A Tangible Architecture for Creating Modular, Subsumption-Based Robot Control Systems

Tim Gorton and Bakhtiar Mikhak
MIT Media Laboratory, 20 Ames Street E15-354, Cambridge MA 02139 USA
{tgorton, mikhak} @ media.mit.edu

ABSTRACT
We present a new modular, reconfigurable architecture for building and interacting with subsumption-based robot controllers. A set of modules – with embedded sensing, communication and processing capabilities – divide up the subsumption architecture into self-contained behavioral layers that subsume each other according to the way the modules are physically stacked. On-board indicators of the internal state of each module help in programming and understanding the behavior of the robot in real-time. This offers novices, in particular children, a new environment to learn about a powerful metaphor for programming robots and other interactive systems in a hands-on and exploratory manner. We will also discuss a prototype implementation of this architecture and the results of a preliminary user study using the prototype with a group of children.

Author Keywords
Subsumption architecture, tangible programming, modular robot controllers, interactive system for children, toy

ACM Classification Keywords
C.3: Special Purpose and Application-based System, I.2.9. Artificial Intelligence: Robotics. K.3. Computers and Education.

INTRODUCTION AND OVERVIEW
Subsumption architectures [1] have proved to be a powerful means of organizing simple computational elements in order to build complex functionality in a robot controller. In this software architecture, behavioral layers selectively override the outputs of lower levels to control a set of mechanical actuators. Designing and constructing robots is a popular hobby activity for adults and children, but the controllers built for these activities are generally either pre-designed electronics or built with an embedded controller such as the LEGO Mindstorms™ RCX brick. Alternate representations are needed in order to leverage the benefits of subsumption architecture in controllers built by children and novice hobbyists. This paper presents a new tangible, modular, reconfigurable architecture for addressing this need.

CHI 2004, April 24–29, 2004, Vienna, Austria.
ACM 1-58113-703-6/04/0004.

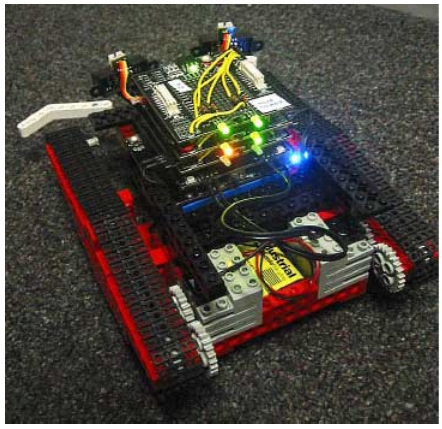

Figure 1: A prototype Implementation of a subsumption-based robot controller

In order for a novice to understand the operation of a subsumption-based controller, the user must be able to construct new controllers in order to explore their behavior. However, the common representation of a subsumption architecture controller, a network of augmented finite state machines, is difficult to construct or manipulate in practice. We explore an alternate architecture obtained by dividing the subsumption architecture into behavioral layers – each represented as an intuitive tangible module with computing, sensing, communication, and display capabilities on-board for encapsulating the layer's functionality – that subsume each other. These layers can be then added, removed, and reordered easily to build different controllers.

Transparency is a crucial factor in the design of rich programming environments for novices. [5] For a modular robot controller, this implies that the operation of each layer must be easily understandable in isolation and in a full controller. Also, the link between a layer's sensors and its outputs must be simple enough to be comprehensible while powerful for implementing sophisticated robot controllers. The relationship between the layers must also be clear, as well as how particular observed behaviors emerge from the interactions between layers. Our approach is to connect directly the modules used in constructing the controller with their outputs using an arrangement which makes it clear how the layers' outputs lead to the observed behavior.

RELATED WORK
The Mobile Robot Group at MIT AI Lab headed by Rodney Brooks both invented the subsumption architecture [1] and

has demonstrated its success in creating robust robot control systems for a variety of robot types. [2] Though we have chosen a different means of implementation in our work, our approach still aims to take advantage of the robustness and incremental development described by Brooks.

This work is inspired by Tim McNerney's work on tangible programming [5] and an earlier implementation (by one of the authors) of the robot controller presented in this paper using McNerney's tangible programming blocks. In that implementation, the order of the behavior blocks was read by a base unit, which contained all of the logic and sensors for the robot controller. It was not fully modular and lacked the transparency and extensibility of the current work.

