Journal on
Perceptual Control Theory
Vol 1 No 1, 1999

Control of a multi-legged robot based on hierarchical PCT

by

Richard Kennaway
University of East Anglia
Norwich, U.K.

Last Revision Date: March 20, 1999 ISSN 1437-9325

Control of a multi-legged robot based on hierarchical PCT

Richard Kennaway, University of East Anglia, Norwich, U.K.

10 March 1999

Abstract

We describe an experiment in robot control, using a simulated six-legged robot. The robot, herein called Archy, is capable of standing, resisting disturbing forces, and walking over uneven terrain and up and down a staircase. It can continue to stand up and control its body position and orientation even when some legs are removed. The control systems are all simple PID controllers, with a couple of fuzzy controllers for navigating when walking.

1 Introduction

The purpose of this research is to explore the effectiveness of HPCT as a control architecture for a complex non-linear control problem. The problem is to enable a six-legged robot to stand, adjust its body to any desired position and orientation (with respect to its footprint), to walk, and to navigate towards a landmark. It should be able to cope with externally applied disturbing forces and uneven terrain.

There are many hexapod robot projects, to survey which would require a lengthy article, and we shall not do so here. The HPCT control architecture is described in detail in other sources ([2, 3, 4]), and for brevity we assume familiarity with its fundamentals.

A basic principle of this organisation is that only the bottom layer of controllers send their output to the physical actuators. All higher levels send their outputs to the reference inputs of the next level down. In effect, each layer of control creates a virtual machine whose "actuators" are the reference inputs for that layer.

As yet, Archy exists in simulation only. This has the advantage of allowing greater flexibility in its design, but the disadvantage that it is difficult to simulate mechanical systems closely enough to be confident that the behaviour is close enough to that of a physically constructed robot. However, the robustness of our design to large variations on the physical properties of the robot gives us some assurance that our design will transfer to a physical robot.

Figure 1 shows Archy as it appears when the program is running. Its geometry is roughly similar to that of a stick insect. Each leg consists of an upper and a lower leg, with an elbow joint having one degree of freedom (open-close), and a shoulder joint with two degrees: pitch (up-down) and yaw (left-right). This makes a total of eighteen degrees of freedom of articulation. (Although we refer to the appendages as "legs", we call its joints the shoulder and elbow.)

There are many parameters of the simulation which the user can vary. The screenshot in Figure 2 shows in the lower half just one of 15 separate panels of controls.

2 Bottom-level control

For each of the 18 degrees of freedom of articulation there is a PID (proportional-integral-differential) controller. The controller senses the rate of change of the angle of the joint, compares that with a reference value, and outputs a torque tending to bring the perception closer to the reference.

Each PID controller has a response determined by four parameters, according to the following formula. The quantities are p, the perception, r, the reference, $e = p - r$, the error, and o, the output. t is the time. K_P, K_D, K_I, and K_L are constants.

$$o(t) = -K_P\, e(t) - K_D \frac{de(t)}{dt} - K_I \int_{-\infty}^{t} e(t') \exp(K_L(t' - t))dt'$$

