Robot Builder

The Official Publication of the ROBOTICS SOCIETY OF SOUTHERN CALIFORNIA
Post Office Box 26044, Santa Ana, CA 92799–6044

PRESIDENTS MESSAGE

by Jess Jackson

Quite unusual weather the south land is having this year. The snow began to gently fall just outside my office window. Joan and I stepped outside for a few moments to enjoy the winter season before the snow fall passed.

Tom Carroll, our past president, and I were concerned about how to get RSSC to grow and prosper.

I decided to change the format of a few meetings. I chose an open topic interactive forum so that you, the membership could get more involved.

I selected a dozen different topics to be discussed during the meeting. The topics were scheduled for 15 minutes each but the discussion was lively and the items covered all required more time. Only about half of the items were covered.

A WarBot SIG was established and will meet at Jerry's place with first meeting on Mar 2. Mark Thorp, of Robot Wars will be there to discuss the autonomous class rules and to let us know the latest news from the founder. ROBOT WARS 96 is scheduled for Aug 9, 10, and 11 at San Francisco.

There are a number of members that have ROBOTS with 68HC11 controller boards provided by Don Golding, our past technical VP. After talking with various members, sufficient interest surfaced to warrant a refresher course on programing of these boards. Don agreed to teach three 1 1/2 hour sessions.

First session will cover hardware I/O, language basics and commands to control hardware. Second session will cover the instinct and behavior levels of control. The third session will cover writing goal level commands and some problem solving algorithms.

The tentative schedule for the March meeting will be as follows:

11:00	fair planning meeting
12:00	RSSC business meeting
12:30	Break for Lunch
1:00	General meeting
2:30	Whiskers programing
4:00	adjourn

Bring your bots, completed or not, your work projects and what ever to the meeting to share.

Jim Benson, our news letter editor stated that he wants interested and timely articles.

The ROBOT FAIRE96 is now scheduled for Saturday, Oct 5 and Sunday Oct 6. Joe McCord is heading up the fair organization and will need your help to get all the tasks accomplished.

HISTORY: Some time ago I was reviewing old club membership records. I was interested who were the founding members of the RSSC. I was surprised at who is still attending after six years.

Of the 64 that signed up and joined RSSC during late 1989 and 1990, only 10 are still active in the club.

Tom Thornton	8/29/89
Don Golding	8/29/89
Thomas Carroll	9/01/89
Joseph McCord	9/01/89
Jerry Burton	9/05/89
George Ronnquist	11/8/89
Tim Lewis	1/8/90
John Sprinkle	1/22/90
Jesse Jackson	8/13/90
Peter Cresswell	11/7/90

Bring your own discussion topics for the meeting and see what a brain storming cession can do for obtaining answers. I'll see you all at the next meeting

In This Issue

Submission Request

We now have the capability to scan photos and publish them in the newsletter! If you have a photo of your current project — or your past projects — please submit them for publication.

Along with the photo, please give at least some information about what it is and what it does (or is designed to do) :–)

Also in the works is the design and building of an Internet “World Wide Web Home Page” that would let computer users around the world get a taste of what we are doing out here in Southern California . . .

Of course, if you do not want your submission to reach such a broad distribution, just let me know.

World Wide Web

Some sites I have seen.

US&R
http://www.traveller.com/~insecta/

Digital Storage Scope faq
http://www.mv.com/ipusers/wd1v/

Robot News
http://www.robotic.com/robonews.html

OPIE the robot
http://www.islandnet.com/~pmd

Rockies Robotic Group
http://www.he.net/~roundy/RRG.html

Robotics Society
of
Southern California

President
Jess Jackson

Past President
Tom Carroll

Vice President Programs
Henry Arnold

Vice President Faire
Joe McCord

Secretary/Treasurer/Editor Pro Tem
Tom Thornton

Newsletter Editor
Jim Benson

Newsletter Production
Scott McGillivray

Membership
Pete Cresswell

Hospitality
Joan Jackson

The Robot Builder is published monthly by the Robotics Society of Southern California.

The yearly membership fee includes a subscription to this newsletter and is available for $20.00.

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 materials.