Peat Wyeth and Helen Purchase have reported success in building tangible programming elements appropriate for children less than eight years old, a target audience for this work. [8] Patten, Griffith, and Ishii have also explored other means of using tangible interfaces to program a robot controller, linking events to actions using physical strings. [7] The logic created through these connections was then downloaded onto a Lego Mindstorms™ brick, preventing real-time analysis of the robot controller.

Lund and Pagliarini have developed an environment for 7 to 14-year-olds to build autonomous robot controllers for a "RoboCup Junior" robotic soccer tournament. [3] Their on-screen design environment allowed users to select from and order high-levels behaviors such as "find the ball" and "circle around the ball." Lund and Pagliarini report that children were able to build successful controllers within 60 minutes using these behaviors as building blocks. Like Patten's work, the controller logic was downloaded onto a Lego Mindstorms™ brick for the competition.

ARCHITECTURE OVERVIEW

The architecture that we have created to address the design concerns described in the introduction consists of a base unit and a set of layers that can be stacked atop the base in any order. Each layer implements some simple behavioral primitive and contains or is attached to the appropriate sensors needed for this behavior. (Figure 2) The vertical ordering of the layers determines which layers inhibit which other layers; in particular, layers may inhibit the outputs of layers above them. For example, in the arrangement shown in Figure 2, the "Seek light" layer could inhibit the outputs of the "Move randomly" layer.

To allow arbitrary orderings of these layers, we have imposed constraints on the general form of subsumption architectures. Specifically, each layer must use the same set of outputs with a common set of tokens. (See Figure 3) Because the robot's actuators are located on the base unit, we have also reversed the traditional ordering of the layers, so lower layers override the outputs of those above them.

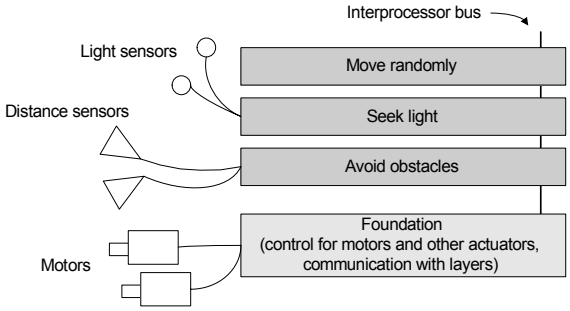

Figure 2: Architecture Concept

Each output must also have a default, in case none of the layers outputs a token for a particular output. (For example, see output 3 in Figure 3.) In the current version of our architecture, layers are unaware of the outputs of higher layers that are being inhibited. We will discuss the effects of this and other limitations to the subsumption architecture more fully elsewhere.

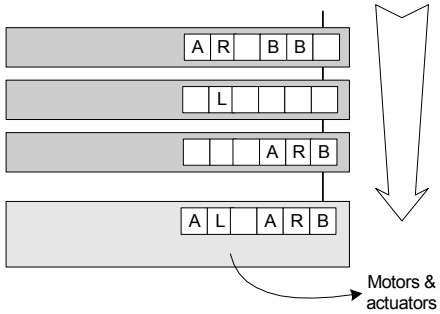

Figure 3: Subsumption Mechanism

PROTOTYPE DESIGN

The prototype for this system, shown in Figure 1, is a controller for a simple tracked LEGO robot chassis. The robot chassis has two motors, one for each track, and two touch sensors attached to bumpers in front of the robot. Figure 4 illustrates the possible motions of this robot. Each of the three illustrated motions can be reversed in order for the robot to move backwards or turn in the opposite direction.

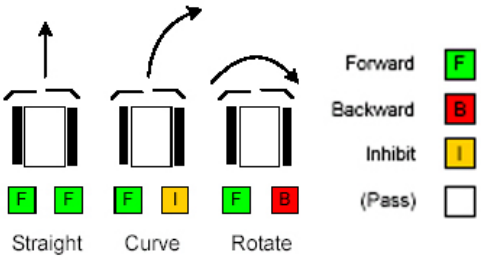

Figure 4: Prototype Robot Movements and Tokens

The prototype controller built for this robot (Figure 5) uses the four tokens shown in Figure 4: Forward, Backward, Inhibit, and Pass. The difference between the Inhibit and Pass tokens is that the Inhibit token indicates that the layer wishes to stop the motor, while Pass indicates that the layer

has no preference. The system has only two outputs, one for each motor, which are made visible on each layer by two small lights called LED's. Each LED can be lit red, green or amber, corresponding to the tokens above.

