The Robot Builder

Volume Twelve Number Two February 2000

Notices

· Practical Mobile Robotics Class - 11:00AM - 12:00PM

· Business Meeting - 12:30 - 1:00

· General Meeting - 1:00 - 3:00

Distribution

If you would like to receive The Robot Builder via e-mail, contact the editor at:

apendragn@earthlink.net

Inside this Issue

Animated Artificial Fishes

By Arthur Ed LeBouthillier

Researchers at the Computer Science Department of the University of Toronto have been developing artificial fish which operate in a simulated world; they are seeking to create life-like artificial animals. Much of what they learn can be applied to real-world robots.

Simulated World

The simulated world is a product of a photorealistic 3-D graphics generation program which models many features of a real fish's marine habitat. There are simulated plants, simulated plankton, and other marine features such as currents. Many physical forces are modeled within this simulated world allowing realistic models of interaction between simulated fish and their environment. For example, hydrodynamic forces are modeled allowing the forces due to water currents to be modeled. Additionally, within this environment are many simulated fishes. These fishes are highly detailed models which include their appearance, their movement capabilities and their sensation and behavior capabilities. This approach of simulating an animal is called the "animat" approach. An animat is a simulated artificial animated animal.

Artificial Fishes

As the authors states in their report [1], "Artificial fishes may be viewed as animats of unprecedented sophistication. They are autonomous virtual robots situated in a continuously dynamic 3D virtual world." Their functional design, including motor control, perceptual modeling, and behavioral simulation presents hurdles paralleling those encountered in building physical autonomous agents." Many aspects of a fish's appearance, perception, behavior, habits, learning and motor control are intimately modeled.

The appearance of a simulated fish in this simulated environment is modeled to a high degree. The shape of the fish is first encoded as a 3D model; actual images of fish are texture-mapped on this model so that it appears like whichever fish is being modeled. Underlying this shape model is a detailed dynamic structural model which models individual muscles and the interaction of body parts with water. When a fin moves, the individual forces it generates due to interaction with water are dynamically modeled.

Motor Control

Three motor controllers are implemented in software to generate the fish's motions. The Swim-MC (swim motor controller) generates sequences of muscle activity that operate the dynamic body model to generate the forces necessary for forward swimming motion. Left-Turn-MC and Right-Turn-MC motor controllers generate muscle patterns that produce turning behaviors. Although these motor controllers have been hand-coded, learning algorithms have been applied allowing the fish to learn how to swim in improved ways. By developing evaluation functions which determine the swimming

