FINAL TECHNICAL REPORT
WALKING MACHINE CONTROL PROGRAMMING

20 September 1983
SSA #2054

DARPA Order Number: 4456

Name of Contractor: Sutherland, Sproull and Associates, Inc.

Effective Date of Contract: 1 February 1982

Contract Expiration Date: 31 August 1983

Reporting Period: 1 February 1982 – 31 August 1983

Contract Number: MDA903-82-C-0102

Principal Investigator: Ivan E. Sutherland (412)621-5014

Short Title of Work: Walking Machine Control Programming

CONTENTS

Sponsored by
Defense Advanced Research Projects Agency (DoD)
DARPA Order No. 4456

Under Contract No. MDA903-82-C-0102 issued by
Department of Army, Defense Supply Service-Washington,
Washington, DC 20310

SUMMARY—WALKING MACHINE FINAL REPORT

by Ivan E. Sutherland
Sutherland, Sproull and Associates, Inc.
SSA #2054
20 September 1983

The objective of this investigation was to evaluate a 1600-pound, six-legged, gasoline-powered, self-contained, man-carrying, hydraulically-actuated, walking machine built by Sutherland, Sproull and Associates. We were interested in the task of programming the control computer, the problems of operator control, and the performance of the vehicle.

The walking machine presents two technical problems. The first is how best to permit the operator to control it. A control stick and two foot pedals provide operator inputs, and we connected them in various ways to give the driver control of the machine's path, direction, and speed. In addition, we experimented with different algorithms to select the particular uses of particular legs.

The second technical problem posed by the walking machine comes about because of the topology of the hydraulic circuits that actuate it. Rather than using a separate servomechanism for each joint, the hydraulic circuits of this machine permit actuators from several of its joints to be made a part of a single hydraulic circuit. Parallel connections of actuators permit leg to share load equally; series connections force them to move synchronously. By setting valves to establish series and parallel hydraulic circuits, the control computer can obtain coordinated joint motions without further direct action. We learned a great deal about the strengths and weaknesses of this approach to leg coordination.

Our general methodology was direct experimentation. The machine was used for experiments with walking from October 1982 until August 1983. We operated the machine in a parking lot adjacent to the laboratory that housed it. Three operators became reasonably adept at controlling it. We made two video tapes of its operation and had numerous demonstrations for government, industry, and academic visitors.

This report presents the technical results of this work. In order to avoid duplication of material already published, this report presumes its reader to be familiar with the material covered in the book, *A Walking Robot*, which describes the machine. This book is published by The Marcian Chronicles, P.O. Box 10209, Pittsburgh, PA 15232. Detailed material about the control programming is included in the body of this report. As an appendix, we have included a paper called "Footprints in the Asphalt," which is soon to be published in *The International Journal of Robotics Research* by The MIT Press. This paper is the best simple summary of the research project.

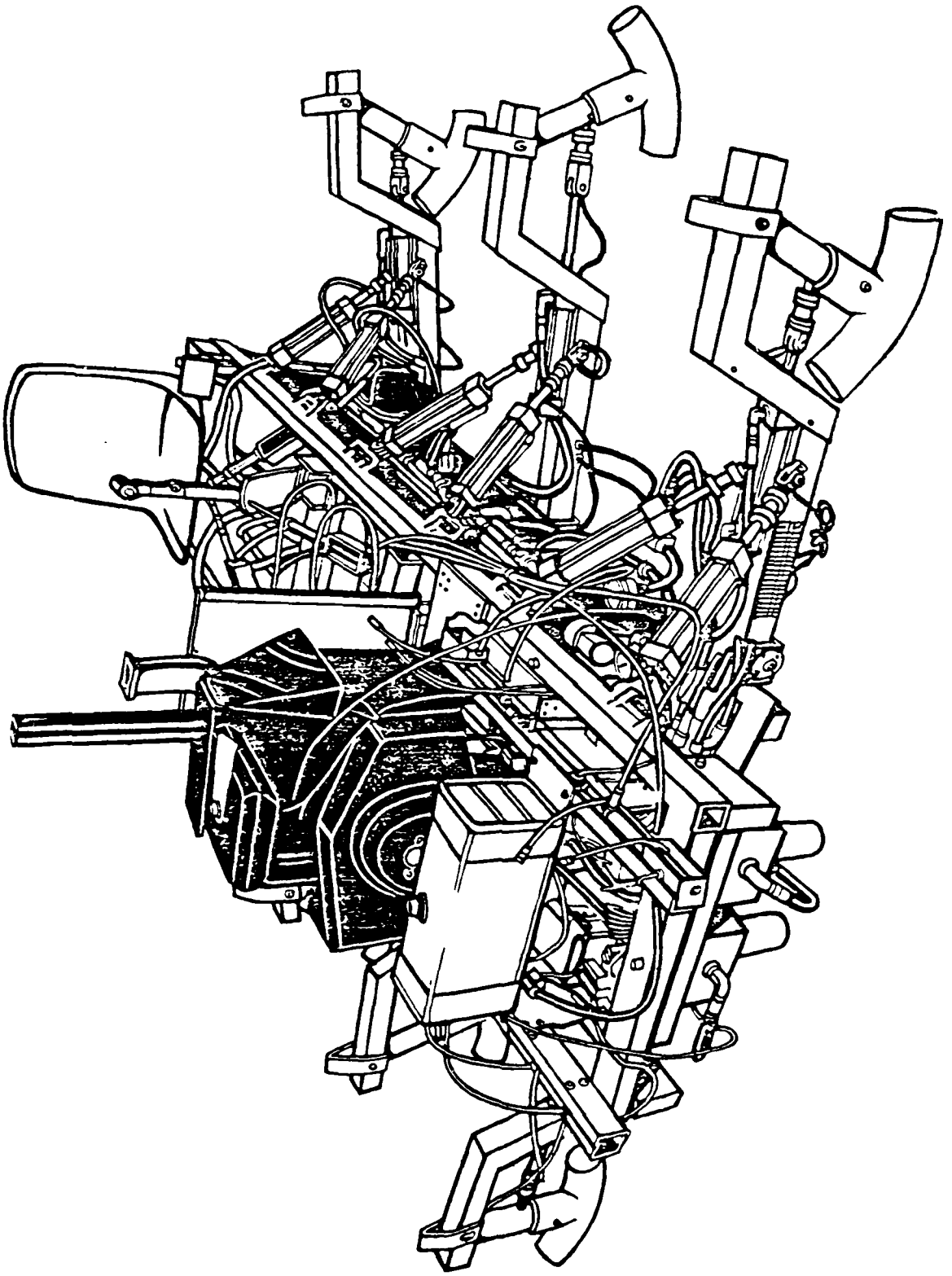
The central finding of our work is that because inertia, and thus balance, play a very important role in a large machine, six is the wrong number of legs to use. Insects have six legs, apparently so that they need never balance, but insects are so small that inertial forces can be ignored relative to gravity, windage, and traction forces. Moreover, there is some evidence from their ability to walk on walls and ceilings that insect feet grip the surface to provide extraordinary traction. The common wisdom that by using six legs one can avoid the need to balance is simply wrong in a machine of the size and speed we have built. Inertial forces when starting and stopping have often caused the machine to tip and bump its underside on the pavement.

A second finding of our work is that the parallel connection of actuators used in the machine works very well. With a parallel connection, load forces are shared equally, making two or more legs act together as if they were a single leg centrally located. We have come to call this notion the "virtual leg" concept. It appears to us that virtual legs could be used to provide very simple control over a complex walking machine.

A collection of findings about independent control of the individual legs will be published separately as a PhD dissertation by Marc Donner. Marc has implemented a concurrent programming language for programming the machine—a language that is interesting for many applications in addition to control of walking. This work was supported in part by IBM.

The hardware development required for this activity was funded by Sutherland, Sproull and Associates. This contract covered only the evaluation of the machine.

A-1



WALKING MACHINE CONTROL PROGRAMMING

FINAL TECHNICAL REPORT

September 20, 1983

SSA #2054

Sutherland, Sproull and Associates, Inc.
1035 Devon Road
Pittsburgh, PA 15213

A-1

Table of Contents

SUPPORT PROGRAMMING

The program running in the on-board walking machine microprocessor has several main functions. First, it decodes values from the various sensors throughout the machine and encodes commands for the valves. That is, it serves as an interface between algorithm and mechanism. Second, it provides a fairly direct way for the operator to control the valve settings manually. This facility is useful for performing various housekeeping chores like lifting the machine up onto its jacks and testing out the motion of individual legs. Third, the code implements a variety of algorithms for walking. The driver can cause the machine to walk forwards, backwards, and sideways, and also to make turns, including turns in place. Fourth, the code transmits valve commands and sensor values between the walking machine hardware and an OWL program running in its own processor. OWL is described elsewhere in this report.

The walking machine program accepts commands from the phone buttons and control stick switches mounted on the machine. It displays its current status on a terminal placed nearby. Because of the limited supply of input switches, the program is organized as a collection of subprograms. Each of these subprograms, or modes, has a unique function and interprets the switches somewhat differently. The top level program displays a menu of the available subprograms, shown below, which can be activated by pressing the appropriate button on the phone dial.

1=Switch,	2=Sensor,	3=Normal,
4=Gait,	5=Quick,	6=Mancon,
7=Servo,	8=Histry,	9=Walk,
*=Tidy,		#=Fowl.

The walking and manual control programs mentioned earlier are examples of subprograms that can be selected in this manner. Pressing the two-step switch on the control stick all the way down causes the currently active subprogram to terminate and return control to the top level menu. The fact that the two-step switch has this behavior throughout the entire program has been useful for running without the display screen because the driver is always able to place the program into a known state.

LOW LEVEL INTERFACE

As described in Ivan Sutherland's book *A Walking Robot*, the control microprocessor is connected to the valve electronics of the walking machine by means of a long serial shift register. The microprocessor shifts valve control bits out into the machine, and it shifts digitized sensor readings in from the machine. In preparation

for this shifting operation, the processor must convert a valve command from the ASCII letter used to represent it in the program to the combination of bits that control the six valves for each leg. It accomplishes this encoding by means of a table lookup. Next, the processor must complete the hydraulic circuit by choosing an appropriate setting of the compensator valves. The theory behind this choice is covered in *A Walking Robot*, but for the purposes of this report it is enough to say that the compensator setting is determined from the combination of leg valve settings by means of another table lookup. Finally, once it has finished encoding the valve commands, the processor shifts them out to the valves and shifts in a fresh set of sensor readings.

The operation of preparing the raw sensor readings for use by the walking algorithm and other control algorithms is slightly involved. After accepting the raw sensor values from the shift register, the processor must scale them so that each lies within some known, fixed range. Moreover, many of the walking machine's joints are monitored by two sensors, and these redundant sensor readings must be averaged to determine the position of the joint. A separate calibration program determines the maximum and minimum sensor readings observed at the extremes of motion of each joint. The resulting calibration tables are usually updated about once a week. The difference between the maximum and minimum calibration values for a particular sensor gives the domain of values that can be expected from that sensor. The average of the minimum and maximum values gives the center of travel of a joint. Another table contains the range of values expected by the various control algorithms. This range table generally remains fixed, although it may be altered manually in order to disable broken sensors or to change the sign of readings from sensors that are wired up backwards. To convert raw sensor readings into scaled values that can be used by walking algorithms, the processor subtracts the average of the minimum and maximum calibration values, multiplies by the desired range, and divides by the observed domain. After scaling, the processor averages redundant sensor readings merely by adding them. The missing factor of two was included in the range table.

All of the tables needed to convert raw sensor readings into values suitable for use by the control algorithms may be examined from the terminal. One of the subprograms selected from the main menu can display the raw sensor readings, the minimum and maximum calibration data, the average and domain tables, the range table, and the scaled sensor readings. The driver selects which table to display by means of the phone dial. Another subprogram displays the completely processed sensor data for selected legs.

The overall operation of the walking machine code may be regarded as a sequence of operations repeated indefinitely. The program first encodes the set of valve commands as a sequence of bits. It then shifts the valve control bits out into the walking machine while shifting in the raw sensor readings. Next, it scales and combines the sensor readings, converting them into a standard form. Finally, it uses the new sensor values as input for a walking algorithm or other control algorithm to determine a new selection of valve settings, and the loop repeats. This main operation loop has been observed to run in as little as 25ms with a very simple control algorithm. By way of comparison, the time required for hydraulic valves to

switch is about 100ms.

MANUAL CONTROL

The manual control program provides a means for the driver to command the motion of the legs in a fairly direct manner. It is useful for a variety of tasks, including climbing up onto jack stands for repair and maintenance, raising the body of the machine high enough to remove the wheels in preparation for walking, checking out leg motion and sensor calibration, and separating legs when they collide during walking.

Like the other subprograms in the walking machine code, the manual control program accepts commands from the control stick and the phone dial. The driver can use the buttons on the phone dial to select the set of legs to which subsequent motion commands apply. Pressing a digit from 1 to 6 adds the corresponding leg to the set of active legs, pressing 0 clears the set of active legs so that no leg is selected, and pressing the digit 9 selects all six legs. The terminal shows which legs are currently selected.

After selecting a set of legs with the phone dial, the driver can specify their motion by means of the control stick switches. Unselected legs remain frozen in the *Hold* valve setting. Selected legs respond to the control stick switches according to the following table.

Control Stick Switch	Valve Setting	Resulting Leg Motion
Trigger	*Energize*	Horizontal Fore/Aft
Thumb Switch	*Place*	Compliant Down/Up
Coolie Hat Up	*down*	Diagonal Down/Up
Coolie Hat Down	*Down*	Diagonal Down/Up
Coolie Hat Left	*Right*	Knee Left/Right
Coolie Hat Right	*Left*	Knee Right/Left
Thumb Switch and Trigger	*Lift*	Compliant Up/Down
Coolie Hat Up and Trigger	*Up*	Diagonal Up/Down
Coolie Hat Down and Trigger	*up*	Diagonal Up/Down
Coolie Hat Left and Trigger	*Left*	Knee Left/Right
Coolie Hat Right and Trigger	*Right*	Knee Right/Left

Notice that when the trigger is combined with some other switch, the action of the other switch is reversed in some sense.

The *Place* and *Lift* settings are most useful for doing "push-ups" in which the walking machine uses its legs to press the body into the air. The machine will typically do a push-up before and after walking in order to permit removal and reinstallation of its wheels. Also, when repairs are necessary, the machine can do a higher push-up by standing on four wooden boxes, or "footstools." This maneuver permits jack stands to be inserted beneath the body, and once the machine has settled onto the jacks and the footstools have been removed, its legs can no longer touch the ground. Besides making it easier to work on the individual legs, walking algorithms are simpler to debug when the machine is in the air.

The remaining valve commands are less commonly used in manual control. The driver will sometimes attempt to adjust the position of the lower leg using the *Left* and *Right* settings, because push-ups place less strain on the leg when the lower leg is perpendicular to the upper leg. The *Energize* setting is most often used to demonstrate the "hydraulic ellipse," which is described in Sutherland's book. The diagonal motion settings, *Down*, *down*, *Up*, and *up*, will move a leg in a definite direction and can therefore be used to position a leg more precisely. When legs collide during walking, one of these commands can be used to get a leg out of trouble.

It is possible to drive the compensator manually in or out when none of the legs are selected. To extend the compensator, the driver must push the coolie hat up while depressing the thumb switch. Pushing the body of the machine up off the ground tends to cause the compensator to be pulled in, so it is useful to be able to push it back out in order to avoid pumping oil through a relief valve. If the driver pushes the coolie hat down while depressing the thumb switch, the compensator will be retracted. Unfortunately, the valve configuration for withdrawing the compensator works by pumping oil through a relief valve—a property that detracts from its usefulness. When the legs of the walking machine rise with respect to the body, the compensator will be driven out. After the compensator runs out of travel, the driver has two choices. He may either continue driving the legs up, which forces oil through a relief valve, or he may retract the compensator, which also forces oil through a relief valve. In either case, the oil temperature will rise. It is thus not generally useful to drive the compensator in manually unless it must be precisely positioned for hardware maintenance, program testing, or some similar operation.

Another manual control program is known as the "Tidy" program because it serves to tidy up the leg configuration. The Tidy program can perform three operations: tuck, knee center, and jack. In the tuck configuration, the upper legs are raised as high as possible and the lower legs are pulled in close to the body. This configuration is useful for two reasons: first, the machine won't fit through the garage door unless it is in the tuck position, and second, a principal way to drive air from the hydraulic system is to drive the compensator in while the machine is tucked. The knee centering routine simply positions the lower legs at the approximate center of their travel. Finally, the jack program was intended to be used for doing push-ups in a semi-automatic manner. The original thought was that the coolie hat would control which pair of legs was driven into the ground, thereby eliminating some button pushing that would be needed if the straight manual control program were used. In practice, manual control is simple enough to use for this purpose.

OWL CONTROLLER

OWL is a programming language designed by Marc Donner to facilitate the description of walking algorithms. Because of OWL's processing and memory requirements, it is implemented in its own microprocessor. Part of the task of the walking machine microprocessor is to communicate between the OWL micropro-

cessor and the control electronics of the walking machine. The walking machine code makes sensor readings continually available to the OWL microprocessor, and in addition, when the OWL server has been activated, it will relay valve commands to the walking machine.

The driver can select the OWL server by pressing the appropriate button of the phone dial when the main menu is active. After selecting the server, the driver can activate it by pulling the trigger. At this point, the walking machine microprocessor will no longer respond directly to driver commands, relying instead on a running OWL program to provide this function. When it has finished, the OWL program can terminate the OWL server so that the driver can use the walking machine code. Alternatively, typing the sequence "FOWLOUT*" on the phone dial will cause the server to disconnect itself from OWL.

The OWL server communicates with a running OWL program by means of a shared memory. The memory of the walking machine microprocessor occupies a small part of the address space of the OWL microprocessor. Several tables and flags are used to accomplish communication between the two processors. Two of the tables are large enough to hold the sensor values that are passed from the walking machine microprocessor to the OWL microprocessor. The other two tables are large enough for the valves commands that are passed in the opposite directions. Pointers serve to identify the active tables, and flags insure exclusive access to the pointers.

There are three synchronization flags for controlling the interactions of the two microprocessors. Two of them arbitrate access to the valve and sensor tables, while the third can be used to terminate the OWL server running in the walking machine microprocessor. Either processor is permitted to examine a synchronization flag at any time, but modifications are restricted. When the value of a synchronization flag is zero, it is considered to be owned by the OWL microprocessor, and when it is non-zero, the walking machine microprocessor has jurisdiction. Neither processor may modify a flag that is does not own. For example, the walking machine microprocessor signals that the OWL server has been activated by clearing the OWL server termination flag. When the OWL microprocessor has finished with the server, it will set this flag to some non-zero value in order to terminate the OWL server and release the walking machine microprocessor.

Sensor values flow from the walking machine microprocessor to the OWL microprocessor by means of a double-buffering technique. The processor begins by determining whether it owns the sensor value synchronization flag. If it does, it fills an available table with fresh sensor readings, places the address of that table into a known pointer location, and finally clears the synchronization flag. By clearing this flag, the walking machine microprocessor agrees not to modify either the flag or the pointer until the OWL microprocessor has returned ownership by setting the flag to some non-zero value. At this point, the OWL microprocessor is said to have accepted the table of fresh sensor readings. The walking machine microprocessor further agrees not the change the sensor reading table until it has passed another one to the OWL microprocessor and the OWL microprocessor has accepted. Valve commands pass from the OWL microprocessor to the walking machine microprocessor by a similar set of flags, pointers, tables, and protocol.

The walking machine microprocessor supplies sensor information to the OWL

microprocessor on a continuous basis, whether or not the OWL server is running. This has been useful because it allows the OWL microprocessor to monitor the performance of walking algorithms running in the walking machine microprocessor. The OWL microprocessor has substantially more memory, so it can record sensor values that are considered to be interesting.

Besides merely acting as a transducer between the walking machine hardware and the OWL microprocessor, the walking machine microprocessor helps to shoulder a bit of the computational load where this is appropriate. First, the sensor values passed between the processors have already been scaled and averaged, relieving the OWL microprocessor of quite a few division operations. Second, the OWL server accepts valve commands in a symbolic form, locally converting them to the bits needed to control the physical valves. It determines the required setting of the compensator valves as well. Finally, it continually monitors the positions of the legs for impending collisions.

WALKING PROGRAMMING

The walking algorithm is based on the premise that the code for deciding *when* to recover a leg can be separated from the code controlling *how* to recover a leg. Recall that leg recovery is the act of moving a leg into position for another step at the end of the stride. This separation of function has two main advantages. First, the two smaller programs are easier to write and maintain than a single larger program would have been. Second, and more significant, because leg sequencing is separate from leg control, it is a relatively simple matter to experiment with a variety of leg sequencing algorithms, or gaits. Indeed, the driver can select one of six primary gaits and then refine it for even more flexibility.

At the heart of the walking algorithm are six independently running finite state machines corresponding to the six legs of the walking machine. The states in each of these finite state machines represent the various phases of motion in the stride and recovery of a leg. A single, independent gait program coordinates legs by forcing selected legs into the initial recovery state at the proper moment. The six finite state machines choose hydraulic valve settings that will result in motion that is appropriate to the individual states. Since more than one leg may be active at a time, however, the independent finite state machines might choose a combination of valve settings that is undesirable or even dangerous. To correct this situation, a series of post-processing programs examine the valve settings chosen by the finite state machines and eliminate improper combinations. Each of the post-processing programs need detect and correct only a single difficulty. Once again, this separation of control has worked out very well because it reduces the amount of coordination needed between the six finite state machines, and it reduces the complexity of the individual modules.

To aid in the debugging and analysis of the walking code, there is a facility for recording transitions between states of the finite state machine. Whenever the finite state machine for a leg switches from one state to another, the new state is entered into a circular buffer that occupies whatever memory is otherwise unused. In addition, some of the states record numeric data like leg positions. After walking has taken place, it is possible to examine the recently recorded history in order to analyze the behavior of the machine. Playback may be restricted to selected legs, and everything but numeric records may optionally be filtered out.

FINITE STATE MACHINE

The primary task of the six finite state machines is to recover legs, that is, when a leg has reached the end of its stride, to reposition it in preparation for the next step. The gait algorithms, which will be discussed later, determine when one step

is finished and recovery should begin. The recovery process tries to place legs at a target chosen independently for each leg. At present, this target is fixed, but it might depend on drive pump displacement, for example. The target position is with the foot at ground level, the lower leg perpendicular to the upper leg, and the upper leg perpendicular to the body. That is, when the leg is in the target position, the upper leg sticks straight out, and the lower leg sticks straight down. In practice, because of the sluggish response times of the valves, legs generally overshoot the target, thereby lengthening the stride somewhat.

The driver can control the finite state machines by means of switches on the control stick. The trigger serves as a dead-man switch. When it is pulled, the finite state machines are permitted to run and to select valve settings for the legs. When the trigger is released, the finite state machines stop and all valves are forced into the *Hold* setting, so that the machine will not move. The display screen is normally blanked when the finite state machines are running because the terminal I/O necessary for continual display of the program's status represents a substantial drain on the microprocessor. This drain turns out to be fairly significant because the performance of the walking machine is determined to a large extent by how quickly the processor can use sensor information to control valve settings. For debugging purposes, the finite state machines will, however, display their status if the driver pushes the two-step switch on the control stick half way down.

The next several paragraphs describe the various states of the finite state machines. The general design philosophy was to keep the algorithm as simple as possible and not to consider cases that had not been demonstrated in the operation of the machine. Thus, rather than being designed to handle every imaginable eventuality, the code was written to cope only with situations that had actually been observed to occur. This approach has certainly contributed to the ease and relative speed with which the code was written and debugged.

A leg will be in the "energize" state when it is on the ground and expected to drive the machine forward or backwards. When a leg is in the "energize" state, its valve setting will be *Energize*, so that the action of the drive pump will result in smooth, horizontal leg motion. The machine is able to turn because the legs on its left and right sides are driven by two independent pumps. The "energize" state remains active until the gait program decides that it is appropriate to recover the leg, whereupon the gait program will force a change to the "lift" state.

The purpose of the "lift" state is to relieve the load being borne by a leg. Valves are placed into the *Lift* setting, which causes both of the upper leg hydraulic actuators to shorten and the leg to rise. The "lift" state remains active until the vertical force detected by the leg's sensors drops below some fixed threshold. The intent here is that so long as the leg is in contact with the ground, it will move generally upwards, but it will also be able to move forwards and backwards on the ground in response to the motion of the rest of the machine. When the leg is no longer in contact with the ground, the state will switch either to "upD" or "upE," depending upon the relative positions of the leg and the target. If, in order to reach the target, the leg must move toward the driver, the next state will be "upD." If it must move toward the engine, the next state will be "upE."

The "upD" and "upE" states lift the leg high off the ground and cause it to

move either forwards or backwards. Consider first the "upD" state, which moves the leg up and toward the driver. It uses the *Up* valve setting to shorten the hydraulic actuator at the driver end of the walking machine. The "upD" state is active until the driver end actuator has become short enough for the leg to be able reach the target position by lengthening only its engine end actuator. The program makes this test by determining whether the leg is above or below a line that passes through the target and is parallel to the leg's direction of travel when only the engine end actuator is moving. Also, the leg must have risen above some minimum height threshold before the "upD" state will terminate. Once the driver end actuator has shortened sufficiently, the "upD" state switches to "top." The "upE" state works in an analogous manner, using the *up* valve setting to shorten the engine end hydraulic actuator. It, too, switches to "top" when the engine end cylinder is sufficiently short.

The "top" state is degenerate in the sense that it makes no attempt to command any valves. Instead, it immediately selects one of two successor states on the basis of the lower leg position. If the lower leg is too far to the right of the target, the next state will be "topL," otherwise "topR."

The "topL" and "topR" states move the lower leg left or right in an attempt to bring it close to the sideways target position. "topL" moves the lower leg with the *Left* valve setting until the leg has traveled to the left side of the target. "topR" uses *Right* to move the leg right until it is on the right side of the target. Notice that this simple algorithm can cause the lower leg to move alternately left and right in successive walking machine steps. The problem could be corrected by including a dead zone about the target in which the leg moved neither left nor right, but this extra complication seemed unnecessary because the knee valves respond quickly enough to minimize the overshoot that would cause this oscillation. Any oscillation that does occur is not significant enough to disrupt the walking algorithm. Another thing to notice is that sideways motion of the lower leg occurs at the apex of the leg recovery motion. This prevents the scuffing that might otherwise occur if the leg were driven to a target while it was on the ground. The states following "topL" and "topR" are either "downD" or "downE," depending on whether the leg is recovering toward the driver or toward the engine. That is, if the leg was moved upward using the "upD" state, it will move down using the "downD" state. The proper direction was stored on entry to the "top" state, so that no further comparison of the target and leg positions is required.

The two states, "downD" and "downE," are analogous to the "upD" and "upE" states except that they move the leg down instead of up. "downD" uses the *Down* valve setting to move the leg down and toward the driver. "downE" moves down and toward the engine using the *down* setting. Downward motion ceases when a leg begins to detect more than some threshold of force, or when it has moved down too far. More precisely, the "downD" and "downE" states terminate when the following quantity becomes positive.

$$a(\text{Vertical Force} - \text{Force Threshold}) + b\,max(0, \text{Position Threshold} - \text{Vertical Position})$$

The next state for either recovery direction is "place." In the expression above, the magnitude of the force required to sense contact with the ground diminishes

linearly as the leg continues to drop below some fixed position threshold. The leg behaves as if the missing force were supplied by a spring attached to be bottom of the foot. After touching the ground, the spring would apply more and more force as it continued to be compressed. The position threshold controls the length of this simulated spring, while the constants a and b determine its strength.

"place" is the state that powers the legs into the ground in order to raise the body of the machine. Unless the foot was placed into a hole, "place" is not usually activated until the foot has reached the ground, so it uses the *Place* valve setting to push the leg generally down while permitting it to move forwards and backwards in response to the motion of the walking machine's other legs. The "place" state remains active until the leg has moved below some vertical position threshold, whereupon the state switches back to "energize" to move the leg forward and back under control of the drive pumps.

POST-PROCESSORS

The six finite state machines choose valve settings for each leg independently of the others. A collection of post-processing programs adjust these valve settings to satisfy constraints on groups of legs. Each processor is specialized to a specific task and is independent of the others.

