Through careful study, scientists can understand the designs and algorithms used in creation....Engineers can emulate these designs and algorithms in the machines they develop.

— Phillip John McKerrow (1991)

2. Conceptual Design

Ultimately, design is a selection process. At the basic conceptual level, designing the crawling vehicle consisted of selecting the physical structure of the robot and determining how that physical structure would move in order to accomplish some desired locomotion. These selection decisions were made with the overall goal of developing a new category of vehicle that would be able to traverse difficult, uneven terrain in a manner similar to caterpillars. Specifically, in order to achieve this design goal, the design choices for the crawling vehicle were guided by the hypothesis that the robot would achieve caterpillar-like mobility if its body structure and methods of locomotion could be made to emulate those of the caterpillar.

The resulting conceptual design has both a body structure and method of locomotion that are radically different from most previous legged vehicles. However, in some other respects, such as sensors, terrain perception, and navigation, this design will still rely heavily on technologies developed by previous mobile robot researchers. This chapter explains the conceptual design of the multibody passive-legged crawling vehicle, focusing on the morphology and motion programming aspects of the design. In addition, it gives an overview of design issues that are common to all legged mobile robots, including brief introductions to power supplies,

sensors, navigation, and controls so that the reader will have an appreciation for the overall design of legged vehicles.

2.1 Basic Components of a Legged Robot

While the vehicle's body structure and method of moving are the most basic parts of its conceptual design, to complete a working design, other components are required to support the locomotion function. The fundamental building blocks of a functioning autonomous legged vehicle include:

1. Physical structure (including actuation system),
2. Motion program(s),
3. Power source(s),
4. Sensors,
5. Navigation and Control system(s).

Note that these subdivisions of the design are intimately related and interdependent. Each subdivision puts design constraints on the others. Thus, in order to complete the design of a successful robot, ALL of the design constraints will need to be understood and taken into account simultaneously. This interdependency makes it difficult to find a simple, logical sequence in which to describe the conceptual design. This chapter will take a generally bottom-up approach in its discussion of the basic components in the overall design of the crawling vehicle.

2.2 Physical Structure

The structure of this robotic vehicle evolved from research on Variable Geometry Trusses (VGT's) conducted at Virginia Polytechnic Institute and State University. In fact, the impetus for this entire project was the suggestion made by Dr. Charles F. Reinholtz to the author in September 1988 that a long-chain VGT manipulator could be disconnected from its ground link and used to make a snake or worm-like vehicle. During this same brainstorm meeting, the author proposed using the caterpillar as the natural analogue for such a vehicle, citing its mobility and the benefit of avoiding the friction and wear that would occur with a legless vehicle.

Looking at the caterpillar as a prototype for the physical structure of the robot, we note that the caterpillar has a long, slender, and generally cylindrical body structure that is divided up into several segments. These segments can contract, expand, and tilt independently of each other. Most of the segments have a lateral pair of legs (e.g. placed side by side relative to the longitudinal axis of the caterpillar body). As a result, both the mass of the caterpillar and its legs are distributed along its length.

Therefore, to enable caterpillar-like mobility, the robot physical structure needs to have places distributed along its length for mounting the lateral pairs of legs. Furthermore, each pair of legs must be able to contract, expand, and tilt independently of the pairs ahead of and behind it. This leads to a repetitive design that imitates the lengthwise uniformity of the caterpillar body structure. Such an arrangement is shown in Figure 2.1. Specifically, the conceptual design of the crawling vehicle body structure is based on the idea of using an alternating longitudinal sequence of "leg pair assemblies" connected to each other by "actuation units". The leg pair assemblies are the points of attachment for the pairs of legs; the actuation units are flexible, actuated sections that can contract, expand, and tilt in a controlled manner.

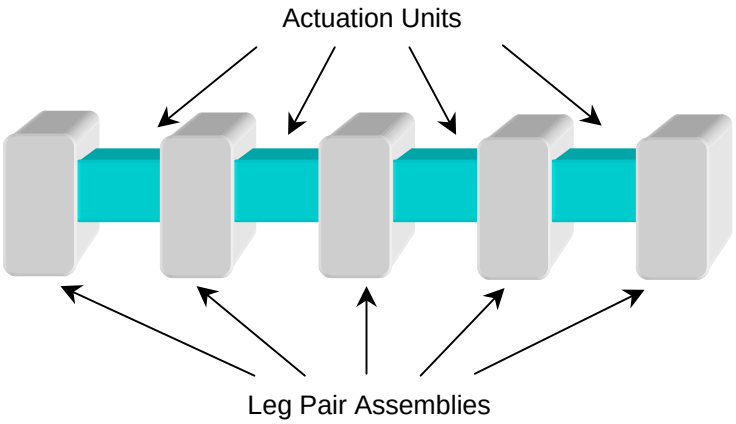

Figure 2.1 — The Basic Physical Structure of the Crawling Vehicle: An Alternating Sequence of Leg Pair Assemblies and Actuation Units

Having this basic topology, what remains of the conceptual design of the physical structure of the vehicle is to determine the characteristics of these actuation units and leg pair assemblies.

2.2.1 Actuation Units

The selection of the actuation units is fundamental to the design of the vehicle structure because they provide the mechanical means for producing the necessary relative motion between the leg pair assemblies. This design decision determines the mobility of the vehicle and its success when operating over extreme terrain.

2.2.1.1 Selection Criteria for the Actuation Unit Mechanisms

In order to have the necessary mobility to enable motion similar to that of caterpillar segments, the actuation unit mechanism design must have a number of important attributes (many of which are derived from caterpillar characteristics, as will be discussed in Ch. 3).

First, it is desirable for the actuation units of the vehicle to all have identical workspaces. Such an arrangement assures that if the front of the robot can travel along a particular path, then the trailing "segments" of the robot will be able to follow, thus simplifying path planning, trajectory planning, and controls.

Second, it is desirable for each actuation unit to have a large workspace so as to allow large relative motions between its adjacent leg pair assemblies. This will enable longer strides, resulting in greater mobility, higher speeds, and better energy efficiency.

Third, it is desirable for the selected actuation unit mechanism design to have a workspace that is symmetric about its longitudinal centerline. This is important because it will allow the robot to travel in one direction as well as it travels in the opposite direction. Specifically, the robot should be able to yaw to the left or yaw to the right, roll clockwise or counterclockwise, pitch upwards or downwards, extend left or right, and extend up or down, all with equal facility, respectively.

In addition to these workspace requirements, the actuation unit mechanism should be simple from the kinematics analysis viewpoint. Ideally, the mechanism should have simple, closed-form solutions for both its forward and inverse kinematics problems. Where these basic kinematics analysis problems can be defined simply as:

- The ***forward kinematics*** problem (sometimes called the direct kinematics problem) is the task of determining the position and orientation of the output link of a mechanism relative to its base, when given the set of input joint values (e.g. joint angles or offsets depending on the type of joints).

- The ***inverse kinematics*** problem entails determining the set of joint values that will place the output link of a mechanism at a specified position and orientation relative to the mechanism base frame of reference.

(It should be noted that it is often desirable to obtain higher-order derivatives for both the forward and inverse cases and that sometimes multiple correct answer sets exist, which indicate alternate assemblies of the mechanism.)

While simple closed-form solutions to these kinematics analysis problems are desirable for the actuation unit mechanisms, more complicated closed-form solutions or even iterative numerical solutions are acceptable if they can be computed with sufficient resolution and rapidly enough to support real-time control functions. Donner (1987) describes adequate real-time performance as follows:

> Adequate real-time performance produces latency small enough for the task at hand. Latency is the amount of time that elapses between two successive opportunities for any specific process to receive CPU cycles and affect the behavior of the system.

The selected actuation unit mechanism should also be both stiff and lightweight. The stiffness is necessary to enable cantilever and bridging maneuvers for crossing large ditch-like obstacles. Having lightweight actuation units is desirable because it reduces the overall weight of the vehicle and thus generally improves the energy efficiency of its locomotion.

Furthermore, the actuation unit mechanism should be designed with regard to mechanical advantage so that excessive amounts of torque and power will not be required to cause the vehicle to move. This will prevent the need for unnecessarily large actuators.

Finally, while seeking to achieve all these design goals, it is important to select an actuation unit mechanism design that is as simple as possible.

2.2.1.2 Variable Geometry Truss Technology

Judging from the criteria discussed in the prior sub-section, it is clear that the mechanisms used for the actuation units will each require more than one degree of freedom of mobility. Thus, these "mechanisms" are more properly termed *manipulators*, since the term, mechanism, in its strictest sense, means a device having only one degree of freedom.

Manipulators are comprised of two or more generally rigid links that are connected to each other by actuated joints. By controlling the actuators, prescribed relative motion can be produced between the manipulator base-frame of reference and either an end-effector link or a platform. Manipulators can be categorized as serial, parallel, or as a hybrid of the two. As their name implies, serial manipulators consist of links that are connected in series by their joints so as to make an open-loop chain. In contrast, parallel manipulators have links that are connected in parallel, so that one or more closed-loop chains are formed. Hybrid manipulators possess both open- and closed-loop chains.

Serial manipulators were eliminated from consideration as candidates for the actuation unit mechanisms because, as Salerno (1993) explains it:

> Each link of the [serial] manipulator is, in effect, a cantilever beam carrying the full load of all the links further out in the chain. The individual members of such a chain may be subjected to all possible types of loads: bending, torsional, axial, and shear. Of these possible loads, the design is usually limited by the effects of torsional and bending loads. Consequently, most purely serial manipulators are inherently compliant and have relatively poor load carrying capacities compared to their overall weight.

Thus, serial manipulators fail to meet the stiffness-to-weight requirements for the actuation unit mechanisms. Therefore, the search for an appropriate mechanism focused on parallel manipulators. The parallel links of such manipulators create multiple load paths, resulting in a number of advantages. As Tsai (1999) states it:

> In general, a parallel manipulator has the advantages of higher stiffness, higher payload capacity, and lower inertia to the manipulation problem than a comparable serial manipulator, at the price of a smaller workspace and more complex mechanism.

Considering these facts, a class of parallel manipulator called a Variable Geometry Truss, or VGT, was initially chosen as the prime candidate for the actuation unit mechanisms because of its superior stiffness-to-weight ratio and its potential mobility.

As their name implies, Variable Geometry Truss manipulators are based upon truss structures. For many years, engineers and scientists have recognized the excellent strength and stiffness properties of static trusses (Williams, 1979). A truss may be defined as a device whose links carry only axial (pure tension or compression) loads. To achieve this load-carrying property, all the links that comprise the truss must be joined together in such a way that no bending moments can be transmitted from one link to the next. These requirements lead to a variety of deltahedron-based structures whose links always form a set of triangular faces (see Fig. 2.2). Although a variety of truss configurations are possible, most common statically-determinate trusses are formed from tetrahedrons, octahedrons, decahedrons, dodecahedrons, or some combination of these (Arun, Reinholtz, and Watson, 1990). These unit cells can be concatenated to make larger structures.

A VGT results when one or more links of a statically determinate truss are made extensible. (Kinematically speaking, when a truss link is made extensible it becomes a kinematic or prismatic pair, that is, two rigid links connected by a prismatic joint. However, because the mechanisms described in this work are derived from trusses and for the sake of simplicity, in this work, these extensible members will often be referred to as simply “links”, “extensible links”, “variable length links”, or “prismatic links”.) Controlling the length of these extensible members enables the shape of the truss to be manipulated. For example, a tetrahedral truss composed of six links can

be provided with as many as six DOF's without altering its basic truss structure. Clearly, many combinations of basic truss units and actuation schemes are possible. Furthermore, when two or more of these unit cells are connected along their longitudinal axes to produce a long-chain VGT, the total number of variable-length members is equal to the number of degrees of freedom of the VGT. Note that making some links variable in length does not change the basic geometric structure of the truss. Thus, the links of a properly designed VGT will still be primarily loaded in tension or compression, and the VGT will retain the excellent stiffness-to-weight characteristics of static trusses.