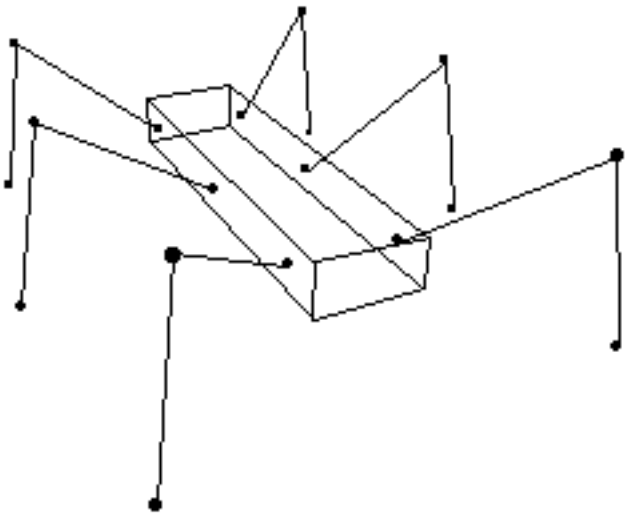

Figure 1: Archy

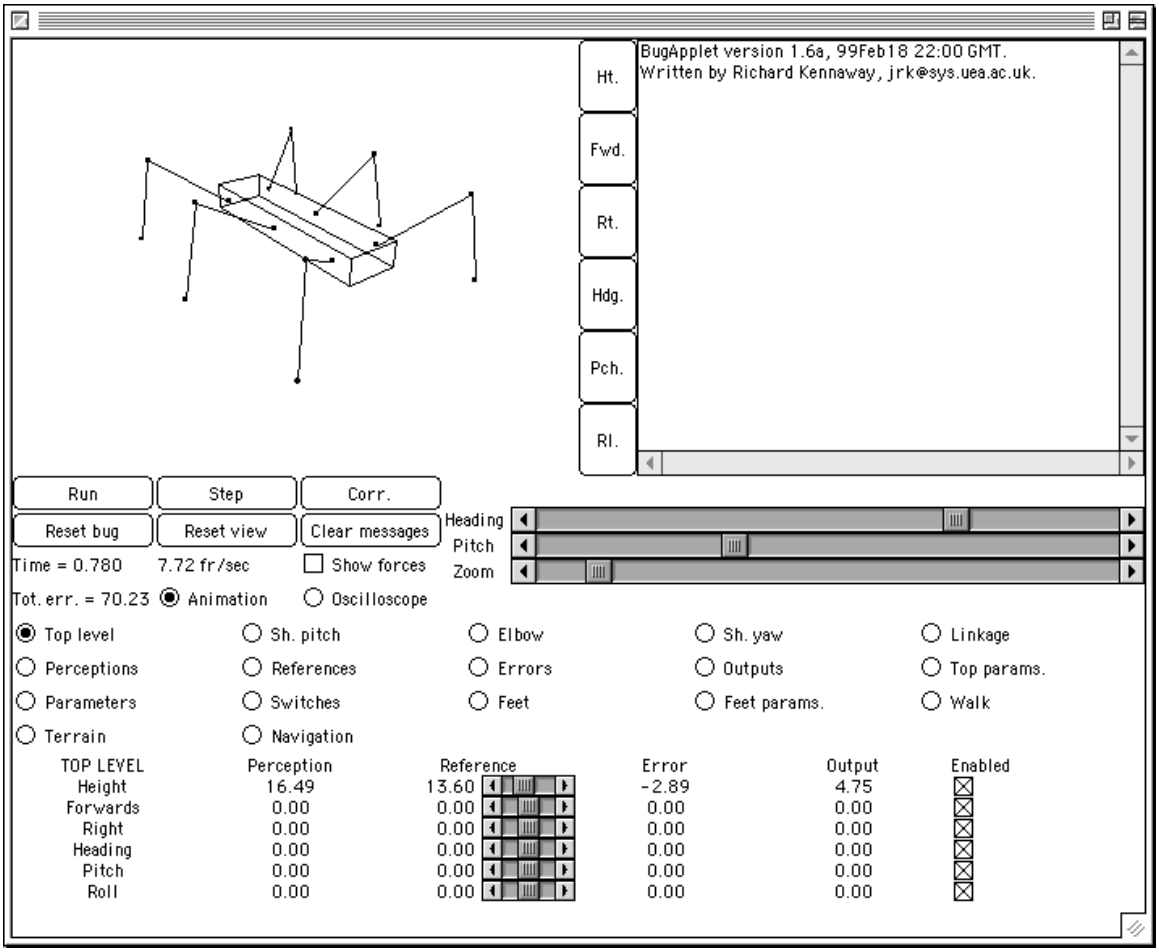

Figure 2: The whole applet window

	height	forwards	rightwards	heading	pitch	roll
left 1	$-+0$	$0\,0+$	$0+0$	$0+0$	$-+0$	$-+0$
left 2	$-+0$	$0\,0+$	$0+0$	$0\,0\,0$	$0\,0\,0$	$-+0$
left 3	$-+0$	$0\,0+$	$0+0$	$0-0$	$+-0$	$-+0$
right 3	$-+0$	$0\,0-$	$0-0$	$0+0$	$+-0$	$+-0$
right 2	$-+0$	$0\,0-$	$0-0$	$0\,0\,0$	$0\,0\,0$	$+-0$
right 1	$-+0$	$0\,0-$	$0-0$	$0-0$	$-+0$	$+-0$