The first post-processing program applied to the result of the finite state machine detects and resolves improper combinations of valve settings. Recall that some valve combinations either force oil into or suck oil out of a part of the hydraulic circuit known as the 'B'-line. This can happen when lower-case valve settings like *up* and *down* are mixed with upper-case valve settings like *Up* and *Down*. In order to resolve this difficulty, the post-processor first determines whether any of the six finite state machines have requested one of the lower-case settings. If not, there is no problem and the post-processor does not alter the requested set of valve settings. If, on the other hand, the finite state machine for some leg has requested a lower-case setting, the post processor will make the following conversions for all of the legs: *Place* to *place*, *Lift* to *lift*, *Up* to *Hold*, and *Down* to *Hold*. Thus, upper-case settings will be converted to the equivalent lower-case setting, or to *Hold* if there is no equivalent. The intent of this method is to eliminate the troublesome lower-case settings as quickly as possibly by diverting oil to them. Also, since *Lift* and *Place* have lower-case equivalents, if all of the legs are recovering in the same direction, it will not be necessary to switch any leg into the *Hold* setting.

The second post-processing step switches lower legs into the crab circuit, which distributes sideways forces between legs during turns, and which can be used to make the machine walk sideways. The post-processor examines each leg independently. If the hip valves are set to *Hold*, the lower leg post-processor assumes that the leg was meant to be frozen in place, and it likewise sets the knee valves to *Hold*. If the hip valves are not in *Hold*, the program examines the knee valve setting as produced by the finite state machine. If the knee valves are something besides *Hold*, the post-processor assumes that the finite state machine has them under control and does not modify the setting. This situation will occur when the finite state machine

is in the "topL" or "topR" state attempting to bring the lower leg close to its target position. Finally, if the knee valves are in *Hold*, the hip valves are not in *Hold*, and the leg senses more than some threshold of vertical force, the knee is switched into the *Crab* valve setting. Recall that the *Crab* setting causes the lower legs to be connected in parallel through the crab pump. Thus, if a leg that was not bearing any of the machine's weight were switched into the crab circuit along with some loaded legs, the unloaded leg would have no way to resist the sideways forces transmitted by the loaded legs, and the entire machine would slump to one side or the other. The crab post-processor prevents this undesirable situation, while allowing crab motion for sideways walking and turns.

The next post-processor checks for end of travel in the fore, aft motion of the legs. When legs on one side of the machine are being driven forwards and backwards, their hydraulic actuators are connected together in series so that each leg will move at the same speed. The walking machine can turn by driving the legs on its two sides independently at different rates. When one of the hydraulic cylinders in the drive series reaches the end of its travel, none of the other cylinders in the series is able to move, and any further attempt to drive them will cause oil to flow through a relief valve. To prevent this from happening, the post-processor searches for legs with their valves set to *Energize* that have also reached the end of their travel. Note that in order to do so, it must consider not only the position of a leg, but also the direction in which the leg is being moved by the corresponding drive pump. If it finds a leg that has reached its end of travel, it switches that leg into *Hold*, along with any other legs on the same side of the machine that are in the *Energize* setting.

Another post-processor controls whether driving will occur at all. When the driver pushes the coolie hat upwards, the program sets a flag enabling the *Energize* valve setting. When he pushes down on the coolie hat, the flag is cleared to disable *Energize*. A post-processor serves to implement this capability. If the energize flag is clear, the post-processor will change any *Energize* valve settings to *Hold*. This feature is useful for disabling energize when walking sideways, for example, so that the driver need not be concerned with the exact position of the control stick.

A final post-processing step prevents the machine from stamping its feet. This potential problem shows up when hydraulic actuators are connected in parallel, as they are in the crab circuit. Recall that when a leg is moving up or down during recovery, the hip actuators of all recovering legs are connected together in a single parallel circuit. This means that if any one of those legs is bearing weight on the ground, and another one is in the air, oil will flow rapidly from the grounded leg to the raised leg until both bear an equal share of the total load. This flow of oil causes the raised leg to stamp the ground rather violently. To eliminate this undesirable action, the final post-processor first determines whether any leg sensing more than some threshold of vertical force is also in one of the parallel actuator settings: *Place*, *Lift*, *Up*, *Down*, *place*, *lift*, *up*, or *down*. If there aren't any such legs, then there is no potential difficulty and the post-processor can terminate. On the other hand, if such a leg does exist, the post-processor must switch any parallel-connected leg not bearing weight into the *Hold* setting to eliminate the possibility of foot stamping. This technique is based on the assumption that load-bearing legs will soon either cease to bear load or be switched into some other valve setting.

There is one monitoring routine that, while not strictly a post-processor, does operate pretty much independently of the rest of the recovery algorithm. It monitors position of the compensator, adjusting it as required to insure that oil need never be driven through relief valves unnecessarily. Walking on the ground tends to drive the compensator in, and when it gets too far in, the monitoring routine freezes all of the legs and extends the compensator. The fact that the machine comes to a hold is moderately annoying to the driver, but it is tolerated as a relatively infrequent occurrence.

Another monitoring routine watches for leg collisions. The three legs on each side of the machine are close enough together that it is possible for a middle leg to interfere with one of the end legs. This is something to be avoided, since mechanical damage could result. The leg collision detector keeps track of leg positions, and if it detects an impending collision it will abort the walking program, freezing the legs as they are. When this happens, the driver can use the manual control program to disentangle the offending legs. The collision detection program is written to be fairly paranoid. When position sensors are redundant, instead of using the averaged reading, as does the rest of the walking code, it selects the worse case of the two redundant readings. In practice, the frequency of leg collisions detected by the program is fairly closely related to the quality of the calibration tables. If the calibration is stale, spurious leg collisions occur with annoying frequency. With fresh calibration data, on the other hand, detected leg collisions are acceptably rare.

GAIT ALGORITHMS

The six finite state machines and the set of post-processing routines control the motion of legs during recovery, but it is the task of the gait programs to determine when a leg should be recovered. The driver can choose from among the available gait algorithms by first activating the gait selection mode from the main menu. When this mode is active, pressing a button on the phone dial while holding down the trigger determines which of the available gaits will be used for walking. The display shows which gait has been selected.

Besides the gait algorithm, the user must specify a sequence of legs. The exact meaning of this leg sequence is gait-dependent, but it is generally interpreted to be the order in which the legs will either be recovered or considered for recovery. Once again, the driver specifies a leg sequence by means of the phone buttons. There are six legs in the sequence, and new legs are shifted into the sequence at the end, causing the leg formerly at the beginning of the sequence to be dropped. The driver can append a leg to the sequence by pressing the corresponding digit on the phone dial. The current leg sequence is displayed on the terminal for reference.

Three of the gaits available to the driver cause the walking machine to recover its legs in a known, fixed sequence according to the contents of the leg sequence table. These three gaits recover either one, two, or three legs at a time, and each waits until no leg is recovering before it permits the next set of legs to recover. The three fixed gaits cause legs to recover strictly in the order that they appear

in the leg sequence table. When the algorithm encounters the end of the table, it simply wraps around to the beginning again. For example, the two-legged fixed gait initially recovers the first two legs mentioned in the sequence table. When these have completed recovery, it causes the legs specified as the third and fourth entries in the table to recover. Finally, the fifth and sixth legs in the leg sequence table will recover, whereupon the algorithm will start over at the beginning of the table.

Each of the three fixed gaits is capable of making the machine walk, but they have somewhat different properties. The one-leg-at-a-time gait works pretty much equally well with any leg sequence that includes every leg because there will always be at least five legs on the ground to support the weight of the machine. Unfortunately, though, the machine makes only painfully slow progress when recovering just a single leg at a time. The two-legged fixed gait is a bit faster, but it requires the driver to exercise some care when selecting the leg sequencing. In particular, if the two legs at either the engine end or the driver end are recovered at the same time, the temporarily unsupported end of the machine will drop to the ground. Similarly, lifting the middle and an end leg on one side of the machine is probably not a good idea. Also, the legs remaining on the ground will be in a better position to bear the weight of the machine if they recover in order from back to front so that recovering legs will be placed near to legs already on the ground. One leg sequence that satisfies these constraints is 152436. Notice that although this sequence satisfies the back to front propagation constraint when the machine is walking in the direction of the engine, it does not do so when the drive pumps are reversed. Another leg sequence, 135246, results in a kind of two-legged tripod gait that satisfies both constraints regardless of the direction of travel. The same sequence, 135246, used with a three-legged fixed gait results in a true tripod gait. Besides satisfying the support constraints, it is successful because it moves the maximal number of legs at a time. Indeed, some of the walking machine's best performances have been controlled by this simple tripod gait.

One problem shared by all of the fixed gaits is that legs recover on schedule whether they need to or not. That is, legs will recover even if the body of the walking machine isn't moving forward or backward. The design of the algorithms to control leg motion during recovery, combined with the sluggish valve response times, can cause legs to recover back and forth about the target. It is particularly annoying when a leg recovers in the wrong direction. A second set of three fixed gaits, analogous to the original three, eliminates this problem by recovering a leg only when it is moving away from the target. That is, any leg that comes up in the fixed recovery sequence loses its turn if it is not moving away from the target position. The gait algorithm determines a leg's direction of travel by examining the sign of the corresponding drive pump displacement. If the driver has used the coolie hat to manually disable the drive motion, the gait algorithm will always permit a leg to recover when it is encountered in the leg sequence. This behavior was included so that sideways walking would still be possible. This modified fixed sequencing algorithm shares the advantages of the original algorithm while preventing legs from recovering in the wrong direction.

Another gait, which is a variation of the fixed gaits described above, sequences legs on the left side of the machine pretty much independently of those on the right

side. That is, there is a separate but fixed leg sequence for each side of the machine, and the gait algorithm selects one leg from each sequence for recovery. The first half of the leg sequence table applies to the 123-side of the machine, and the second half applies to the 456-side. When a leg finishes recovering on, for instance, the left side of the machine, the gait algorithm selects the next leg in the left side sequence without regard for whether the leg on the right side has finished recovering yet. This selection is subject to the constraint that neither the front pair nor the back pair of legs should be recovered simultaneously. One additional complication is that the order in which legs are chosen from their respective sequence tables is dependent on the sign of the drive pump displacement for the corresponding side of the machine. When the machine is driven backwards, the leg sequencing will be reversed. If supplied with an appropriate choice of leg sequence, like 123654, this two-legged gait can effectively support the machine in any direction of motion, including turns.

The gaits described so far have all used the leg sequence table to specify the exact order in which legs will be recovered. The driver may also choose gaits in which the order is determined dynamically. The first such gait recovers just a single leg at a time. The algorithm identifies the leg most in need of recovery, that is, the one that is closest to the end of its travel. If the leg position exceeds some fixed threshold, then the algorithm will cause the leg to recover. The set of legs considered for recovery are those that appear in the leg sequence table. Like almost any single legged gait, this algorithm is capable of making the machine walk, but only very slowly. It works marginally better in turns than a fixed sequence gait because the legs on the outside of the turn are permitted to recover more frequently than those on the inside.

Another gait in which the leg recovery order is determined dynamically may recover up to three legs at once. This algorithm works by continually considering the legs that appear in the leg sequence table but are not currently recovering. It removes a leg from consideration if either of the legs adjacent around the perimeter of the machine is recovering. From among the remaining legs, it selects the single leg most in need of recovery, again by determining which leg is most near its end of travel. If the position of that single leg exceeds some threshold, the leg will begin to recover. This gait tends to resemble a tripod gait, and it works fairly well. The major difficulty is that because legs do not recover in a fixed pattern, the driver cannot always predict when legs will recover, and the program is consequently a bit harder to drive than the plain tripod gait.

In addition to merely being able to select a gait algorithm and leg sequence, the driver can use the thumb switch on the control stick to determine whether legs will recover automatically. When the thumb switch is depressed, the selected gait algorithm will be active, and legs will begin to recover. If the thumb switch is not depressed, legs already recovering will finish doing so, but no new legs will begin to recover. This switch turns out to be quite useful for keeping the machine under control. Fairly often, the driver will momentarily press the thumb switch to release a set of legs, and then wait until all of those legs have completed recovery before pressing the switch again. Operating the machine in this manner allows the driver to separate the control of drive and recovery. When this separation of control is

not necessary, the driver can hold down the thumb switch to enable continuous recovery.

The buttons on the phone dial permit still further control over recovery. Pressing the asterisk forces the finite state machines for all six legs into an initial state that will lift the machine up to walking elevation. Button number 7 performs a similar action, except that only the four corner legs are used to elevate the machine, and the center legs remain frozen. This feature, together with an appropriate combination of values in the leg sequence table, can be used to experiment with four-legged walking. Finally, the driver can force a leg to begin recovery at any time by pressing its corresponding button on the phone dial. This works regardless of gait or whether the thumb switch on the control stick is pressed. Being able to manually recover legs has been quite helpful in several situations. First, it is often useful during testing and debugging to be able to force a leg to recover on demand. Second, since repeated recovery of a single leg will bring it closer and closer to the target position, manual recovery can serve to neaten the leg configuration. Third, when the machine first begins to walk, the lower legs are sometimes at one or the other extreme of their motion. Manually recovering such a leg straightens out the lower leg, putting it in a better position to bear load. Fourth, sometimes the walking algorithms get into trouble and need help. When this happens, the driver can spot an offending leg and force it to recover in order to eliminate the difficulty.

OWL WALKING ALGORITHMS

The structure of the nervous system of insects has been the guide for the design of algorithms for walking in this research. Insects have a single ganglion associated with each leg and this ganglion is responsible for the bulk of the behavior of the leg. There is communication between legs, but it is responsible for only a small amount of the control of walking.

Each leg of an insect exhibits steady repetitive behavior. The foot is placed on the surface, driven rearward until it is as far back as it can go, removed from the surface, and recovered to the forward position, where the process repeats. Behavior of this sort is not sufficient by itself to ensure walking by the insect, there are phase relationships between the legs that are functions of the speed of walking and the radius of turning. These phase relationships are not rigidly fixed, but are rather 'encouraged' by simple communication between the legs. There are two purposes to this communication. The first is to ensure that there are enough legs in contact with the walking surface to permit the insect to maintain its stance over the surface. The second is to encourage rear-to-front waves of leg recoveries.

All walking animals with four, six, or more legs exhibit rear-to-front waves of recovery. In walking, the rearmost leg is picked up and moved forward and placed on the walking surface, usually right behind the footprint of the next leg forward. Once the rearward leg is in contact, then the forward leg is recovered and so on. There are several desirable properties of this pattern of leg motions. The most important is that it permits the animal to walk on insecure terrain without having to have an elaborate vision system and the memory to maintain a map of the terrain. The front pair of legs can be placed carefully; after that each of the trailing legs is placed as near as possible in the same safe place. This behavior is true of both six legged and four legged animals when walking. Dynamic gaits like trotting and running in quadrupeds don't have this property; however, quadrupeds have sophisticated vision systems to help them avoid obstacles and difficult terrain when they are moving fast. There is substantial evidence that six legged animals do not use dynamically stable gaits.

The physiology literature is a bit vague about the details of the communication among legs, but we have found that only two low bandwidth channels are needed from each leg to each of its two or three neighbors. With one channel the leg transmits the fact that it is recovering and that its neighboring legs should be discouraged from initiating recovery. We call this information the *inhibition* signal. With the second channel the leg transmits to its forward neighbor the fact that it has completed recovering and that the forward leg should be encouraged to recover. This signal is called the *excitation* signal. In addition, the rear pair of legs transmit excitation to one another, with each leg sending the excitation when it has reached about the midpoint of the drive stroke.

The excitation provides the rear-to-front waves of leg recoveries and keeps the two sides approximately 180 degrees out of phase. The inhibition is responsible for maintaining a reasonable stance.

Following this concept, the main part of the program that is used to control walking is composed of six independent processes. Each process is a loop in which the leg is first loaded, then driven rearward, then unloaded, and finally recovered to its most forward position. Thus, each process imitates the behavior of a single leg. When a leg is about to begin recovering, it adds a constant to the inhibitions of each of its neighbors. When it has finished recovering, it subtracts this constant from the inhibitions. Also at the end of recovery, it sets the forward neighbor's excitation to a different constant. When a leg is about to begin recovering, it sets its own excitation back to zero.

A leg decides when to end its drive phase and begin recovering based on the inhibition and excitation signals and on the load on the leg. If the load on the leg is less than a computed threshold, then the leg is considered to be available to be recovered. Inhibition decreases the threshold, thus reducing the likelihood that the leg will be lightly enough loaded to recover. Excitation increases the threshold, hence increasing the likelihood that the leg will be willing to recover.

There is a separate process for each phase of the leg cycle and these processes are activated one after another for each leg. When the walking program is started up, all the cycles are started in the 'Load' phase. The load process takes its leg and moves it toward the ground using the 'P' or place valve setting. The load process decides that it is done when the combination of load on the leg and vertical position of the upper leg segment indicate that a satisfactory stance has been achieved. This is done by each leg independently. One consequence of this behavior by the load processes is that there is no special initialization code required. The walking program can take the machine with the legs in any configuration and get into a good stance. The same feedback mechanism that is used to maintain a good stance once the machine is walking also serves to get the machine properly oriented at the beginning.

After the load phase is complete, the 'Drive' phase is initiated. In this phase the leg valve is set to the Energize, or 'E,' setting. If the driver releases the trigger on the joystick handle, then the drive process will place the leg valve in the Hold, or 'H,' setting. This 'dead man' mechanism was intended as a safety provision, but it is also used in practice by the driver to regulate the forward motion. The termination of the Drive phase is, as described in an earlier paragraph, when a load on the leg is less than a function of the inhibition and excitation signals for that leg.

After the drive phase terminates, the 'Unload' phase is initiated. In this phase the upper leg valve is put in the 'L' or lift state and the leg moves upward until the load on it is removed.

Once the unload phase is complete, the 'Recover' phase begins. The motion of this phase is somewhat complex, but basically the leg is moved first upward and forward by shortening the forward of the two upper cylinders, and then downward and forward by lengthening the rearward of the cylinders. The recovery phase is complete when the leg is near the target position or the leg has hit the ground and

begun to take up load. The end of the recover phase causes the beginning of the load phase for the next step.

Other independent processes in the walking program handle the control of the lower leg, or knee, joint as the leg cycles through the various phases of stepping. The knee control process has a particularly simple algorithm. When the foot is on the ground and bearing weight, as reported by the load sensors, then the knee joint actuator is put into the 'C' or 'crab' state. In this state the knee actuator is connected passively to all of the other similarly connected knees and to the crab pump. This passive connection permits the knee to flex as the leg moves, thus permitting the leg to conform both to the immobility of the ground and to the circular path of the knee about the hip joint. Whenever the foot is not bearing weight and the knee is too far from the center, quiescent position, then it is moved toward the center using 'Right' or 'Left' knee commands. When the leg is near enough to the center position, then it is left in the hold setting.

Beyond the processes directly concerned with the control of legs, there are independent processes that are responsible for communicating with the service processor. One set of processes copies all the sensor values from the service processor each time they are updated. Another process transmits the vector of valve commands from the control processor to the service processor every time they are updated. Other processes keep track of some hydraulic constraints that are imposed by the parallel connections made possible by the design of the walking machine. There are a collection of data gathering processes that are used to monitor the state of the machine during walking. They store data in the uninitialized region of RAM that sits after the walking program. The data space there is suitable to hold about ten minutes of detailed data.

We have conducted various experiments in which we varied the relative importance of excitation and inhibition. In our earliest tests, we set the inhibition and excitation constants to zero, thus requiring that each recovery be initiated by hand. The program is so written that pressing the leg's button on the keypad on the walking machine overrides the threshold test and forces the leg to proceed from drive to recovery. Running in this mode we were able to get the code controlling an individual leg debugged and running.

In the next set of tests we enabled the rear-to-front waves of excitation, but disabled the inhibition mechanism and the rear pair cross excitation. We were thus able to demonstrate the forward progress on the ground with only a minimum of operator intervention.

Later experiments with both waves and inhibition turned on provided superior quality motion because the inhibition constraints reduced the occurrence of simultaneous recovery of adjacent legs and the consequent tipping of the machine.

The best walking was produced by runs in which both inhibition and the wave and cross excitation mechanisms were all turned on.

An interesting result was achieved when walking was attempted with only the inhibition mechanism enabled. The machine walked quite well. On some occasions it walked more smoothly than with excitation turned on, on others it walked less smoothly. The details of the walking behavior are different in the two modes of walking. The walking with excitation turned on seems to be closer in detailed struc-

ture to that exhibited by insects. There is better minimization of total 'footprint' size for the excitation produced walking and that should be important in difficult terrain.

The parking lot in which we ran all of our tests has several sizable depressions where drains are located. The machine negotiated these obstacles with no difficulty. It was clear when near these dips that the stance was related to the local terrain, not to some horizon or other global feature.

DATA COLLECTION

In support of his PhD thesis research, Marc Donner has implemented a facility for recording data during the operation of the walking machine. We have used this facility to detect faulty sensors, measure the performance of hydraulic circuits, and monitor the behavior of walking programs. This section is intended to be an exhibition of the kinds of data that we have collected. Of course, it would be beyond the scope of this report to include all of the data that we have recorded, but additional data together with a more detailed analysis will appear in Donner's PhD dissertation.

The recording program is written in OWL and runs in the OWL microprocessor. Because it is an OWL process, it can monitor not only the operation of the walking machine microprocessor, but also the operation of other OWL processes. The program periodically samples sensor readings, valve settings, and program states. It stores the samples within a large buffer in the OWL processor's memory until the recording session is complete, when it uses a terminal line to transmit the data to a host computer for display and long-term storage. The processor has enough storage to hold about five minute's worth of samples, which take about 20 minutes to transmit up to the host.

DATA FOR DEBUGGING HARDWARE

Some of the earliest data that we collected were intended merely to help debug the sensor systems of the walking machine. Recall that in a leg of the walking machine, motion in each degree of freedom is measured by a redundant pair of sensors. During normal operation of the machine, the two redundant sensor readings are averaged, but for testing purposes, data from the two sensors were collected independently. Thus, it is possible to assess the operation of any particular sensor by checking it against its counterpart.

Figure 1 illustrates how redundant sensor values may be compared graphically. This particular example shows readings obtained from the Y and Z axis sensors of Leg 2. Readings from one of the redundant sensors are plotted against the corresponding readings from the other redundant sensor, and successive sample points are connected by lines. Since each of the two sensors is measuring the same physical motion, they should always produce duplicate values, and therefore the plotted points should all be colinear. Unfortunately, the reality of the situation is somewhat less than perfect, however, as is evident from Figure 1. The Z sensor exhibits some non-linear behavior near the origin.

Figures 2 show data taken as Leg 2 was being moved in a cyclical fashion. The vertical leg position has been plotted along the vertical axis, and likewise for the

horizontal position. It is readily apparent from this plot that Sensor 1 is malfunctioning. The hysteresis in the curve suggests that the sensing potentiometer might be loose in its mounting. By examining these curves, we have also been able to detect cracks in the potentiometer substrate. Before these plots were available, we found it very difficult to track down broken sensors. Figures 3 and 4 show data taken from the same leg and same motion axes after repairs had been made.

Another kind of plot is useful for measuring the time lag between when the program commands a leg to move and when the leg begins to respond. Figure 5 plots the position of a hip joint against time. The square wave drives the valve that controls the motion of the joint. It is readily apparent that motion in the Z direction begins sometime after the leg is commanded to move, and motion stops sometime after the command is withdrawn. We believe that most of this delay is caused by the finite response time of the valves. It takes time for the current in the valve control solenoid to build up, causing a similar delay in the motion of the valve spindle.

To help maintain a machine of this kind, we recommend that a direct measurement of the valve spool position be provided. Such a measurement would, for example, provide direct evidence of a stuck valve. A simple switch connected to the valve spool could easily provide this information. We looked for some effect of spool motion on the coil EMF, but were unable to reliably detect any.

DATA FOR ANALYZING WALKING

Besides aiding in sensor debugging, plots derived from collected data are useful for analyzing and illustrating the behavior of the machine while it is walking. The rest of the plots shown in this report reflect data taken during walking experiments.

One of the more straightforward plots illustrates the distribution of vertical ground forces as sensed by the load cells in each leg. In Figure 6, force on each leg is plotted against time, and Figure 7 shows the sum of the forces on various legs. The top curve in Figure 7 is the sum of the forces on all six legs. As one might expect, the sum of the loads remains relatively constant over time. The departures from constant load can be explained by the fact that the leg forces are sampled in sequence, not simultaneously. For example, if one leg were in the process of picking up load from another, the sampling process might happen to catch the first leg while it was still loaded and the second leg after it had assumed its load. In this case, the sum would appear too large.

Figure 8 shows the settings of operator controls during the course of a walk in the parking lot. There are curves representing, from top to bottom, the two axes of the stick, which control the displacements of the two drive pumps, and the right foot pedal, which controls the recovery pump. The setting of the crab pump is not shown. It is interesting to note that the driver is actually exercising quite a bit of control while the machine is walking. In particular, we have found that drivers learn to use their control over the recovery pump quite a bit in order to smooth out the ride. Large recovery pump settings are needed to drive feet into the ground as they assume load, but smaller settings are more suitable for the motion of elevated

legs during the recovery phase.

Figure 9 shows the "progress" of the machine as it walks. This plot is the result of a "dead reckoning" computation based on the motion of legs that are bearing weight. In order to compute the "progress number," the length of the load-bearing stroke of each leg is first calculated from the raw position data. Next, the strokes of legs on the same side of the machine are averaged, since they work together to propel the machine forward. Finally, the left and right side progress numbers are merged to form the progress value. Obviously, the sum of the left and right progress numbers says something about the distance traveled over the ground, and the difference of the two numbers says something about the orientation of the vehicle. We have not yet made plots of the heading of the vehicle, nor of its absolute position in space based on these computations, but only of its "forward progress."

The final and perhaps the most interesting plot is shown in Figure 10. In this plot, the state of each leg has been separated into a "load bearing" and an "elevated" case, and these states have been plotted against time for each of the legs. The separation is based on a cut-off value for the sensed leg loading, but similar plots based on the intent of the control program show similar patterns of activity. The form of the "gait" being used is quite clear in Figure 10. One can see waves of activity progressing from the rear legs, 3 and 4 in this case, toward the front legs, 1 and 6 in this case. We expect a more detailed analysis of this kind of plot to be an important part of the dissertation soon to be published by Marc Donner.