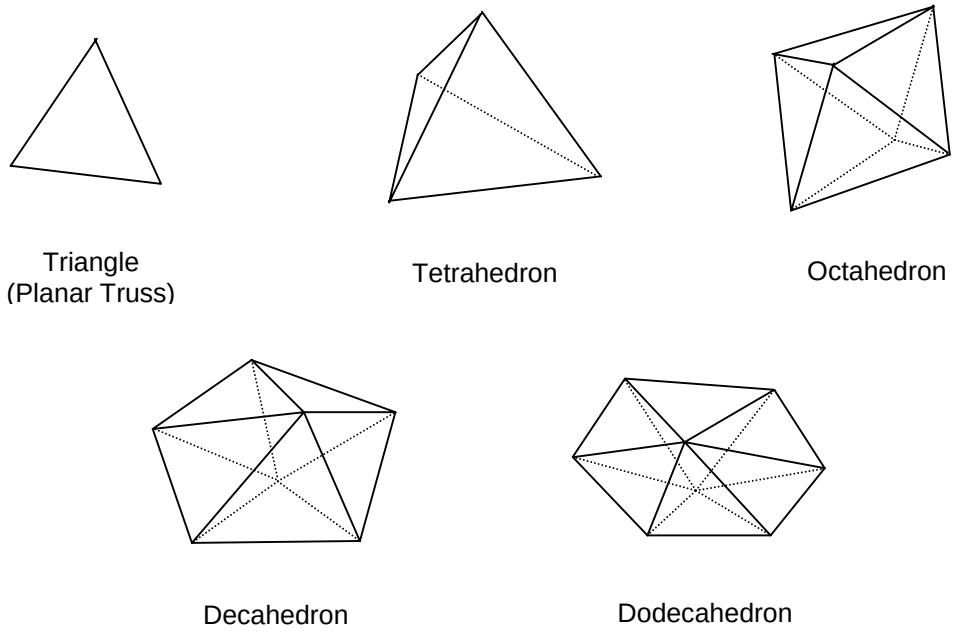

Figure 2.2 — Basic Truss Unit Cells

Much of the early work in adaptive (variable geometry) trusses was sparked by NASA's interest in foldable, deployable space structures (Salerno, 1993). Many schemes were devised to enable the trusses to collapse for compact storage. Furthermore, VGT's have also been previously used for actively damped structures,

robotic manipulators, and antenna controllers (Wynn, 1990; Warrington, 1991, Reinholtz and Gokhale, 1987; Padmanabhan, 1989; Salerno, 1989 and 1993; Tidwell, 1989).

2.2.1.3 Candidate VGT Unit Cells

As was shown in Fig. 2.1, it is required that each actuation unit mechanism be rigidly attached at two opposite ends to its adjacent leg pair assemblies. For the sake of stiffness, these connections should consist of three truss links composing one triangular face of a VGT cell, rather than the structurally less stiff options of connecting at a single joint or via a single truss link. Consequently, two planar faces of the VGT must be allocated for use as connections to the leg pair assemblies.

When they are evaluated in the light of these connection requirements and the workspace requirements described in sub-section 2.2.1.1, several of the basic truss unit cells can be eliminated as the possible basis for a VGT-type actuation unit.

A VGT mechanism based upon a simple triangular truss might be adequate for some laboratory tests and proofs of concept, but it is wholly inadequate for use on a vehicle intended to move over 3-D undulating terrain, because such a mechanism would only allow planar motions.

While tetrahedral VGT's can be very stiff, they are eliminated from consideration for two reasons. First, they have an insufficient number of links. Thus, they cannot support enough actuators to satisfy the mobility requirements. Second, tetrahedral cells lack the symmetry needed to make them usefully stackable. That is, they cannot be stacked in a manner that makes it convenient to attach them to their adjacent leg pair assemblies.

The decahedron and dodecahedron cells have more links than are desirable and would not have the desired workspace symmetry. Hence, they would be awkward and unnecessarily complicated for use with the actuation units.

An octahedral cell has twelve links that form eight triangular faces. Hence, two of these faces can be rigidly attached to leg pair assemblies, and six links will still be left over that could be made to be extensible. Furthermore, the octahedral cell is symmetric so that long-chain structures can be formed by simply stacking units end-to-end, as shown in Fig. 2.3.

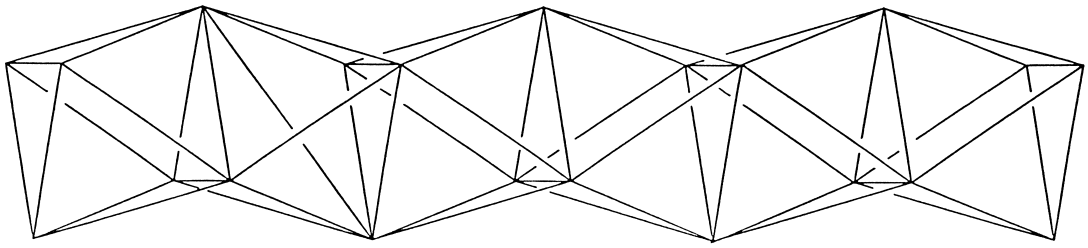

Figure 2.3 — A Long-Chain Truss Structure Based Upon Six Octahedral Unit Cells

Thus, an octahedral cell is the best candidate for use as the basis for a VGT-type actuation unit mechanism for the crawling vehicle. Other researchers have also selected the octahedral cell for their own applications (Miura, et. al., 1985; Sincarsin and Hughes, 1987).

2.2.1.4 Candidate Octahedral VGT's

Having selected the basic VGT unit cell, the next step is to determine how many octahedral cells will be used for each actuation unit and which of their rigid links will be replaced with prismatic joints. Looking at octahedral-based VGT designs and limiting the selection to fairly simple designs that have the required workspace

symmetry, two leading concepts emerge, the double octahedron VGT and the single octahedron VGT. These concepts are illustrated in Fig. 2.4, where the thickened sections on the links indicate the presence of linear actuators that enable those links to extend and contract.

The first VGT concept is referred to as the double-octahedron VGT because its overall geometry is formed by combining two octahedral unit cells. These two cells share a common lateral triangular face. The three links that form this intermediate triangular truss are the variable length members in this design, while the lengths of the remaining links are all fixed (see Fig. 2.4). The double-octahedron VGT can be used as an actuation unit mechanism by attaching a leg pair assembly to the rigid triangular trusses on either end. This design for would enable the actuation units to produce three DOF's of mobility between successive leg pair assemblies.

Double Octahedral VGT

Single Octahedral VGT
(a subclass of the
Stewart-Gough Platform)

Figure 2.4 — A Comparison of the Double Octahedral and Single Octahedral VGT's

Thus, when using this VGT to manipulate the position of one leg pair assembly relative to another, a motion planner can specify the location (x, y, z) of the leg pair assembly to be controlled (the position problem) or the relative angle (α, β, γ) of the leg pair assembly (the gimbal problem). The necessary lengths of the prismatic links that will achieve a given set of desired inputs (location and/or angle) are determined

by means of the inverse kinematics. For the double-octahedral VGT, the solutions to both the position problem inverse kinematics and the gimbal problem inverse kinematics are available in closed-form, so long as the two octahedral cells of the VGT are identical (Padmanabhan, et. al., 1992). These inverse kinematics problems can have up to 8 different solutions that represent different assemblies of the VGT manipulator. The forward kinematics solutions for the position and gimbal problems involve iterative calculations using a quasi-Newton method (Padmanabhan, et. al., 1992; Reinholtz and Gokhale, 1987). However, these can be computed rapidly enough to enable real-time control.

In the VGT illustrations presented so far, the nodes of the trusses have been rendered as simply meeting at a point. However, for an ideal truss to be formed, no bending moments or torques can be transmitted through these nodes. Hence, the nodes of ideal trusses allow relative rotation between their connected links so that the links will carry only pure tension and compression loads. For a functioning VGT, the connecting nodes require large angles of relative rotation between many of the links. Hence, on VGT's, some of the node connections must be replaced with revolute joints. Figure 2.5 shows a double-octahedral VGT that was designed and constructed by Paul Tidwell (1989). Looking closely at the figure, it can be seen that there are revolute joints located at most of the link connection nodes. Also seen in the figure are the DC motor-powered lead screws that actuate the three prismatic joints of this VGT design.

As the three lateral joints of a double-octahedral VGT are extended, the truss will become progressively wider and shorter. By doing so, a properly designed double-octahedral VGT can fold into a nearly flat configuration. This collapsibility feature would help make the crawling vehicle easier to store in confined spaces.

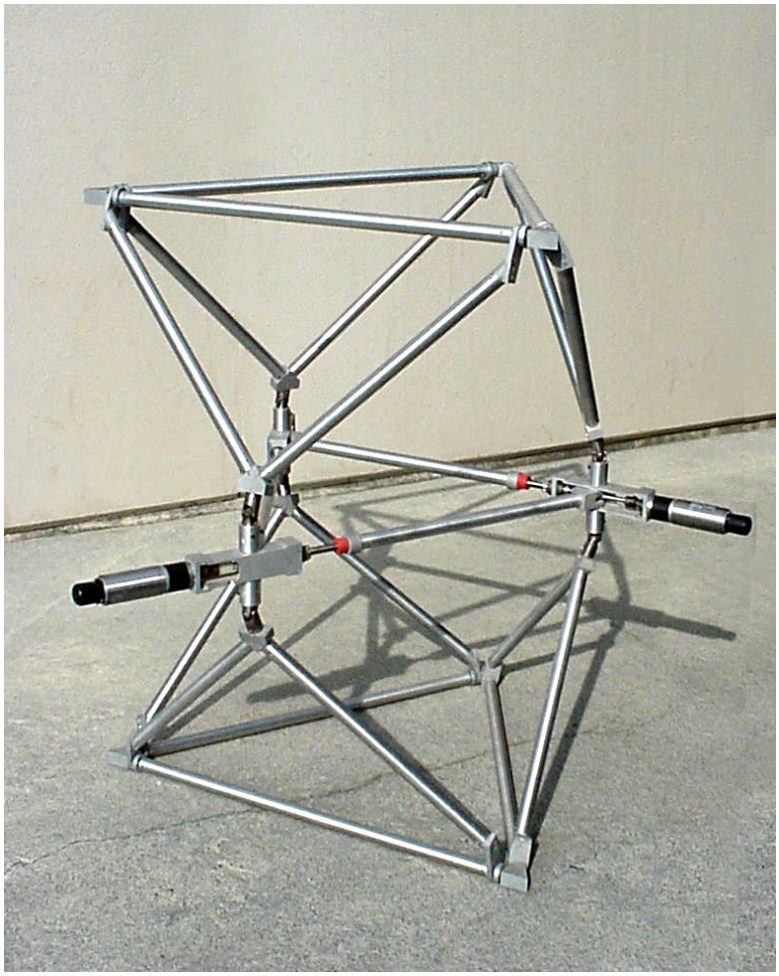

Figure 2.5 — A Functional Double-Octahedral VGT

The workspace of the double-octahedron also enables it to produce long strokes along its longitudinal axis, which would allow the vehicle to make long strides. Furthermore, its workspace also enables large yawing and pitching angles that could help the vehicle perform sharp turns.

However, despite its excellent workspace in some respects and its collapsibility, the double-octahedral VGT design was found to be inadequate. Specifically, having actuation units based on this design would probably be sufficient to enable

earthworm-type locomotion, but a study of caterpillar locomotion (described in the next chapter) indicated that the 3 DOF's provided by this design would be insufficient to enable the robot to emulate the mobility of caterpillars.

Therefore, because mobility considerations are preeminent in this conceptual design, the second VGT concept was deemed to have better potential for use in the actuation units.

The second concept for a VGT-based actuation unit mechanism uses only one octahedral unit cell and is therefore referred to as the single-octahedron VGT. In this design, the triangular truss faces on both ends of the octahedral unit cell are rigidly attached to its adjacent leg pair assemblies. All six of the longitudinal links of the truss are made to be variable length links (see Fig. 2.4). The prismatic joints of the variable length members are powered by linear actuators. This arrangement produces six DOF's contained within a single octahedral cell.