See ***Fish***, Page 2

Fish from Page 1

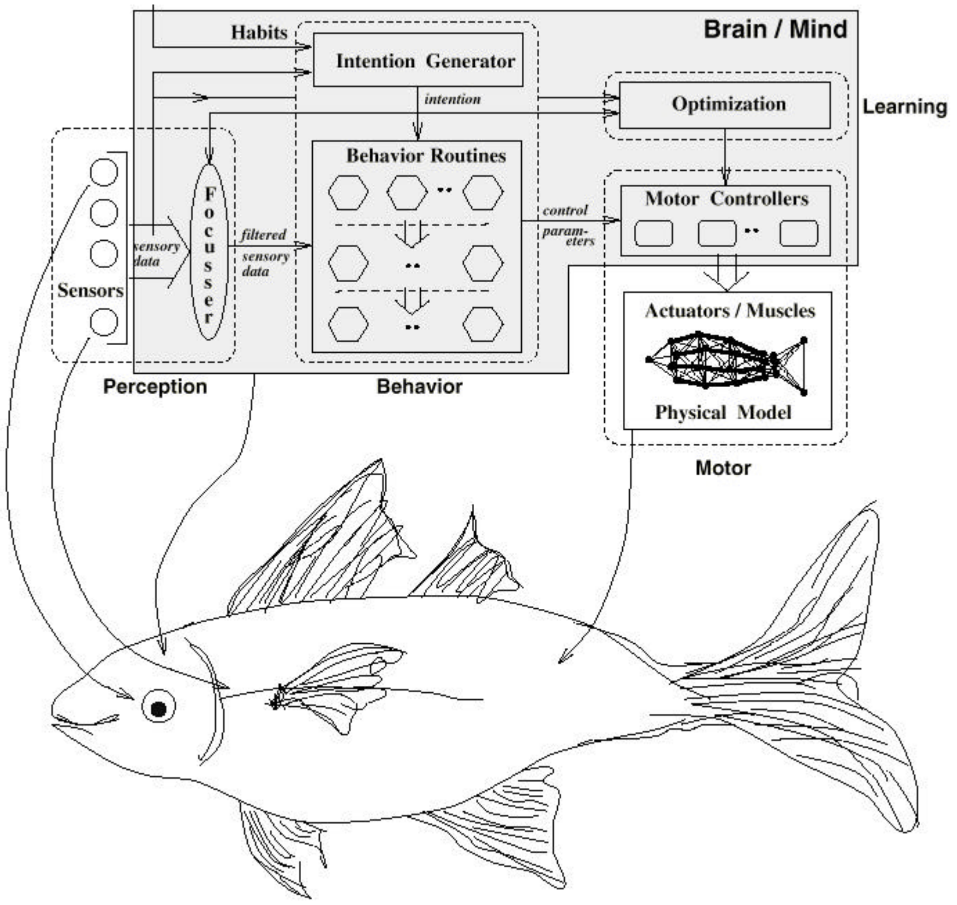

Figure 1 - Anatomy of an artificial fish

speed obtained by command signals, parameters specifying movement patterns are optimized to be efficient and fast.

Perception Models

Many aspects of a fish's visual perception are modeled. Occlusions are modeled so that fish are not able to see behind solid objects. Each fish has a limited field of vision which is consistent with its real-life capabilities. The range at which objects are visible is consistent with the clarity of the water. Additionally, an ability to characterize an object based on its unique pattern of colors has been developed allowing the artificial fishes to identify food, mates, and predators. Each artificial fish has binocular vision and is able to use this capability to determine the range to various objects. Using stereoscopic and image-flow measurements, the fish is able to focus on objects and stabilize its vision as it moves. A perceptual filter allows sensory information which is not vital to immediate behavioral needs to be reduced. Because of these features, the vision system provides sufficient information to locate, attend to and track various features in its environment.

Behavioral Model

The behavioral model attempts to create realistic interactions between the artificial fish with its environment by mediating between the perceptual system and motor system. It consists of an intention generator which takes into consideration the fish's habits, mental state, and incoming sensory information to produce an intention. There are eight basic behaviors which can be intended: avoiding-static-obstacle, avoiding-fish, eating-food, mating, leaving, wandering, escaping,and schooling.

The innate behavior of a fish is dependent upon which type of fish it is and is defined by establishing a number of "habits." A habit is a weighting parameter pattern which specifies how sensory

See *Animat*, Page 3

Animat from Page 2

stimuli relate to three state variables representing the fish's "emotional" state: hunger, libido and fear. Different fish have different habit parameters which are enduring qualities.

The intention system first checks for stimuli that might elicit fear and adjusts the fear state variable appropriately. If fear is below a threshold, then the hunger and libido state variables are computed. If the greater of these two is above a threshold, then either eat or mate behaviors might be initiated. If none of the three state variables is above their thresholds, then a wander behavior is initiated. After a behavior is intended, the perceptual focus system is tuned to support activated behaviors. Behavior routines use the focussed perceptual information to select the proper motor controller activity (Swim-MC, Left-Turn-MC, or Right-Turn-MC).

Results

Complex habitat systems composed of predators and prey, male and females have been created. Using the behavior system described above, prey fish feed, school, wander and mate while predators search for them. Simulating this artificial underwater environment has allowed researchers to develop and test computational models of life-like simulated animals. Complex visual algorithms allowing sophisticated vision-based behaviors have been developed. Robotics researchers can learn from these techniques and validate certain theories about perceptual, behavior and motor control systems in a realistic underwater environment.