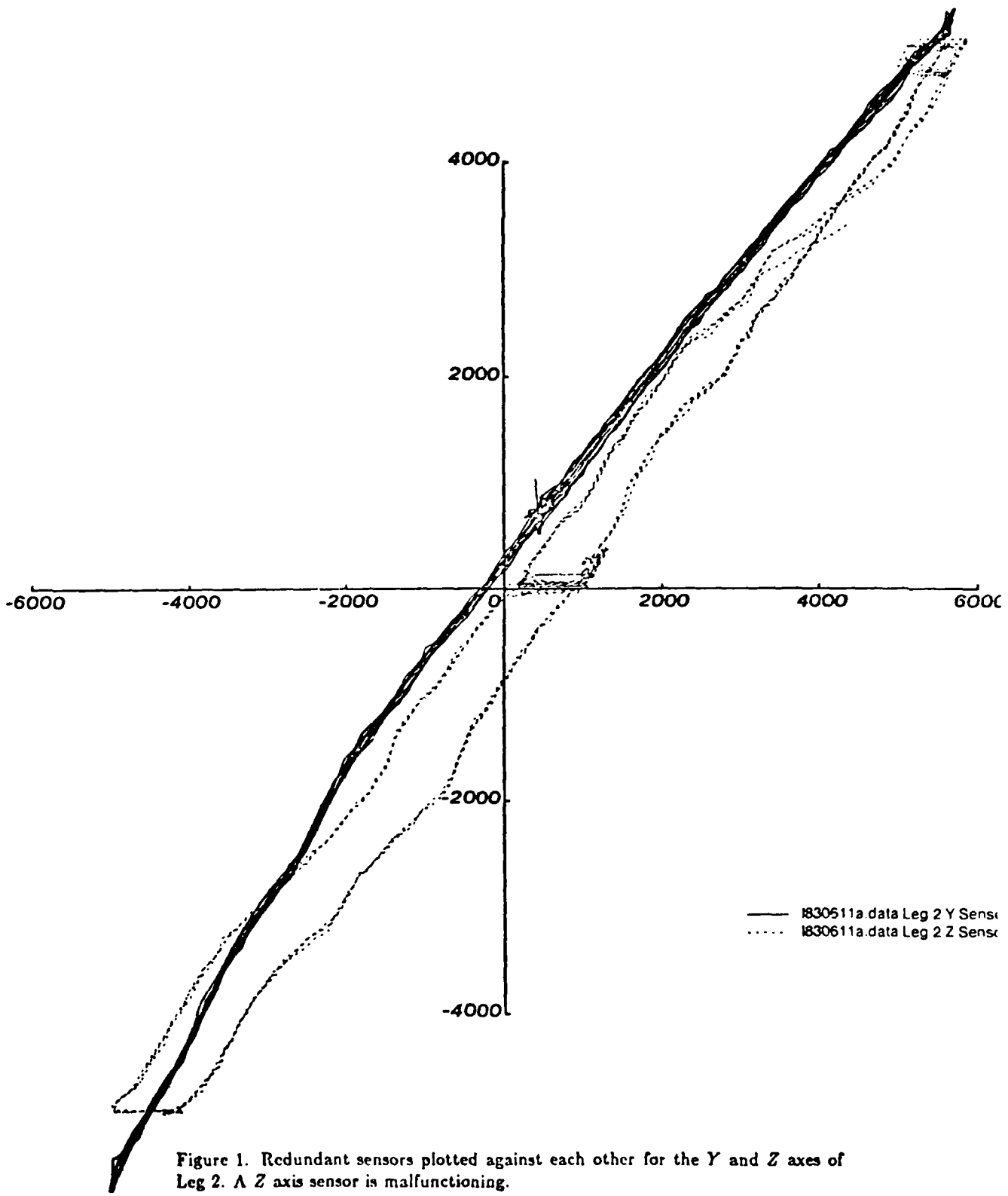


Figure 1. Redundant sensors plotted against each other for the Y and Z axes of Leg 2. A Z axis sensor is malfunctioning.

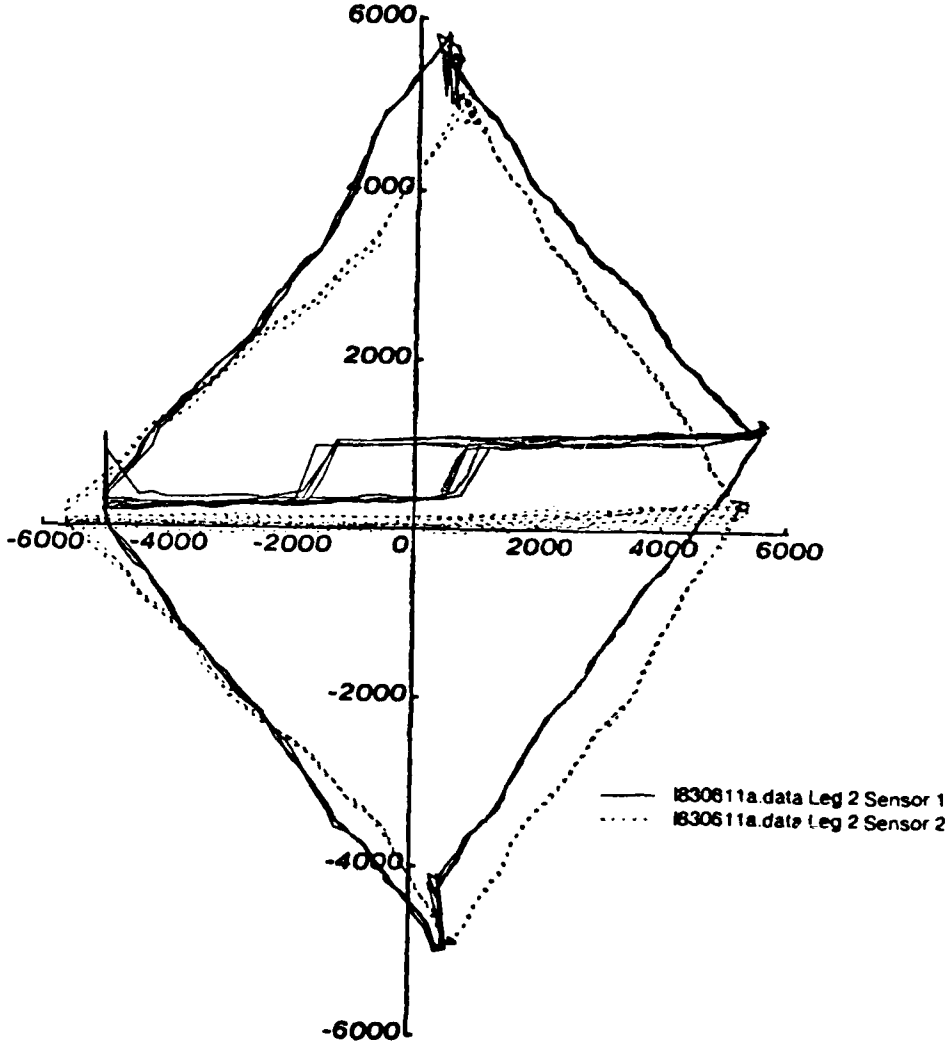


Figure 2. *Y* and *Z* axis sensors plotted against each other for Leg 2. The redundant curve corresponds to the redundant pair of sensors. Sensor 1 is malfunctioning.

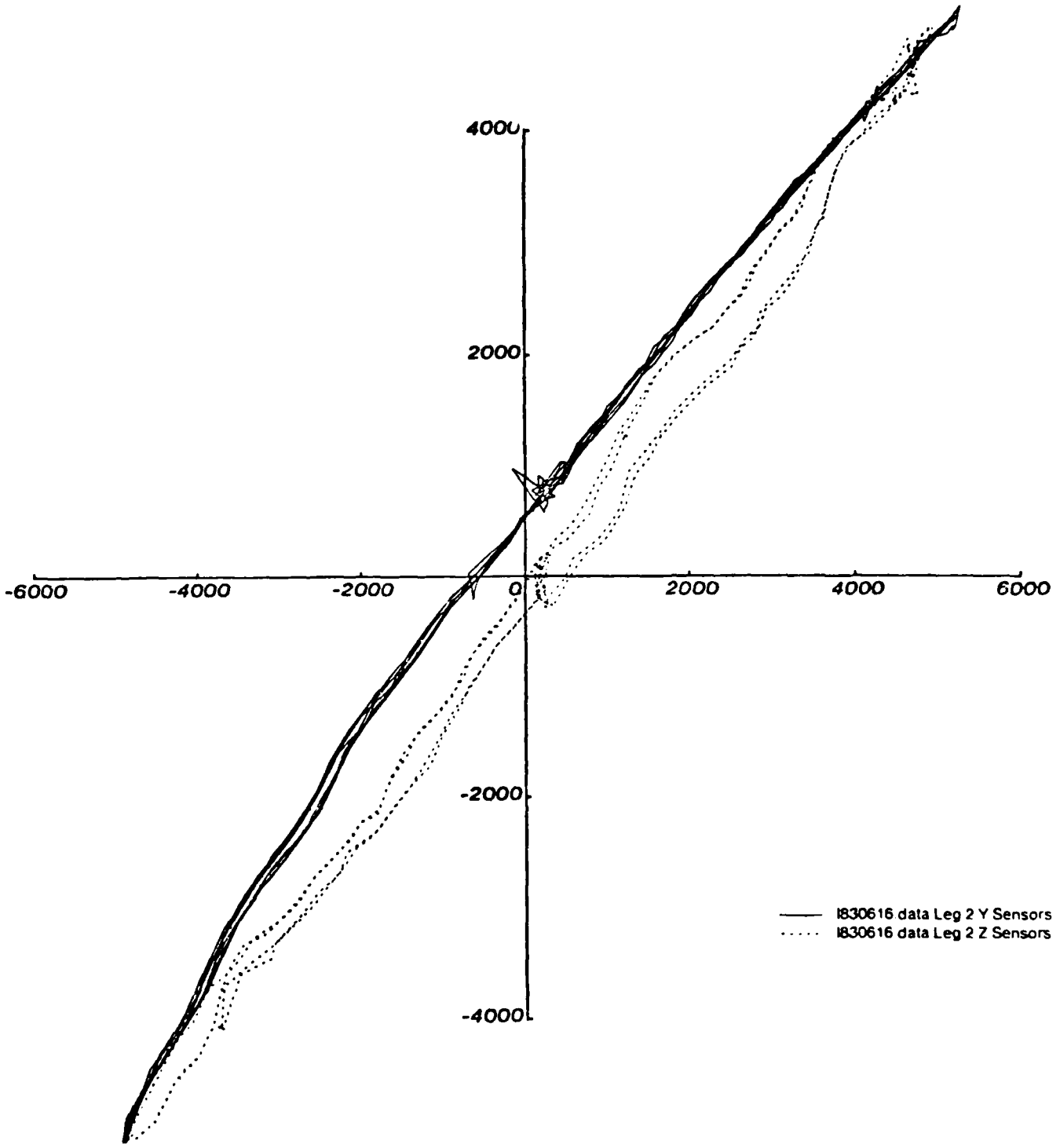


Figure 3. Redundant sensors plotted against each other for the Y and Z axes of Leg 2 after repairs have been made.

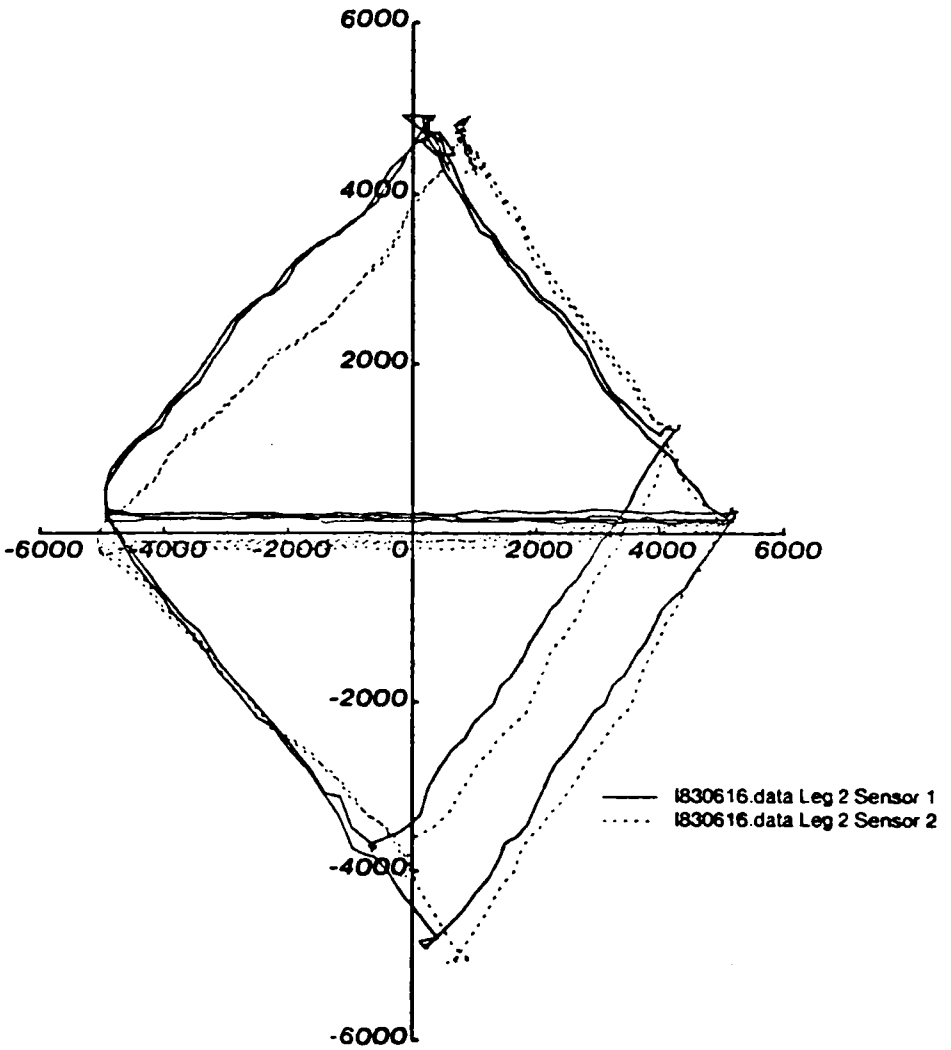


Figure 4. Y and Z axis sensors plotted against each other for Leg 2 after repairs have been made.

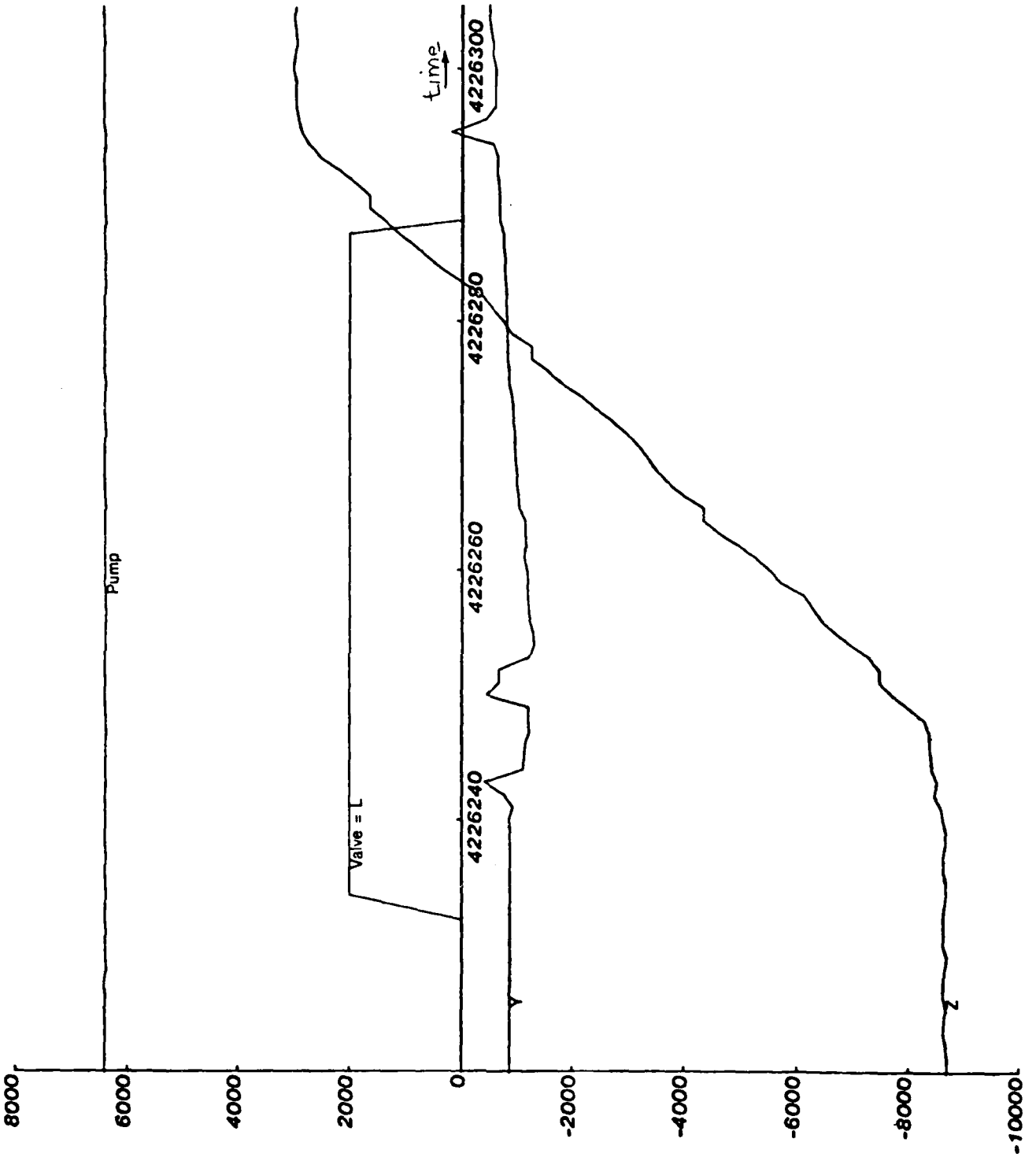


Figure 5. Hip position and valve setting plotted against time to estimate the valve response.

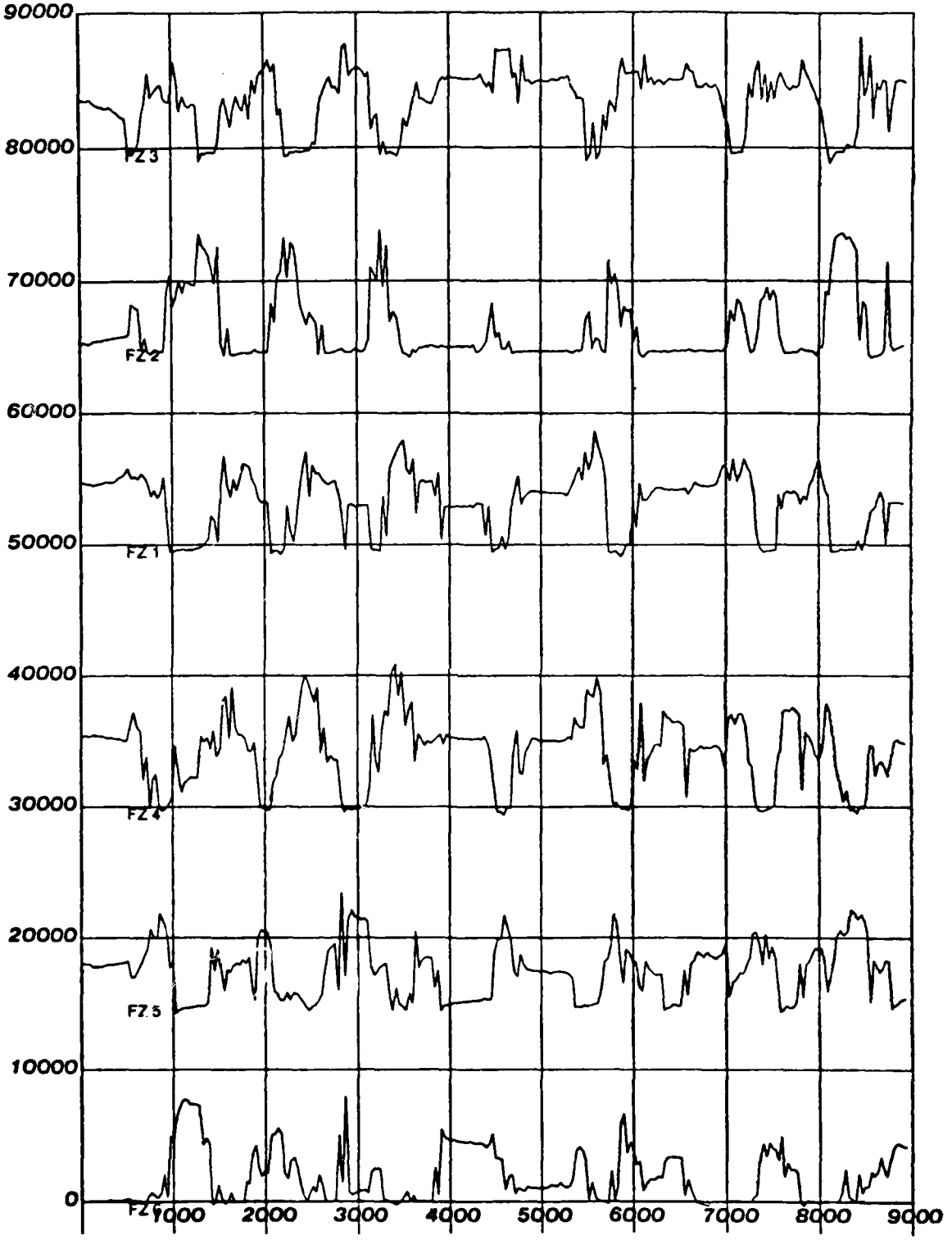


Figure 6. Vertical forces plotted against time for all six legs.

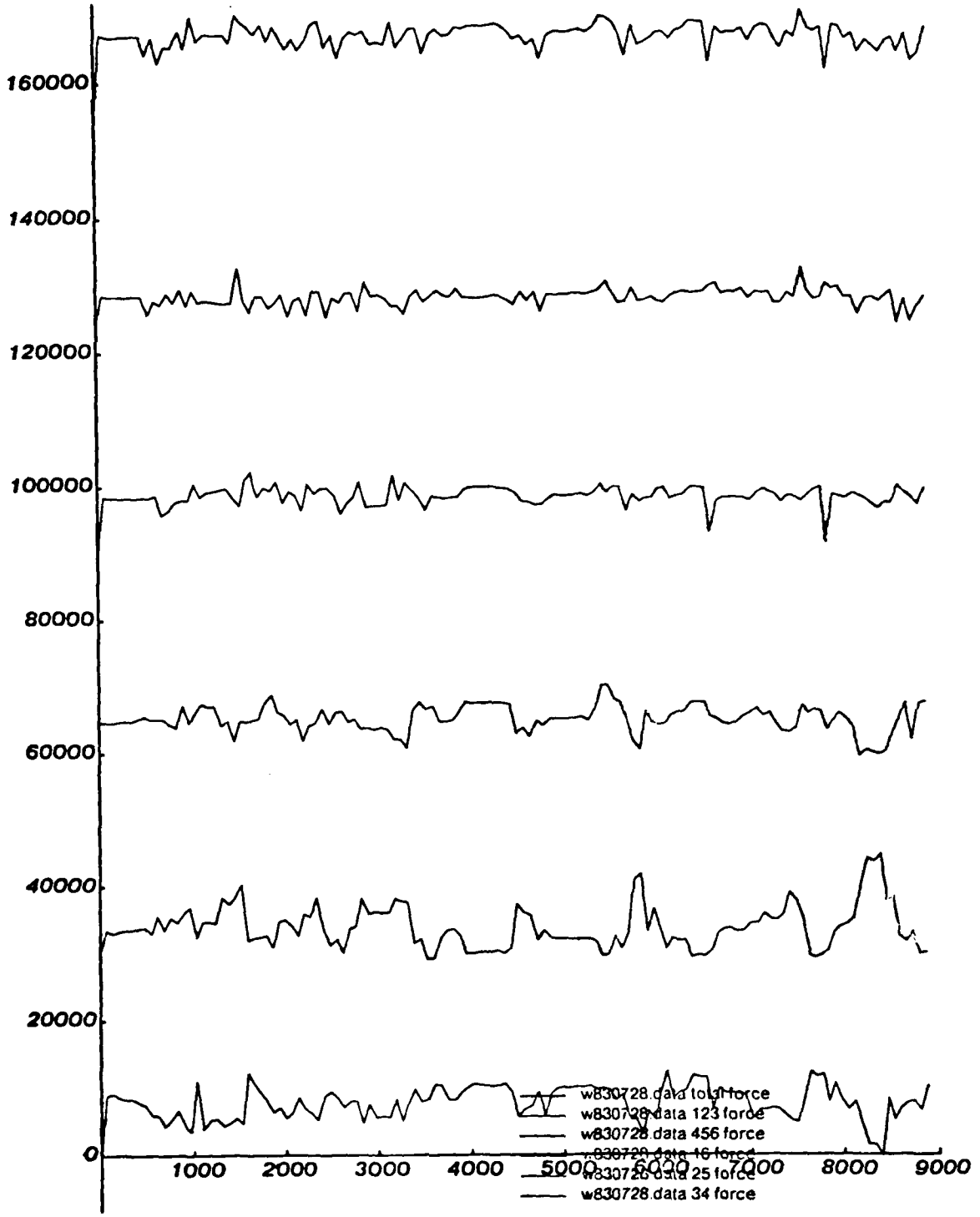


Figure 7. Sums of vertical leg forces plotted against time. Top to bottom: sum of all six legs, sum of legs 123, sum of legs 456, 16, 25, 34.

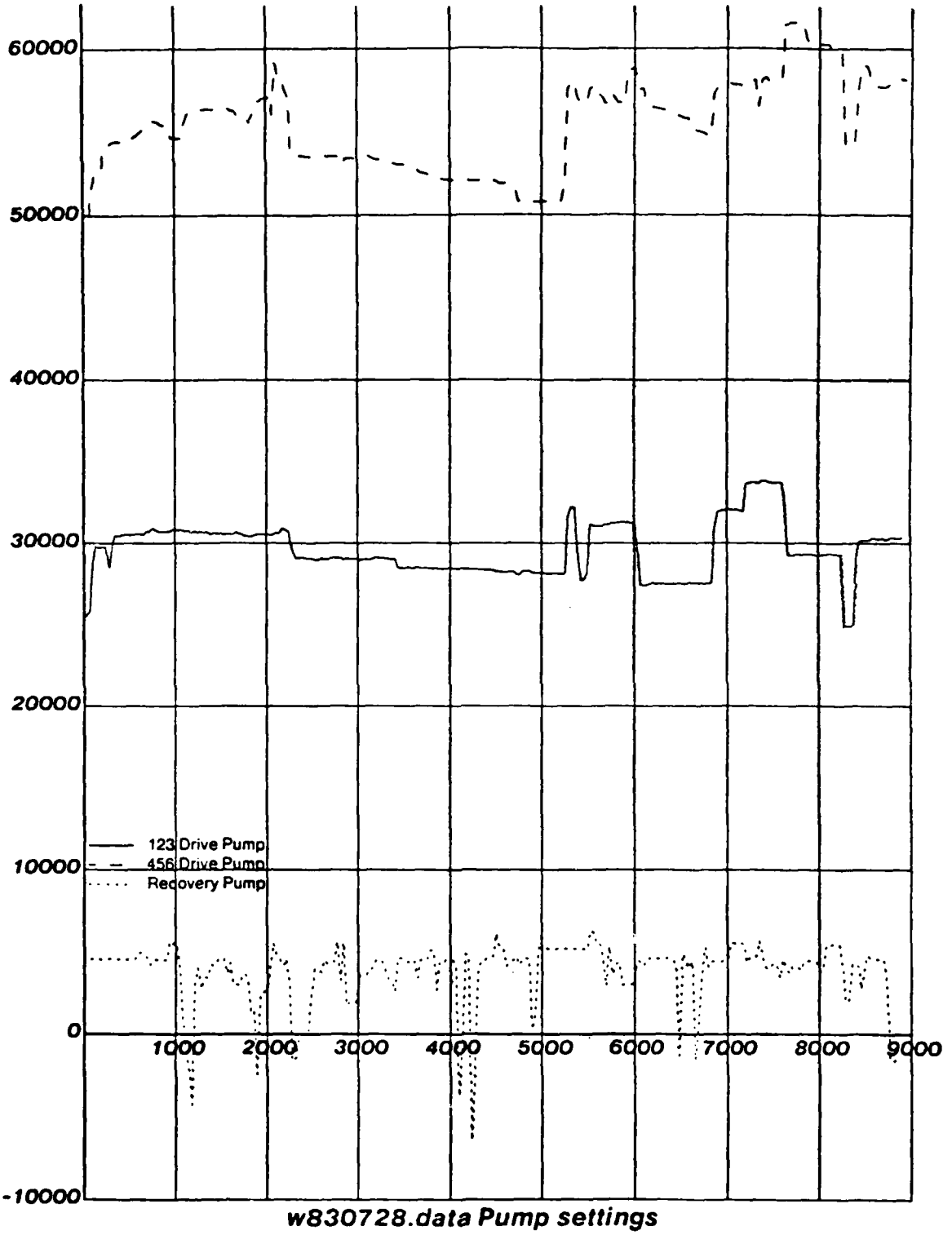


Figure 8. Driver control settings plotted against time. Top to bottom: 456-side drive pump setting, 123-side drive pump, recovery pump.