The single-octahedral VGT can also be considered as a special truss-like case of the general Stewart-Gough platform mechanism (Stewart, 1965-66). In this actuation unit application, the base and the moving platform of the Stewart-Gough mechanism are the leg pair assemblies attached to both of its ends. Within a limited workspace, each leg pair assembly of the robot can take any arbitrary position and orientation relative to its adjacent leg pair assemblies, thus giving the crawling vehicle increased range of motion and terrain adaptability.

Thus, when using this mechanism to manipulate the position of one leg pair assembly relative to another, a motion planner can simultaneously specify *both* the location (x, y, z) and relative orientation (α, β, γ) of the leg pair assembly to be controlled. Solving the inverse kinematics to determine the lengths of the prismatic links required to produce a specified relative position is a remarkably simple computation for the single octahedron VGT and all forms of the Stewart-Gough platform. Specifically, the inverse kinematics solution is available in closed-form and

can be computed very rapidly for use in real-time control (Stewart, 1965-66, Tsai, 1999). (Note that the inverse kinematics solution will be presented in Ch. 4.)

In contrast, the forward kinematics problem, that is, determining the position of the moving platform relative to the base when given the lengths of the six prismatic links, can only be solved by iterative means (Padmanabhan, et. al., 1992). The general case Stewart-Gough platform has 40 forward kinematics solutions (Raghavan, 1993). Wen and Liang (1994) presented a closed-form governing polynomial equation for the forward kinematics, but because it is a 20^{th} degree polynomial, it is itself not solvable in closed form. Wen and Liang (1994) have developed a program that can solve this equation to 7 digits of accuracy within 4 seconds when run on a personal computer that uses an 80386 CPU. In view of this computational burden, it may be preferable to solve the forward problem using a numerical method that iterates upon inverse kinematics solution (hence, taking advantage of the rapid, closed-form inverse). Salerno and Reinholtz (1990) described such an iterative approach, which used a quasi-Newton method to solve the 3-3 case of the Stewart-Gough Platform. Similarly, Nguyen and his colleagues (1991) described an iterative forward solution that used the Powell Direction Set Method to solve the 6-6 case of the Stewart-Gough Platform. However, at this stage of the research, it appears unlikely that the forward solution will need to be done very often. It may only be used to recalibrate the actuators after initially powering-up the robot, after an unusual movement (such as a fall), or if errors accumulate during the robot's operation. Hence, rapid calculation of the forward problem is not as critical as it is for the inverse problem.

In Fig. 2.4, pairs of longitudinal members are shown meeting at points. Mechanically, this could be realized by having coincident spheric joints (e.g. one spheric joint would be mounted inside another and they would have the same center of rotation). Such an arrangement would make the single-octahedral VGT an ideal truss. However, having coincident spheric joints limits the angles of rotation for the links,

which in turn limits the workspace of the VGT. Therefore, as a practical matter, it is often best to use a separate spheric joint for connecting each of the variable length members to the rigid end-planes of the truss. These non-coincident spheric joints allow some non-axial loads to be transmitted through the rigid links of the single-octahedral VGT, thus preventing it from being an ideal truss. However, by placing the spheric joints close together, these non-ideal effects can be minimized, and the VGT will still retain most of the high stiffness-per-unit-weight of static trusses.

Figure 2.6 explicitly shows the joint arrangement used with a single-octahedral VGT or truss-like Stewart-Gough platform. As shown in the figure, each of the six kinematic chains connecting the base and the moving platform is a spheric-prismatic-spheric (SPS) limb. However, because this construction possesses a passive rotational freedom about the limb axis, the same effective mobility can be also obtained by using a spheric-prismatic-universal limb (SPU) arrangement.

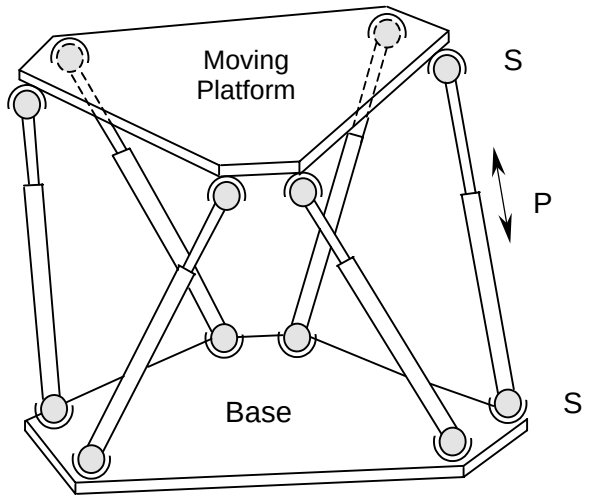

Figure 2.6 — A Stewart-Gough Platform

The actuated prismatic joints can be realized by using either hydraulically or pneumatically driven cylinders or motor-driven lead screws or ball screws. If screws

are chosen, it can be helpful to use double nuts preloaded with spring washers to eliminate backlash (Fichter, 1986).

Because universal joints (i.e. Hooke joints or gimbals) generally have greater angular workspace than spheric (ball) joints, it may be desirable to have U-joints on both ends of the limbs. For the case of hydraulic or pneumatic piston prismatic joints, this could be accomplished by using a universal-prismatic-revolute-universal (UPRU) chain, where the additional non-actuated revolute rotates about the prismatic axis. Or, in the case of lead screw or ball screw prismatic joints, a USRU chain could be used, where again the additional revolute joint has the same axis as the screw so as to decouple the screw angle and extension (and thus avoid having to complicate the kinematics by accounting for extension/rotation coupling when the platforms roll relative to one another).

2.2.1.5 Alternate Stewart-Gough Platform Mechanisms

Having decided upon the single-octahedron VGT (a subclass of the Stewart-Gough platform) for use as the actuation unit mechanisms, the selection of the specific configuration of the joints still remains. As discussed in the previous section, the joints could be placed close together to preserve truss-like stiffness, which is important for cantilevering and bridging across large ditch-like obstacles (these operations were shown in Fig. 1.7 and Fig. 1.8). However, it should be recognized that other, less truss-like Stewart-Gough platform designs can also have the necessary mobility and that mechanical advantage is also an important factor. Therefore, rather than limiting the design to only the truss-like subclass of Stewart-Gough platforms, the more general cases of the Stewart-Gough platform should also be considered.

Considerable freedom exists in the choice of the specific configuration of the spheric joints. Even if we limit the locations of the spheric joints on each end of the Stewart-

Gough platform to be co-planar (which simplifies the forward kinematics (Tsai, 1999)), there still remain two infinities (one for *x* and one for *y*) of possible locations for each spheric joint, for a total of 24 infinities of possible configurations for the Stewart-Gough platform mechanism. However, some care must be taken in these joint location selections to avoid designing in the architectural singularities described by Ma and Angeles (1991).

The actuation unit design that was initially considered is a truss-like design that separates the six spheric joints on each side of the actuation unit into three widely separated pairs. The two joints in each pair are placed as close as possible to each other without causing interference between the attached links so as to approximate meeting at a single point. Thus, this design approximates an octahedral VGT cell and consequently retains much of the high stiffness to weight ratio of an ideal truss. Figures 2.7 and 2.8 illustrate this joint configuration.

The locations of the joint pairs were chosen so as to make the bottom and top links parallel to the ground when the vehicle is in a neutral stance. This constant ground clearance design requires the two diagonal links (seen in Fig. 2.7) to have longer, more powerful actuation strokes than the other four prismatic links.

Another candidate actuation unit was developed by Paul Mele (1991) in his baseline detail design of the physical structure of a multibody passive-legged crawling vehicle. The resulting "Mele configuration" design is shown in Fig. 2.9 and Fig. 2.10.

In his design, priority is given to using the same components for all of the prismatic links, while attempting to maximize the mechanical advantage and the workspace of the mechanism. Therefore, the variable length links of the actuation units were constrained to have the same range of motion.

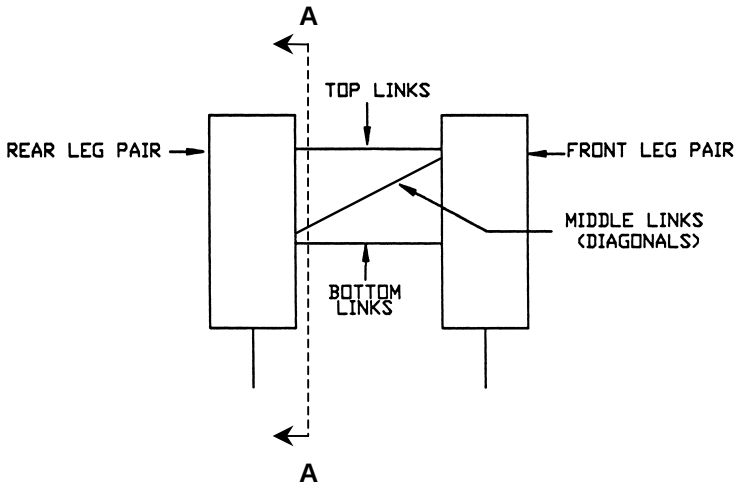

Figure 2.7 — Side View of the Initial Configuration of the Stewart-Gough Platform Spheric Joints (Illustration after Mele (1991))

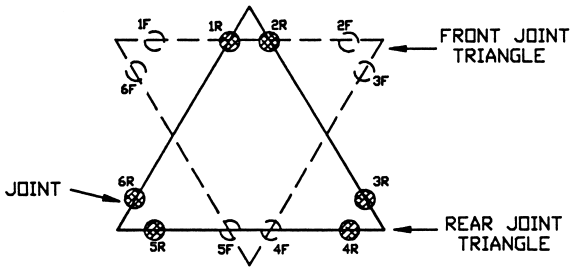

Figure 2.8 — Section A-A View of the Initial Configuration of the Stewart-Gough Platform Spheric Joints (Illustration after Mele (1991))

By careful placement of the spheric joints, the Mele configuration allows the use of interchangeable linear actuator components while having better mechanical advantage when initiating lifting motions and sacrificing little, if any, of the workspace or stiffness of the initial joint configuration design (Mele, 1991).

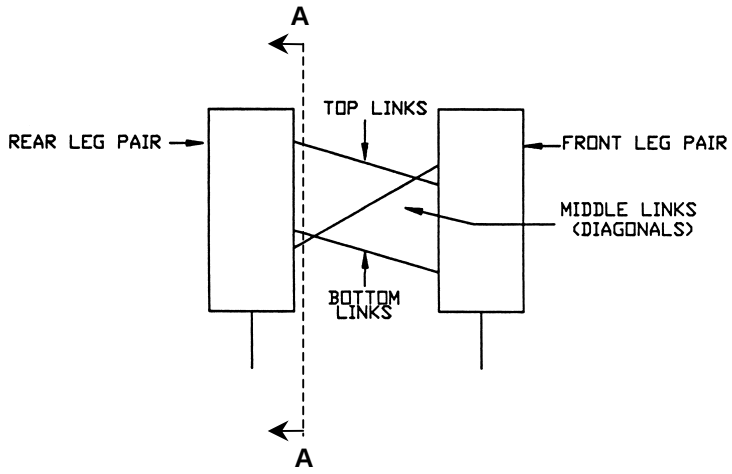

Figure 2.9 — Side View of the "Mele Configuration" of the Stewart-Gough Platform Spheric Joints (After Mele (1991))

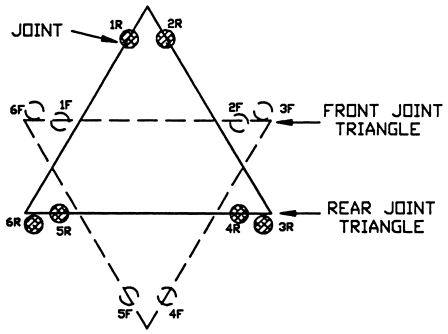

Figure 2.10 — Section A-A View of the "Mele Configuration" of the Stewart-Gough Platform Spheric Joints (After Mele (1991))

2.2.2 Leg Pair Assemblies

Recall from Section 2.2 that the leg pair assemblies are intended to provide places for attaching lateral pairs of legs and feet and also a means of connecting successive actuation units of the vehicle body to each other. Recognizing that the vehicle also requires containers for carrying and protecting power components, sensors,

controls, and payload (which could be cargo, instruments, manipulators, etc.), and that the vehicle mass should be distributed along its length, it seems logical to combine all these functions into the design of the leg pair assemblies.

