and Implement Performance Tests in Alternate Architectures**

  In this phase I will modify CREAL so that alternate architectures may be implemented, as outlined above. Two months will be given to this phase because this will contain most of the difficult research that is to be done.

- [May 1 - May 25] **Phase 4: Final Writing Phase**

  A final month will be given for writing and presentation of data collected over the previous months.

References

[1] Ronald C. Arkin and Tucker R. Balch. Aura: principles and practice in review. *JETAI*, 9(2-3):175–189, 1997.

[2] Rodney Brooks. A hardware retargetable distributed layered architecture for mobile robot control. In *IEEE Transactions on Robotics and Automation*, pages 106 – 110, May 1986.

[3] Rodney A. Brooks. A robot that walks; emergent behaviors from a carefully evolved network. *Nerual Computation 1:2*, pages 253–262, 1989.

[4] Rodney A. Brooks. The behavior language: Users guide. Technical Report 1227, MIT Artificial Intelligence Laboratory, April 1990.

[5] Rodney A. Brooks. Intelligence without reason. In John Myopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595, Sydney, Australia, 1991. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[6] Rodney A. Brooks. Creature language. Technical report, MIT Artificial Intelligence Laboratory, November 2002.

[7] Mario Ferrari. *Building Robots with Lego Mindstorms: The Ultimate Tool for Mindstorm Maniacs*. Syngress Media Inc., 2001.

[8] Living Machines Group. Alive stack user manual. Technical report, MIT Artificial Intelligence Laboratory, November 2002.

[9] iRobot Corporation. *Roomba Intelligent FloorVac Owner's Manual*. http://www.roombavac.com, 2002.

[10] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The robot world cup initiative. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340–347, New York, 5–8, 1997. ACM Press.

[11] Fred G. Martin. *The Handy Board Technical Reference*. http://handyboard.com/techdocs/, November 2000.

[12] Fred G. Martin. *About Your Handy Cricket*. http://handyboard.com/cricket/docs/, July 2002.

[13] M. Mataric. Integration of representation into goal-driven behavior-based robots. In *IEEE Transactions on Robotics and Automation*, volume 8 No. 3, pages 304 – 312, June 1989.

[14] Rabbit Semiconductor. *RabbitCore RCM2300 User's Manual*. http://www.rabbitsemiconductor.com/docs/, 2001.

[15] K Team. *Kephera II user manual*. http://www.k-team.com/download/kephera.html, March 2002.