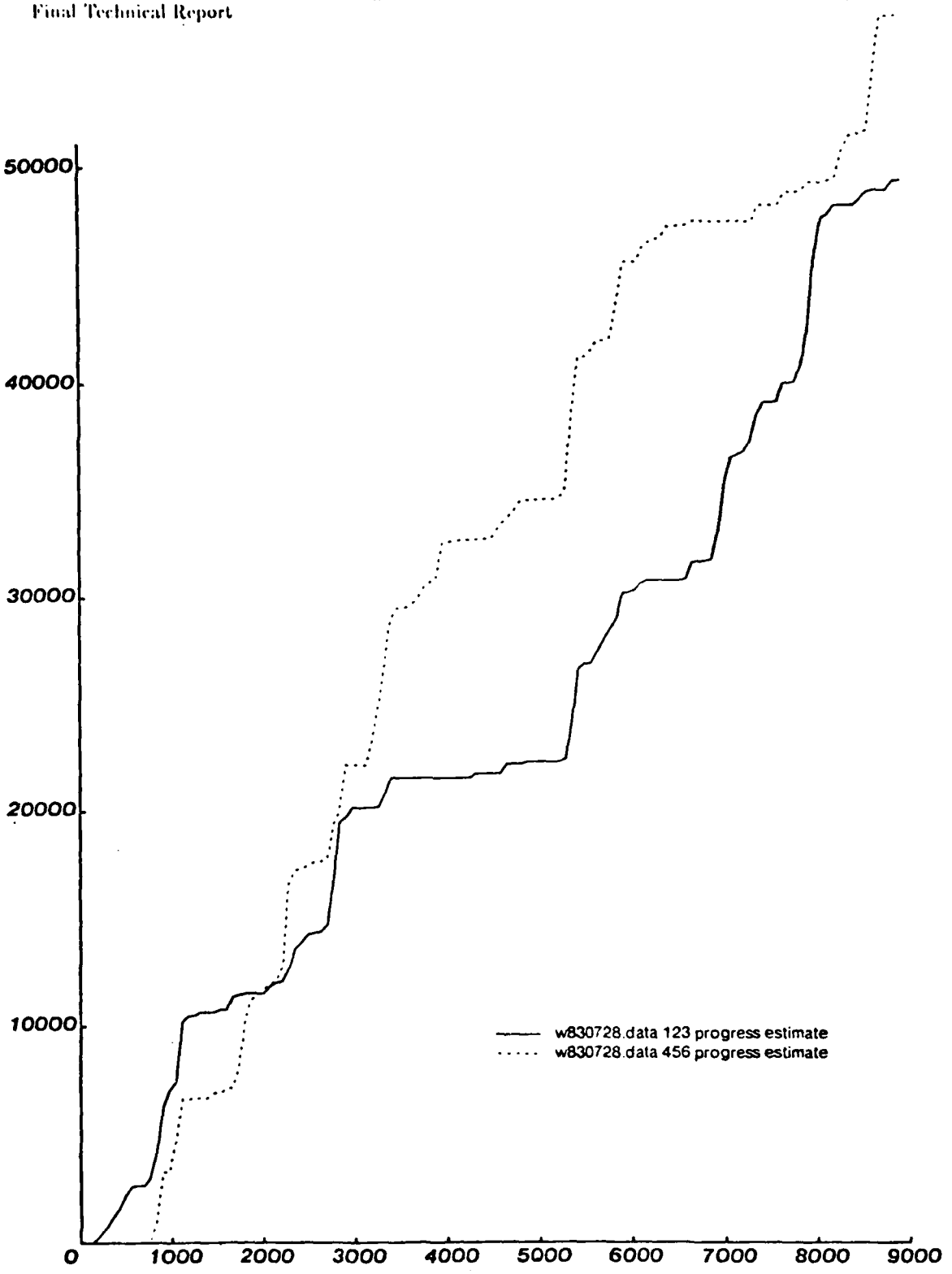


Figure 9. Progress estimates plotted against time for the two sides of the machine.

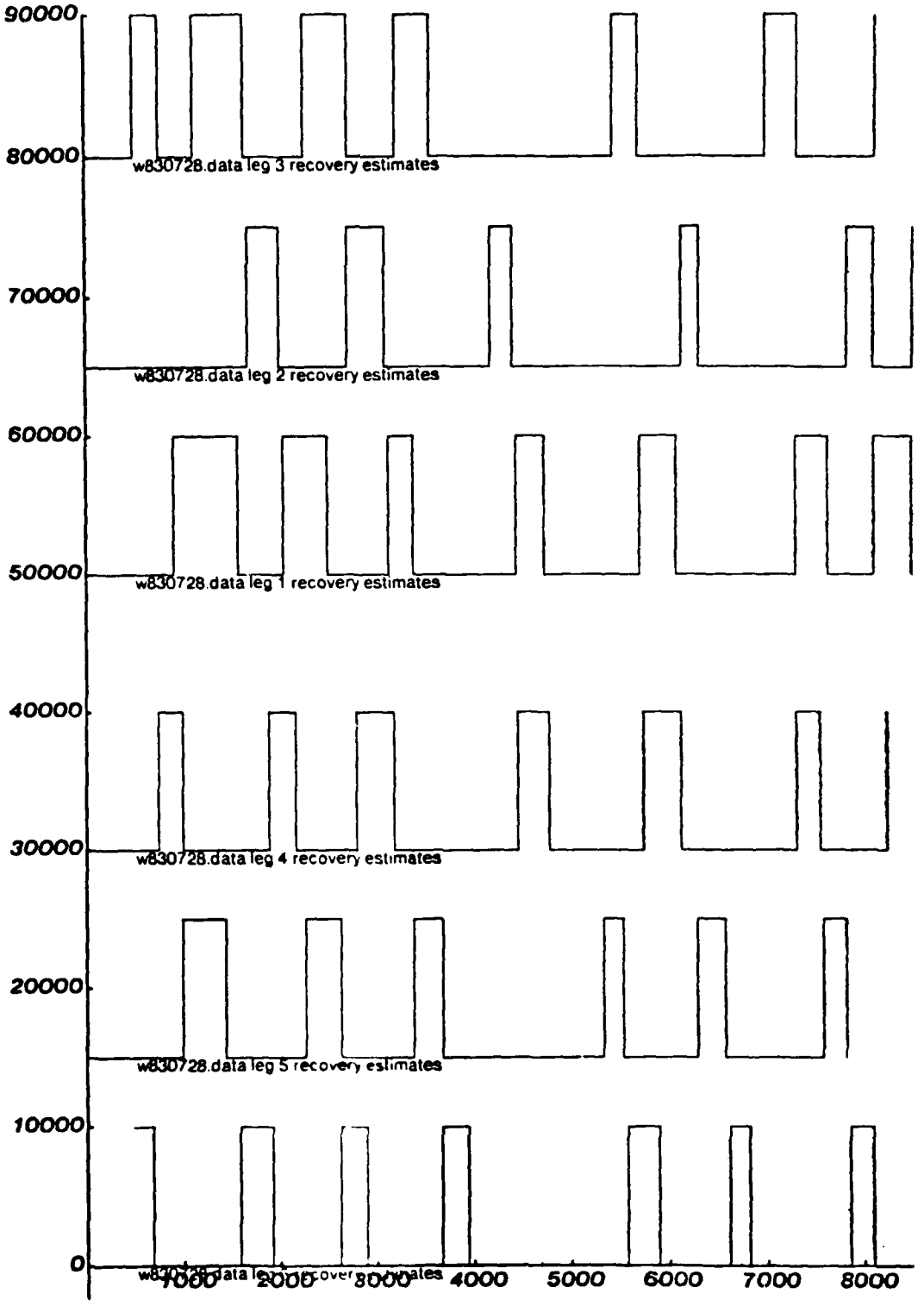


Figure 10. Estimated state of Donner's walking algorithm, plotted against time.

FOOTPRINTS IN THE ASPHALT

(To appear in *Robotics Research*)

by Ivan E. Sutherland
Sutherland, Sproull and Associates, Inc.
SSA # 1853

ABSTRACT

This paper describes our experiences with a man-carrying, 1600-pound, gasoline-powered, hydraulically-actuated, six-legged walking robot. An on-board computer uses leg position and loading information to select one of a few possible hydraulic connections of the leg actuators by switching directional control valves. Passive parallel connections of hydraulic actuators distribute loads equally among several legs, creating the effect of a smaller number of "virtual legs." Series connections of hydraulic actuators achieve coordinated leg motions during walking. The robot uses no sophisticated servos.

Operation of the robot has taught several lessons. For a machine of this size, the arguments favoring six legs rather than four are weak at best. The virtual leg mechanism has proven to be quite effective: when the hydraulic actuators of several legs are connected in parallel, leg motion is coordinated well without computer intervention. Steering with differential drive using variable displacement pumps is quite effective. The slow response time of the valves severely limits coordination and results in a very rough ride.

This paper presents the major lessons learned from operating the machine. Details concerning its design, construction, and programming are given in [1] and [2].

ANECDOTE #1: Students at Carnegie-Mellon University have named the machine "The Trojan Cockroach."

BACKGROUND

This paper is about the operational experience we have gained with the six-legged walking robot shown in Figure 1. I started building the machine in the spring of 1980 as an educational exercise in hydraulic mechanisms. It took its first steps in October 1982. I think of this project as an initial exercise in development of walking vehicles, and of construction of the machine as an apprenticeship. I have rebuilt some parts of it several times, learning through experience the weaknesses of the first designs. The machine has proven to be surprisingly reliable, but it is

very difficult to fix when things go wrong. Overall, it walks convincingly, though roughly. The design of the robot is adequately covered in my book, *A Walking Robot* [1], so I shall review here only enough of its design to make the main lessons comprehensible.

Power and steering control for the machine are provided by four variable displacement hydraulic pumps driven by an 18 horsepower gasoline engine. The rate and direction of oil flow through these pumps may be varied by moving a lever attached to the pump. A control stick and two foot pedals, shown in Figure 2, determine the displacement settings of the four pumps and thus the rates of oil flow. The driver establishes the speed of leg motion manually using these controls. We usually operate with the stick controlling forward speed and turning rate, with the right foot pedal controlling the speed of the elevated legs and the left foot pedal controlling the speed of sideways motion of the knee joints.

The machine has an on-board computer to sequence its leg motions. The computer obtains information from position and force sensors that measure the location and loading of the legs as well as the displacement settings of the pumps. By controlling the setting of directional control valves, the computer can select from a small repertoire of stereotyped leg motions. Because the valves have only three positions each, the computer cannot control the speed of leg motion in a given direction; it selects only the direction of motion. Nor can the computer control the leg motion with any great precision, because the directional control valves take about 100 milliseconds to respond to control signals. The speed of the motion is set by the pump displacement established manually by the driver.

The passive hydraulic circuits built into the machine simplify its control. Once the computer has selected a set of leg motion directions from the limited repertoire available by setting the valves in a given configuration, the nature of the resulting hydraulic circuits coordinates the leg motions without further computer intervention. By connecting hydraulic actuators in parallel, the computer can cause them to share loads equally. By connecting hydraulic actuators in series, the computer can cause them to move at equal rates. It was part of my design objective to explore the use of such passive hydraulic circuits as opposed to active servo control of the legs. Much of this paper will focus on what I have learned about such passive hydraulic interconnections.

MASS AND INERTIA

My first surprise about the machine was how large it turned out to be. I wanted a machine that would carry a man and be self-contained, and I wanted to gain experience with hydraulic mechanisms. The smallest readily available variable displacement pumps were Sundstrand units rated at 10 horsepower each. They determined the size of the engine, valves, filters, actuators, and other components. The readily available hydraulic components turned out to be quite massive; the complete machine weighs about 1600 pounds and is over 8 feet long.

The weight of the machine is clearly evident in its behavior. The dynamic forces required to accelerate the machine from a standing start are substantial. The driver

is clearly aware of the acceleration and can sense the mass of the machine through its response to his control actions. He feels the acceleration directly through the seat of his pants because he is, of course, also accelerated. He is aware of the mass of the machine in two ways. First, because the variable displacement pumps provide a control loading that is roughly proportional to the oil pressure required to move the machine, the driver can feel the acceleration forces directly in his controls. Second, because the machine responds to the onset of fore and aft control commands by accelerating, taking a noticeable time to reach a given speed, and continuing to move unless decelerated, it is clear to the driver that the dynamic forces involved in moving the machine are substantial.

The size of the machine has a strong effect on its balancing behavior. Balance experiments have shown that the driver is almost able to balance the machine on its center two legs using its relatively crude manual controls. Because the machine is quite long, it topples forward and aft quite slowly. It is easy to bring it to the balance point on its two center legs and to tip it back and forth by using small control motions to move the center legs slightly forward and aft. Given slightly improved controls, I think it would be possible to balance the machine manually. Given a servo system of modest performance, automatic balance of this kind should be easy.

The size and weight of the machine have led me to a first and perhaps most important conclusion: six legs are too many for a machine of this size. The argument for using six legs, namely that the center of gravity can easily be kept within a static base of support, is wrong when dynamic forces are considered. Any advantage of static stability for such a massive machine is available only at very low speeds, because dynamic forces dominate the actions of a vehicle of this size moving at the speeds one might normally expect. Moreover, the task of balancing a large machine on a few legs appears to be quite simple. Thus I conclude that at this scale, four legs are more appropriate than six.

The size and weight of the machine have also had a significant effect on our experimental technique. The machine is heavy enough to require a crew effort during experiments, some prior planning for each run, and considerable care to prevent personal injury. As we have continued to work with the machine, we have improved our ability to lift heavy parts into place by using better lifting machines and jacks. We have built and regularly use a set of four removable casters on which to roll the machine out of its garage. We call the garage a "laboratory" even though the shorter term is more apt. We regularly run the machine in the parking lot nearby, and since we roll the machine in and out by manually pushing it, we are well aware of every slope, bump, and hole in the pavement along our usual route. On hot days, the casters sink into the asphalt, making the machine particularly difficult to move.

We generally gather a crowd of onlookers when we run. Crowd control has not been a problem, though people seem remarkably unaware of the real dangers of coming too close to the machine. On one memorable occasion, a young mother put her child between the legs of the machine to take a picture, and removed it only after a forceful warning of the personal danger involved. We have enjoyed the reactions of casual observers, especially their reactions to the loud backfire that

usually accompanies shutting off the engine. Newcomers jump, to the considerable delight of the initiated.

We have used four kinds of support for the machine. First, to move the machine about, we use the casters previously mentioned. Second, for walking experiments, as shown in Figure 1, the machine rests only on its own legs. We remove the casters entirely, as shown in Figure 3, to improve the ground clearance, leaving only a disposable wooden undercarriage to protect the bottom of the machine should the walking program permit it to tip over or fail to support the weight on the legs. Third, for convenience and safety when debugging the computer programs, we rest the machine on four adjustable jack stands as shown in Figure 4. When supported by the jack stands, the machine can wave its legs in the air as though walking, without moving away from the computer terminal and with relative safety. Fourth, not illustrated, we have hung the machine from an overhead frame on chains attached to the lifting loops on its knees. This permits the legs to bear weight without moving the machine forward. We don't like the hanging mode of operation very much because the machine swings alarmingly on the chains.

To change the support form, we have the machine do a push-up that raises the body off the ground. The driver switches off the valves to lock the machine in the raised position and then raises his hands as a signal to the crew that it is safe to approach the machine to remove or install the wheels. To remove or install the jack stands, we have the machine do a push-up with four of its feet on four wooden "footstools" each about 8 inches high, as shown in Figure 5. This raises the body sufficiently so that when the body is supported on the stands and the footstools have been removed, the feet cannot touch the ground.

THE FEET LOOK LIKE GALOSHES

I have tried several kinds of feet on the machine. The first set of feet were simple stubs of tubing extending down from the knee. It was my intention to include a ball joint ankle and a flat pad foot for traction. Early tests with this design showed that because of the position of the knee joint such simple feet would cause the machine to twist sideways so as to lower its center of gravity. To avoid this problem I made the present feet out of curved sections of large electrical conduit whose outside radius matches the length of the lower leg. Thus the feet are actually sections of circles, so that the height of the knee does not change as the machine moves sideways.

In an attempt to improve traction and to soften the ride, we attached sections of automobile tire to the feet. The result looks a little like children's galoshes. The tire sections do improve traction somewhat, but do little for the smoothness of the ride.

One problem with the present feet is that the ends of the circular section can get caught on the ground when the machine comes down from a high push-up. The natural position for the feet as the machine comes out of the garage is the "tuck" in which each knee is bent in as far as possible. If the feet are placed on the ground in this position, their outer ends will dig into the ground, leaving footprints in the asphalt. We have learned to center the knee joints before letting the feet support

weight.

STEERING

ANECDOTE #2: Kevin to Bert, ex-Navy pilot, as he sits in the driver's seat fondling the F-4 control stick handle used for steering: "Don't you think it's time we discussed the armaments package?"

Forward or backward motions of the load-bearing legs on the two sides of the machine are separately powered to provide for steering control. The main fore and aft drive circuit for the left legs has a pump to power the left leg actuators; a separate pump is used to power the right leg actuators. By setting the displacements of these two pumps to different values, the driver can make the machine turn. We can connect either the foot pedals or the driver's control stick to the displacement controls of these pumps in order to try different steering control methods. We usually operate with the control stick connected to these displacement controls as shown in Figure 6, so that the machine will move forward when the driver pushes on the stick, and backward when the driver pulls on the stick. Moving the stick to the right makes the right legs drive backward and the left legs drive forward, causing a right turn in place.

Control seems quite natural with this steering connection when moving forward or when turning in place. Turning motion when moving backwards is less natural, however. Unlike an automobile, in which the steering wheel controls the radius of the path of motion, sideways displacement of the stick on the walking machine controls the angular rate of turn, independent of the direction or speed of linear motion. Thus when backing up, steering control is the opposite of that with which automobile drivers are familiar. Military tanks have this same property.

Our operating experience with this form of control has been quite satisfactory. The driver is able to control the motion of the vehicle easily in spite of a sizable dead zone at the center of travel of the stick caused by backlash in the control mechanism and friction in the hydraulic actuators. Because the variable displacement pumps provide a force feedback on the controls roughly proportional to the pressure of the oil being pumped, the driver feels the accelerations of the vehicle through the control stick and pedals as well as through the seat of his pants.

We generally operate with the right pedal connected to the pump that moves elevated legs and forces them down to pick up weight. Drivers develop considerable skill in controlling the displacement of this pump to meet the conflicting needs of rapid motion for elevated legs and slower motion for legs about to strike the ground. The driver feels the weight of the machine being taken onto each new leg or set of legs as a force feedback in this control. This control also is used to establish the height and rate of a manually controlled push-up.

We generally operate with the left foot pedal connected to the crab pump that drives the knee joints. With this connection, pushing on the left pedal with the toe causes the machine to walk sideways to the right and pushing on the left pedal

with the heel causes the machine to walk sideways to the left. This control is hard to remember, but fortunately the machine moves sideways quite slowly, and has caused no damage.

We have also used a different arrangement of the controls in which we used the pedals for steering and the stick to control the speed of elevated legs and sideways motion. This connection improves the control for sideways motion by making the obvious connection between sideways stick motion and sideways vehicle motion. Because the forward drive mechanism moves the machine quite rapily, we have found that steering with the pedals is less satisfactory than steering with the stick; we like the finer control of drive and steering available with the stick.

To simplify the rearrangement of the controls, we connect the controls to the displacement mechanism of the pumps with a passive hydraulic system much like the hydraulic brake system on a car. A small master cylinder connected to the control transmits motion to a similar slave cylinder at the pump through fluid motion in a pair of small plastic tubes. The fluid connection transmits the commands to the pumps, returns the control loading forces to the driver, and provides a comfortable damping on the control system. To simplify cleanup after rearrangement of the controls, we use water rather than oil as the fluid in this system. In cold weather the water freezes after about 20 minutes of exposure, thus limiting the length of the run. We fill the system from an elevated bucket of water with a siphon.

ANECDOTE #3: Marc, graduate student, to Ivan as he attempts to clear the siphon tube of water by blowing into it: "You'll never be able to blow the water out of that tube." Mike, previously graduated, as Ivan succeeds: "Never underestimate the power of the professorial windbag."

HYDRAULIC ACTUATORS IN SERIES

In order to coordinate the motion of the sets of legs on the left and right side of the machine, I built the machine with two series hydraulic circuits for the forward drive, one of which is shown in Figure 7. Oil from the pump causes the front leg to move back, displacing oil from the other chamber of its hydraulic actuator. This displaced oil drives the next leg back, displacing oil from the other chamber of its actuator, oil that is used in turn to move the third leg back. Thus the legs in each series circuit move back in lockstep without any computer intervention. Indeed, the left drive circuit couples the left legs together hydraulically and the right drive circuit couples the right legs together hydraulically.

Valves permit the computer to include or remove individual legs from this series drive circuit. When a leg is taken out of the drive circuit, the valve diverts oil past the actuators of the removed leg so as to maintain the series circuit for the other legs. Thus the legs that are driving will move in a coordinated fashion, but not all legs need drive simultaneously.

The series connection of hydraulic actuators has caused us considerable diffi-

culty. The trouble comes from the internal friction of the actuators. Because of the nature of the piston rod seals, their friction drag is proportional to the pressure that they bear. Thus even with no external load, the first of several hydraulic actuators connected in series will operate with high friction, because it is operating at the sum of the pressures required to overcome friction in all of the actuators. The pumps in our system are able to move the machine forward in spite of the friction, but it requires considerable energy to do so. Were I to rebuild the machine, I would abandon the series connection altogether and use instead the parallel connections that I shall describe next.

PARALLEL HYDRAULIC CIRCUITS

The parallel connection of actuators shown in Figure 8 provides for the actuators so connected to share load forces equally because of the equal pressure of the oil in each of the actuators. The detailed motion of the actuators, however, will depend on the external constraints imposed on them. As is obvious from the figure, all of the oil might flow into one actuator if it were easier to move than its mate, or one actuator might even move in the opposite direction if it were loaded substantially more heavily. In fact, the parallel connection guarantees only that the sum of the motions of the actuators will be controlled by the pump; the difference in the motions is not controlled.

I selected a parallel connection of actuators to drive the knee joints on the machine because it can achieve passively the complex knee motions that must occur as the machine walks. Because the knee of a driving leg moves in a circular arc, whereas its foot must move along a line parallel to the motion of the body, each knee flexes in and out slightly with each step. When the machine turns a corner, the knees of forward legs flex in one direction while the knees of the rear legs flex in the other direction. All of these knee motions are accomplished without computer intervention by passive oil flow that equalizes the knee loading.

The knee drive actuator for each load-bearing leg is switched into the parallel knee drive circuit as shown in Figure 9. All such knee drive actuators must support the same sideways load, or nearly so, because otherwise oil would flow away from the legs carrying the greater sideways load. The knees adjust their positions automatically to distribute the sideways load equally. On level ground where there can be no net sideways force on the machine, the total sideways force must be zero, and thus the sideways force on each leg is also near zero. On sloping ground, each leg will bear an equal share of the lateral force required to support the machine.

The knee drive cylinder for elevated and lightly loaded legs must be isolated from the knee drive circuit. If elevated legs were included in the parallel circuit, they would rapidly move to an extreme position whenever other legs supported sideways loads. The knee drive cylinder of elevated legs is driven in a different hydraulic circuit to preposition the knee in anticipation of its required motion during the next step. The computer uses data from the leg loading sensors to select which circuit controls the knee motion.

This knee coordination mechanism has proven to work remarkably well. If the

knee drive pump displacement is not zero, the machine moves smoothly sideways. Most important, the knee drive mechanism presents no resistance to turning; the rate of turn is set only by the differential fore and aft motion of the left and right sets of legs. When the machine turns, the front knees flex in one direction and the rear knees in the other direction to accommodate the turn. No computation is required on the part of the computer to control the knees other than to switch the knee actuators into and out of the knee drive circuit as the legs rise and fall.

Another parallel connection of actuators occurred because of the configuration of the vertical motion actuators for the legs. All legs that are being forced downward share a common source of oil and happen to be connected in parallel. This parallel connection has some interesting implications that I would like to cover in some detail, because it sheds light on the nature of the passive load sharing that is possible with simple hydraulic circuits.

Consider two legs that are being moved downward simultaneously in order to make the machine do a push-up. Because their hydraulic actuators are connected in parallel, they must each carry the same load. If they did not, the one with the lesser load would move down faster and pick up a larger share of the load, or the one with the greater load would move upward and cause oil to flow into the actuator of the other until the load was equally shared. Similarly, if more than two legs are simultaneously driven downward, they will all move so as to distribute the load equally. This load equalization occurs, of course, passively and without the intervention of the computer other than to select the settings of the valves so as to connect the hydraulic actuators in parallel.

In order to think effectively about the behavior of such a parallel connection, we have devised the notion of a "*virtual leg.*" To see how the virtual leg comes about, consider a set of legs being used to do a push-up with their vertical actuators connected in parallel. Such a set of legs can contribute nothing to the balance of the machine, because each such leg must carry equal load. If the machine begins to tip with such a collection of legs on the ground, the body will descend with respect to some of the legs and rise with respect to others, redistributing the oil amongst their actuators, but all of the feet will continue to support equal weight on the ground. Only minor frictional forces will resist the tip.

The lifting force provided by such a set of legs is equal to the sum of the lifting forces of the individual legs. This composite force can be thought of as a single force located at the geometric center of the positions of the feet involved in the parallel lifting set. Thus if two legs connected in parallel are lifting the machine, they behave like a single leg located midway between them, and if three connected in parallel are lifting the machine, they behave like a single virtual leg located at the center of the triangle formed by their feet. For a six-legged system, at most four legs can safely be connected in parallel, because two others must remain independent to retain the balance of the machine. If all weight-bearing legs are connected in parallel, the machine will tip over.

This load-sharing mechanism is remarkably useful. We often raise the machine off the ground by connecting the three left legs in parallel and driving them downward. Because of the parallel connection, they all remain in contact with the ground and share the load equally. By considering them as a single virtual leg at the loca-

tion of the center one, it is easy to see that the front and rear legs on the other side machine, which are locked in position, must take up unequal loads to compensate for any fore and aft imbalance of the machine. We also sometimes tuck the center legs up and out of the way and use the two front legs or two rear legs to lift the machine, using the pair at the other end to control sideways tipping. The virtual leg concept makes it easy to see that this connection is equivalent to providing a tripod of support. We can also do a four-legged push-up, as shown in Figure 10, in which case, remarkably enough, the load on each of the six legs is approximately equal!

This load-sharing property of the parallel connection can create problems as well. The most significant of these we have come to call the "foot stamp." To see how this comes about, consider a situation starting with an elevated center leg and weight-supporting fore and aft legs on one side of the machine. Now suppose that the valves for vertical control of all the legs on that side are suddenly switched so as to connect the actuators of the two weight-supporting legs in parallel with those of the elevated leg. Because the load must be equal on all legs in a parallel circuit, the weight-supporting legs move up quickly as the body falls, and the elevated leg drops down very rapidly, striking the ground with substantial force. The body of the machine also drops, though of course by a lesser amount. No one has yet been hurt when this occurs, but the very rapid leg motion that results is a potential danger of the parallel connection. Our programs avoid the foot stamp by using leg loading data to determine which legs are elevated and must not share parallel circuits with loaded legs.

The result of connecting hydraulic actuators in parallel is very like the result obtained for the back wheels of an automobile by the differential gear in the back axle. Like the hydraulic actuators in parallel, the differential gear makes the force generated by each wheel be the same, regardless of their rates of rotation, thus producing a virtual drive force mid-way between the wheels. When the automobile goes around a corner, requiring one wheel to turn more rapidly than the other, the differential gear permits the wheels to accommodate to the needs of the ground. When one wheel loses traction, in snow for example, the other cannot drive the vehicle forward. Parallel connections of hydraulic actuators permit load sharing when it is appropriate, but with simple valves the load sharing can easily be disabled when necessary.

PARALLEL RATHER THAN SERIES DRIVE

In retrospect, I wish I had used a parallel hydraulic connection for the right and left side leg drive motion instead of a series connection. Whereas the series connection provides that the feet move aft at the same rate, regardless of the drive force required, the parallel connection would provide for an equal sharing of the required drive forces, permitting minor accommodation of the motion to equalize the drive forces. The coordinated motion of the feet would be provided by the ground on which they rest; there is really no need to provide an additional mechanism to synchronize leg motion. Rather, one wishes to have the legs share the drive

force equally, assuming none of them slip. I would, thus, have made three parallel drive circuits, one for the fore and aft motion of the right legs, one for the fore and aft motion of the left legs, and one for the sideways motion of all legs. Such a connection provides exactly three drive circuits that correspond to the three degrees of freedom involved in motion across a plane surface.