Therefore, each leg pair assembly consists of the following components: a payload box, two legs, two feet, and either six or twelve spheric joints (depending upon whether the leg pair assembly is mounted at one of the ends of the robot or between two other leg pair assemblies). See Fig. 2.11.

As shown in the figure, the two legs are attached to the bottom of the payload box, one near each lateral edge (for greater lateral stability). Also, six spheric joints (represented in the Fig. 2.11 by small ellipses) are attached to both the front and the rear sides of the payload box.

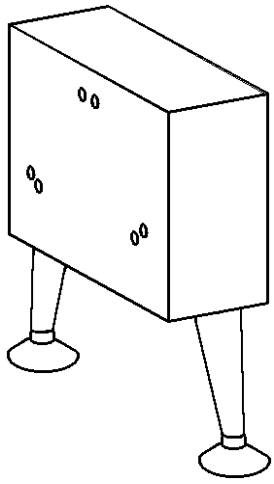

Figure 2.11 — A Leg Pair Assembly

The payload boxes are crucial to the design of the leg pair assembly because all of the other components are either mounted onto them or carried inside of them. Structurally, the payload boxes must be stiff and strong to provide the hard-points for

mounting the legs and also to support the weight of neighboring actuation units and leg pair assemblies during cantilever and bridging maneuvers.

The payload box consists of a framework overlaid by a skin material. It may also have an internal truss for reinforcement. However, the locations for the truss nodes, and hence, the overall shape of such a truss, are dependent upon the final configuration of the Stewart-Gough platform spheric joints that are mounted to the payload box. (Note that, since the payload box will be a rigid frame in order to protect its contents, the non-ideal truss characteristics resulting from the non-coincident spheric joints become less important.)

The two legs are not actuated. Additional actuation for the legs is not required for mobility because the flexible "body" of the vehicle can already position the leg pair assembly with 6 DOF's. For the sake of energy efficiency, at some future date it may be desirable to add a single DOF to each leg pair assembly for lifting the legs independently of the payload boxes. However, this dissertation assumes a conceptual design with non-actuated legs.

During locomotion there will be inevitable uncertainties concerning the relative positions of the feet and the terrain. Therefore, the leg design should include a vertically aligned, spring-loaded shock absorbing translational freedom. The goal of this feature is to protect the vehicle in cases when, at the end of a stride, the feet contact the ground sooner (and hence at a higher velocity) than anticipated by the control system. Specifically, such a design should lower the force of impact to a safe level, decelerate the falling leg pair assembly, and give the control system time to react and adapt to any terrain irregularities.

The simplest way of accomplishing this is to add spring-loaded prismatic joints to the legs. Specifically, this can be implemented by constructing each leg using two hollow, telescoping sections. The protected cavity formed within these sections would contain an appropriately sized captive cylindrical helical spring. In order to

avoid large leg bending moments and foot shear forces when the leg pair assembly is lowered to the ground, the portions of the two legs that telescope must be parallel.

As a leg pair assembly is lowered to the ground, the leg prismatic joints will be progressively compressed and the shock-absorbing springs will convert some of the leg pair assembly gravitational potential energy and store it as strain energy. Because of the damperless spring-loaded prismatic leg design, virtually all of this energy will later be released when lifting a leg pair assembly at the beginning of the next stride. Thus, the spring forces reduce the initial lifting force required and accelerate the stride motion.

However, some constraints must be imposed on the allowable spring stiffness and stroke length of the shock absorbers. First, if the spring forces are too high, it is possible that, when supporting leg pair assemblies are supposed to be firmly planted on the ground, their upper legs and payload boxes could bounce because they would be suspended by the springs. These oscillations could cause difficulties in the trajectory execution of the neighboring leg pair assemblies. Instead, the shock absorbers should become rigid whenever the full weight of the leg pair assembly is upon them.

Therefore, the prismatic joint design will use a spring with an appropriate operating stroke length and spring constant so that its peak force ($F = kx$) will be less than the minimum weight that will be exerted upon the leg when its leg pair assembly is placed on the ground. Furthermore, the design will feature a mechanical stop that halts the compressive strokes and locks the prismatic joint to make the leg rigid. Rather than simply using a pin or flat plate for the stops (which could allow some unwanted translation of the lower leg within the upper leg, thus reducing rigidity), a mechanical stop design that includes mating conical surfaces (hence restraining 5 DOF) should be used. Finally, the mechanical stops should be designed with

sufficient strength to withstand the large weight forces they will be required to support.

Second, there needs to be some means of limiting the stroke length so that the lower leg does not extend indefinitely and possibly allow the foot to collide with the terrain when it is supposed to be off of the ground. This is accomplished by having a second mechanical stop that halts the extension stroke of the leg when the leg pair assembly is lifted off of the ground.

Third, when the feet are lifted off of the ground while a leg pair assembly is being moved, inertial forces might induce foot bouncing, resulting in unwanted vibrations. This can be prevented by sizing the relative lengths of the captive spring and the cavity it operates within so that the spring will exert a moderate force against the second mechanical stop when the lower leg is fully extended.

Finally, the stroke length of prismatic joint should be limited because the longer the joint stroke, the higher the leg pair assembly mass must be lifted in order to prevent the feet from colliding with the ground unintentionally. While the springs store some energy that help lift the leg pair assembly on each stride, they do not provide all of the energy necessary for lifting. Therefore, the use of the shock absorbing springs does cost some energy in terms of work against gravity, so they should not be made longer than necessary to perform their shock protection function.

Note that many of the proposed applications of the crawling vehicle entail operating within “dirty” environments involving exposure to dust, moisture, and radioactive, toxic, and corrosive substances. Therefore, when operating in such environments, it may be desirable to add flexible, protective rubber boots to seal the annular opening of the prismatic joints. These cylindrical boots should be molded with a bellows-like shape to allow the free movement of the joints.

The two feet are connected to the legs by ankle joints having two relative, passive rotational DOF's to accommodate surface irregularities. These ankle joints can be spring-loaded so as to restore the feet to a neutral orientation whenever they are lifted off of the ground. This will help enable the feet to adapt to terrain irregularities at the next foothold. Once again, when operating in dirty environments, it may be desirable to add flexible, protective rubber boots to cover the ankle joints.

For most operating environments it will be desirable to imitate animals by including compliant soles in the design of the feet. Properly specified, this compliance will help reduce shock and will damp out chatter whenever the feet impact the ground. The final foot design will depend upon the terrain over which the vehicle is expected to operate. If indoors, then the feet may have non-marking synthetic rubber treads; if operated mostly over pavement, concrete, or smooth rocky ground, then carbon rubber soles would probably be appropriate; on common soils and turf, rubber cleats should work well; and on snow and very soft sand, each foot could have a single, large, spade-like cleat for longitudinal traction with a horizontal disk mounted just above it to prevent the foot from sinking down too deeply (see Fig. 2.12).

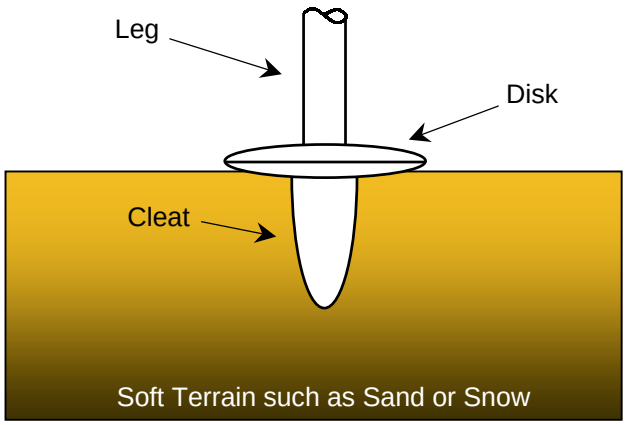

Figure 2.12 — A Foot Design for use on Soft Sand or Snow

This assembly of payload box, two legs and feet is rigid, and moves as a unit; for convenience, it will be referred to as a "leg pair assembly", or simply, a "leg pair".

2.2.3 Summary of the Physical Structure

The primary difference between this vehicle and other legged robots is that, while most legged robots have a single rigid body with actuated legs, this robot has an actuated multibody structure with passive legs. This arrangement imitates the body structure of caterpillars. Specifically, the proposed crawling vehicle is constructed of a lengthwise chain of at least 4 “leg pair assemblies” that are connected to each other by “actuation units” composed of Stewart-Gough platform mechanisms. These Stewart-Gough platforms enable each leg pair to move independently of the leg pairs ahead of and behind it with 6 degrees of freedom. This relative motion of the leg pairs can be used to bring about locomotion.

When operating on rough terrain, this design benefits from being a legged vehicle (as described in Section 1.4) and from having a multibody structure (as described in Section 1.5). This concept also has a number of other advantages. Like its biological counterpart, the caterpillar, the vehicle has an approximately even lengthwise distribution of both its mass and its supporting legs. This reduces the peak ground pressure and soil damage, simplifies path planning, and helps enable bridging and cantilever maneuvers for crossing large obstacles. The lengthwise uniformity of the design simplifies the manufacture of the hardware, allows parts to be interchangeable, and results in the actuation units having identical workspaces, which, in turn, simplifies the motion planning and control of the vehicle. These actuation unit workspaces are symmetric about their longitudinal centerlines and are similar to the “workspaces” of caterpillar segments (as will be discussed in Ch. 3). The Stewart-Gough platform mechanisms that are used for the actuation units are stiff, lightweight, and possess an inverse kinematics solution that can be computed

very rapidly. Furthermore, the design has a small frontal cross-sectional area relative to its mass, enabling it to travel through narrow passages and closely spaced obstacles.

Figures 2.13 and 2.14 show conceptual models of the multibody passive-legged crawling vehicle using two different configurations of the Stewart-Gough platform mechanism. In Fig. 2.13 the robot uses a configuration where the diagonal links of each actuation unit are longer than the other four prismatic links. In Fig. 2.14 the robot uses the "Mele configuration" for the spheric joints, which enables the use of identical actuated links throughout the robot and results in greater mechanical advantage for lifting leg pairs (Mele, 1991).

When operating in dirty environments, it may be desirable to add flexible coverings (e.g. bellows-like coverings, or boots, perhaps made of rubber) to cover all of the spheric, revolute, and prismatic joints individually. Alternatively, a large, flexible, tubular boot could be attached from each leg pair to the next so as to surround and protect the entire actuation unit between them. (Such a "skin" arrangement would make the vehicle resemble its natural counterpart even more strongly.)

In addition to the examples shown in Fig. 2.13 and Fig. 2.14, many other Stewart-Gough configurations are also possible. The exact specification of which particular Stewart-Gough platform would be best for a particular application of the multibody passive-legged crawling vehicle is within the domains of the configuration and detailed design stages rather than the conceptual design addressed here. Making an informed design decision on this matter requires extensive analysis. While not selecting a particular Stewart-Gough platform configuration from the myriad of possibilities, this work supports the *configuration design* function by creating tools for synthesizing, analyzing, and simulating a virtually limitless variety of multibody passive-legged crawling vehicle designs. These configuration design tools will be described in detail in Chapters 4, 5, and 6.