Figure 3: Linkage matrix

The constants are the same for each of the 18 controllers, and can be adjusted by the user. They are all non-negative. By default, K_I is zero.

With just these controllers, and the references set to zero, when the simulation is initiated, Archy falls slowly to the ground, at a rate limited by K_P and K_D.

3 Second-level control

There is a second level of control, which controls the position and orientation of Archy's body. This contains one controller for each of the six degrees of freedom: height, forwards, rightwards, heading, pitch, and roll. The perception of height is the height of the centre of the body from the ground immediately below. Forwards and rightwards are defined relative to the centroid of the displacement vector of each foot, i.e. the vector from the foot's default initial position to its current position. The perception of heading is relative to the legs, and is computed from the shoulder yaw angles: if all six legs were turned rightwards from their default initial position, Archy would perceive its body as being turned leftwards. Pitch and roll are perceived relative to the best-fit plane through its feet.

In an earlier version, pitch and roll were perceived relative to gravity. When Archy was given the task of climbing stairs, this proved ineffective, since the geometry makes it impossible for a vehicle of this shape to climb stairs unless its body adopts an orientation more or less parallel to the stairs. The pitch and roll perceptions were therefore changed to be relative to the footprint plane.

These controllers are again PID controllers, and by default, K_I is zero. The user can vary these parameters independently for each controller.

The six outputs of these controllers provide the 18 reference inputs of the bottom level controllers, via a fixed linkage matrix displayed in Figure 3. Each column corresponds to a top level controller, and each row to a leg, numbering the legs from 1 to 3 front to back. The three values in each entry correspond to the three leg joints, in the order shoulder pitch, elbow, and shoulder yaw. Each element of the matrix is either 1, 0, or -1.

For example, the height controller's output is delivered negatively to each shoulder pitch joint, positively to each elbow joint, and not at all to the shoulder yaw joints. This means that if Archy's height is below the reference height (implying that the error is negative, and the output signal positive), it will attempt to raise itself by swinging all six shoulder joints downwards and opening all six elbow joints. Similarly, pitch errors will be corrected by means of the shoulder pitch and elbow joints of the front and rear legs. Forwards errors are corrected through the shoulder yaw joints, and so on.

Notice that most of the joint controllers will combine inputs from several of the top-level controllers. The front elbow joints, for example, take input from the height, rightwards, heading, pitch, and roll controllers. This creates complicated couplings between the controllers. Nevertheless, each of the upper-level controllers succeeds in controlling its perception. As far as each controller is concerned, the couplings have the effect of disturbances which the controller resists, just as it would resist any other disturbance.

The user interface allows the user to set the reference levels for each of the upper-level controllers. As the user drags the relevant scroll-bar thumb, Archy responds in close to real time. With the control parameters tuned appropriately, there is very little overshoot or lag, and the whole system remains stable.

The default values for the control parameters for all the controllers were initially guessed by physical intuition, and then adjusted by trial and error to obtain stability and good response to disturbances. There is no autonomous adaptation in the current version.