There are two weaknesses in this parallel design. First, should a leg slip, all legs will loose traction. It would be a prime objective of the computer to detect slipping legs and remove them from the parallel drive circuit. The existing programs do this for sideways motion quite effectively by switching legs into the parallel drive only when they bear a good deal of weight. Lightly loaded legs are permitted to coast. I believe that the problem of leg slipping could easily be handled by a combination of this leg loading rule and programs to detect leg slipping explicitly.

Unfortunately, the second weakness of a simple parallel connection requires the use of additional hydraulic components. The rate of motion of a set of actuators connected in parallel and driven by a common pump depends on how many actuators there are in the set. As more actuators enter the circuit, the average rate of motion decreases. To obtain smooth motion as actuators are switched into and out of the circuit, one really wants the oil flow rate to be proportional to the number of actuators being driven. A hydraulic flow divider can be used to provide a constant rate of motion even as actuators are switched into and out of use. Flow dividers are more often used as shown in Figure 11 to provide for synchronous motion of multiple actuators, but they can also be connected as shown in Figure 12 to provide for parallel leg drive for a walking robot.

I believe that it is possible to design the drive circuits for a walking robot so that the parallel connections provide for coordinated motion of the legs in all required directions. For example, in addition to the lateral drive circuits that can control forward and sideways motion as well as yaw, other parallel circuits can be used to control height, tilt and pitch. With such circuits in place, the displacement of a few variable displacement pumps can provide adequate control for the machine. The computer need only select which of the legs to include in which of the circuits.

COMPUTERS

The on-board computer is an MC68000 microprocessor development system. It is connected to the sensors and to the valves that it actuates by a serial CMOS digital communication link. There are eight analog to digital converters in the system, one associated with each leg, one for the driver station, and one associated with the engine and pumps. These analog to digital converters can be connected to any of several position and force sensors in each of the legs. To permit a degree of self test, each leg joint has two independent position sensors. The control switches on the F-4 handle at the top of the stick and an additional set of buttons from a telephone dial provide digital signals that the computer can sense. The computer controls the setting of six valves associated with each leg, each of which has three possible positions. Although there are twelve valve control bits per leg, only about a dozen of the combinations make meaningful hydraulic interconnections.

Three sets of programs have been written. My own code was written in 68000 assembly language, assembled on a VAX and downloaded into the 68000 over an RS-232 line. It provided the basic I/O routines and a simple form of walking. Improved code by Mike Ullner, also written in assembly language, has permitted us to try several different gaits and has improved the manual control of the machine. In order to explore the problems of independent control of each of the six legs, a graduate student, Marc Donner, has developed a concurrent programming language for programming the walking machine [2]. This language compiles on the VAX and downloads into a second on-board 68000 with a larger memory. This language and Marc's experiments with the machine will be the subject of his dissertation at Carnegie-Mellon University [3]. When his programs are running, the smaller 68000 handles I/O control of the walking robot, and the larger 68000 operates the walking algorithm.

We have found a need for considerable feedback from the computer to the driver. In some early walking experiments we ran the machine in an entirely self-contained mode without connection to fixed computers or power. We have found, however, that it is valuable to be connected to terminals during a debugging session, and thus have equipped the machine with a dual RS-232 umbilical to the terminals. We have also used a portable TRS-80 terminal held in the driver's lap to give him more direct indication of what the computers are doing. It is evident from our need for the computer terminals that we did not include enough computer-controlled operator feedback. In a new design I would provide a few lights per leg, and perhaps a computer controlled analog indication for each leg. It would be useful for the driver to know the load on each leg as well as its current state in the walking cycle. Making these indicators bright enough to be read in sunlight is obviously important.

The most annoying part of debugging the code for this machine has been the "leg bump" problem. The mechanical design permits legs to collide and therefore requires some monitoring by the computer to prevent mechanical damage to the machine. The simplest forms of these monitoring programs simply stop the operation of the machine when legs get too close together. The driver must infer which legs are in trouble from the leg position or by examining the computer terminal. He uses manual control to separate them.

The hydraulic circuits in this machine make the legs interdependent in several ways. The most obvious interdependency is the series drive circuit. The series circuit produces a control interdependency because when one leg in a series circuit reaches the end of its travel none of the other legs will move until it is removed from the circuit. Similarly, the parallel connection of actuators that result when several legs are involved in lifting the machine produces a control interdependency because oil can flow directly from one such actuator to another. The control computer must account for the actions of other legs in selecting the action for a particular leg lest the "foot stamp" problem result. Less obvious is the fact that some combinations of leg motions require the use of a "hydraulic compensator" to equalize the oil flow in different parts of the hydraulic circuit. The hydraulic compensator has limited travel and thus puts arbitrary restrictions on the possible leg motions that are an annoyance to the programmer. Some combinations of leg motions are simply impossible because of the nature of the hydraulic circuit and must be avoided by the

control computer. An improved design would include as few restrictions as possible to permit algorithms to operate the legs independently.

We have evolved a useful operational rule for who drives the machine. The rule is: "If you programmed it, you drive it." This rule makes good sense for several reasons. First, the bumpy ride is distributed among the entire group so that no one's posterior suffers unduly. ("Posterior" replaces a shorter Anglo-Saxon term to which some reviewers objected.) Second, the programmer is maximally aware of the problems with his program. Third, the rule permits enough concentration of the driving activity to develop a good level of driver skill. And fourth, the rule provides a simple way to distribute what is, after all, the payoff of the project: that sense of control one gets when driving a powerful machine. We have had a few guest drivers.

ANECDOTE #4: Claude Shannon visited, drove a little, observed the machine doing a maneuver that he described as a "pirouette," although a turn in place that takes a full minute might better be described otherwise, and offered the following eight lines:

The Trojan Cockroach

Higgledy Piggledy
Ivan E. Sutherland
Built a huge cockroach—twelve
Horsepower clout!

The roach, waxing vengeful for
Massive roach genocide,
Hexapediacally,
Stamped Ivan out.

—Claude E. Shannon

ACKNOWLEDGMENTS

I would like to thank the many people who have helped with the construction, operation, and programming of the machine. Especially Marc Raibert and Bob Sproull for encouragement and moral support, Mike Ullner, Marc Donner, and Kevin Nolish for major programming efforts, Bert Sutherland for the control station design and construction, and David Douglas for maintaining the machine during much of the experimental work. Support for building the machine came from Sutherland, Sproull and Associates. Support for programming, operating and reporting on it came from DARPA under contract #MDA903-82-C-0102. Marc Donner's work was sponsored by IBM. Facilities to house the machine and tools to fabricate it were provided by the Robotics Institute at Carnegie-Mellon University.

I gratefully acknowledge the material support of these sponsors. Most of all, thanks to Marcia Sutherland whose moral support and frequent contributions of time and energy have made the whole project possible.

POSTSCRIPT

I agreed with Marc Donner that I would keep the machine working until August 15 so that he could finish the work required for his dissertation. On August 18 leg #5 bent during a walking session and we decided to "terminate the operational part of the program." The machine is currently serving as a source of parts for another machine. Stay tuned...

REFERENCES

[1] Ivan E. Sutherland, illustrated by Frederick H. Carlson. *A Walking Robot*, The Marcian Chronicles, Inc. 1983. Mail orders: P.O. Box 10209, Pittsburgh, PA 15232, $25 plus shipping.

[2] Marc Donner, "The Design of OWL, A Language for Walking," in *Proceedings of the SIGPLAN'83 Symposium on Programming Language Issues in Software Systems*, published as *SIGPLAN Notices 18*, 6 (June 1983), 158–165.

[3] Marc Donner, PhD thesis dissertation, in process.

[4] Final report on DARPA contract #MDA903-82-C-0102, Programming and Evaluating an Untethered Walking Machine, Sutherland, Sproull and Associates, Inc., 1983.

[5] Marc H. Raibert and Ivan E. Sutherland, "Machines That Walk," *Scientific American*, January 1983, 44–53.

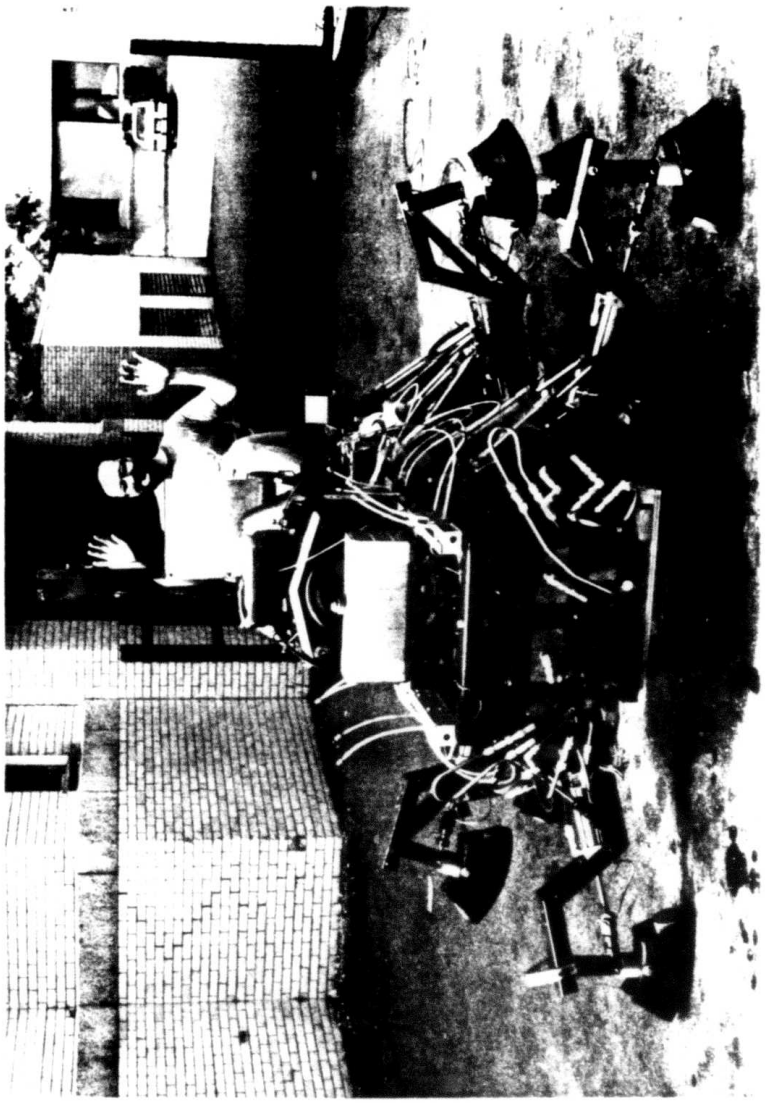


Figure 1. Photograph of walking robot doing a four-legged push-up.

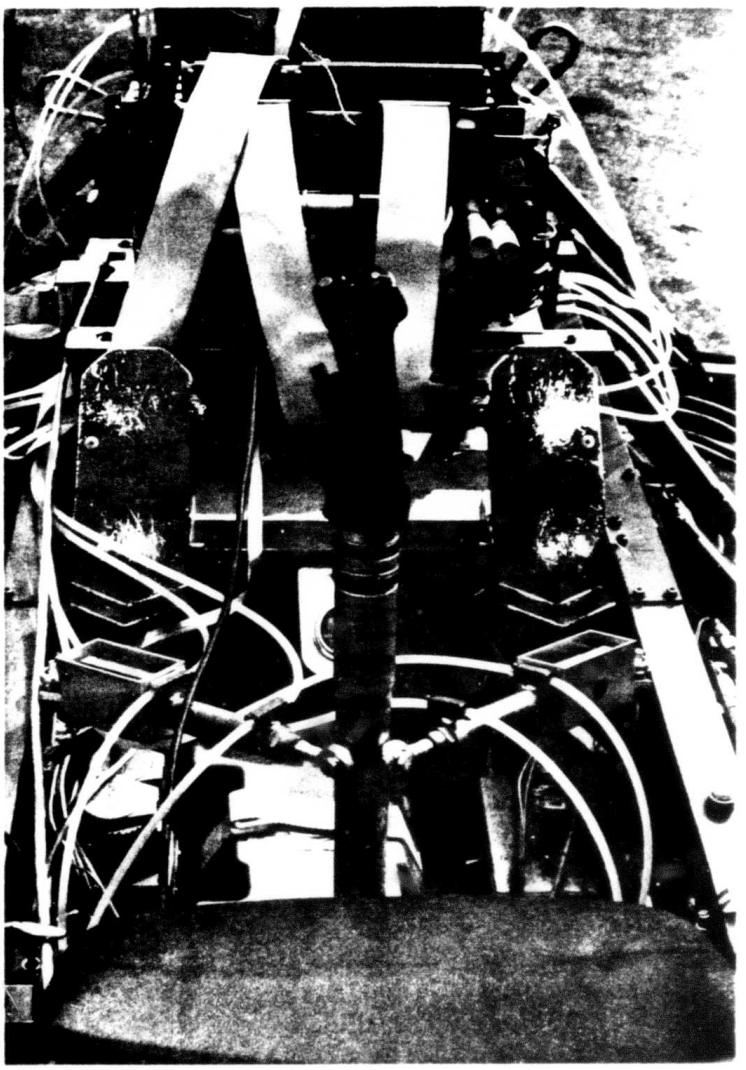


Figure 2. Driver's station.

Figure 3. Installing wheels while the machine is doing a push-up.

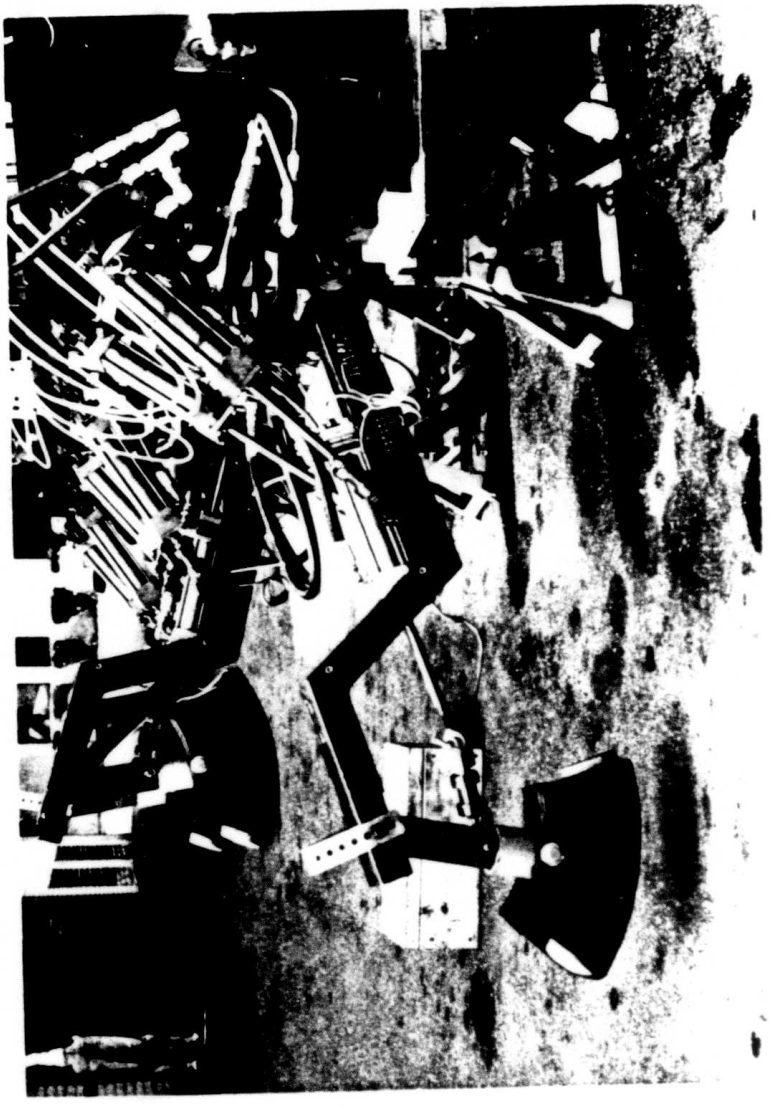


Figure 4. Machine supported by jack stands with its legs waving in the air.

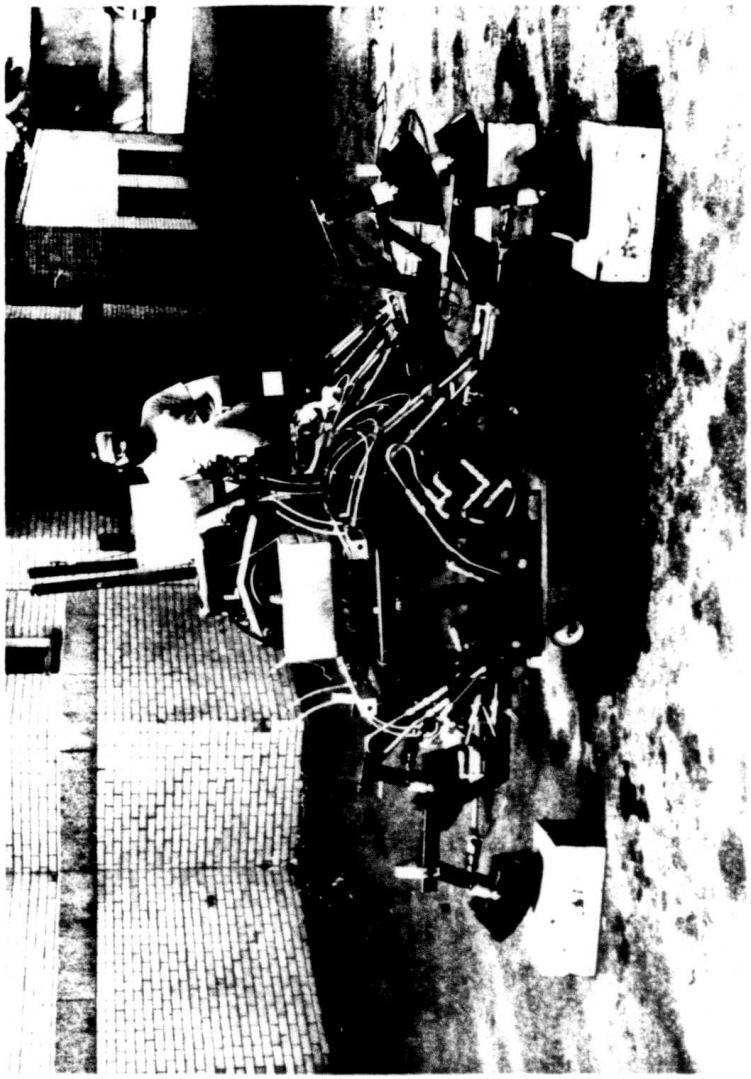


Figure 5. Machine doing a push-up on the wooden crates, or "footstools."

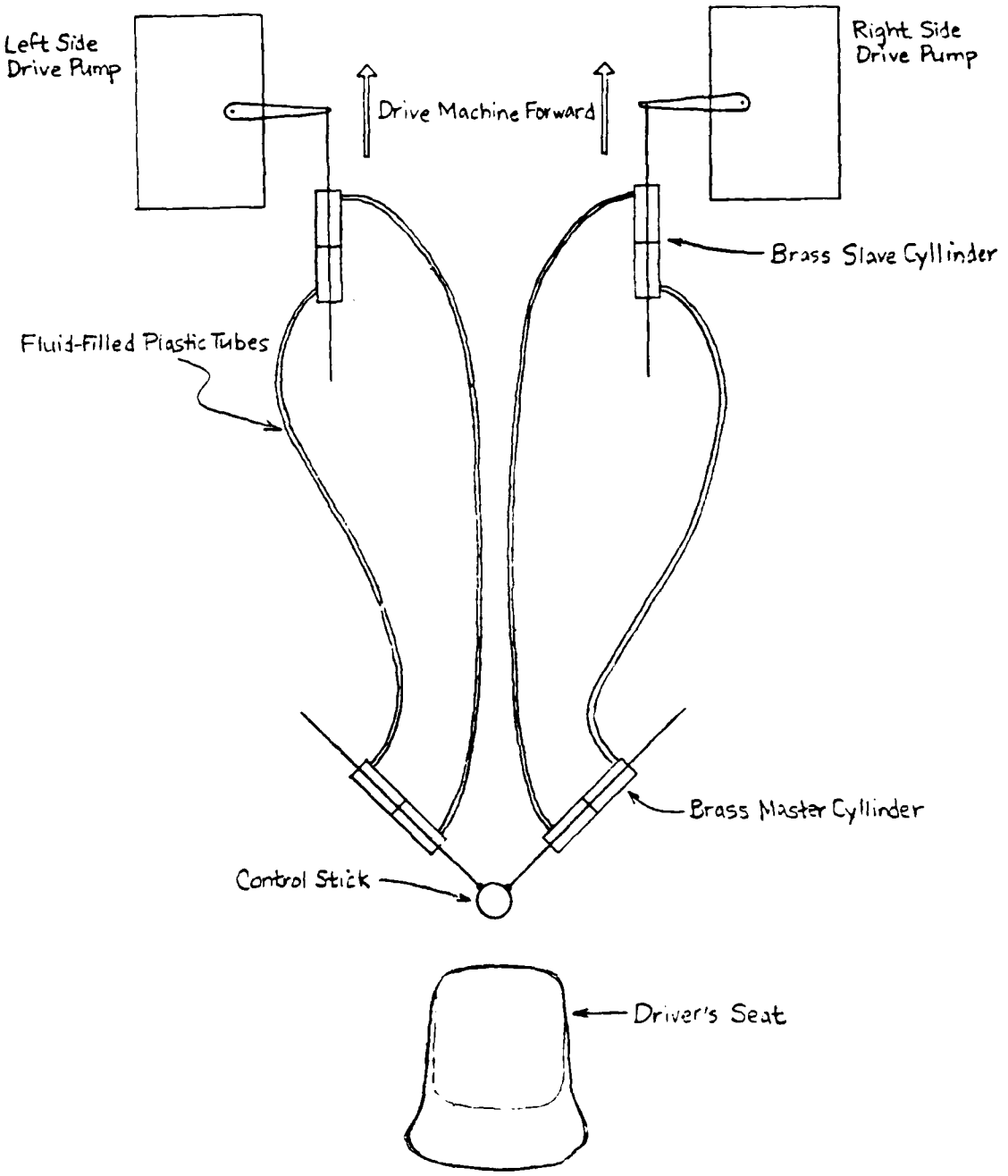


Figure 6. Control stick connections to displacement settings of drive pumps.

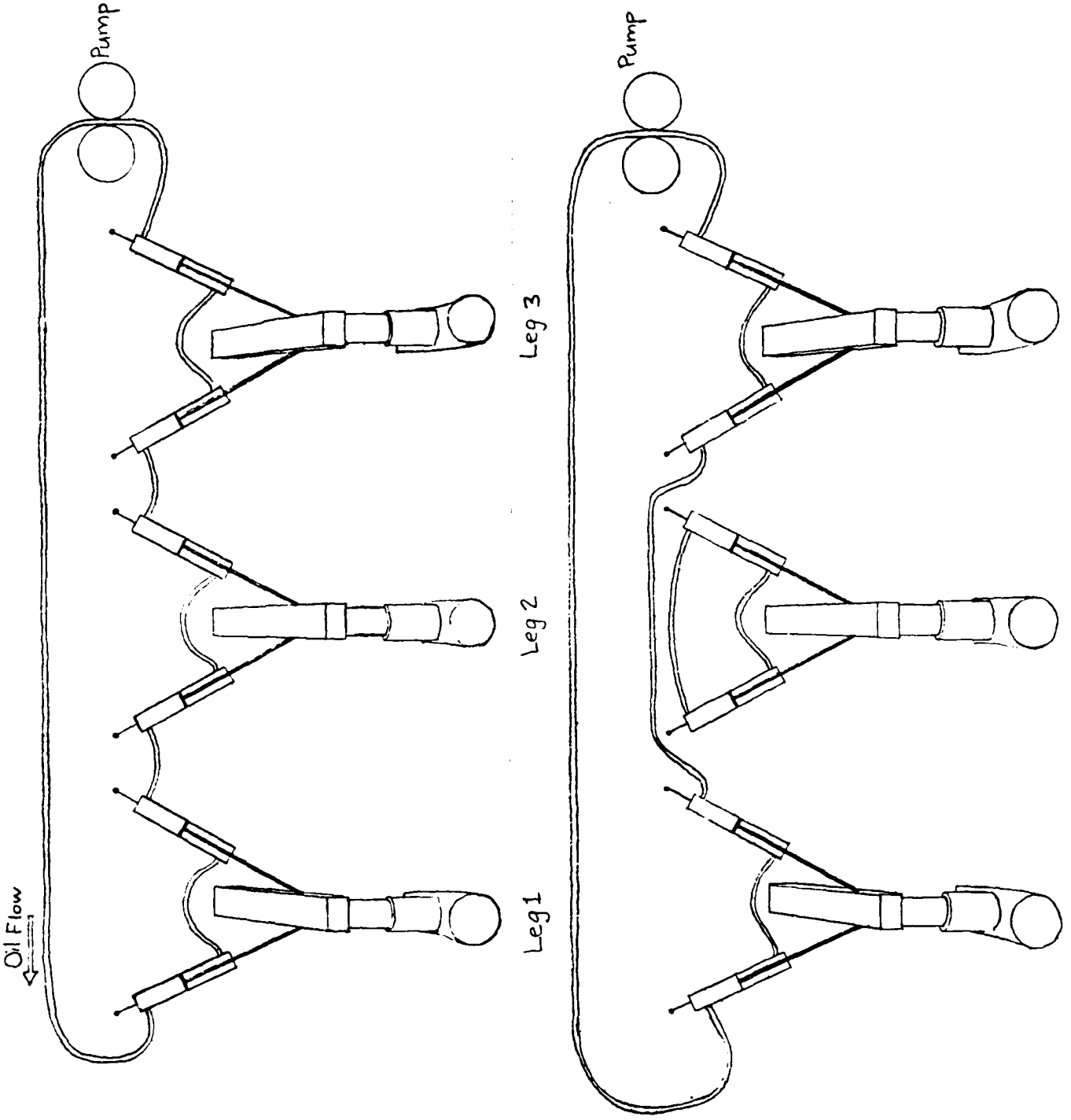


Figure 7. Forward drive circuits. A: Forward drive circuit with all legs driving. B: Forward drive circuit with Leg 2 coasting.

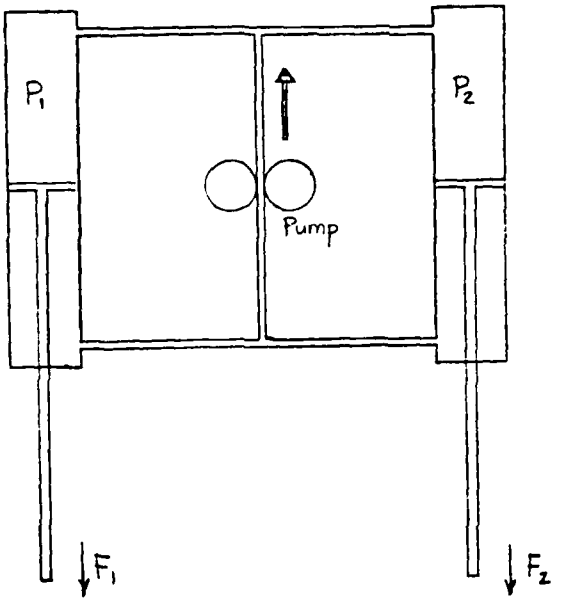


Figure 8. Parallel connection of actuators. $P_1 = P_2$, therefore $F_1 = F_2$.