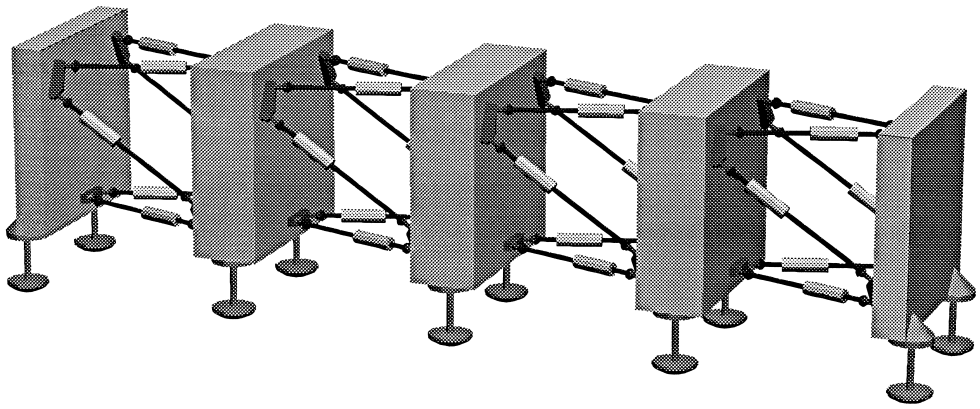

Figure 2.13 — A Multibody Passive-Legged Crawling Vehicle using a VGT-like Configuration for the Spheric Joints of the Stewart-Gough Platform Mechanisms

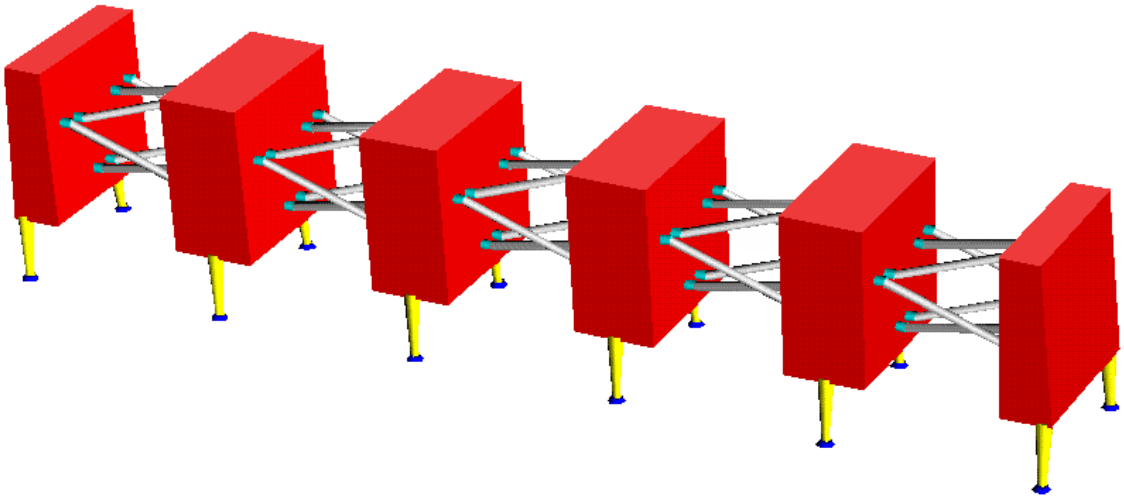

Figure 2.14 — A Multibody Passive-Legged Crawling Vehicle using the "Mele Configuration" for the Spheric Joints of the Stewart-Gough Platform Mechanisms

2.3 Introductory Motion Programming

Beyond selecting the physical structure of the robot, the other major conceptual design issue involves deciding how that physical structure should move to produce effective locomotion. This section gives an overview of this topic and introduces some nomenclature for describing the robot's motions. A specific algorithm for motion programming will be presented in Ch. 5.

As discussed in Section 1.8.3, there are two main approaches for programming the motions a caterpillar-like flexible robot such as the multibody passive-legged crawling vehicle: motion programming based upon shape control methods, such as those developed for long-chain VGT/hyper-redundant robot research, and stride-oriented motion programming methods similar to those used by walking machines. This research will pursue the latter approach.

2.3.1 Hierarchy of Motion

The following "hierarchy of motion" is a set of useful definitions for describing legged robot locomotion. Note that there is an analogous hierarchical arrangement in the control system, as will be shown in Section 2.6.

The motion hierarchy ascends from an individual movement, through a leg step, through a locomotion cycle, and up to a gait. An individual movement is an incremental displacement of a leg pair. A leg step is the completion of a sequence of individual movements that moves a leg pair one step forward. A locomotion cycle, or machine step, is when all of the machine's legs have each completed a step. A gait is one or more locomotion cycles that use the same distinct leg step sequence.

2.3.2 Basic Maneuvers

The physical structure of the multibody passive-legged crawling vehicle is geometrically capable of performing many types of gaits. Two of the classes of gaits, wave gaits and sidestepping gaits, are introduced via several examples. For the sake of simplicity, the example leg steps are broken down into a sequence of three rudimentary movements: lifting a leg pair, moving it forward, and lowering it to the ground (see Fig. 2.15).

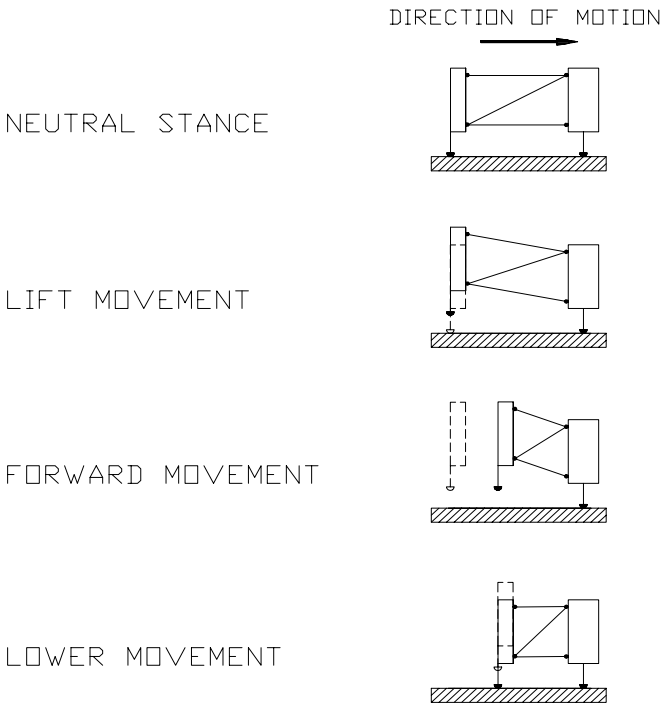

Figure 2.15 — A Simplified Leg Step

Many animals, including the caterpillar, use wave gaits. Wave gaits have been shown to have optimal longitudinal stability for hexapod vehicles (McGhee, 1985; Song and Waldron, 1989). The following three examples demonstrate basic maneuvers using a very simple wave-type gait.

Forward motion, shown in Fig. 2.16, is initiated by lifting the rearmost leg pair, moving it forward a specified step length, and then lowering it. As the feet of the rearmost leg pair touch the ground, the second leg pair from the rear is lifted and goes through the movements of its leg step. Then the third leg pair moves, and so on. To an observer, this motion looks like an actual wave traveling up the body of the vehicle.

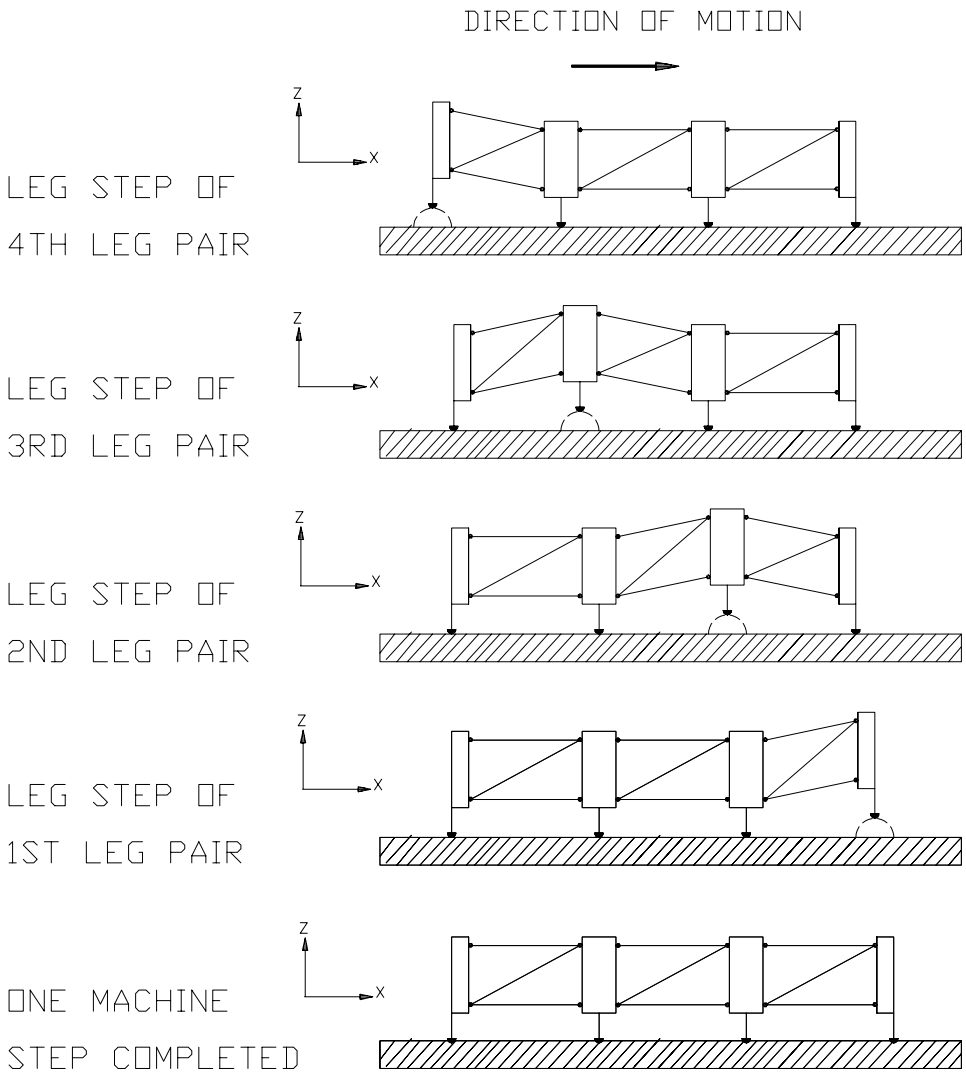

Figure 2.16 — Forward Motion Using a Simple Wave Gait

Reverse motion can be implemented in a similar manner, only the leg steps start with the front leg pair and progress towards the rear of the vehicle. To achieve higher speeds, the controller could initiate new locomotion cycles before the first locomotion cycle is completed, causing several "waves" to progress along the vehicle simultaneously.

Turning motion, shown in Fig. 2.17, can be used to navigate curved paths. When a leg pair reaches a curve in the vehicle path, its leg step is modified.