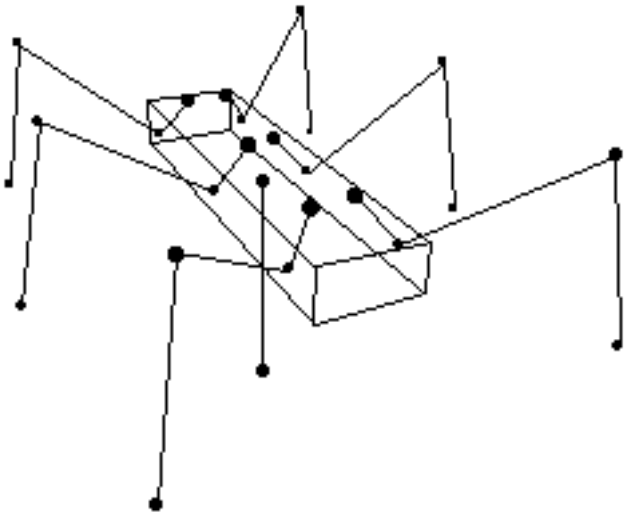

Figure 4: Force vectors

4 Resistance to disturbing forces

Simulated disturbing forces and torques can be applied. These can be randomly time-varying or regular oscillations. Very little appears to happen when the user switches these on. The robot remains motionless. This is typical of well-functioning control systems — they often don't look as if they are doing anything. An option to display all the forces and torques in the animation as vectors illustrates more clearly what is happening.

In Figure 4, the disturbing force and the total force are displayed as vectors based at the centre of the robot, and ending in blobs. The forces exerted at each shoulder are displayed as vectors based at the shoulders. The long downwards line is the disturbing force — something is pushing downwards on the robot. The blob at the robot's centre is the (near-zero) net force. The vectors upwards from the shoulders are the forces being exerted on the body by the legs. As the disturbing force increases, the forces exerted by the legs also increase, keeping the body stationary.

Each of the top-level controllers can be switched on or off by the user. If the height controller is switched off in the situation of Figure 4, Archy immediately yields to the disturbance and drops to the ground. When the height controller is turned back on, Archy immediately lifts itself to its former position. The response to disturbances of the other degrees of freedom is similar.

Besides disturbing forces, individual legs can be disabled. As long as Archy has enough legs to balance on, it is still able to control its body position and orientation in the face of disturbances, although not so well as when it has all its legs. The total number of legs is a compile-time constant in the code; 4-legged, 8-legged and 10-legged versions also work well.

5 Walking

We assume a repertoire of basic actions for each leg: lift the foot off the ground, place the foot on the ground, and (if the foot is off the ground) set the yaw angle to Δ, 0, or $-\Delta$, where Δ is a fixed value of 0.6 radians. The physical model here is simplified: each of these actions is assumed to happen instantaneously. A more sophisticated simulation would model the mass of the leg and move it by means of the joint angle controllers.

Walking is performed by executing a sequence of these actions at regular time intervals. Raise and swing forwards the front legs. Lower the front legs. Raise and swing forwards the middle legs. Lower the middle legs. Raise and swing forwards the rear legs. Lower the rear legs. This results in forward motion, because when a pair of legs is swung forwards and placed on the ground, this moves the centroid of the footprint forwards. Archy now perceives its body to be to the rear of the centroid, and acts to move its body forwards, by swinging all six legs backwards.