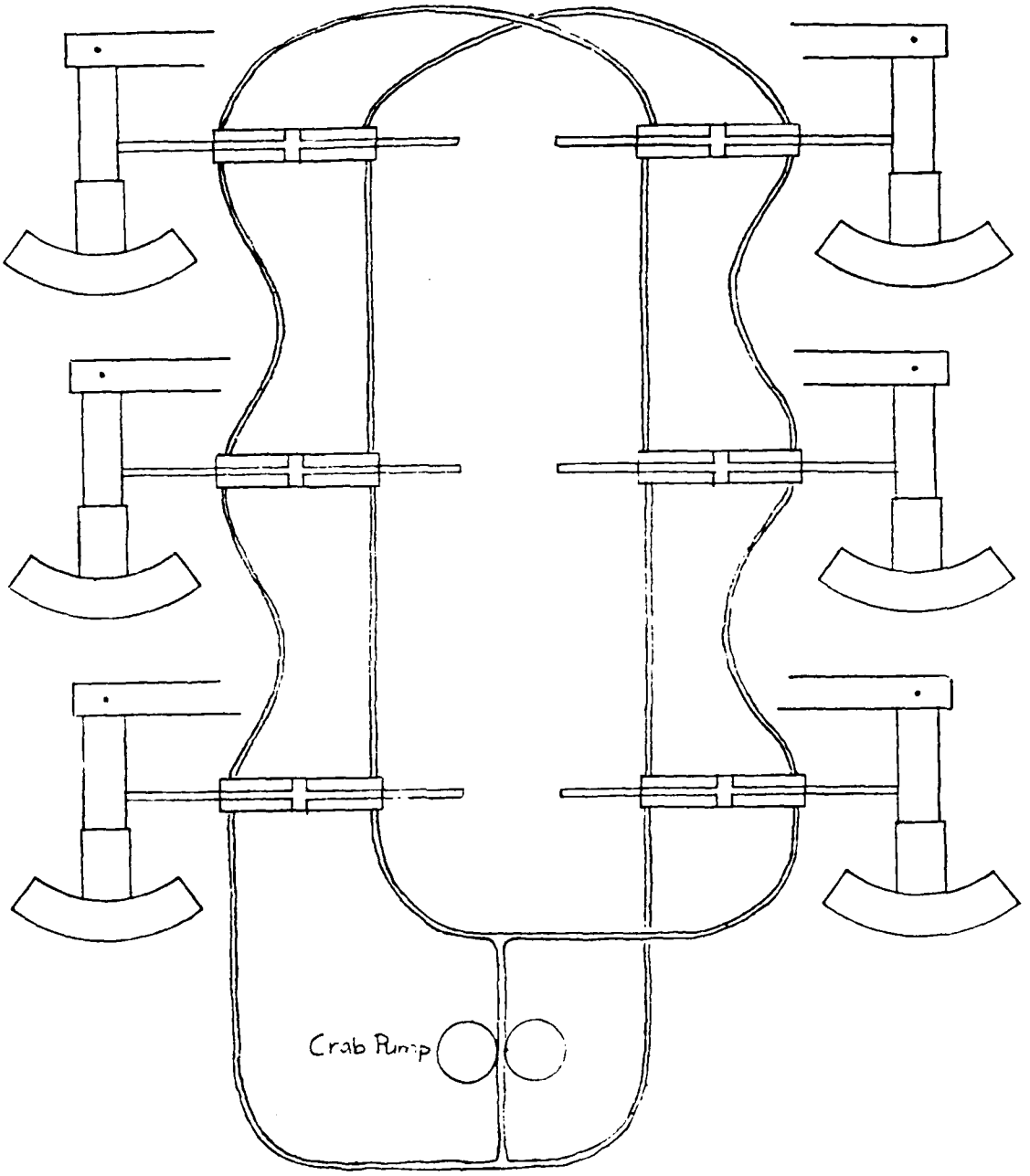


Figure 9. Parallel knee drive circuit.

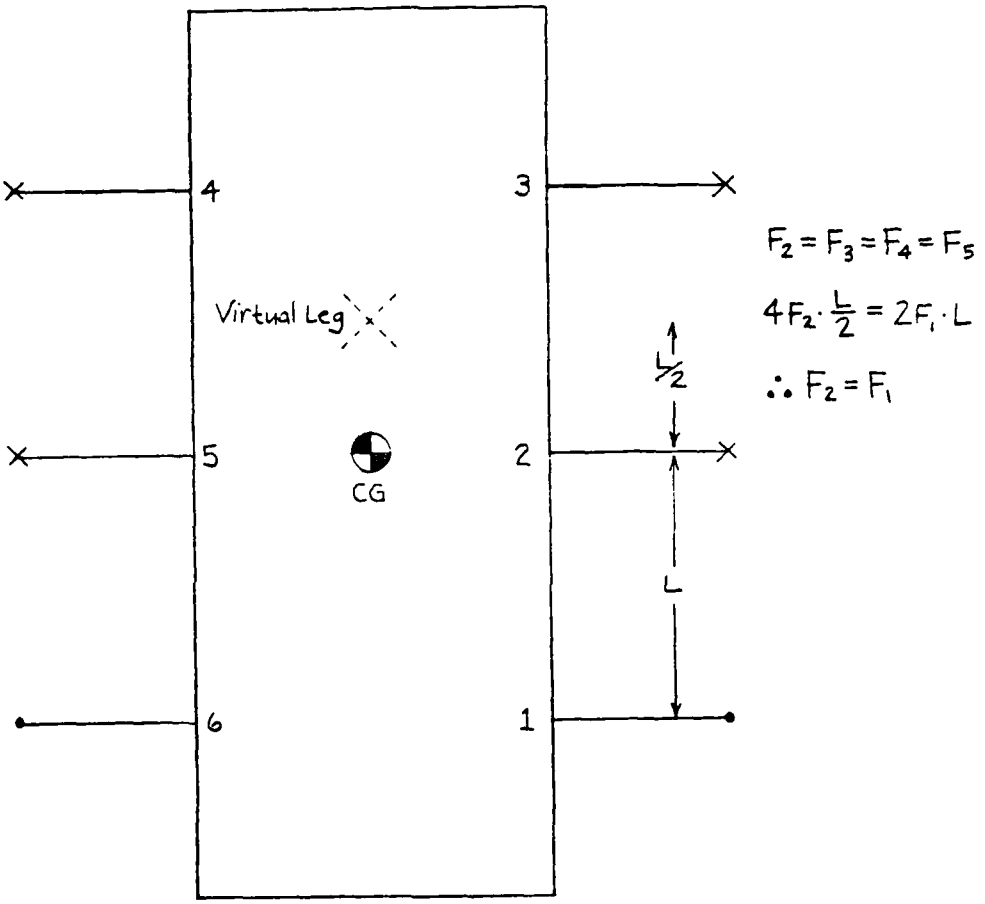


Figure 10. Plan of a push-up with four lifting legs. Legs 1 and 6 are fixed. Legs 2, 3, 4, and 5 are *lifting* in parallel.

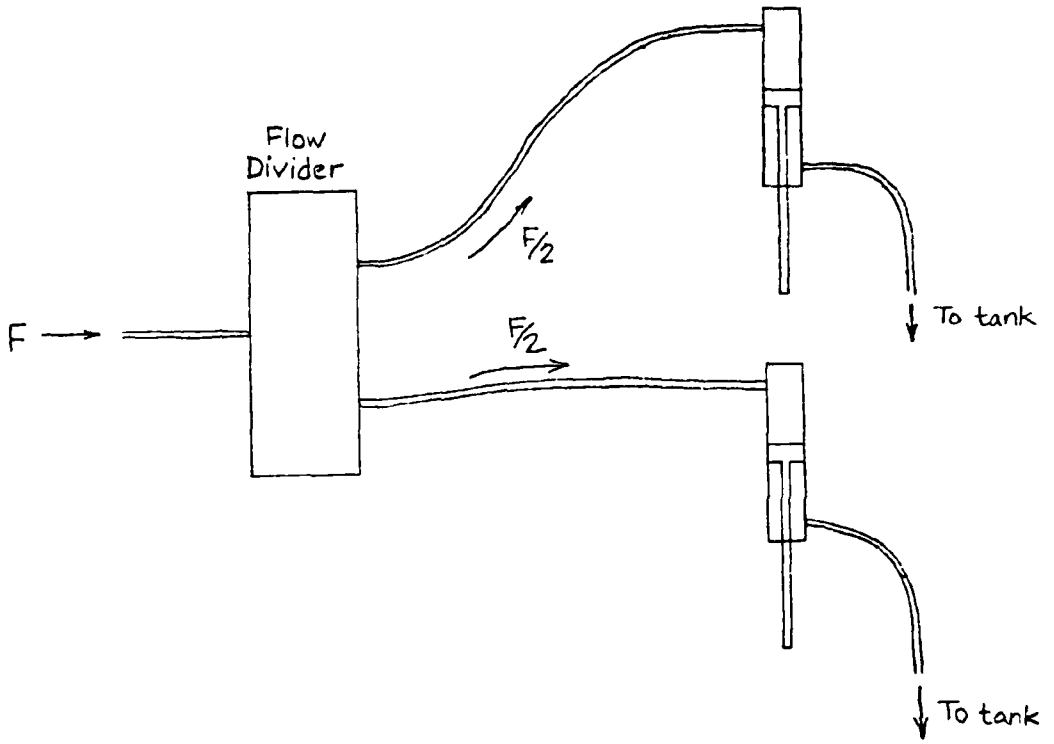


Figure 11 Conventional flow divider circuit.

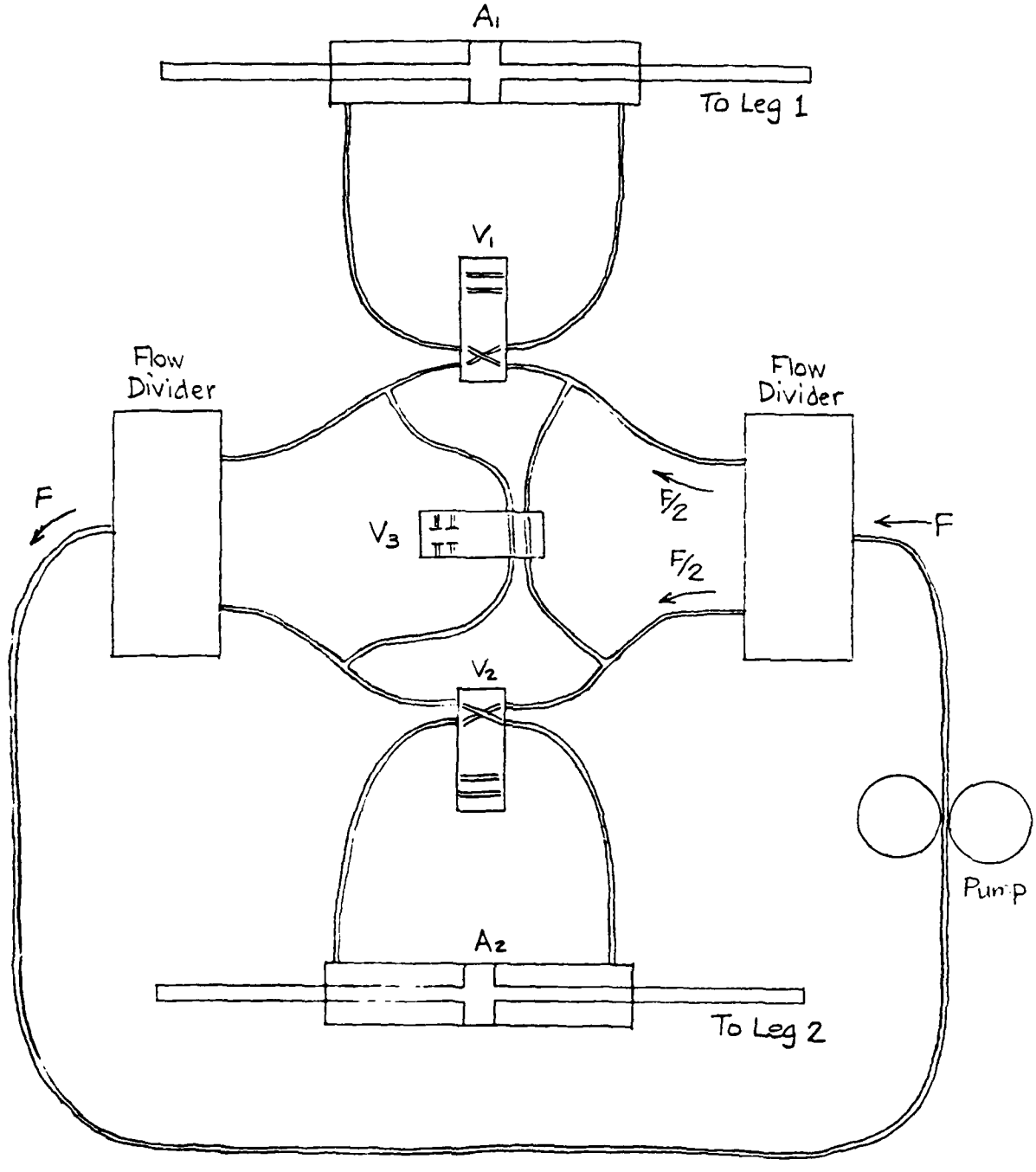
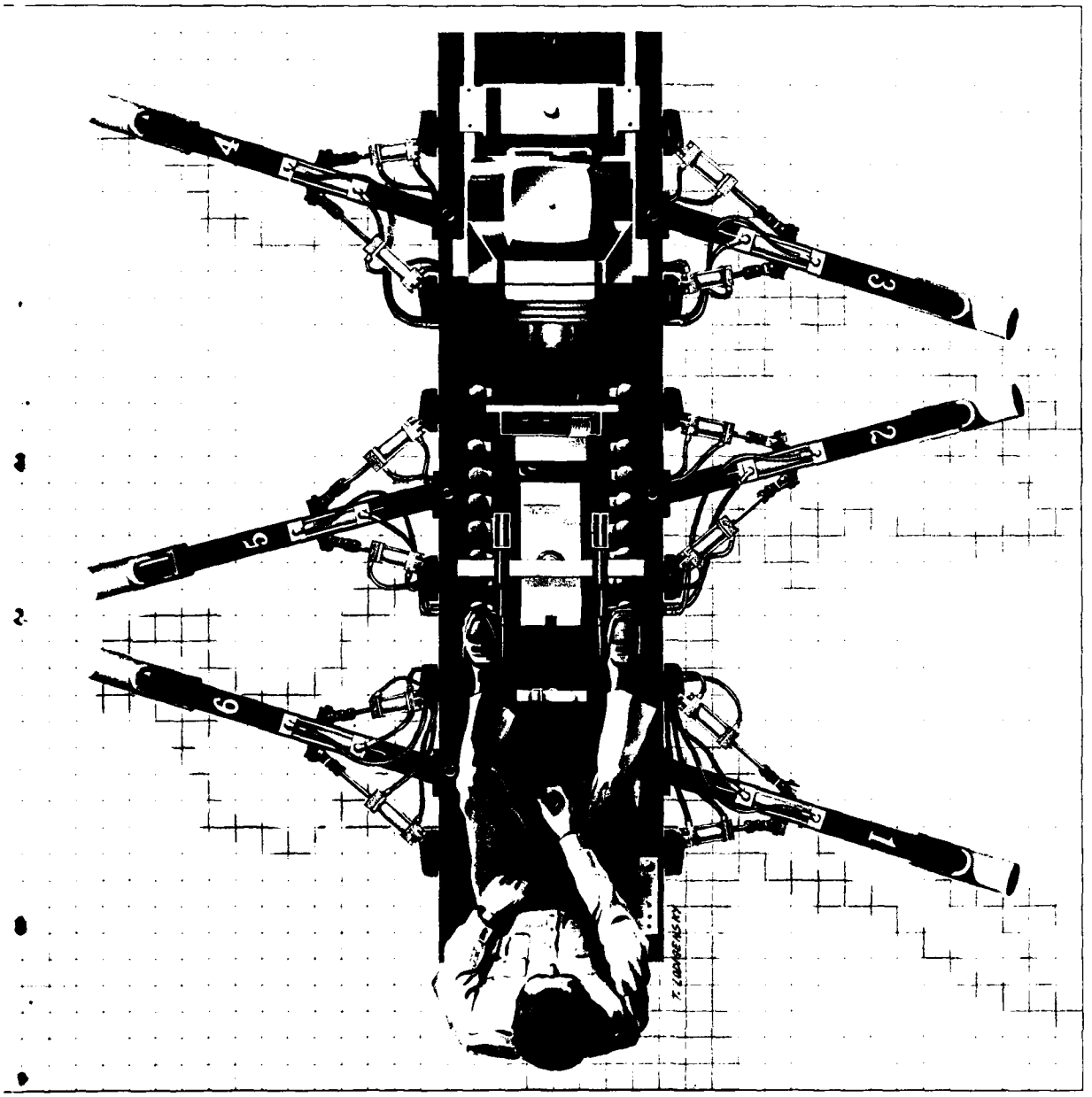


Figure 12. Flow divider used in a leg drive circuit. V_1 and V_2 select which leg to drive. V_3 connects legs in parallel if both are driving.

SCIENTIFIC AMERICAN



MACHINES THAT WALK

$2.50

January 1983

Information, please.

Or, how the IBM Personal Computer can bring you the world.

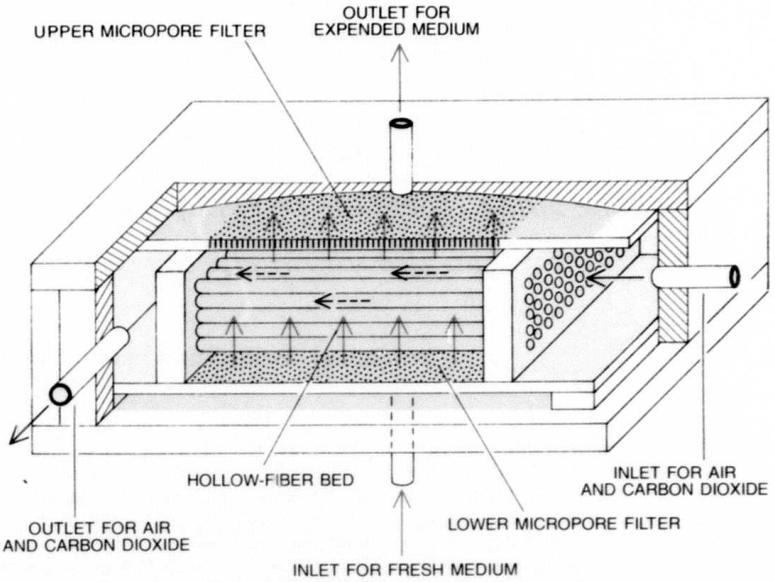
Modem shown not supplied by IBM

the filter. The recycling pump moves the medium, with its suspended cell-encrusted beads, toward the satellite vessel. Once in the settling bottle, removed from the area where the medium is being stirred, the beads tend to settle and aggregate into a dense slurry that gradually slides back into the main vessel; medium that is essentially bead-free goes on to the satellite filter vessel and is either recycled or removed from the system.

In the settling process the beads and their cells come in close contact and continue to be perfused with fresh medium from the main vessel. Cells bridge from one microcarrier to another, forming large clumps of beads and cells, with the result that there can be up to four times as many cells as one would expect to find on a given bead-surface area. Another advantage, which we had not foreseen, arises from this highly aggregated condition: the separation of the cells from their microcarriers is facilitated, solving the scale-up problem. In the course of aggregation the attachment of individual cells to the beads is somehow reduced, to be replaced by multiple cell-to-cell attachments. When the clumped beads and cells are pumped out of the reactor, a brief treatment with the enzyme trypsin releases the cells from the microcarriers in good condition.

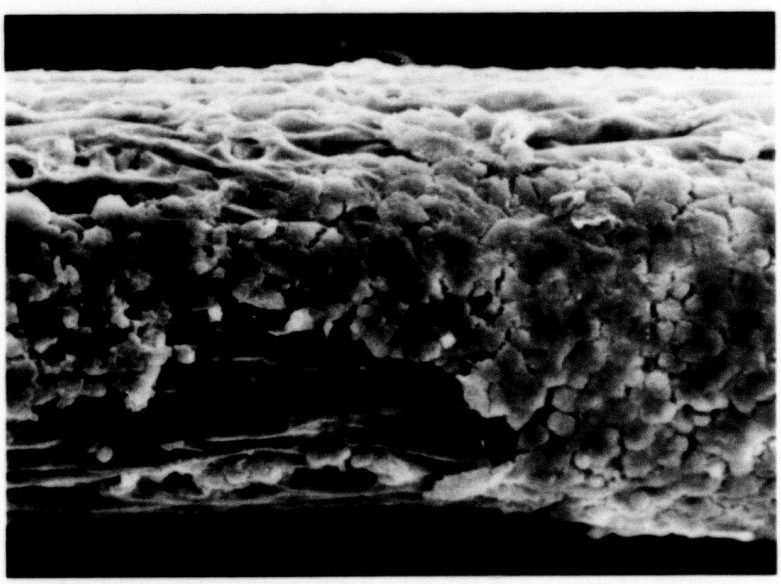
The microcarrier-perfusion reactor has proved to be an efficient system for the large-scale culture of anchorage-dependent cells. Microcarriers have always had the potential of providing a very large surface area. Now all that area can be exploited (actually overexploited, given the bridging effect) because perfusion maintains an optimum environment for cell growth even at extremely high cell densities, which in this system are from five to 10 times as high as before.

The combination of high-density cell growth and efficient scale-up leads to impressive results. In one experimental run of a small (four-liter) microcarrier-perfusion reactor we grew 40 billion human fibroblasts, the equivalent of a normal harvest from 1,300 roller bottles. Those cells were removed from their carriers and were inoculated in turn into a 44-liter reactor along with 400 grams of fresh microcarriers, which provided a surface area of 188 square meters. In the ensuing run 340 billion cells were grown, a harvest that would have required 11,000 roller bottles. The harvested fibroblasts served for the production not only of interferon but also of urokinase and an angiogenic factor. The best measure of the cost-effectiveness of the microcarrier-perfusion system is the yield of cells per liter of medium expended. In all our experiments the yield has been about four times as high as it is in roller bottle systems.



HOLLOW-FIBER PERFUSION REACTOR grows cells on a flat bed of porous plastic fibers .3 millimeter in diameter. The ends of the fibers are open to a supply of air and carbon dioxide, which moves through the fibers (*black arrows*); the outside of the fibers is bathed in a culture medium whose flow (*colored arrows*) is kept uniform by two micropore filters. Cells inoculated into the reactor attach themselves to the surface of the fibers and proliferate, in time forming what is in effect an artificial tissue that can be maintained for several months.

Efficient large-scale systems for the culturing of a wide range of mammalian cells in the absence of antibiotics and at a reasonable cost are necessary to meet the needs of investigative laboratories as well as for production on a commercial scale. It seems clear that effective new technology can expand knowledge of biological processes and also provide significant quantities of cell products that promise to be important therapeutic agents.



HUMAN CANCER CELLS (HeLa cells) are seen growing on the surface of one of the hollow fibers in this scanning electron micrograph. In addition to eventually covering the entire surface of the fiber the cells penetrate its lobed structure. The enlargement is some 250 diameters.

Machines That Walk

Locomotion on legs resists imitation, but modern control technology should be able to solve the problem. Experiments with machines that hop and crawl can also illuminate the mechanisms of natural walking

by Marc H. Raibert and Ivan E. Sutherland

Many machines imitate nature; a familiar example is the imitation of a soaring bird by the airplane. One form of animal locomotion that has resisted imitation is walking. Can it be that modern computers and feedback control systems make it possible to build machines that walk? We have been exploring the question with computer models and with actual hardware.

So far we have built two machines. One has six legs and a human driver; its purpose is to explore the kind of locomotion displayed by insects, which does not demand attention to the problem of balance. The other machine has only one leg and moves by hopping; it serves to explore the problems of balance. We call the first kind of locomotion crawling to distinguish it from walking, which does require balance, and running, which involves periods of flight as well. Our work has helped us to understand how people and other animals crawl, walk and run.

Unlike a wheel, which changes its point of support continuously and gradually while bearing weight, a leg changes its point of support all at once and must be unloaded to do so. In order for a legged system to crawl, walk or run, each leg must go through periods when it carries load and keeps its foot fixed on the ground and other periods when it is unloaded and its foot is free to move. This type of cyclic alternation between a loaded phase, called stance, and an unloaded phase, called transfer, is found in every form of legged system. As anyone who has ridden a horse at a trot or a gallop knows, the alternation between stance and transfer can generate a pronounced up-and-down motion. We believe legged machines can be built that will minimize this motion.

Our work and related work by others may eventually lead to the development of machines that crawl, walk and run in terrain where softness or bumpiness makes wheeled and tracked vehicles ineffective and thus may lead to useful industrial, agricultural and military applications. The advantage of legged vehicles in difficult terrain is that they can choose footholds to improve traction, to minimize lurching and to step over obstacles. In principle the performance of legged vehicles can be to a great extent independent of the detailed roughness of the ground. Our objective has been to explore the computing tasks involved in controlling and coordinating leg motions. It is clear that very sophisticated computer-control programs will be an important component of machines that smoothly crawl, walk or run.

As we have indicated, locomotion is possible with or without dynamic balance. The animals that crawl avoid the need for balance by having at least six legs, of which at least three can always be deployed to provide a tripod for support. High-speed motion pictures of insects show that they commonly crawl with an alternating tripod gait.

Although a crawling machine that does not need dynamic balance can be built with four legs, such a machine performs awkwardly because its weight must be shifted at each step to keep it from tipping over. Satisfactory performance without active balance calls for at least six legs, since six is the smallest number of legs that always provide a tripod for support even when half of the legs are elevated. Several six-legged machines have now been built, each differing in size and in mechanical design. All of them depend on computer control of the legs.

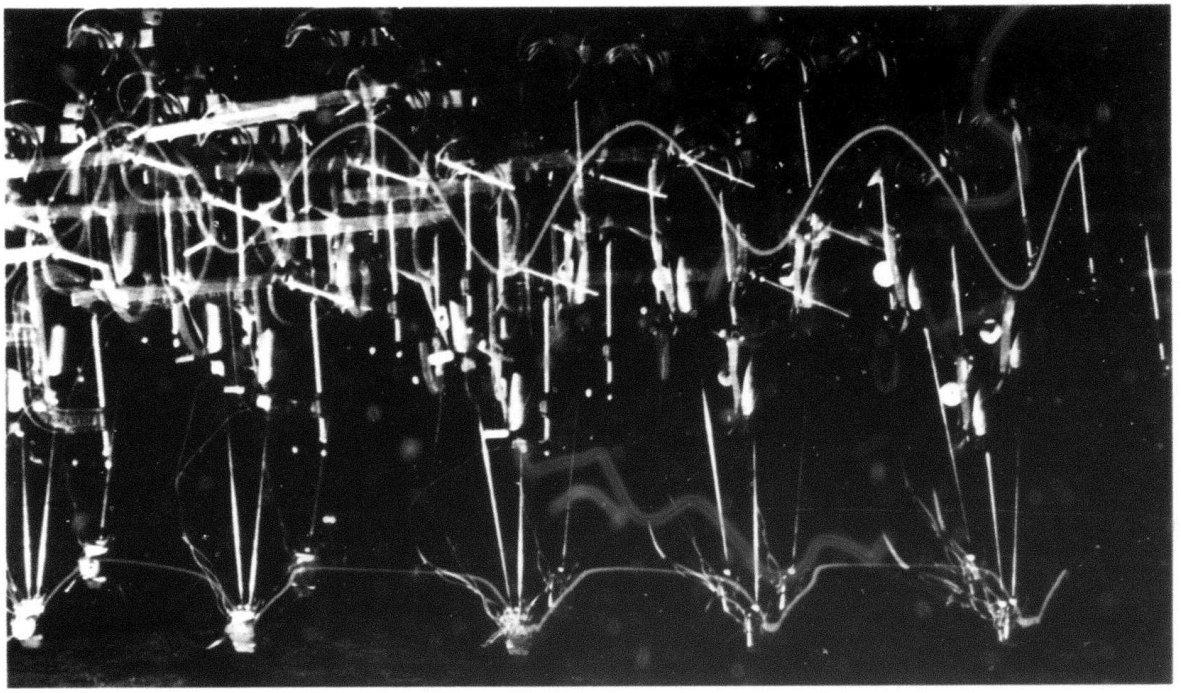
A computer program that controls such a machine accomplishes five tasks. First, it regulates the machine's gait, that is, the sequence and way in which the legs share the task of locomotion. Six-legged machines work with gaits that elevate a single leg at a time or two or three legs simultaneously.

The simplest gaits involve a regular sequence of leg motions. A gait can be described by noting the sequence. For example, the tripod gait can be recorded as (1,5,3;6,4,2;), with the commas designating the concurrent use of legs and the semicolons sequential use. Similarly, gaits that elevate a single leg at a time such as (3;2;1;4;5;6;) and (3;4;2;5;1;6;) are useful. A gait that elevates several legs at once generally makes it possible to travel faster but offers less stability than a gait that keeps more legs on the ground.