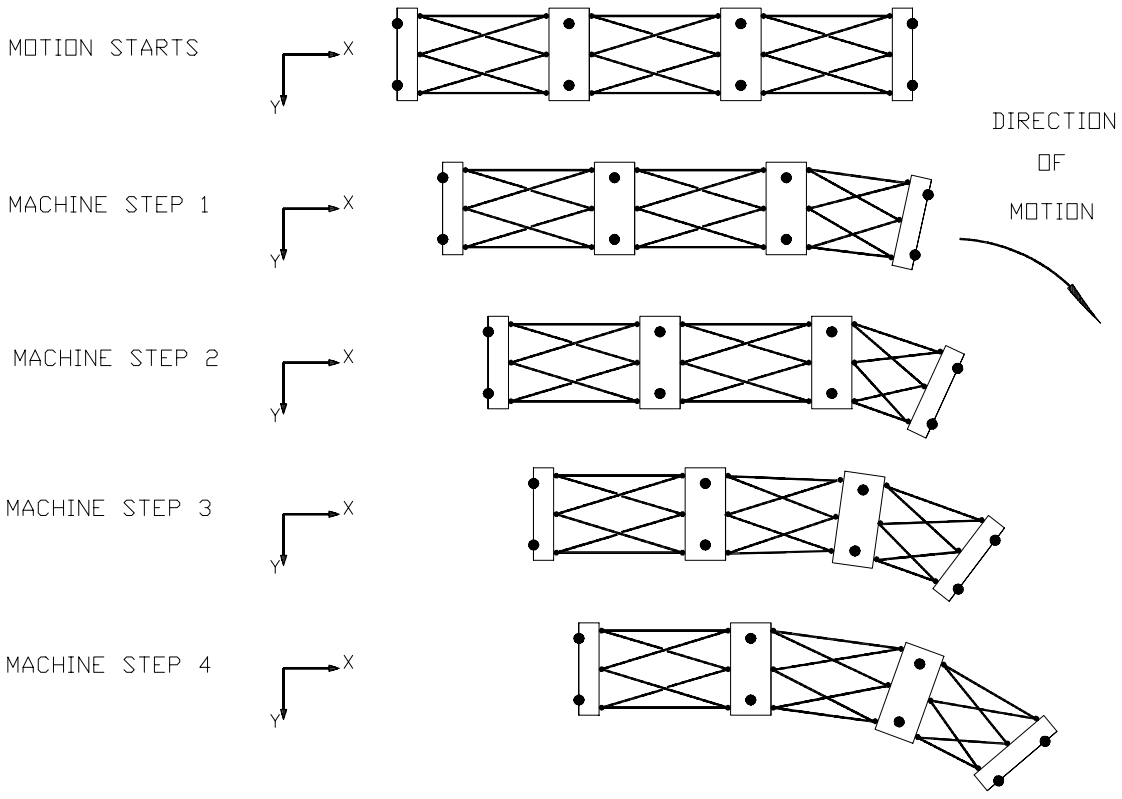

Figure 2.17 — A Simple Turning Maneuver

Specifically, turning motion is produced by lifting the leg pair, rotating it through an angle, translating it along a chordal direction, and then lowering it.

Climbing up and down over smooth slopes is accomplished in an identical fashion to forward motion. However, when small step-obstacles must be crossed, a simple climbing motion can be used, as shown in Fig. 2.18.

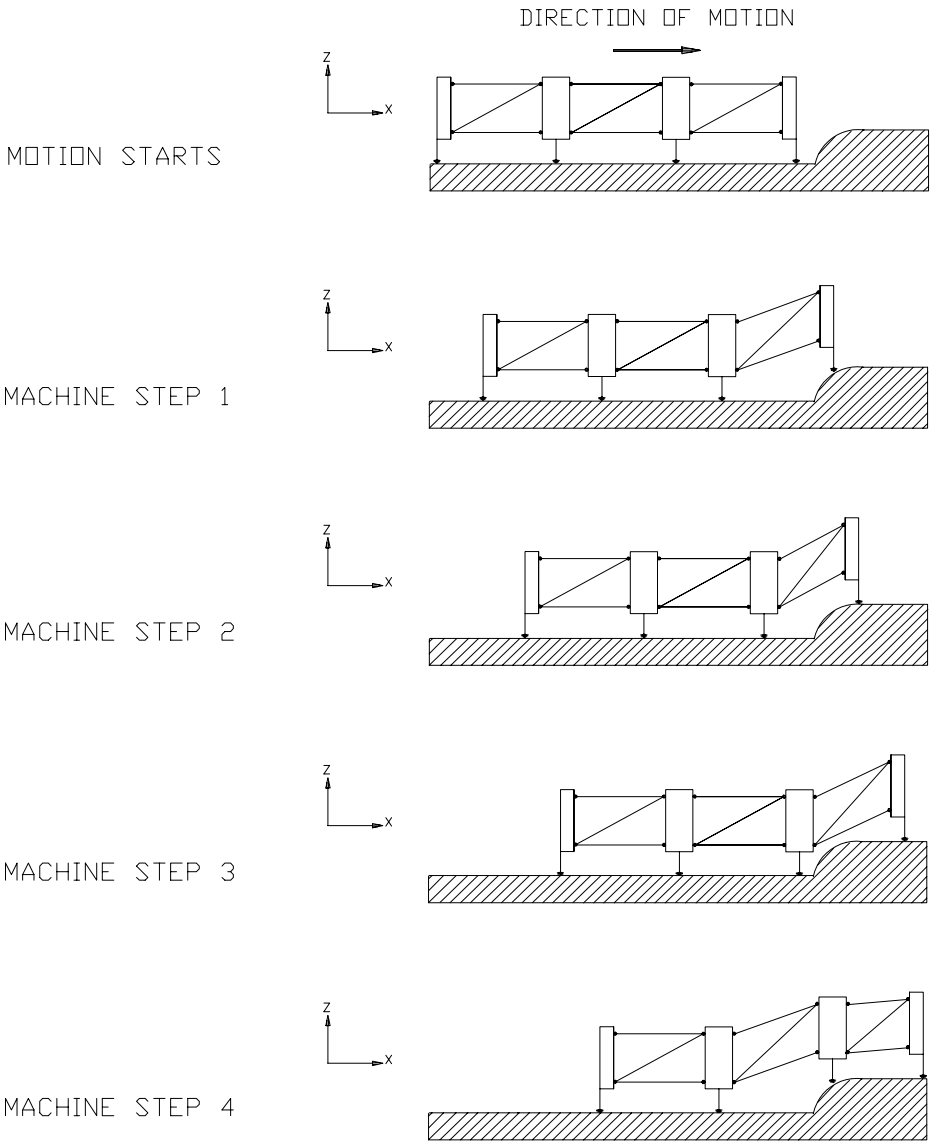

Figure 2.18 — Climbing Over a Small Obstacle

This simple climbing motion is produced by bringing the rear leg pairs forward, lifting the front leg pair and extending it over the obstacle, and then repeating this pattern until all the leg pairs have cleared the obstacle. These steps are fundamentally the same as the forward motion sequence except when the lift movement must be increased to clear the obstacle.

The vehicle should also be capable of traversing larger obstacles by extending two or more consecutive leg pairs off of the ground at the same time. These bridging and cantilever maneuvers were illustrated in Fig. 1.7 and Fig. 1.8. However, such advanced maneuvers may require the use of shape control methods in addition to the stride-oriented motion programming methods presented in this work.

Although the author has not observed the caterpillars doing so, the proposed crawling vehicle would also be capable of performing sidestepping gaits. Sidestepping locomotion, shown in Fig. 2.19, can be produced by simultaneously lifting the two inner leg pairs, swinging them to the side, lowering them, then lifting the two outer leg pairs, swinging them to the side, lowering them, and then repeating the sequence. Note that other types of sidestepping gaits are also possible.

One application where this design has a disadvantage relative to wheeled vehicles and other walking machine designs is in passenger transport. For example, with most single-body walking robots, one of the motion programming objectives is to smooth fluctuations in the velocity of its rigid body so as to reduce power requirements and make a more comfortable ride. Hence, the velocity of their rigid body can be more or less continuous. In contrast, the individual bodies of a multibody legged vehicle have, by definition, an intermittent forward motion. In addition, if the legs of the multibody vehicle are passive, each of the bodies must also be lifted from the ground with every step.

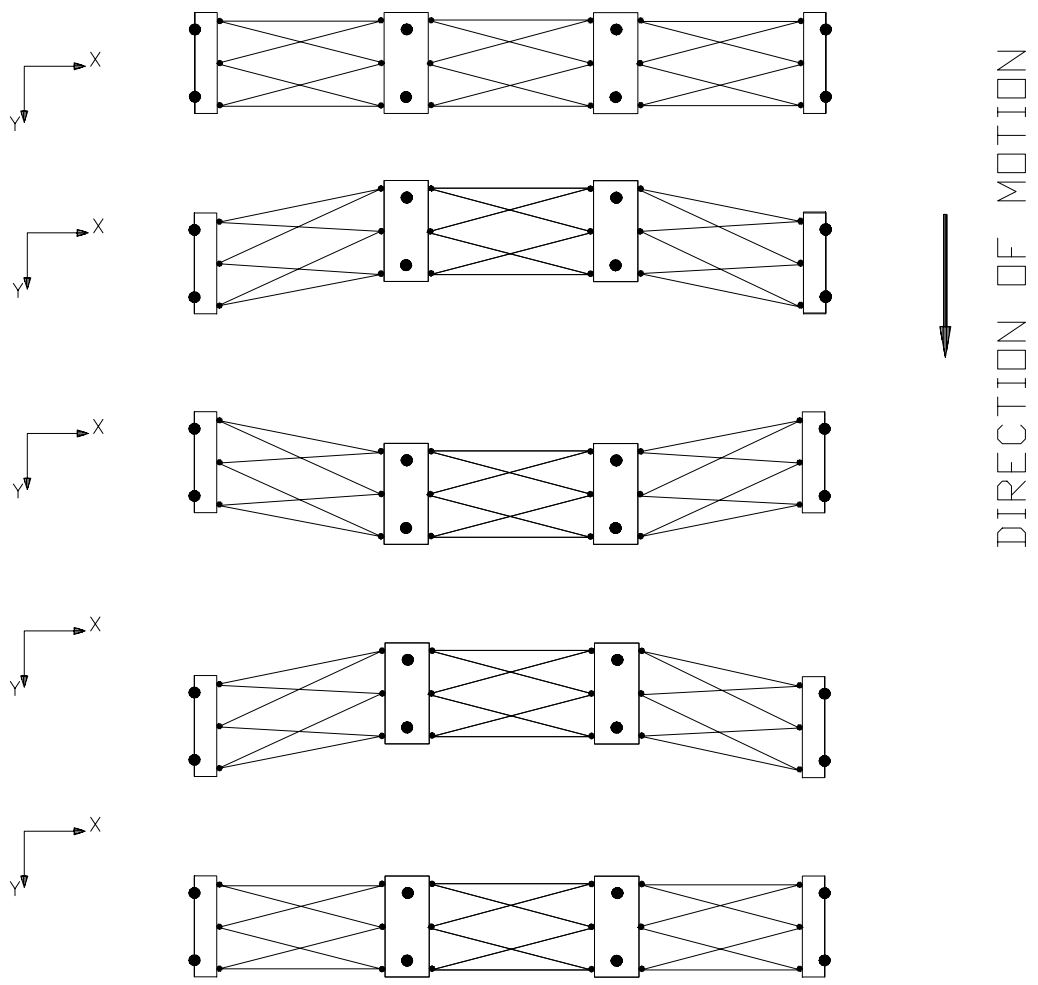

Figure 2.19 — The Sidestepping Gait

Thus, a passenger occupying one of the individual bodies would be exposed to a possibly disconcerting intermittent motion as the body rises from the ground, swings forward, and then returns to the ground with each step. Hence, while the crawling vehicle might be appropriate for an amusement park ride, it is an unlikely candidate for serious passenger transport. Note, however, that this intermittent motion also has some compensating advantages for sensing when the robot operates autonomously (these will be discussed in Section 2.5).

2.4 Power Source

In order for the crawling vehicle to operate, power in electrical form must be provided for the on-board sensors and control processors. Furthermore, power, in some form, must also be provided for the vehicle actuators. As explained in Section 2.2.1.4, either electric motors, hydraulic pistons, or pneumatic pistons could be used to actuate the prismatic joints of the vehicle's Stewart-Gough platform mechanisms. In a laboratory environment, a test vehicle could be "tethered" to a power supply by an electrical cable or perhaps hydraulic or pneumatic hoses. Such an arrangement may also be sufficient for some practical applications, such as fire, mine, or earthquake disaster rescue, where the crawler could operate while tethered by an umbilical cable reeled out from a "mother" transport vehicle.

To travel long distances, however, an on-board power source is required. For terrestrial applications, an air-breathing engine can be used as the prime mover to drive either an electric generator (for electric motors), a pump (for hydraulic pistons), or a compressor (for pneumatic pistons). For non-terrestrial applications, such as a planetary exploration rover, air-breathing engines may not be a possibility (at least not in our solar system). Radioisotope Thermoelectric Generators (RTG's) have been successfully used on numerous interplanetary missions, including the Apollo Moon landers and Viking Mars landers, among others (Cassini Public Information [Internet], 1998). However, according to a study by the Space Exploration Initiative Synthesis Group (1991), plutonium has become so scarce that using fuel cells may be preferable. With such a system, the rover vehicle will need to periodically go to a base camp to return its exhaust product (water) so that it can be regenerated into oxygen and hydrogen fuel.