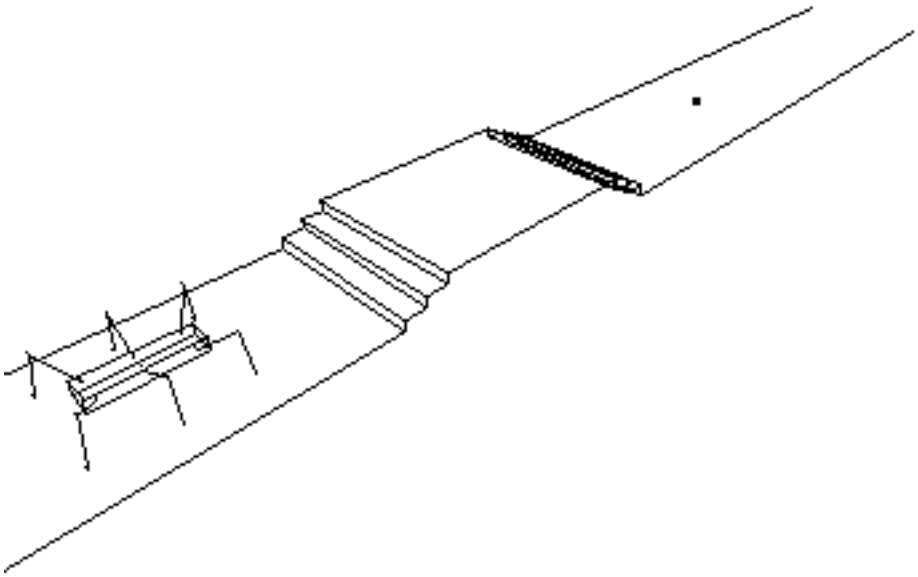

Figure 5: Archy, stairs, and a landmark

Turning round is performed by a similar sequence of actions. To turn to the right, raise the front legs and swing them clockwise, lower them, and repeat with the middle and rear legs. The heading controller will perceive (by sensing the shoulder yaw angles) that the body is pointing to the left, and will adjust the references for rate of change of shoulder yaw so as to swing the body to the right.

The above mechanism is simply a blind sequence of actions. Disturbances, uneven terrain, and rounding errors will all tend to make Archy's path when walking forwards wander from a straight line. We add to the environment a landmark, and provide Archy with sensors that tell it what direction the landmark is, and how far away. Two additional controllers sense the direction and distance, and produce as their output a selection from the possible walking programs: forwards, backwards, turn right, turn left, and stop. The controllers are fuzzy: direction is categorised into ahead, to the left, to the right, and out of sight. Distance is far or near. The combined output selects turning left when the landmark is to the left, right when it is to the right or out of sight, ahead when it is ahead and far away, and stop when it is ahead and near.

When the landmark is placed on the opposite side of a short staircase up and down, as shown in Figure 5, Archy will walk towards it, successfully going up and down the stairs. Archy does not know that the stairs are there; so long as it lifts its legs high enough to get over the steps, it will make it to the other side.

Figure 6 shows various stages in walking over the stairs. These are taken from a single run, with the view angle adjusted for clarity for each snapshot.

6 Conclusion

The robot control architecture is noteworthy for what it does not contain. Archy has no model of its surroundings or itself. Its perceptions are built solely from its joint angles, their rates of change, the position of its body relative to its feet, its height from the ground beneath it, and the distance and direction of the landmark. It knows nothing else about its environment. It performs no forward or inverse kinematic calculations. (Such calculations do occur in the code that simulates the physics, but if Archy were physically constructed, none of that code would be present.) It does not perform any planning.