[1] Terzopoulos, Tu, and Grzeszczuk, "Artificial Fishes: Autonomous Locomotion, Perception, Behavior, and Learning in a Simulated Physical World." *Artificial Life*, **1**(4):327–351, 1994.

[2] Demetri Terzopoulos, Tamer F. Rabie. "Animat Vision: Active Vision in Artificial Animals." Videre, Volume 1, Number 1. The MIT Press

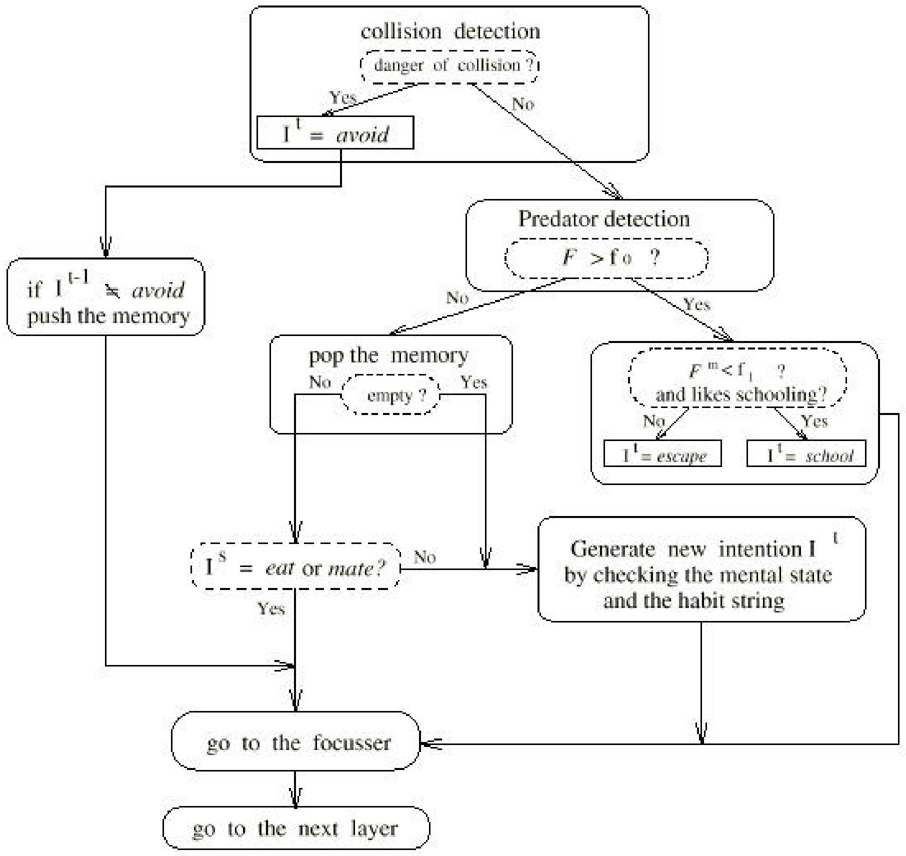

Figure 2 - an example of an intention generator

IBM's Zero Instruction Set Computer
By Arthur Ed LeBouthillier

Neural Nets excite many as a possible technique for developing adaptive capable robot controllers. Although the basic idea of their operation is simple and their use is efficient, the biggest problem has been in training them. A product developed by IBM called the Zero Instruction Set Computer (ZISC) provides hope that neural net technology might be reasonably used in robots.

Basic Neural Nets

Neural Nets are networks based on a simple model of biological neurons. They simulate the capability of neurons to take many inputs and to generate outputs based on simple math models. In these models, each neuron consists of a number of inputs (i1, i2, …, i*n*), going through a number of weights (w1, w2, … , w*n*) which are summed together and sent through a sigmoid function to generate an output. Figure 1 shows this model.