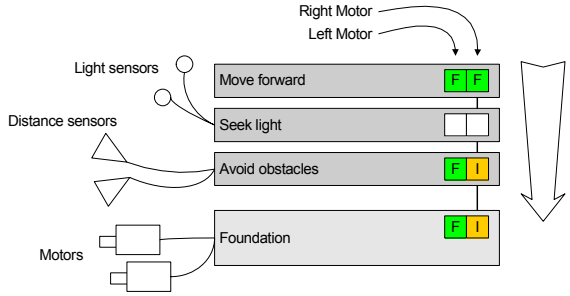

Figure 5: Prototype Controller

The base unit and layers were built using components of the Tower System. [4] Each subsumption layer is implemented as a circuit board with the necessary sensors and LED's attached. We have built layers for moving forward, moving randomly, seeking light, avoiding obstacles, and recoiling when colliding with an obstacle. (Figure 6) These layers can be reconfigured arbitrarily and follow the subsumption rules described in the previous section.

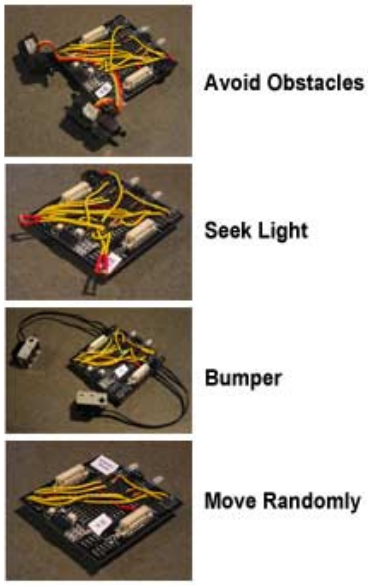

Figure 6: Prototype Behavior Layers

Building Complex Behaviors

The most interesting behaviors occur when a few layers are combined. For example, consider the arrangement shown in Figure 7. The robot moves forward, unless it sees a light or an obstacle. Avoiding an obstacle will inhibit the movement towards a light source. This leads to a behavior where the robot repeatedly turns towards the flashlight just enough to see the box underneath it and then turns away. Finally the robot no longer sees the light after turning away from the box and heads away in a straight line. (Figure 8)

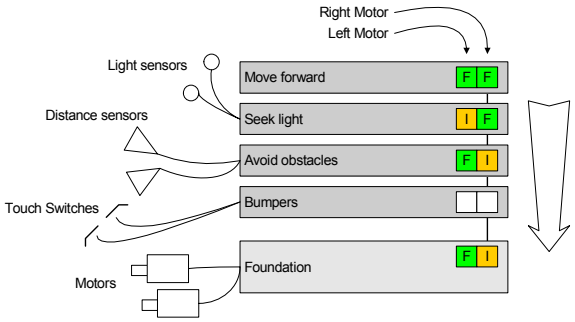

Figure 7: A complex controller

Figure 8: Observed movement of a robot using the controller shown in Figure 7

However, when the order of the Avoid Obstacles and Seek Light layers are reversed (Figure 9) the desire to seek light inhibits the higher layer's desire to turn away from the box. This results in a movement, in which the robot avoids obstacles until it sees the flashlight, which it moves towards until it finally touches the box and turns away. (Figure 10)

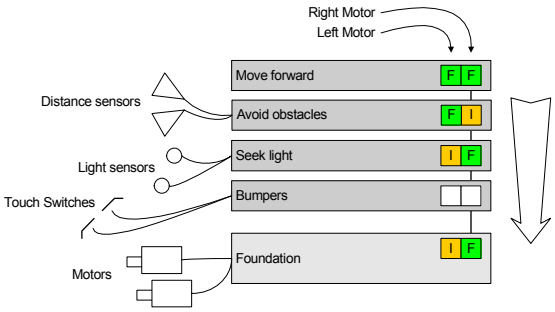

Figure 9: Another complex controller

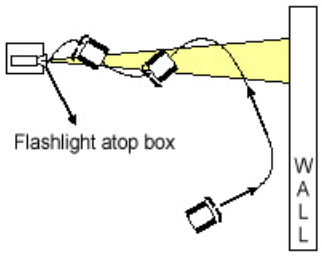

Figure 10: Observed movement for a robot using the controller shown in Figure 9

The behaviors observed by the controllers described in this section make it clear that it is possible to create interesting, complex behaviors by combining primitive behaviors in layers such as those created for this prototype. It is also clear that rearrangements of these layers can yield new behaviors. However, using a number of these layers makes it more difficult to examine the layers' interactions by observing the LED's on the controller.