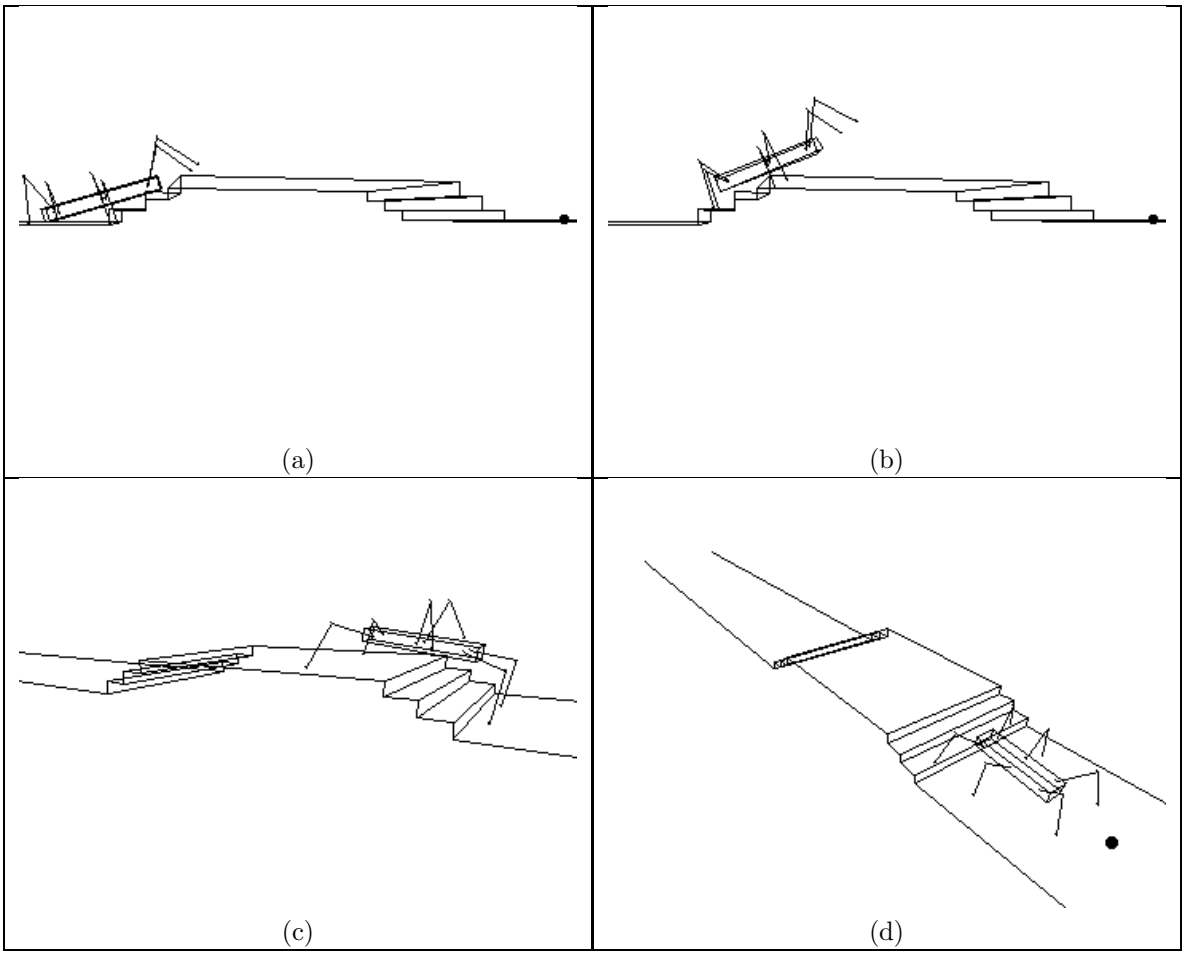

Figure 6: Walking up and down stairs

The only similar approach to robot control is Rodney Brooks' subsumption architecture [1], which shares many of these features. Subsumption, like HPCT, builds complex behaviours from layers of simple controllers. However, subsumption makes no prescriptions about how those controllers may be connected together. In principle, controllers at all levels may have access to the actuators, with higher-level controllers being allowed to override the outputs of lower level controllers. HPCT provides a rationale for a specific organisation of the interconnections.

We intend to make the physical simulation more accurate by modelling the mass of the legs. This will allow us to model walking in more detail, by using the joint actuators to lift and move the legs, instead of simply placing them in the desired positions. Physically simulating such a complex system is quite a difficult task. There are fourteen mechanical elements (six upper legs, six lower legs, the body, and the ground), with many kinematic loops. We hope to find off the shelf software for modelling such systems, although this may mean departing from Java.

However accurate the simulation can be made, it will never be as detailed as the real world. Therefore one can only be sure that this approach to robot control will work by demonstrating it in a physical robot. We are currently looking at possible methods of low-cost construction. The fact that the control architecture is robust to quite gross disturbances suggests to us that complex high-precision engineering will not be necessary.

The version of Archy described in this paper is available on the web at
http://www.sys.uea.ac.uk/∼jrk/PCT/Archy/Archy.html.

References

[1] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[2] W. Powers. *Behavior: The Control Of Perception*. Benchmark Publications, *http://members.aol.com/BenchmkPub/index.htm*, 1998. Originally published by Wildwood Press, 1974.

[3] W. Powers. *Making Sense of Behavior: The Meaning of Control*. Benchmark Publications, *http://members.aol.com/BenchmkPub/index.htm*, 1999.

[4] Control Systems Group web site. *http://www.ed.uiuc.edu/csg/*.