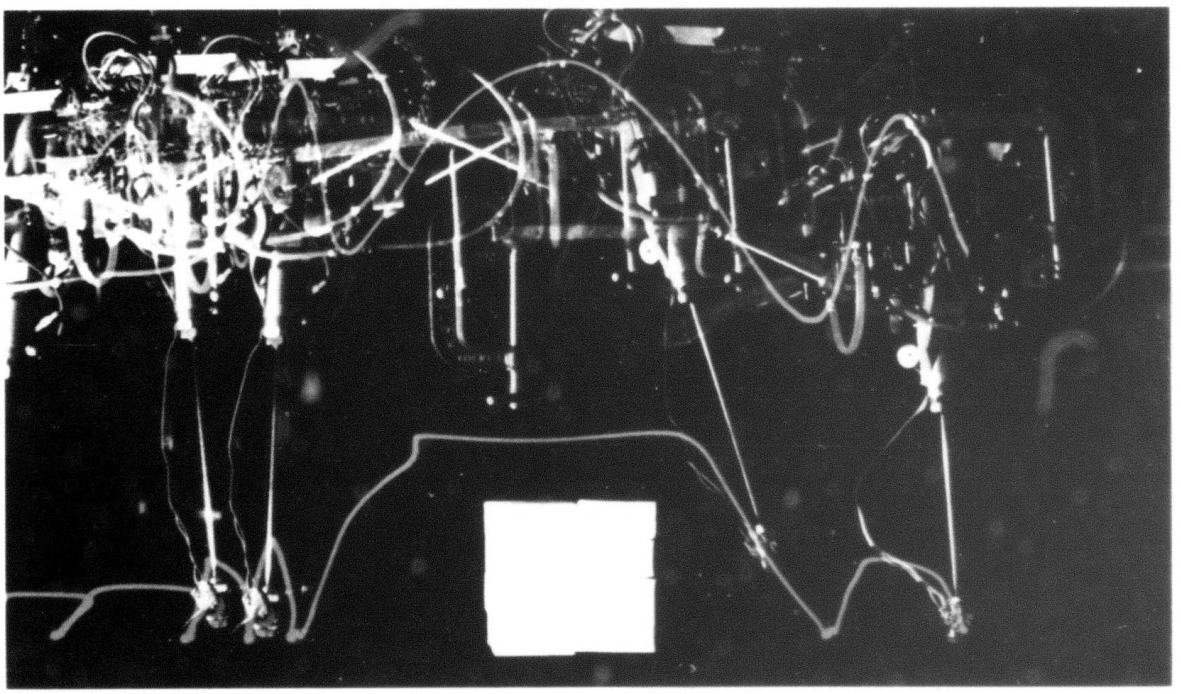
A second task of a computer program controlling a crawling machine is to keep the machine from tipping over. If the center of gravity of the machine moves beyond the base of support provided by the legs, the machine will tip. The computer must monitor the location of the center of gravity of the machine with respect to the placement of the feet to ensure that the base of support is always large enough. For simple gaits the geometry of the legs may suffice to keep an adequate base of support, but for more complex gaits a careful computation of static stability may be critically important.

Since many legs share the support of the machine, a third task of the control computer is to distribute the support load and the lateral forces among the legs. In the tripod gait, of course, the distribution of the support load is set by the geometry of the three supporting legs. With more than three supporting legs, however, the control computer must decide how to manage the distribution of loading in order to achieve higher-level objectives such as smoothness of ride and minimal disturbance of the ground.

Even when only three legs are supporting the machine, the control program must distribute the lateral foot forces. One way of looking at this task is to consider that the control system must keep the machine from simply doing isometric exercises against the ground. The amount of sensing and computation that is needed to distribute the lateral loads among many legs can be formidable. We have reduced this burden for the crawling machine we are building by providing passive hydraulic circuits that automatically distribute the sideways loads.



HOPPING MACHINE was built by one of the authors (Raibert) to explore the problems of controlling a machine that must balance as it moves. Its leg is actuated by compressed air and its motions are controlled by a computer that obtains feedback from position sensors. This version is held by a tethering arm and so balances in a single plane. A series of hops is recorded in this photograph, which was made while the camera lens was kept open. Red lights attached to the body and foot of the machine delineate the path of the hops.



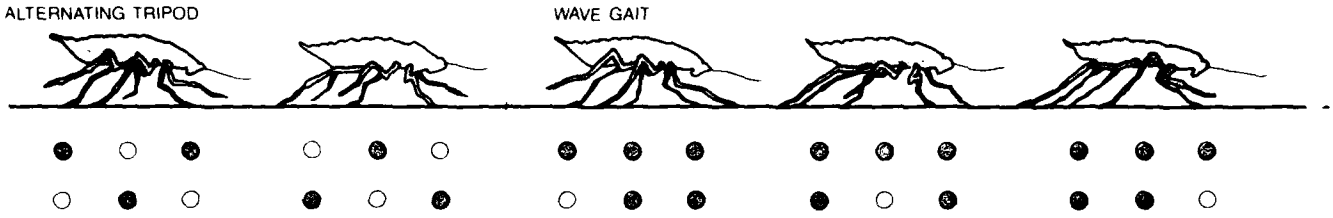
JUMPING MODE of the hopping machine shows it leaping over an obstacle. The machine approaches the obstacle from the right. When it is one step away, the operator pushes a "leap" button. As a result the maximum tension is generated in the drive actuator so that the altitude of the next hop will be increased. In flight the leg is shortened and its normal swinging motion is delayed to provide better clearance of the obstacle. A servomechanism controlling balance moves the leg to the correct landing angle and the leg is lengthened in preparation for landing. Thereafter the machine continues its normal hopping. The obstacle was 15 centimeters (six inches) high.

A fourth task of the control computer is to make sure the legs are not driven past the limits of their travel. The geometry of the legs may make it possible for one leg to bump into another; if legs can collide, the computer must limit their motion to prevent damage. To maximize the usefulness of each leg its placement on the ground must take into account the limits of the leg's motion and the expected motion of the machine during that leg's stance period. For example, if the machine is turning to the right, the forward legs should be placed farther to the right so that their sideways travel can be accommodated during the turn. For a vehicle with autonomous control the placement of the legs can be based on the planned future path of the vehicle. For a vehicle with a human driver the proper placement of each leg requires a prediction of the driver's commands for the next stance period.

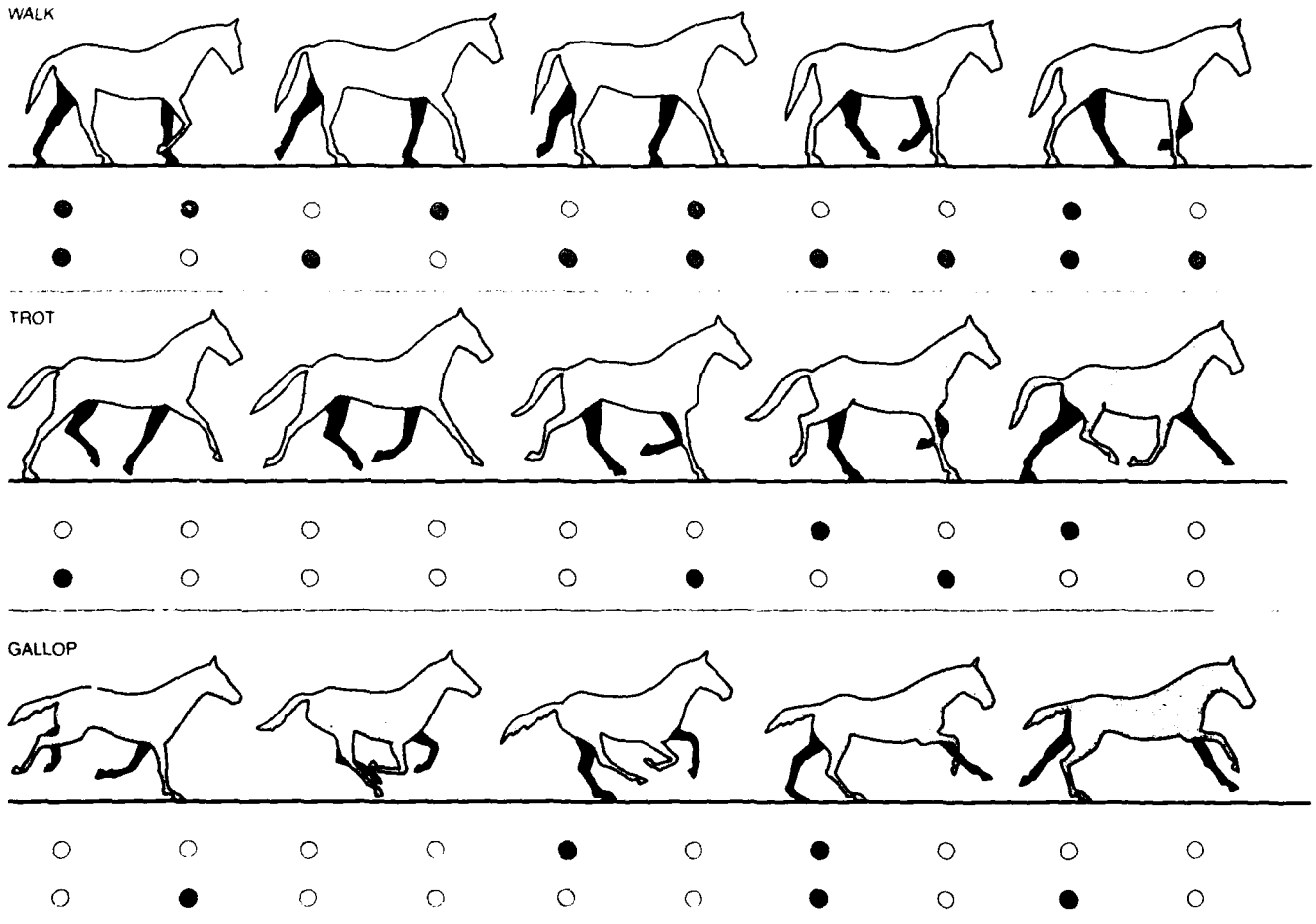
A fifth task for the control computer is to choose places for stepping that will give adequate support. On smooth ground the task is easy, but on rough terrain it may be exceedingly difficult. No system has yet been built that accomplishes this task. One can envision a terrain-scanning system that would survey the ground ahead of the machine and choose likely footholds. To make use of such a scanner the control computer would build an internal digital model of the terrain. Such a model would have to account only for bumps that are about the size of the machine's feet or bigger. Human input to the model might help in the evaluation of possible footholds.

One of us (Sutherland) is building a six-legged, hydraulically driven crawling machine. A gasoline engine provides its power and hydraulic actuators move its legs. There are six legs, so that dynamic balance is not needed.

A built-in microprocessor controls the legs by switching on or off the valves that regulate the flow of oil to the hy



GAITS OF INSECTS provided a basis for designing the method of locomotion of the six-legged *crawling machine portrayed on* the cover of this issue. The circles below each drawing show whether the corresponding leg is on the ground or in the air: a filled circle represents



GAITS OF A HORSE represent the kind of locomotion in which balance is a factor. In the walk at least two of the horse's legs touch the ground at all times. In the trot and the gallop the animal periodically leaves the ground. The drawings are based on the stop-motion pho-

draulic actuators. Sensors in each leg report its position and the forces acting on it to the microprocessor. The machine is large enough to accommodate a human driver, who controls its speed and direction of motion and establishes the tilt of its body and its ground clearance. The vehicle's design speed is about two miles per hour.

One objective in the design of the vehicle was to minimize the amount of computation required to obtain a crawling movement. The hydraulic circuits are designed to make the legs move along useful paths without attention from the microprocessor, which merely selects one of the available paths for each leg. Thus the microprocessor is free to concentrate on selecting which legs to use for support and on deciding where to step next; it does not have to spend time computing the details of leg motion.

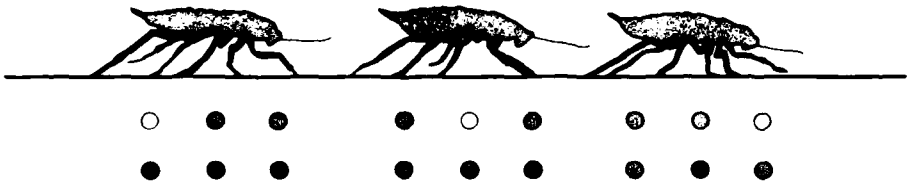
Each leg of the machine can swing fore or aft and up or down on the universal hip joint that attaches it to the frame of the machine. These motions are executed by lengthening or shortening the two hydraulic actuators per leg that are arranged in a *V* configuration above the leg. One setting of the valves provides that oil leaving one actuator will enter the other, so that as one actuator shortens, the other actuator lengthens by the same amount. Because of the geometry of the pivots this connection provides horizontal leg motion.

The horizontal motion can be powered or unpowered depending on the valve settings, so that some legs can serve to drive the machine forward while others coast. As legs are placed on the ground and accept load they are able to coast forward or backward as dictated by the motion of the legs already on the ground and driving. Hence the control computer does not need to compute the precise instant when a leg will touch the ground or the details of the motion required at the time of contact to obtain a smooth forward motion.

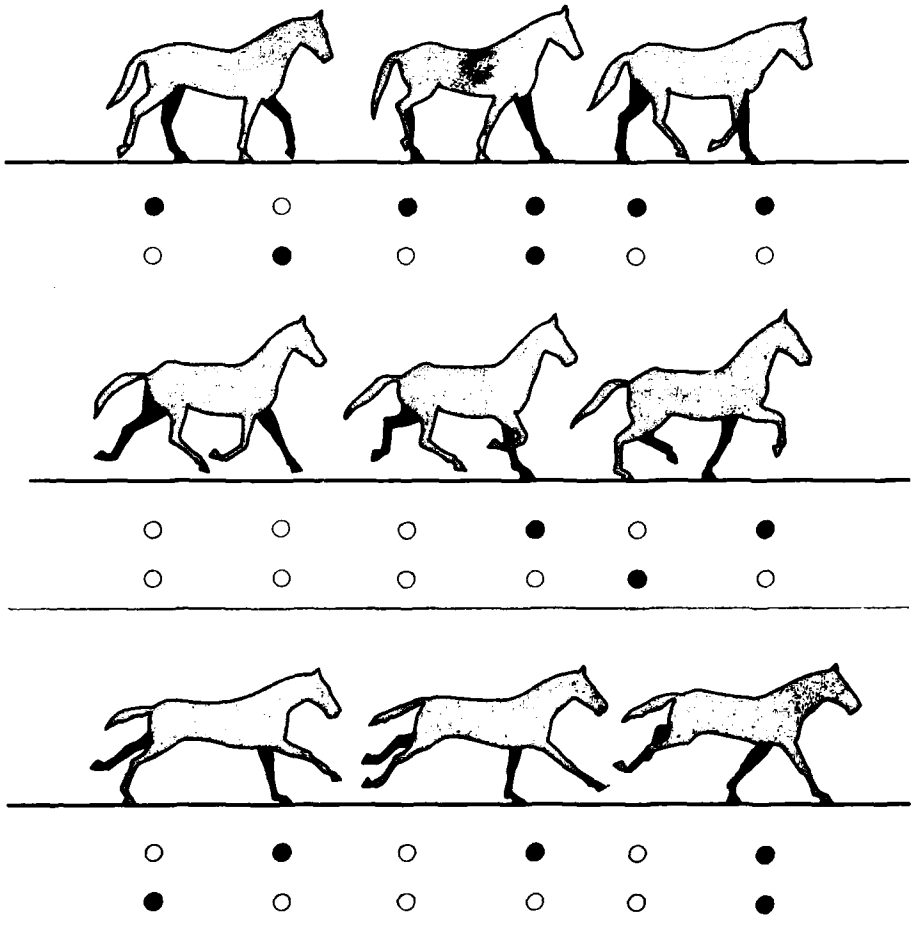
The knee joint of each leg is powered by a separate hydraulic actuator mounted horizontally along the leg. This actuator can be powered while the leg is raised in order to position the foot sideways for the next step. When the foot is on the ground, the knee joint must move slightly to match the circular path of the knee about the hip to the straight path of the foot on the ground. It is a complex motion, but it does not call for action by the computer; instead a simple parallel connection of the knee-joint actuators enables all the knees to accommodate to the average motion of the vehicle. An additional hydraulic pump provided in the system can force a collective sideways motion of all the knee joints, making the machine crawl sideways like a crab.

The human driver of the machine has three kinds of control. First, he can regulate the amount of oil flowing in the system because he can control the displacement of the hydraulic pumps. Separate pumps are provided for the legs on the left and right sides so that the driver can steer by making the machine crawl faster on one side than on the other. The settings of the steering controls are reported to the microprocessor so that it can position the feet properly. For example, if the machine is turning to the right, the front feet must be transferred to the right and the rear ones to the left to accommodate to the turn. If the machine is walking backward, which is achieved by reversing the flow of oil, the feet must be transferred backward with each step rather than forward. As each foot is lifted from the ground the control computer picks a target position for it based on the current rate and direction of oil flow set by the driver. When a supporting foot nears the limit of its travel, the control computer initiates its lift and transfer to a new foothold. If any supporting foot actually reaches the limit of its travel, the microprocessor stops the vehicle until that foot can be lifted from the ground and transferred to a new foothold where it has room to move.

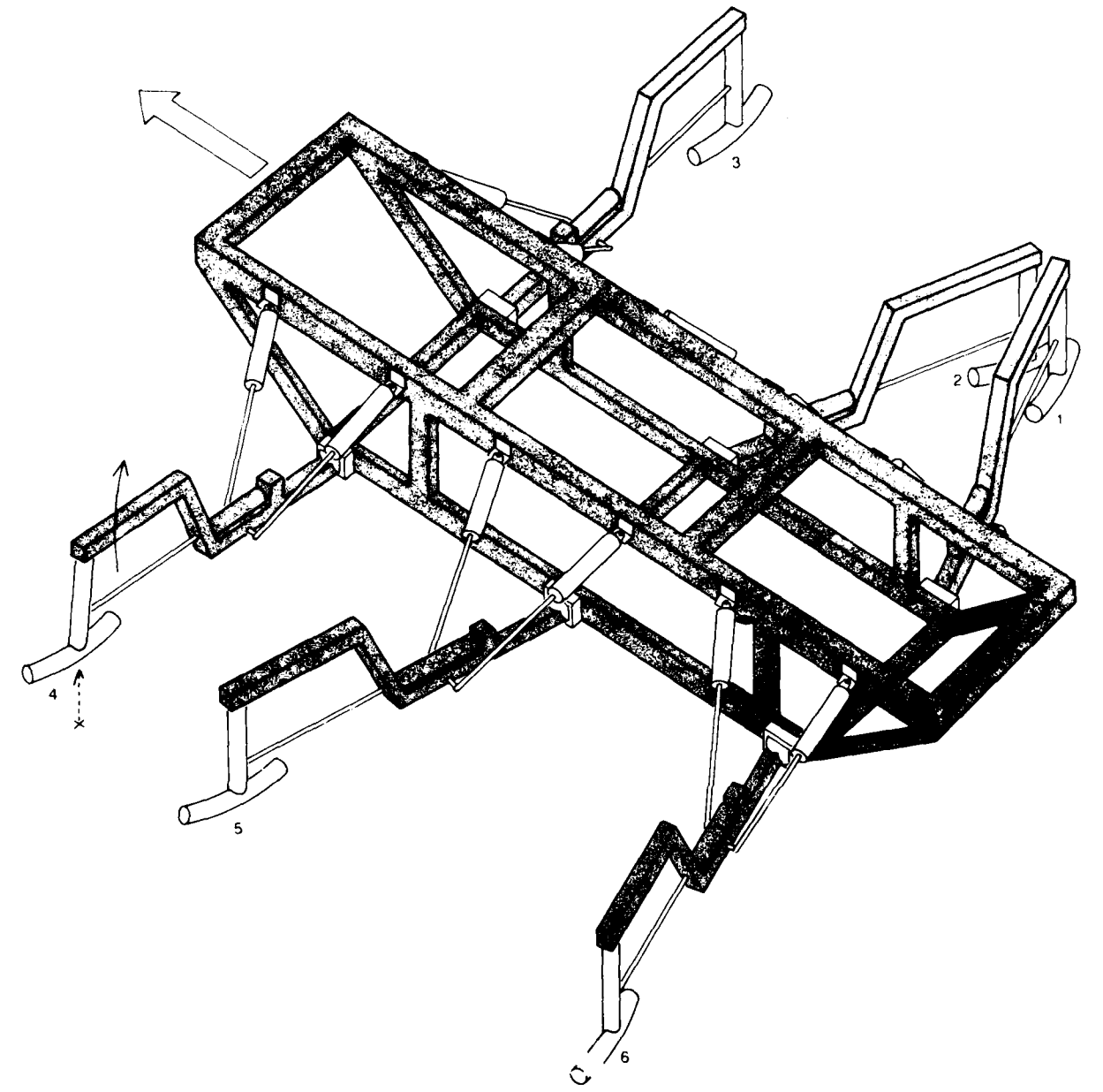
The second kind of human control of the vehicle establishes the attitude and

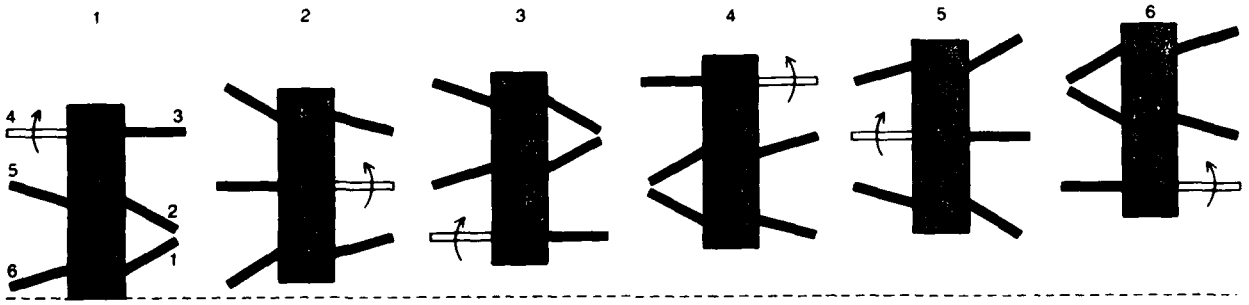


a leg touching the ground, an open circle a leg in the air. The alternating-tripod gait always provides stability. In creeping with a wave gait the insect's adjacent legs move successively.



tographs made by Eadweard Muybridge 100 years ago that settled a long-standing debate on whether a horse in a trot leaves the ground. Many other animals also do so in running.





SIX-LEGGED MACHINE built by one of the authors (Sutherland) moves by crawling. It does not have to balance. The six legs are controlled by a built-in microcomputer. Power comes from an 18-horsepower gasoline engine that drives separate hydraulic pumps for the left and right legs. A human driver steers the machine by making the oil flow at different rates on the two sides. Sensors report the driver's commands as well as the position of each leg and the forces on it to the microcomputer, which employs the information to choose the order and path of leg motion. Six legs ensure stability because at least three are always on the ground. Passive hydraulic circuits simplify the computing task; a leg that is supporting weight can either be connected to the drive unit or can coast, being pushed by the ground according to the motion generated by the other legs. The diagram at the bottom indicates the position of the legs in a walking cycle, which can be denoted as (4;2;6;3;5;1;). A solid rectangle represents a leg touching the ground and an open rectangle represents a raised leg moving forward as indicated by the arrow. In the numbered walking cycle the semicolons denote the sequential use of the machine's six legs.

ground clearance of the machine. The driver can set a control that changes the vertical support position for the left and right feet to make the vehicle roll left or right. Similarly, he can indicate different vertical support positions for front and rear feet to make the machine pitch forward or back. Another control enables him to indicate that the vertical support positions for all six legs should be raised or lowered collectively to change the ground clearance of the vehicle.

The third kind of human control will achieve careful placement of the feet for operation on very rough terrain. We have not yet decided how to provide this kind of control. A walking truck built some years ago by Ralph Mosher at the General Electric Company depended exclusively on manual control of foot placement and was therefore quite tiring to drive. We believe selection of the gait may also be important, but we have not yet had enough experience to know whether it could be done automatically or whether human inputs will be needed. It is precisely to answer such questions that we have built the machine.

The other subject of our attention is walking and running where balance plays a role. Until a century ago people still debated whether or not a horse in a trot had all its legs off the ground simultaneously. The stop-motion photography of Eadweard Muybridge settled the debate, showing that a horse does leave the ground entirely during a trot. A running person does so too, as do the dog, the cheetah and of course the kangaroo. Such animals not only walk, which requires dynamic balance, but also run, employing ballistic motions effectively to increase their rate of travel.

There are two fundamental differences between a crawling vehicle that is statically balanced and one that is dynamically balanced. The first difference is in the definition of stability. A crawling vehicle is stable if its legs provide at least a tripod of support at all times to ensure that it does not tip over; a dynamically balanced walking or running vehicle can be allowed to tip for brief intervals. Motions of the legs and the body ensure that a single tipping interval is brief and that an adequate base of support is maintained on the average. For example, a running man touches the ground alternately with his two legs, providing a base of support for his body only over time.

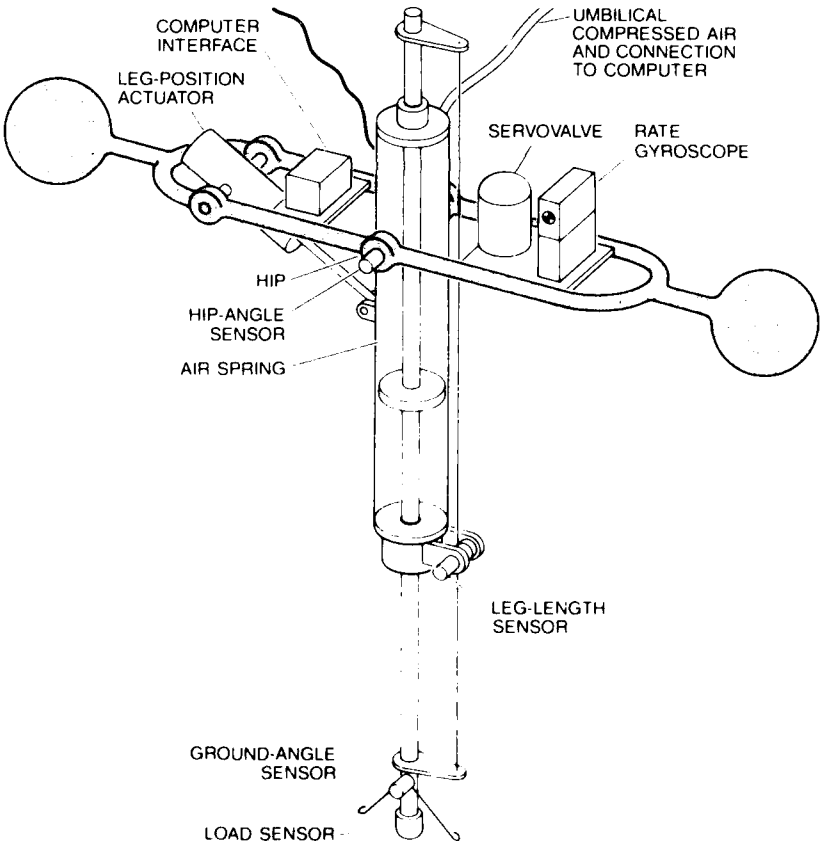
The second difference between static and dynamic balance is in the consideration of speed and momentum. Static balance assumes that the configuration of the supporting legs and the position of the center of gravity are adequate to specify stability; it ignores the vehicle's motion. Such static computations are not always sufficient. For example, a fast-moving vehicle might tip forward if it stopped suddenly with the center of gravity too close to the front legs. In order to understand the greater mobility of walking and running systems one must both relax the definition of stability and account for velocity in computing balance.

It is to study the problem of balance in its simplest form that one of us (Raibert) and his co-workers at Carnegie-Mellon University have built and demonstrated a machine that hops on its single leg and runs like a kangaroo, in a series of leaps. The device can be thought of as a computer-controlled pogo stick. We have been encouraged by the remarkable simplicity of the balancing algorithm. In its present form the machine is limited to movement in a single plane, so that it can tip over in only one direction.