PRELIMINARY EVALUATIONS AND FUTURE WORK

The preliminary user testing was performed at the Computer Clubhouse at the Boys and Girls Club in Chelsea, Massachusetts. A group of fifteen children between the ages of 8 and 11 interacted with our prototype robot over a forty-five minute period. We first let the robot roam with the *Bumper* and *Avoid Obstacles* layers, and a number of the children quickly became interested in the robot's movements at their feet, and began trying to characterize its behavior, listing the individual behaviors of avoiding and recoiling from obstacles. We demonstrated to them how to change the robot's functionality by removing and adding layers with a few children and they then began to examine the different functionalities. They were able to describe how the different behaviors they saw were contained in different layers, noting both that a behavior stopped after they removed the associated layer. They explained this by simply pointing to, for example, the distance sensors attached to the Avoid Obstacles layer and saying they were no longer on the robot. Though they were able to create and understand complex behavior like the ones discussed in the last section, they had difficulty with understanding the interactions of the layers and their combined effect on the observed behavior. Further study is required to see if this will prove to be important in their understanding of more general behavior. We suspect that it does and this issue needs to be investigated and addressed more fully.

The architecture described in this paper places several severe constraints on how a layered, subsumption-based controller may be realized in physically distinct layers. Though the prototype controller demonstrated that an engaging set of behaviors could be built despite these constraints, the effects of these limitations on the types of controllers that can be built is certainly worth discussion and investigation. A natural next step in this research is addressing the following three limitations to a full-feature implementation of subsumption architecture. (1) Outputs of each controller are limited to a set of global tokens only. (2) The layers are unaware of the values that they may choose to inhibit, since no communication occurs directly between layers. (3) A single hardware stack requires a universal set of outputs and prevents us from exploring scenarios with two modules on the same "level" as each other. Addressing these limitation will likely require extensions to the communication protocol between layers and our on-board means for visualizing the internal state of each layer.

CONCLUSION

In this paper, taking advantage of modularity and means of controlling complexity in the subsumption architecture, we have presented a new tangible, modular, reconfigurable architecture for building robot controllers that is more accessible to novices and is particularly more engaging for children. Conceptually and physically, this new architecture splits a subsumption-based control system into behavioral layers, each of which is implemented on a microcontroller in a stack of circuit boards.

The prototype implementation using the Tower system demonstrates that the architecture is both conceptually and technically promising. We reported our findings from a preliminary study with a group of children as young as 8 years old. While finding that children in our group partially succeeded in deconstructing the robot's observed behavior into outputs of individual layers, our study indicated that understanding the interactions of the layers and their combined effect on the observed behavior remains difficult. Growing out of this development and evaluation effort, we discussed three fruitful directions for further research: developing and testing new implementations, improving the transparency of these implementations, and extending the means for novices to expand the existing set of controller.

REFERENCES

1. Brooks, R. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1 (1986), 14–23.
2. Brooks, R., Flynn, A. Fast, Cheap, and Out of Control: A Robot Invasion of the Solar System. *Journal of the British Interplanetary Society*, (1989), 478–485.
3. Lund, H., Pagliarini, L. Edutainment Robotics: Applying Modern AI Techniques. *Proceedings of International Conference on Autonomous Minirobots for Research and Edutainment* (AMIRE-2001).
4. Lyon, C. Encouraging Innovation by Engineering the Learning Curve. Master of Engineering Thesis, Massachusetts Institute of Technology, Cambridge, MA. (2003).
5. McNerney, T. Tangible programming bricks: An approach to making programming accessible to everyone. Masters Thesis, Media Lab, Massachusetts Institute of Technology, Cambridge, MA. (1999).
6. Papert, S. Mindstorms: Children, computers and powerful ideas. New York: Basic Books, 1980.
7. Patten, J., Griffith, L., and Ishii, H., A Tangible Interface for Controlling Robotic Toys. In *Summary of Conference on Human Factors in Computing Systems (CHI2000)*, ACM Press (2000) 277-278.
8. Wyeth, P.A. and Purchase, H.C. Tangible Programming Elements for Young Children, *Proceedings of the CHI conference*, ACM (2002), 774-755.