For most power systems, some means of temporarily storing energy will also be needed. This could take the form of one of the various types of electrochemical

batteries, flywheels, hydraulic accumulators, or pneumatic accumulators, depending upon the actuation system chosen.

The performance of the crawling vehicle will depend significantly upon the power capacity per unit of weight, available peak power output, energy efficiency, energy storage capacity per unit of volume, physical size, weight, and reliability of the power system. However, specification of the power source depends upon, among other things, the specific vehicle application, the vehicle mass and volume, and the type of actuators chosen, and is therefore beyond the scope of this dissertation.

2.5 Sensors

As discussed in Section 1.2, terrain sensing has been one of the primary impediments to autonomous locomotion over extreme terrain. This work does not propose new sensing technologies; instead, it proposes a new conceptual design in which mobility performance is more tolerant of sensing errors. (Indeed, one of the main reasons for using the caterpillar as a natural analogue for an all-terrain vehicle was its superb all-terrain mobility despite its limited sensory ability.)

Sensors enable a mobile robot to measure its own operating parameters and the attributes of its environment. Thus, sensors are a necessity if a robot is to use closed-loop controls or react to its environment. The sensors of a robot can be classified as either "internal" or "external", depending upon their role (McKerrow, 1991). Internal sensors measure the operating parameters of the robot itself, such as the positions of all of its body members, thus approximating the proprioceptive sensing and sense of balance of living organisms. External sensors measure the attributes of things outside of the robot, that is, the operating environment of the robot, such as the geometry of the surrounding terrain, thus mimicking the touch, echolocation (used by bats and porpoises), and vision sensing of organisms.

2.5.1 Internal Sensors

At the most basic level of the crawling vehicle control system, all of the prismatic joint actuators must be accurately controlled. Therefore, in order to execute programmed motions correctly, each actuator needs some sort of feedback sensing. The type of sensors used depends upon the form of control selected. Sensors for position, velocity, and force, or some hybrid or combination of these methods are likely candidates. This type of actuator feedback control, entailing the use of encoders, resolvers, tachometers, and strain gages, is well understood and has been successfully implemented in many industrial robots and motion control systems. Some difficulties arise for this vehicle because, even for the simplest locomotion, it is necessary to control twelve actuators simultaneously. Thus, there needs to be a large number of input channels for the servomechanism feedback sensors.

Just as animals and humans are aware of the magnitude of their physical exertions, it may be desirable to add such a sense to the crawling vehicle. For example, electric current sensing could be used to determine when an actuator or amplifier was near saturation.

Leg force sensors can be used to measure the weight supported by each leg, the lateral forces on the legs, and the forces of foot impact, as well as to determine whether a leg is in contact with the ground. For example, in the cases of the OSU Hexapod (Klein, et. al., 1983) and the Dante II robot (Bares and Wettergreen, 1999), lateral leg forces were sensed using two strain gages on each leg, while axial leg forces were measured by piezoelectric load cells. Strain gages are compact and have no moving parts (although they strain along with the underlying part). Piezoelectric load cells are also compact and can handle high impact loads. However, the load cell used with the OSU Hexapod needed to be recalibrated often. Leg force information gathered with these sensors is useful for balancing the vehicle's weight between the supporting feet, estimating the shear forces acting on

the foot/ground interface, detecting leg collisions, and implementing reactive control to limit the contact forces.

Attitude sensors endow the vehicle with the ability to sense the direction of gravity relative to the robot. This information helps the controller predict whether a certain move would cause the vehicle to tip over. For example, Klein, Olson, and Pugh (1983) used both oil-damped pendulums and a vertical gyroscope for attitude sensing of the OSU Hexapod.

Additional internal sensors could be used to determine the heading direction (compass), remaining fuel or battery charge level, and to test and verify the operation of various system components.

2.5.2 External Sensors

In addition to using internal sensors to monitor the operating parameters of the robot itself, it is also useful to perceive the attributes of the surrounding environment and the relative position of the robot within it. The robot could measure the surrounding terrain relative to itself via contact or non-contact sensors.

Contact sensing can be used to determine which parts of the robot are touching the ground or brushing up against some obstacle. Foot contact sensors allow the robot to confirm when a foot leaves the ground or touches back down. A contact sensor can be as simple as a spring-loaded switch on the sole of each foot. (Note that this can also be accomplished using the leg force sensors discussed in the previous section.) Hirose and Umetani (1980) put contact switches on the sides of their PV II robot's feet for detecting when the robot "stubbed" its foot. Just as some caterpillars have whiskers protruding from their bodies to enable them to detect nearby objects, it may be desirable to attach "whiskers" to the crawling vehicle in the form of limit switches with flexible actuation levers that protrude from the robot.

It may also be beneficial to be able to detect objects that are in close proximity to the robot without actually having to come into contact with them. However, the various optical proximity sensing methods are probably not appropriate for the crawling vehicle because of the unstructured, dirty outdoor environment in which the robot is expected to operate. Specifically, the proximity sensors would need to be mounted near the ground to detect most obstacles, but dust or mud would probably occlude their lenses. Likewise, the various proximity sensors based upon electromagnetic induction and magnetism would be unlikely candidates because of their inability to detect non-metallic obstacles and objects that do not have magnetic fields, respectively.

To avoid the necessity of traveling slowly and "feeling" its way along, it would be best to pre-plan the vehicle locomotion based upon an accurate model of the terrain. For simple, well-known operating environments, a precise map of the terrain could be preprogrammed. However, because most applications will involve exploring new terrain or operating within environments that can change over time, it will be desirable to give the robot the equivalent of a sense of sight so that it can make its own measurements of the surrounding terrain.

There are several alternate technologies for producing rough approximations of human "vision" for robots. These rely on triangulation of terrain features from two or more points of view, time of flight measurements of reflected signals, or phase shift measurements of reflected signals. More specifically, these technologies include ultrasonic echolocation, mono and stereo video, omnidirectional video, radar, and scanning laser range finders.

Most researchers using ultrasonic echolocation have used a transducer made by Polaroid. This technique has enjoyed its greatest success when banks of these transducers are used within indoor environments.

Mono and stereo video cameras have become commonplace peripherals on mobile robot prototypes since low-cost CCD (Charge Coupled Device) cameras became commercially available. Omnidirectional video has been implemented using rotating cameras, fish-eye lenses, spherical mirrors, and conic mirrors (Yagi, et. al., 1994). However, while video devices mimic vision most closely, the automated interpretation of the images produced is another major area of research (as will be briefly discussed in Section 2.5.3).

Scanning laser rangefinders have been used by a number of researchers in several major projects (Waldron and McGhee, 1986; Simmons and Krotkov, 1991; Horn and Russ, 1994; Krotkov and Hoffman, 1994; Bares and Wettergreen, 1999). This sensor consists of a laser rangefinder and a motorized mirror system that causes the laser beam to scan across the terrain ahead of the robot. As the laser scans, discrete range measurements are taken. These can then be transformed into a 2-D grid of terrain elevations to form a map (Krotkov and Hoffman, 1994). The most capable of these systems (Horn and Russ, 1994) is said to detect range and surface reflectance out to a range of 15 m, with a range resolution of 0.45 mm.

2.5.3 Sensor Fusion

Terrain perception and modeling involves more than just taking measurements of the terrain. The sensor data must be "digested" so as to produce a terrain model that is useful for motion planning. Specifically, different terrain measurements must be combined, or fused, together to form a terrain map that embodies the best estimates of the terrain geometry.

This process of *sensor fusion* can involve one or more of the following challenges:

1. combining the terrain measurements from different sensors

2. combining terrain measurements that were taken at different times

3. combining terrain measurements that were taken from different vantage points.

Combining these different measurements into a common frame of reference can entail converting between different systems of coordinates (e.g. radial versus Cartesian), magnitudes of uncertainty, resolutions, sampling rates, fields of view, perspectives, and depths of field (McKerrow, 1991). Due to the difficulties of sensor fusion, the top locomotion speed of many mobile robot designs has been limited, not by dynamic mechanical considerations, but by the speed of "vision" processing (Bares and Whittaker, 1989; Roman and Reinholtz, 1998).

However, in the case of the crawling vehicle, the "head" leg pair is expected to move intermittently. Thus, during part of each locomotion cycle, the vision system and/or other 3-D mapping sensor(s) will have a stationary frame of reference for their measurements of the terrain ahead of the robot. This will simplify sensor fusion because there will be fewer temporal and spatial differences in the frames of reference of the various measurements, and because the controller will have more time to analyze the sensor data before moving across it, in contrast to continuously moving vehicles.

2.5.4 Mapping

When newly acquired terrain data is fused with prior knowledge of the terrain, it must be incorporated into a terrain model, or map, for later reference. This map is in actuality a computer software data structure. According to McKerrow (1991), the data structure chosen is crucial to operation of the vehicle because, to a great degree, it determines which sensor fusion and path planning algorithms can be used efficiently, as well as the amount of storage and processing power required for these tasks. McKerrow (1991) classifies these algorithms into four broad types: path maps, free-space maps, object-oriented maps, and composite space maps. He lists 12

characteristics of an ideal map data structure, but concludes that, “no data structure has yet been found that meets all these criteria. Hence, data structure design is an exercise in trading off task requirements against implementation efficiency.” A more recent review of these issues is presented by David Conner (2000).

For immediate path planning, the robot only needs high-resolution information about the terrain underfoot and directly ahead along the path of travel. Because the data structures for such information consume a large amount of computer memory, the robot must "forget" the details of what it has already walked over so it can have enough available memory to store the new information about the terrain it is approaching. Thus, researchers such as Bares and Whittaker (1989) have separated the mapping task into the creation and use of two maps of different scale and structure: a global map and a local map. Points of interest and landmarks used for large-scale navigation are stored in the global map. Terrain data used for immediate path planning (as discussed above) are stored in the “Local Terrain Map”.

Composite space maps appear to be the logical choice for modeling the unstructured, natural terrain of the local maps. However, a variety of data structures can be used for storing them. According to McKerrow (1991), data structures for storing composite space maps include point grids, area grids, quad-trees, and rule-based maps. If modeling volumetric space is desired (e.g. including data for ceiling heights), then a cell decomposition method, such as the oct-tree representation or one of its derivatives, may be the preferred method (Zhu and Latombe, 1991, Rao and Arkin, 1990). Several researchers have advocated storing the Local Terrain Map as 2-dimensional arrays of elevation values (Wu, 1993; Krotkov and Hoffman, 1994), citing the simplicity of the method, its relative ease of sensor fusion, and the ability to derive terrain slopes and curvatures from this representation.

2.6 Navigation and Controls

An autonomous robot must simultaneously control a number of different operations. Humans and animals plan routes of travel, use vision and force sensing to perceive the terrain, control their muscles to produce locomotion, and regulate their digestion, respiration, and heart rate in order to supply the necessary power for their nervous system and muscles. Similarly, mobile robotic systems must not only control their actuators, but also need to make navigation choices, sense and evaluate the immediate terrain ahead, and provide themselves with the power necessary to accomplish their actions. Thus, the control of a locomotion system is inherently hierarchical, or layered.

Recent research concerning the control of autonomous robots is divided into two schools of thought, with a spectrum of hybrids between them. Section 2.6.1 takes the more traditional approach of describing the navigation and controls systems of the crawling vehicle as a hierarchy of functional modules. Section 2.6.2 discusses the alternate approach of defining the navigation and controls systems for the vehicle as layers of behaviors. Finally, Section 2.6.3 presents a conceptual discussion of the controls hardware.

2.6.1 A Functional Decomposition of the Crawling Vehicle Controls

Various other researchers, including Todd (1985) and McKerrow (1991) have described the architectures of mobile robot control systems as hierarchies of functions. Applying this method to the multibody passive-legged crawling vehicle yields the functional decomposition diagram shown in Fig. 2.20.