If possible, and if you want it to look nice in the newsletter, please submit copy in ASCII or RTF via diskette, email or upload.

rssc@netcom.com (Jim Benson)

Robot Dawn BBS, Host of RSSC
714–538–0614 1200/2400/v.32bis 8N1

LEGS

Tom Thornton

The most common reason to build a walking robot is for improved maneuverability. According to the US Army, half of the Earth's land surface is inaccessible to wheeled or tracked vehicles. We who build robots with legs like to think that our creations will be able to traverse at least some of that difficult terrain.

Another good reason to build walking robots is to see what the issues and problems of locomotion really are. For example, when you notice that your walking machine's feet won't track a straight line, you know that animals must have found solutions to this problem, either in the mechanics of their bodies, or with coordination and control.

Terminology – leg links are referred to with the words used by biologists: the femur is the upper link (thigh), the tibia is the lower leg (shin), and the foot is the tarsus. Joints are hips, knees, and ankles.

During locomotion, there are two distinct phases: when the foot is on the ground, the leg is in the stance phase; when the foot is in the air, the leg is in the swing phase.

Single degree of freedom (dof) legs move in only one axis relative to the body of the robot. The author's HexWalker robot uses single (or one) dof legs. That is, each leg has only one articulated joint and that joint has only one axis of motion. On the HexWalker robot the four corner legs move in the longitudinal plane (parallel to the length of the body). In other words, the hip rotates on a vertical axis. The two center legs move in the vertical plane. That is, the hip joint rotates on an axis parallel to the length of the body. This particular implementation uses three motors (rc style servos) to drive the six legs in pairs. This is the simplest walking machine architecture that allows a robot to move forward and backward and to turn either left or right.

The major limitation of this design is that the robot is restricted to flat smooth surfaces. It cannot navigate irregularities in the surface on which it walks. Some surface irregularities can be accommodated by increasing the lift height of the legs.

Two dof legs have motion in two axes relative to the body of the robot. To overcome the surface roughness limitation each leg can be made to move longitudinally and vertically. Two motors are used on each leg. One method is to attach the first motor to the body which then swings the second motor in one axis. The second motor drives the leg in the second axis.

It is often the case in two dof legs that the knee joint is articulated. In this configuration the femur link is can be four bar mechanism.

This allows the tibia to remain vertical throughout the legs full vertical range of motion. This arrangement is capable of navigating relatively rough ground. When a robot equipped with two dof legs is fitted with proper sensors and controllers it can be made to traverse terrain that is impossible for any wheeled or tracked vehicle of comparable size.

Three dof legs move in three axes relative to the robot body. The third direction of motion transverse to the body allows the robot foot to assume any position in a specified volume called the working volume.. This allows the foot to be maneuvered around a small object rather than stepping on it or over it. It also allows the robot more latitude in selecting foot placement. The capability of arbitrary foot placement enjoyed by three dof legs is offset by increased complexity and weight. For all but the simplest robots this tradeoff is well worth the price.

Next time: Geometry. (What an acorn says when it grows up – gee, i'm a tree!)

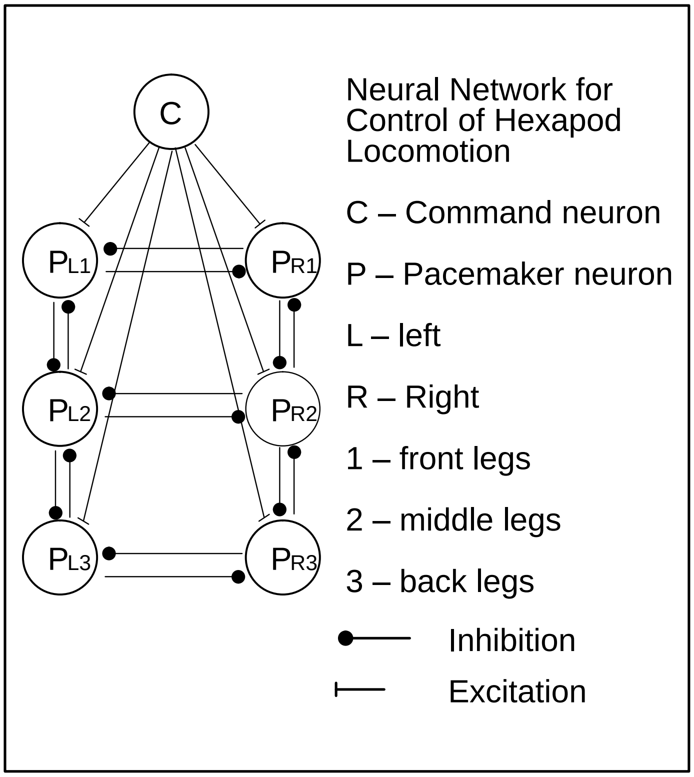
Neural Network Controller for Hexapod Locomotion

from H. J. Chiel and R. D. Beer (1989?)