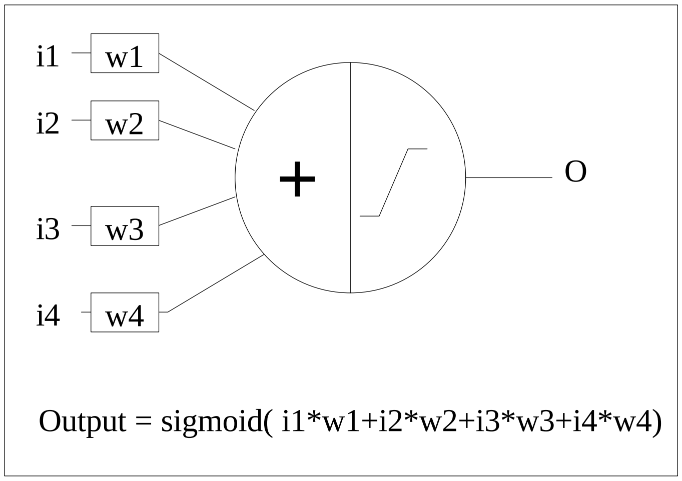

Figure 1 - a simple neuron model

As figure 1 shows, the output is a function of the sum of the inputs times a weight value. The sigmoid function is a non-linear function which creates a quick change in output once the sum of the inputs times their weights is above or below a threshold. Commonly, these neuron model elements are combined together to create a neural network as illustrated in figure 2.

Figure 2 shows a simple feedforward neural net. It is feedforward because all signals flow in the forward direction from input to output without any feedback signals. These kinds of neural nets are very adept at learning to recognize patterns in their input despite missing or noisy inputs.

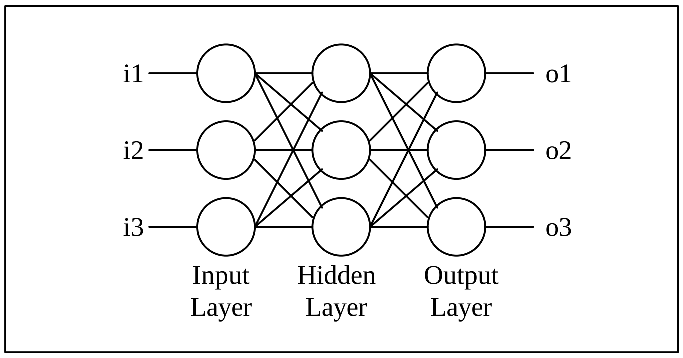

Figure 2 - a simple feedforward neural net

IBM's ZISC

IBM's Zero Instruction Set Computer (ZISC) is a hardware implementation of 36 neurons. It is intended for a wide variety of applications from image recognition to robotics control. It implements a simplified learning model based on the Radial Basis Function (RBF). This model of the RBF limits the total numbers of categories that a neuron can learn to 16,384 but larger networks can show increased learning capability. Although simplified, it still offers powerful capabilities As their literature states:

> ZISC036 is the first of a family of integrated circuit based on neural network designed for recognition and classification applications which generally require super-computers.
>
> ZISC products provide a very cost-effective way to solve such problems with sufficient performance to match real time constraints.

A ZISC036 implements 36 neurons on an integrated circuit which can be connected into any kind of network. Each neuron can have up to 64 inputs. The ZISC036 has a 16-bit data bus which allows loading of input parameters for each neuron. There are additional address and control lines to perform full input and output communications.