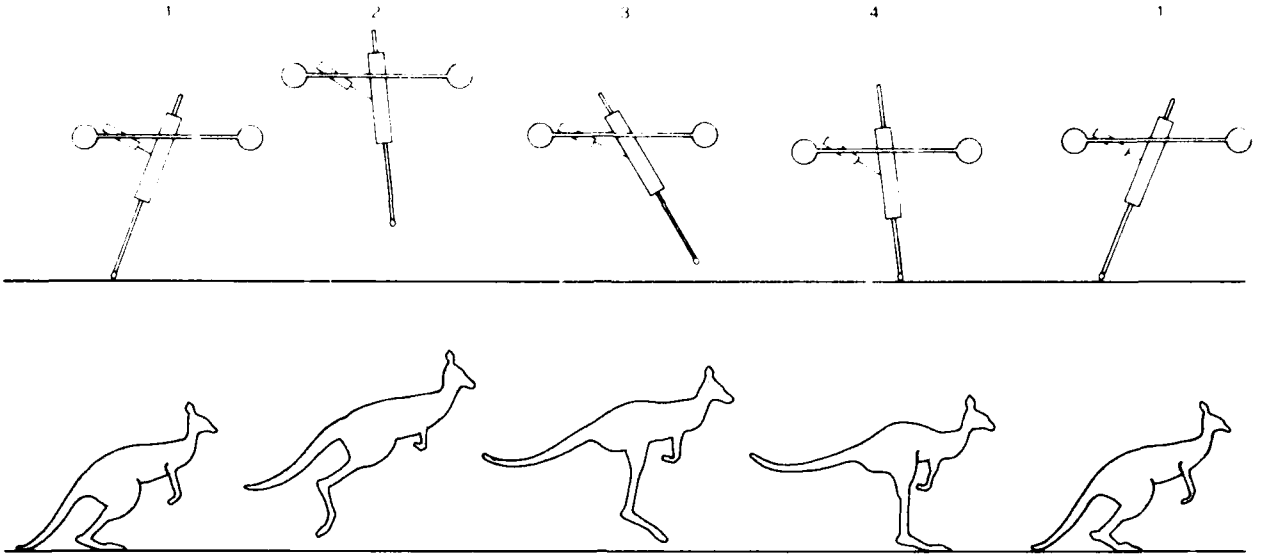
The machine has two main parts: a body and a leg. The body provides the main structure and carries valves, sensors and electronics. The leg is a simple mechanism that not only changes length along its axis but also pivots with respect to the body at a hinge called the hip. The leg bounces on a spring with adjustable tension, much like a human leg with its springy muscles and tendons. The spring is an air cylinder in which pressures are controlled with sensors and valves. At the bottom of the leg is a small foot.

The pivoting motion of the leg is controlled by a second air-operated actuator that applies torques at the hip hinge. A simple on-off valve controls the leg spring, but control of the pivot angle of the leg requires a proportional servovalve, that is, a feedback device that responds in proportion to the strength of the signal it receives. Because the moment of inertia of the leg is less than 10 percent of the body's moment of inertia the leg can pivot during flight without imparting much motion to the body. The tilt of the body is measured by a gyroscope, enabling the control computer to maintain the body in a level attitude. Other sensors measure the angle of the hip, the length of the leg, the air pressure in the leg spring, the angle between the leg and the ground and the force of the leg's contact with the ground.

Three separate servo-control loops



TWO-DIMENSIONAL HOPPING MACHINE serves to study the problems of controlling motion with balance. The machine normally operates while leaning parallel to a tilted wall, separated from it by a cushion of air; the machine can tip only in the plane defined by the wall. A computer receives data from the angle sensors, a pressure sensor and a switch on the foot. It controls hopping by adjusting the pressure in the pneumatic chamber that functions as a spring. The computer also applies torque between the leg and the body to regulate the angle of the hip. This angle determines the horizontal displacement of the foot and therefore influences balance.



HOPPER IN MOTION operates cyclically, as all legged systems do; the leg alternates between periods of support and periods of flight. At the left the machine is about to begin a leap. While it is in the air the leg swings forward at the hip in preparation for the next landing. At touchdown the leg spring shortens to its minimum length to provide for the next leap. A ground-contact sensor acts as a trigger for the vertical-control program. The machine also has feedback loops to control attitude and balance in synchrony with the vertical control. Like a pogo stick, the machine can balance only while it hops. The hop is like the kangaroo's movement Muybridge called a ricochet.

regulate the machine. One loop controls vertical motion, one balance and one body attitude. Each loop is synchronized with the basic hopping motion.

The first loop controls the height of the hopping motion. It adds or removes energy from the motion in order to achieve the correct hopping height and makes up for the energy lost during each hop. The height control does both tasks by periodically adding air to or releasing it from the main drive cylinder to adjust the effective tension of the air spring. In other words, the height control governs the timing and the magnitude of the power delivered to the hopping drive mechanism, thereby achieving the desired hopping height. When a desired hopping height has been achieved, most of the energy needed for the next hop is recovered from the spring, in which it was stored during the previous landing. As long as the hopping motions are relatively stable the task of managing the hopping energy of the machine is not particularly difficult.

The second servo-control loop provides for the balance of the machine by positioning the foot while the machine is in flight so that the next landing is made in a balanced posture. The calculation of the correct foot position takes into account both the forward speed of the vehicle and the inclination of the body. A single computer algorithm for balance works when the machine is hopping in place, accelerating to a run, running at a constant velocity, leaping over objects and slowing to a stationary hop.

When the machine is hopping in place, the leg and foot are moved small distances to compensate for external disturbances and the errors of previous hops. When the machine is to start running, say to the right, the foot is moved first to the left to unbalance the vehicle so that it starts to tip in the desired direction. Stable running is just like hopping in place except that the balancing adjustments supplement large sweeping motions of the leg, which are determined by the rate of travel. Stopping is much like starting except that the machine is made to tip in the direction opposite to the direction of movement.

The third servo-control loop stabilizes the attitude of the body to keep it upright. It provides torques between the leg and the body while the foot is on the ground in order to achieve the desired attitude during the next flight. The effectiveness of this servo depends on good traction between the foot and the ground. The attitude servo that operates when the foot is on the ground shares the hip-drive mechanism employed by the balance servo that operates while the machine is in flight. Certain subtle details of the change from one control mode to the other are associated with detecting the start and finish of each flight. The torquing mechanism must be idle during these events lest the foot slip on the ground.

When an animal runs, its legs swing back and forth through large angles to provide balance and forward drive. We have found that such swinging motions of the leg do not have to be explicitly programmed for a machine but are a natural outcome of the interactions between the controllers for balance and attitude. Suppose the vehicle is traveling at a constant horizontal rate and is landing with its body upright. What must the attitude controller do during stance to maintain the upright attitude? It must make sure that no torques are generated at the hip. Since the foot is fixed on the ground during stance, the leg must sweep back through an angle in order to guarantee that the torque on the hip will be zero while the body moves forward.

On the other hand, what must the balance servo do during flight to maintain balance? Since the foot must spend about as much time in front of the vehicle's center of gravity as behind it, the rate of travel and the duration of stance dictate a forward foot position for landing that will place the foot in a suitable spot for the next stance period. Thus during each flight the leg must swing forward under the direction of the balance servo, and during each stance it must sweep backward under the control of the attitude servo; the forward and back sweeping motions required for running are obtained automatically from the interplay of the servo-control loops for balance and attitude.

We are now building a version of the machine that will balance in three dimensions and therefore be able to move around on an open floor. We have written and tested a computer simulation of the motion of such a machine and have found that control in three dimensions can be broken down into the same three servo-control loops we have described.

Our work in making the one-legged machine run was greatly aided by a

thought that came to us as we went along. It was that running can best be understood by breaking it down into the three parts we have discussed here: height control, balance control and attitude control. Partitioning the control into these three parts has made the complex behavior of legs in walking and running much easier to understand. This insight has led to a fairly simple control system that makes the one-legged machine balance and run.

Our success in this effort encourages us to think about building dynamic-motion machines with more than one leg. We believe the right way to think about such machines is to focus first on their up-and-down and balancing behavior, postponing the complications introduced by forward motion. The notion that hopping is the main activity was natural to the one-legged machine and provided an effective way to think about its behavior, but it seems less natural for machines with several legs. Perhaps it seems less natural because we are accustomed to seeing animals run and want to understand their behavior all at once.

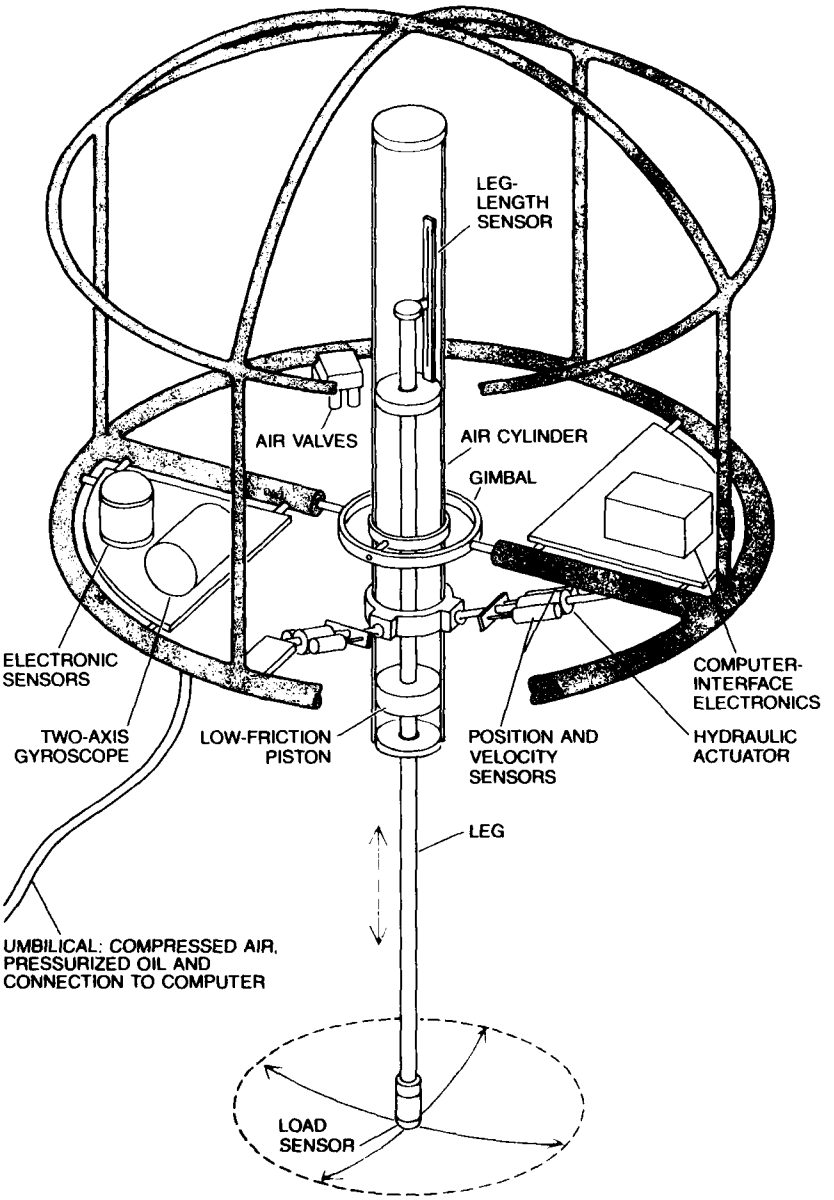
A four-legged machine hopping in place might use any of several sequences of leg activity. The simplest pattern would be to hop simultaneously on all four legs. It is not hard to imagine that the same three servomechanisms that control the one-legged hopping machine might control the motion of a four-legged vehicle in this mode. In fact, the attitude-control loop that keeps the body upright might be substantially simpler because of the broader base of support. When the machine moved forward, the legs would swing together in a pattern of motion that could easily be generated by the same control mechanisms as those that serve the one-legged machine.

Another possible gait for a four-legged machine hopping in one place is bouncing on diagonally opposite pairs of feet. Again it does not stretch the imagination too much to see how one might separate the control of such a vehicle into a height control, a balance control and a body-attitude control. The height control would add energy to the hopping motion to keep the hopping height at the desired level. The balance control would position the two raised legs in such a way as to maintain balance. The body-attitude control would apply appropriate torque to the pair of legs on the ground. The attitude and balance controls would alternate in the use of the same leg actuators, as they do in the one-legged machine. Moreover, just as we have found in that machine, forward motion could easily be accommodated by moving each raised foot to a forward position chosen to make the average balance force of the leg zero during the next stance period. The resulting motion is a trot of the kind common among four-legged animals.

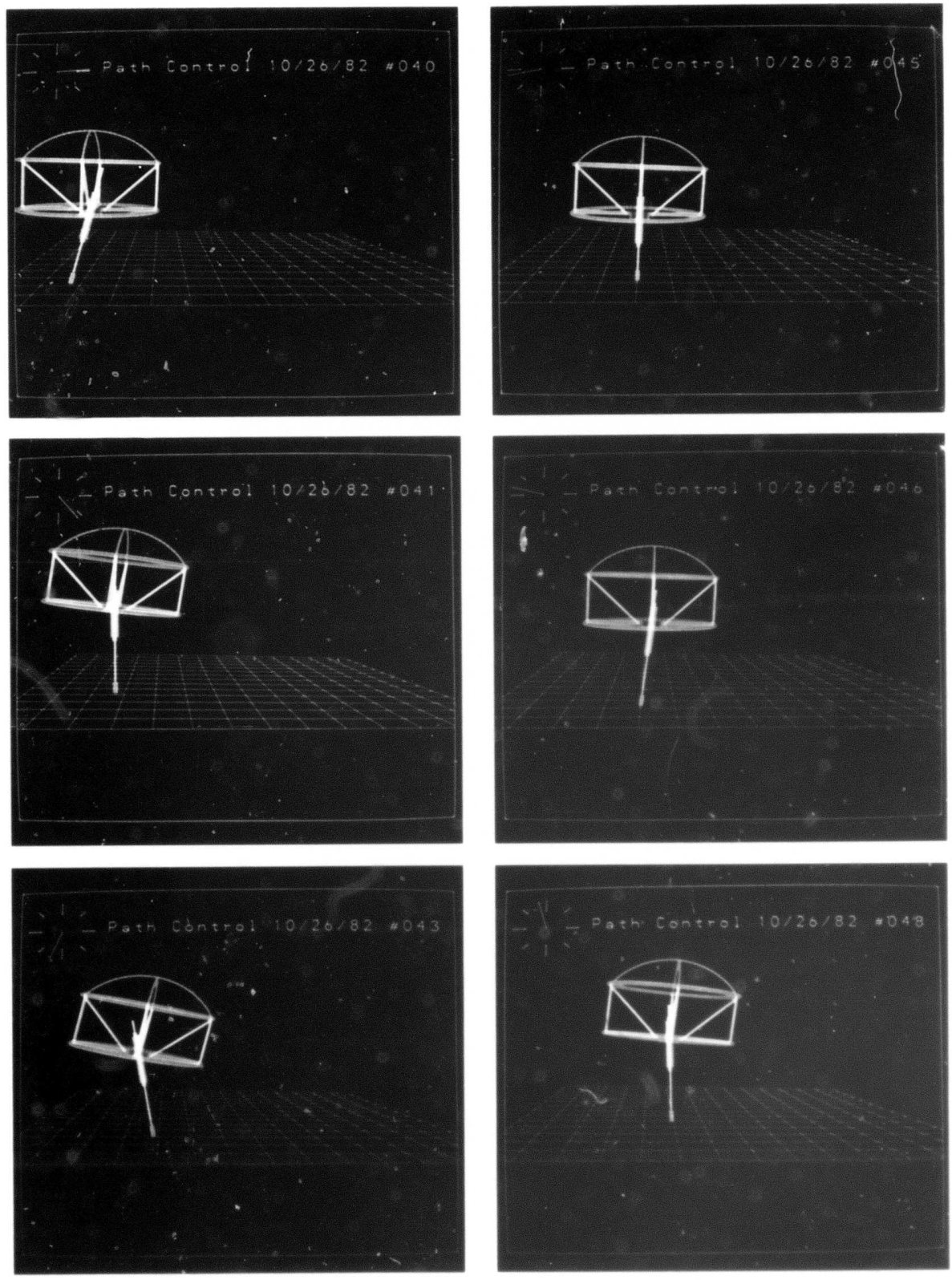
Two other gaits can similarly be understood by separating the control of each leg into vertical, balance and body-attitude components. In the gallop the rear legs land slightly sooner than the front ones. The body attitude is allowed to change during flight so that a nose-up attitude is seen as the rear legs touch down and a nose-down attitude develops as the front legs take off. The bound is a variation of the gallop in which the front legs operate nearly in unison and so do the rear legs but the front and rear actions are equally spaced in time. It is the bound that enables the cheetah to sprint at speeds of more than 60 miles per hour.

Efficient motion over the ground requires that little energy be lost during each motion of the machine. We have already mentioned how the vertical motions of legs can be made efficient by storing energy in elastic elements. What about fore-and-aft leg motions?

At high speeds over the ground the legs of a vehicle will have to move forward and back quite rapidly. Most of the energy expended by a running animal goes into generating these leg mo-



ADVANCED HOPPING MACHINE now under development is designed to operate in three dimensions. It is about one meter high, weighs 20 kilograms and is connected to a nearby computer. Compressed air provides the hopping power and regulates the height of hopping. The actuators that position the foot are hydraulic. During flight they position the foot to maintain balance. When the foot is on the ground, they maintain the machine in an upright body posture.



SIMULATED MOTION of the three-dimensional hopper is shown in these photographs from the display of the computer that worked out the motion. Here over a period of approximately .7 second the machine is shown balancing itself while it lands and takes off again.

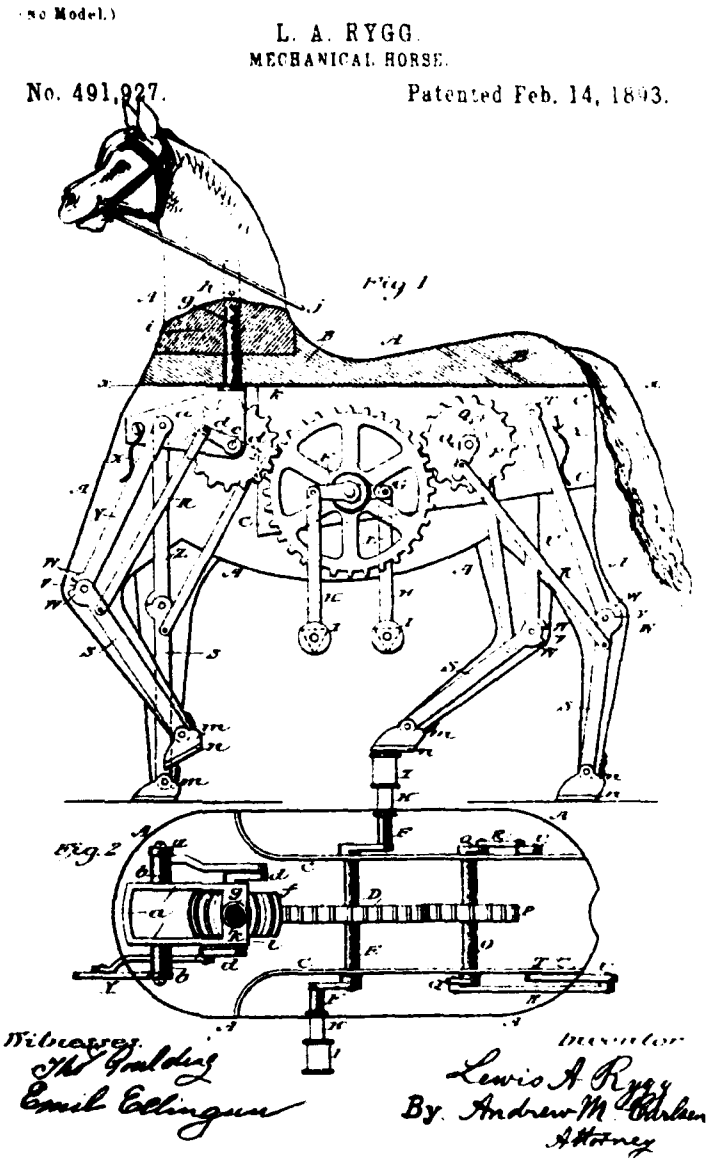
tions. In our one-legged machine these motions are provided by a conventional proportional servomechanism. In such a system the kinetic energy of the leg as it swings is entirely lost as the leg is brought to rest momentarily at each extreme angle. The hopping motion, on the other hand, is obtained by a self-resonant system made up of the leg spring and the mass of the machine, so that the height servo need only add or subtract a small amount of energy to maintain the hopping height. It is obvious that if a machine with several legs is to be made efficient as well as fast, it will have to incorporate some kind of self-resonant system for the fore-and-aft motion of the legs as well. One might design such a mechanism with springs between the legs to make the legs oscillate like a tuning fork at a frequency appropriate to the vertical bouncing rate.

Although we believe we understand how to build a four-legged machine that can run with any of the common gaits we have described, there remain many interesting questions associated with starting and stopping such a machine and selecting its gait. We can easily see how to start forward if the machine is already hopping in place. What is much less obvious is how to coordinate the transition from a standing start to full-speed running. Similarly, how and when should such a machine change from one gait to another? A running horse switches its lead as it turns, that is, it changes which of the two front legs slightly precedes the other. What computations should be done to make such minor changes in the pattern of leg motion? We find these problems fascinating, both as engineering questions in the form "What should we build?" and as scientific questions in the form "How do living systems work?"

A much more difficult problem is how to choose footholds for the machine. The function of vision in walking and running by people and animals, particularly the ability to choose sensible places to put the feet, is not well understood. One can imagine avoiding this problem by having the machine run fast only over smooth ground and by having some kind of human assistance to choose a safe path. It will probably be desirable to scan the ground ahead of the vehicle for holes. Still, just as a galloping horse runs the risk of stepping into a gopher hole, so we must expect that a running machine will also get into that kind of trouble.

The mobility of off-road vehicles is limited by two factors. First, the continuous footprint of wheels and tracks prevents wheeled and tracked vehicles from making use of the discontinuous points of support that are available to a legged vehicle. We are encouraged to think that the state of the computer art is now sufficiently advanced to allow the construction of adequate control systems for legged vehicles, and so the legged alternative for high-mobility vehicles can be seriously considered. Indeed, the Defense Advanced Research Projects Agency is already sponsoring research on such vehicles, including partial support for our projects.

A second source of mobility is narrow width: a motorcycle can get into places a jeep cannot reach. Narrow legged vehicles can be built, but they will have to balance, at least in the sideways direction. Involved as we are both in the construction of a six-legged vehicle that can crawl without attention to balance and in studies of walking and running with balance, we believe the effort to understand balance is much more important. We think the experiments with six-legged crawlers now under way in our laboratory and elsewhere are mainly exercises in the control of multiple legs and are not in themselves useful: such crawlers will ultimately be replaced by machines with fewer legs that can balance. Mastery of balance will be the key to building high-mobility machines that walk and run.



MECHANICAL HORSE was the subject of a patent obtained by Lewis A. Rygg in 1893. The lower drawing is a plan view along the line *x–x* of *Fig. 1*. The stirrups doubled as pedals that were to enable the rider to power stepping motions. Steering was to have been done with reins that moved the head and forelegs from side to side. Apparently the machine was never built. It would have been similar to many modern walking toys. Since they have no sensing or computing facilities, they cannot adapt to variations in terrain. They only crawl on flat surfaces.

The Hair Cells of the Inner Ear

They are exquisitely sensitive transducers that in human beings mediate the senses of hearing and balance. A tiny force applied to the top of the cell produces an electrical signal at the bottom

by A. J. Hudspeth

The sense of hearing, the sense of balance that enables human beings to walk upright, the ability of certain animals to detect vibrations of the ground and the ability of fishes to detect the displacement of water would appear to have little to do with one another. Actually the four senses are closely related. Indeed, each of them is made possible by the same sensory receptor. The receptor is called a hair cell and is named for the hair bundle, a group of fine projections that extend from its upper surface. The hair cell is an extremely sensitive mechano-electric transducer, that is, it converts a mechanical force into an electrical signal. The mechanical force is the stimulus applied to the hair bundle; the electrical signal is the message relayed to the brain.

Each hair cell is sensitive to only a limited range of stimuli. Therefore if the organism is to obtain useful information about its environment and about its own movements the output from thousands of receptors must be combined. The requisite number of hair cells are found in several small sensory organs in the inner ear. In human beings arrays made up of thousands of hair cells with slightly different sensitivities are found in six organs in each ear. The combined responses of the cells yield information about linear acceleration in any direction, about angular acceleration along three perpendicular axes and about acoustic tones with a wide range of frequencies.

The general structure of the hair cell and the receptor's sensory role have been known for many years. How the individual hair cell works, however, is only now becoming clear. To examine events on the cellular scale I have worked with single hair cells from the inner ear of the bullfrog. A microscopic probe is employed to push the bundle back and forth as an electrode records the cell's electrical output. With this setup I have obtained the first recordings made from single hair cells during the application of a precisely controlled mechanical stimulus directly to the hair bundle.

Each hair cell can respond to remarkably small stimuli. Hair cells from mammals begin to respond when the tip of the hair bundle is moved no more than 100 picometers (trillionths of a meter); this distance is about the same as the diameter of some atoms. In addition, recordings from isolated hair cells show that each receptor has one direction of maximum sensitivity. When the hair bundle is displaced in any one direction, the cell responds only to the component of motion that is in the direction of maximum sensitivity. Along with light microscopy and electron microscopy recordings from individual hair cells are beginning to yield a rich understanding of the workings of the sensory receptors of the inner ear. Many questions nonetheless remain, among the most intriguing of which is: What is the exact molecular mechanism whereby the displacement of the hair bundle changes the electrical properties of the cell and causes a message to be sent to the brain?

The hair cell is generally either cylindrical or flask-shaped and is always found as part of an epithelium, a sheet of cells that is at most a few cells thick. It is closely related to the neuron, or nerve cell, although it does not have axons or dendrites, the fibers that extend from neurons and transmit electrical signals in the nervous system. For this reason the hair cell is sometimes referred to as a paraneuron. In the epithelium where the hair cell is found its apical, or upper, end lies flush with the upper ends of the supporting cells that surround it. The apical ends of the hair cells and those of the supporting cells form a smooth surface above which the hair bundle extends. In different cells the length of the hair bundle ranges from about three micrometers (millionths of a meter) to more than 100 micrometers.

The detailed structure of the hair bundle differs among the hair cells of different species and even among hair cells from different organs of the same individual. The basic form of the hair bundle, however, is the same in all vertebrate animals. Each bundle consists of from 30 to 150 thin, rod-shaped extensions. The extensions are of two kinds; they are both called cilia but nonetheless have quite different internal structures. All but one of the extensions are called stereocilia. Stereocilia are cylindrical or club-shaped organelles with a core made up of tightly packed filaments. The plasmalemma, the surface membrane of the cell, extends over the filamentous core of the stereocilium much as the finger of a glove covers the human finger. The diameter of the stereocilium ranges in different cells from about .2 micrometer to one micrometer.

In spite of the term, stereocilia are not true cilia. A true cilium, such as the cilium of a sperm cell, has a complex and highly differentiated central structure called the axoneme. Most true cilia are capable of an independent motion much like that of an oar in rowing; the motion originates in the axoneme. The hair bundle has one true cilium: the kinocilium. It is about .25 micrometer in diameter. Like the axoneme of other true cilia, the axoneme of the kinocilium consists of two central tubules surrounded by nine "doublet" microtubules, pairs of small tubes that share a central wall. The kinocilium is not, however, capable of independent motion in the hair bundle. At the end away from the main body of the cell it is usually attached to the adjacent stereocilia.

The arrangement of the cilia in the hair bundle is highly regular. The stereocilia are in a hexagonal array: each one is surrounded by six others that are equidistant from it. The stereocilia are not of equal length. The hair bundle is circular in cross section and along one diameter of the circle there is a progressive increase in the length of the stereocilia. Along any axis perpendicular to that diameter, however, the cilia are the same length. Thus the hair bundle has ranks of cilia of equal length that are arranged in order of size. As a result of this arrangement the hair bundle has a plane of bilateral symmetry. In the intact cell the stereocilia are not uniform-