In studies of cockroach locomotion, Pearson and his colleagues obtained evidence that the protraction of the legs, i.e., the time during which they swung forward, was probably due to the activity of non–spiking oscillatory neurons (Pearson et al. 19973). their studies did not indicate whether this oscillator was a single neuron or a coupled network of neurons, but the lack of overlap between times of activation of adjacent legs suggested that inhibitory connections existed between oscillators for adjacent legs.

Based on these results, we proposed a neural network architecture for the control of locomotion (Beer et al. 1989). Since the oscillatory system had not been specified experimentally, we created a model pacemaker neuron. Two intrinsic currents controlled the behavior of the pacemaker: One induced it to remain depolarized for a fixed period of time, and a second induced it to remain hyperpolarized. The duration of the second intrinsic current varied as a function of the voltage of the pacemaker, so that when the pacemaker was hyperpolarized, the time between bursts was lengthened, and when it was depolarized, the time between bursts was shortened. In addition, brief excitatory inputs during the time a pacemaker was off could turn it on and reset the phase of its bursting; brief inhibitory inputs when it was on could similarly reset its bursting phase. These pacemaker properties have been observed experimentally in nerve cell (Kandel 1976).

The neural network consisted of sex pacemakers, one for each leg. Interactions between pacemakers were modeled after the interactions that had been deduced to exist between the flexor burst generators: Adjacent pacemakers inhibited one another. finally, as a model of higher–order influences that are known to activate locomotion, we connected a single command neuron to all six pacemakers. Its steady excitation or inhibition of the pacemakers caused them to oscillate at different frequencies, due to their intrinsic properties (see figure).



If the command neuron provided identical inputs to each pacemaker, we found that only one pattern of activity was generated by the network: Sets of nonadjacent pacemakers alternate activity. When one set of pacemakers was on, they inhibited all the pacemakers that could inhibit them, and their intrinsic currents provided self–excitation for the duration of their burst. In fact, when the network was in this state, we found that small perturbations such as injecting small depolarizing or hyperpolarizing currents into the pacemakers did not cause it to change state. Once the depolarizing intrinsic current shut off, however, the other triad of pacemakers was released from inhibition and depolarized in response to the steady input from the command neuron. Since their inhibitory intrinsic current was ready to shut off, their depolarizing intrinsic current turned on, and the network moved to its other stable state. Interpreting the outputs of the pacemakers as motor commands, one would say that the network was generating a tripod gait, in which alternating tripods of legs provide support and push backward, while the other tripod swings forward.

SORTING ALGORITHMS

Tom Thornton

So your new sonar is working fine. Pings on command, reads echoes, scans through the designed angle, and cranks out time of arrival and azimuth data – wonderful. Now you have a table full of data but what does it mean? How can you use the data?

What direction is the closet target (obstacle)? Well, you could scan the list for the smallest time of arrival and read the accompanying azimuth value. But that's too easy (and obviates this space consuming article). A more interesting and compute intensive method is to sort the table in time of arrival order.

It turns out that there a number of ways to do this, and these several ways have names – Bubble, Selection, Insertion, and others. Which of these sort routines is best for you data set and the constraints of your available processing power? This article discusses some of the pros and cons of these sorting algorithms.

Bubble sort – make multiple passes through the data set. At each element, exchange adjacent elements if necessary. When one complete pass through the data is made with no exchanges, the sort is complete. In C it this looks like:

```
bubble(int a[], int N)
{
    int i, j, t;
    for (i=N; i>+1; i––)
        for (j=2; j<+i; j++)
            if (a[j–1] > a[j])
                {t=a[j–1];a[j–1]=a[j];a[j]=t;}
}
```

On the first pass the largest element is swapped until it is in the last position. On subsequent passes the next elements are swapped until they are in proper position in the array. A LOT of work takes place to put everything in place.