See *ZISC*, Page 5

ZISC from Page 4

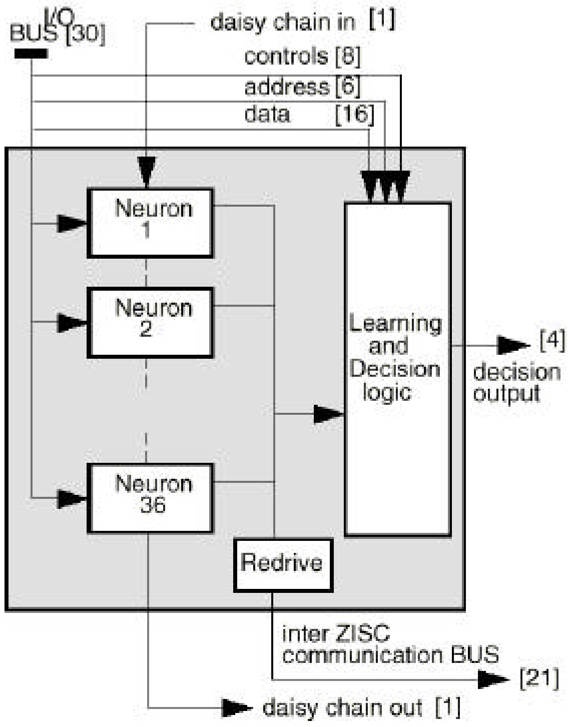

Figure 3 - Block diagram of IBM ZISC036

Using this bus access technique, data can be written to the input registers of the chip, which can become the input for all 36 neurons (or they can be grouped to respond to certain kinds of inputs individually). Once the last input value is entered, the chip computes the outputs for each neuron very quickly; with a 20 MHz clock, the result of the neural calculations is available 0.5 microseconds after the last input data is entered. This allows up to 250,000 evaluations per second (including the time to enter the input data). According to the manufacturer, similar capabilities could only be seen with a 2,000 MIPS Von Neuman architecture. The ZISC can also be trained with new patterns quickly: an individual neuron can be taught in less than a microsecond.

ZISC's can be cascaded into quite large parallel networks. This means that many individual neuron operations can be performed in parallel at full speed. Additionally, each neuron can be individually trained allowing the time-multiplexing of different networks. This allows a chain of ZISC's to be used like a quickly-reconfigurable time-shared neural network. The user has the choice whether to dedicate his neurons to special functions or to use them as general-purpose processing resources. Using this reconfigurability, a network can be a perceptron for a certain amount of time, become a sequencer circuit for another period of time and then go back to being a perceptron. Different operating modes allow access to parts of the neuron calculation. This can allow you to utilize the processing capabilities of the ZISC for different kinds of neural net algorithms.

Summary

IBM's ZISC036 is an exciting product which allows real-time processing of neural net applications. For those seeking fast neural computations, it might offer just the right tool for the job.

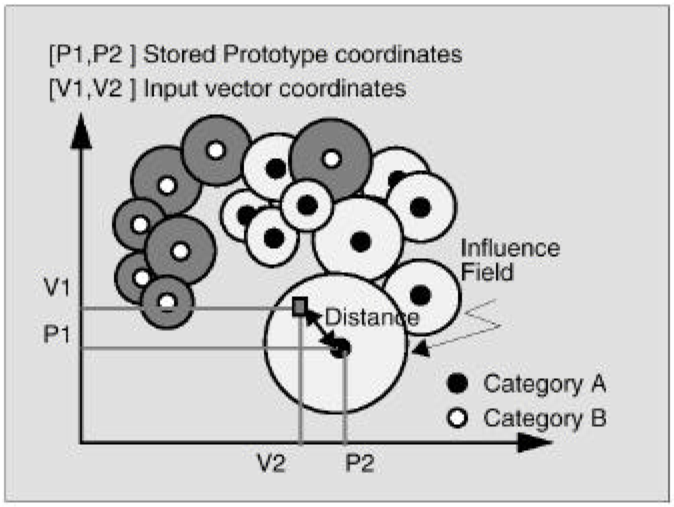
[P1,P2] Stored Prototype coordinates
[V1,V2] Input vector coordinates
V1
P1
V2
P2
Distance
Influence Field
● Category A
○ Category B

Figure 4 - RBF neurons compute the distance of the input vector from a prototype vector in up to 64-dimensional space. They can output a 14-bit category descriptor if the input is within a certain distance of the prototype.

Basic C Programming (Part 3)

By Arthur Ed LeBouthillier

The printf Function

Last month we introduced the printf function. printf is known as a formatted output function because it allows one to present data in specific formats. The general syntax of a printf function is:

```
printf( control_string, arg1, …, argn );
```

The first argument that the printf command takes is a control string. A string is a sequence of printable characters in between double quotes (“). The printf also can take any number of arguments after the control string.

The printf Control String

The control string allows you to place a sequence of printable characters, control sequence characters and conversion characters together for sending to the computer’s display. Let’s suppose you wanted the words ‘hello world’ to appear on your computer screen. To do this, your printf statement would look like this:

```
printf(“hello world”);
```

Were you to include this line in a program, and run it, the words ‘hello world’ would appear on the screen. Let’s assume you wanted to use two printf statements one after the other to print different strings on different lines:

```
printf(“Hello Dave,”);
printf(“I’m sorry I can’t do that.”);
```

The resulting output would look like this:

```
Hello Dave,I’m sorry I can’t do that.
```

This is not quite what we intended because the outputs of the two printf statements appeared immediately one after the other. We would need to have some sort of way of telling the printf statement to put the output from the second printf on another line. C provides this capability by using character sequences to signify special operations.

Escape Characters

In order to allow the sending of non-printable characters to the display using the printf function, C uses the ‘\’ (backslash) character to signify an escape sequence. There are a few predefined escape sequences:

\n	line feed
\r	carriage return
\t	tab
\0	null
\\	backslash
\’	singe quote

In general, C allows you to send any ascii character using an escape sequence by writing:

```
\ddd
```

where ddd are three octal digits (digits from 0 to 7). Therefore, you could also send a formfeed by sending the escape sequence, ‘\014’, which is an octal 14 (or decimal 12). You can include these escape sequences inside of the control string of the printf function in order to control how your control string is printed. The proper way to print those two strings on separate lines would be to include a ‘\n’ escape sequence in the control string:

```
printf(“Hello Dave,\n”);
printf(“I’m sorry I can’t do that.\n”);
```

The printf function has very sophisticated display capabilities which we’ll look at more next month. As a refresher, here’s the ‘hello world’ program from last month. The printf statement should make more sense now:

```
#include <stdio.h>

void main()
{
        printf(“Hello world.\n”);
}
```

See *Basic C*, Page 6

Robotics Society of Southern California

The Robot Builder (TRB) is published monthly by the Robotics Society of Southern California. Membership in the Society is $20.00 per annum and includes a subscription to this newsletter.

Membership applications should be directed to:

Robotics Society of Southern California
Post Office Box 26044
Santa Ana, CA 92799-6044

Manuscripts, drawings and other materials submitted for publication that are to be returned must be accompanied by a stamped, self-addressed envelope or container. However, RSSC is not responsible for unsolicited material.

We accept a wide variety of electronic formats but if you are not sure, submit material in ascii or on paper. Electronic copy should be sent to:

apendragn@earthlink.net

Arthur Ed LeBouthillier - editor

The Robotics Society of Southern California was founded in 1989 as a non-profit experimental robotics group. The goal was to establish a cooperative association among related industries, educational institutions, professionals and particularly robot enthusiasts. Membership in the society is open to all with an interest in this exciting field.

The primary goal of the society is to promote public awareness of the field of experimental robotics and encourage the development of personal and home based robots.

We meet the 2nd Saturday of each month at California State University at Fullerton in the electrical engineering building room EE321, from 12:30 until 3:00.

The RSSC publishes this monthly newsletter, The Robot Builder, that discusses various Society activities, robot construction projects, and other information of interest to its members.

Membership/Renewal Application

Name ______________________________

Address ______________________________

City ______________________________

Home Phone ( ) - Work Phone ( ) -

Annual Membership Dues: ($20) Check #
(includes subscription to The Robot Builder)

Return to: RSSC
POB 26044
Santa Ana CA 92799-6044

How did you hear about RSSC? ______________________________

RSSC
POB 26044
Santa Ana CA 92799-6044

Please check your address label to be sure your subscription will not expire!