Bubble performance – bubble sort uses about $N^2/2$ comparisons and $N^2/2$ exchanges on the average and in the worst case. When the file is in reverse order (worst case) the *i*th pass requires N–*i* comparisons and exchanges. Running time of this sort depends on input – only one pass is required if the file is already in order.

Selection sort – find the smallest element in the array, exchange it with the element in the first position. Then find the 2nd smallest element and swap it with the second element. Continue until the data set is sorted. This is called selection sort because is repeatedly "selects" the smallest remaining element. In C it can be written as:

```
selection(int a[], int N)
{
    int i, j, min, t;
    for (i=1; i<N; i++)
    {
        min=i;
        for (j=i+1; j<=N; j++)
            if (a[j]<a[min]) min=j;
        t = a[min]; a[min] = a[i]; a[i] = t;
    }
}
```

As you pass down through the array elements above you are already sorted. When you get to the end of the array the data are fully sorted.

Selection sort performance – process uses about N2/2 comparisons and N exchanges. For each *i* from 1 to N–1, there is one exchange and *N–i* comparisons, so there is a total of *N–1* exchanges and *N(N–1)/2* comparisons. This holds no matter what the input data is: the only part of selection sort that does depend on the input is the number of times `min` is updated. In the worst case this could also be quadratic but in the average case this quantity is only *O(N logN)*, so we can expect the running time of selection sort to disregard input data.

Insertion sort – comparable to selection sort but somewhat more flexible. Each element is considered and inserted by moving larger elements one position down in the array and then inserting the element into the empty slot. A C implementation is:

```
insertion(int a[], int N)
{
    int i, j, v;
    for (i=2; i<+N; i++)
    {
        v=a[i]; j=i;
        while (a[j–1]>v)
            { a[j]=a[j–1]; j––; }
        a[j]=v;
    }
}
```

Similar to selection sort, elements above the action point are in sorted order during the sort, but they may have to be moved to make room for smaller elements found later. The array is fully sorted when the index reaches the right end.

Insertion sort performance – this process is linear for almost sorted data. Insertion sort works well for some types of non–random files that often arise in practice. Consider the operation of insertion sort on a file which is already sorted. Each element is immediately determined to be in its proper place in the file, and the total running time is linear. The same is true for bubble sort, but selection sort is still quadratic. Even if a file is not completely sorted, insertion sort can be quite useful because its running time depends quite heavily on the order present in the file. The running time depends on the number of inversions: for each element count up the number of elements above it which are greater. This is the distance the elements have to move when inserted into the file during the sort.

Conclusions – as a rule these methods take about N2 steps to sort N random items. If N is small enough this may not be a problem. If the items are not random some methods may run much faster than more sophisticated ones. These methods should *not* be used for large, randomly arranged files.

So where does that leave our array of sonar sensor data? Sad to say, but none of the above sort processes are suitable for any but the smallest data sets. The method of choice for data sets such as we might be expecting to work with is the Shell Sort – but that is another story.

The above statistics on sort algorithms are found in “Algorithms in C” by Robert Sedgewick, Addison–Wesley 1990, isbn 0–201–51425–7.

The Robotics society of southern California was originally formed in 1989 as a non–profit experimental robotics association. The goal was to establish a co–operative association among associated 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 on the 2nd Saturday of each month at Orange Coast College in Costa Mesa.

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

Membership / Renewal Application:

Name ____________________________________

Address ____________________________________

City ____________________________________

Home Phone (   )   –   Work Phone (   )   –

Annual Dues: Newsletter & Membership ($20)   Check #

Return To: RSSC
Post Office Box 26044
Santa Ana, CA 92799–6044

Area(s) of Interest / Background

How did you here about RSSC?

Robotics Society of Southern California
Post Office Box 26044
Santa Ana, CA 92799–4066

Address Correction Requested

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

The Robotics Society of Southern California meets the Second Saturday of the month at Orange Coast Collage.
Take Fairview Blvd south from the 405 then turn right on Merrimac Way. Free Parking.
** Look for the building with the green glass roof **