Figure 2.20 - Functional Decomposition of the Crawling Vehicle Controls Hierarchy

2.6.1.1 Destination Selection

The topmost functional module shown in Fig. 2.20 concerns the task of destination selection. The artificial intelligence technology for fully autonomous destination goal selection is in its infancy. A basic means of achieving this is via pre-programmed motives and rules. For example:

- To maintain its operational status, the crawling vehicle could be programmed to return to its refueling station whenever it gets low on fuel.

- To map a region, the crawling vehicle could be programmed to synthesize a locomotion path (i.e. select a series of destinations), which when followed, would cause it to carry its sensors to within, say, 5 meters of every point within the area of interest.

- To locate a source of radiation, the robot could be programmed to seek higher levels of radiation by taking a series of measurements over an area, computing a gradient field of the radiation, and then steering to move toward the higher levels of radiation (similar to some optimization algorithms).

For reasons of task flexibility and safety, however, it is debatable whether autonomous goal selection is truly desirable.

On the other hand, using an on-board human driver with the crawling vehicle would be both disconcerting for the driver (because of the intermittent motion) and would defeat the purpose of having a vehicle that could go places too dangerous for people. Thus, this vehicle would most likely be teleoperated, at least initially.

Still, because of its numerous actuators, it would be both tedious and difficult to manually control. Thus, the best method of operating the vehicle would probably be as a "telerobot". As described in Sheridan (1989):

> Add some autonomy to the remote teleoperator and the result is the telerobot. Instead of controlling a telerobot's every move, a human supervisor can simply state an objective. The telerobot then makes decisions and acts based on an on-board computer and signals from its own sensors...

2.6.1.2 Global Path Selection

Given the current location of the robot and its selected destination, the global path selection task of the controls hierarchy determines the basic route to be taken to get there. Hence, this stage embodies the traditional notion of navigation.

Synthesizing a "global" path generally involves planning a path that is beyond the range of the robot's own terrain sensors. Therefore, it must rely upon some form of map (e.g. the global map mentioned in Section 2.5.4) to make its selection.

Assuming there are no obstacles between the current position of the vehicle and its destination, and also that the intervening slope is not excessive, then a "bird's eye" view of the optimal path would be a straight line between them. However, in many cases there will be obstacles and varying slopes, so the shortest path will not be the safest. Hence, path synthesis is usually a challenging design problem. As with other design problems, there are usually numerous, even infinite, feasible solutions.

By a feasible route, it is meant a path from the current position to the selected destination position along which the vehicle will not collide with anything and along which the vehicle can continuously support itself using a sufficient number of safe footholds that are within its reach. To achieve this, the global path plan must take into account the anticipated terrain geometry, the vehicle mobility (actuation unit workspaces) and vehicle stability. In addition, depending upon the vehicle mission, there may also be a transit time constraint.

As in other design problems, while there may be many feasible solutions, in general, some of them will be much better than others. Therefore, it is desirable to choose

wisely from among the feasible paths in order to optimize certain criteria. Depending upon the mission requirements, these criteria can include some combination of the following: minimizing energy consumption, minimizing peak power requirements, minimizing transit time, and/or minimizing the risk of instability. Note that, individually, these criteria often conflict with one another. For example, minimizing the transit time will often entail choosing paths that do not minimize instability risk and moving at speeds that do not minimize the peak power requirements. Also, note that this optimization process may itself have a maximum allowable time constraint.

2.6.1.3 Local Path Selection

Based upon the overall route decided in the navigation planner, the local path selection stage of the controls hierarchy selects and then specifies the precise path over which the crawling vehicle is to travel. In many ways, this task is similar to the global path selection discussed previously. However, at this level, the path is selected using detailed terrain geometry information gathered from the vehicle's own on-board sensors and stored in the Local Terrain Map.

Whenever possible, the path should be chosen to avoid terrain features that would increase the probability of having the vehicle undergo any of the locomotion failures discussed in Sections 1.5 and 1.6. For example, to maximize stability, an ideal path would pass a safe distance from wall-like obstacles, fissures, and steep lateral slopes.

Path optimization at the local path level is similar to that of the global path level. The best path to satisfy the mission requirements is selected based upon a trade-off between desired speed, efficiency, and margin of safety (e.g. for efficiency, the path is chosen to minimize unnecessary climbing, etc.). But if no feasible path can be found, the local path planner can request that a new global path be selected at the prior stage of the hierarchy (see Fig. 2.20).

The path plan can be specified by a set of curves that mark the desired path of the vehicle's centerline. These curves could be either 2-D planar curves that are superimposed onto the 3-D terrain, or 3-D space curves that trace the path along the terrain surface. The initial elevation measurements of a patch of terrain along the path of travel will generally not be as accurate as measurements taken later, after the robot has gotten closer. That is, as the robot travels, it can progressively measure the terrain elevations along the path more accurately, enabling the progressive re-evaluation of the projected local path plan. As a result, 3-D curve path plans would need to be recomputed and updated in light of the new elevation information. Thus, the use of 2-D path plans may be preferable.

Mobile robot navigation and path planning is a vast area of research. Both Larkin (1996) and McKerrow (1991) (in Chapter 8 of his book) provide interesting overviews of the subject. However, the specifics of path selection are beyond the scope of this work.

2.6.1.4 Gait Selection

Having determined the local path, the next task is to select which gait to use to travel across the next segment of that path.

Animals use a variety of gaits. For example, horses can walk, trot, canter or gallop. Similarly, most legged vehicles can perform a huge number of possible gaits (McGhee, 1985). Likewise, the multibody passive-legged crawler is capable of traveling using a variety of gaits. Different gaits have different performance attributes. Some gaits enable faster locomotion than others, some are more stable than others, and some are more efficient. Often these performance goals conflict, and a trade-off decision must be made to select the gait. For example, gaits with the best stability generally maximize the number of legs on the ground in support phase. Gaits with the best speed generally maximize the number of legs moving forward in

swing phase. Thus, higher speed gaits are typically less stable because they require more of the feet to be simultaneously moving, and hence, off of the ground.

In order to select a particular gait from the available choices, the risk associated with traveling over the path segment must be balanced with the needs of the mission. For example, if the mission is to explore an area and there is plenty of time available, then only small risks are tolerable. If, however, the mission is to rescue people who are suffocating due to a mining accident, then much higher risks are tolerable in the interest of locomotion speed.

By "risk" it is meant the probability of locomotion failure. As discussed in Section 1.6, the primary form of locomotion failure "risk" for the multibody passive-legged crawling vehicle is instability. A tip-over event will definitely slow the vehicle down. And depending upon the size of the robot and its design, a tip-over may well cause mechanical failure of the vehicle and thus failure to perform its mission, since it will never reach its destination. Clearly, terrain ruggedness is a key factor in determining risk. However, determining risk involves uncertainty, such as the terrain perception uncertainty and the unforeseen events discussed in Section 1.8.1.2.

How rough can the terrain be for a particular gait? Two methods seem possible for estimating the risk: run-time simulation and rule-based estimation.

With run-time simulation, the controller would test a gait by simulating the next subsequent locomotion cycle based upon the vehicle's perception of the terrain immediately ahead and underfoot, and then determine whether the robot would be in too much danger of getting stuck or tipping over. (A method of determining a "stability margin" will be described in detail in Ch. 4 and other methods will be discussed in Ch. 8.) If the gait tested with the simulation proved to be too risky, then, the planner could choose a more stable gait or request a new path from the local path planner. (Note that the simulation tools discussed later in this work would form the foundation of this technique.)

With rule-based estimation, the controller would analyze the Local Terrain Map and then apply a set of rules to determine the risk/reward of using each possible gait. Such rules might be based upon the ratio of the area without feasible footholds to the total area along the path. The simulation tools discussed later in this work may be helpful in determining the appropriate criteria for such rules.

2.6.1.5 Gait Generation

Gait generation involves timing. While the sequence of leg pair states (i.e. supporting or striding) is defined by the gait selection, the *rate* at which the gait sequence is performed can be varied. Gait generation timing variations can be useful for "pacing" the robot in a number of situations. For example, the robot may need to reduce the gait execution speed in order to limit power consumption. Or alternatively, it may need to delay when unusually long sensor processing computations occur, when difficulties achieving secure footholds cause an extended period of guarded motions, or when one or more actuators experience torque or speed saturation, which might occur when the robot is climbing a steep slope.

2.6.1.6 Foothold Selection

As seen in Fig. 2.20, the foothold selection task of the controls hierarchy is a task that must be performed for *each* leg pair. In order to travel along the selected path using the selected gait, specific foothold locations for each leg pair must be defined. While these are distinct tasks, they are not fully parallel because, due to workspace considerations, the foothold selections of most leg pairs are dependent upon the foothold locations selected for the adjacent leg pair that strides before they do.

An acceptable foothold is a safe location on the ground to place a foot so as to provide firm, secure support and good traction; therefore, holes, loose rocks, and slippery slopes must be avoided. In order to select the two footholds for a particular

leg pair, the robot controller searches the Local Terrain Map of the ground surface immediately ahead of the leg pair's current position. Specifically, it searches for two, safe, acceptable footholds that are within the workspaces of the two Stewart-Gough platform mechanisms that are connected to the front and back of the leg pair. Typically, the expected area is located at approximately 90% of the maximum stride length along the local path. If footholds there are unacceptable, then the controller can search ahead up to the maximum stride length. If acceptable footholds are still not found, the search then proceeds backwards from the 90% area toward the minimum stride length.

Some gaits may allow more leeway in choosing the locations of footholds than others. One advantage of having a long vehicle with many feet is that footholds are not absolutely required for all the feet. In fact, during some locomotion cycles, entire leg pairs can be left "hanging" without supporting footholds, so long as other appropriately placed leg pairs are keeping the robot stable. Stability calculations will probably need to be done here to confirm that there is enough support. Foothold selections may need to be altered if the footholds of adjacent leg pairs must be shifted from their expected areas.

2.6.1.7 Leg Pair Trajectory Specification

Robot motions are defined by trajectories. A complete trajectory specifies all 6 degrees of freedom of a rigid body as functions of time. That is, it defines both the 3-dimensional path (x, y, z) and the spatial orientation (roll, pitch, yaw) of a rigid body as functions of time. In the case of the multibody passive-legged crawling vehicle, the rigid bodies requiring trajectories are the individual leg pairs. Given the current position of a leg pair at its present footholds and the selected destination foothold locations, the task of the leg pair trajectory specification stage of the controls hierarchy is to the synthesize a trajectory that will move each leg pair from its current foothold position to its selected destination foothold position.

The trajectory plan for a leg pair of the multibody passive-legged crawling vehicle will, in some ways, be considerably simpler than the trajectory plan for a leg of a single-body active-legged walking vehicle. Specifically, the leg of a walking vehicle must lift its foot and recover forward rapidly, then stop the foot in mid-air and accelerate it backwards until it has matched the velocity of the ground relative to the moving vehicle chassis. Once it has matched velocity (so as to prevent excess shear forces on the foot at the moment of contact), it can then lower its foot. After its foot is on the ground, the leg continues to move as it sweeps backward to propel the robot body forward. In contrast, a leg pair of the multibody passive-legged crawling vehicle starts from a stationary position, lifts off of the ground, moves forward, decelerates to zero velocity relative to the ground, and then lowers to the ground until its feet touch. Once back on the ground, the leg pair is not required to move (although in some cases it might be desirable to tilt some). That said, the actuators attached to the leg pair will need to move whenever an adjacent leg pair is in motion.

The specified leg pair trajectory plan is subject to the following functional constraints:

- It must produce motions that avoid collisions with terrain, such as running into an obstacle, stubbing a foot, or scraping the side of a leg pair against a wall.
- The leg pair path must be continuously within the workspaces of the adjacent actuation units.

Furthermore, an ideal synthesized trajectory would also have the following characteristics:

- It should produce smooth, continuous curve paths in Cartesian space.

- The path curve should be monotonic so as to minimize vertical motions. That is, it may go up once and then down once, but should not oscillate vertically during a single leg step trajectory.

- It should be smooth in the temporal sense, avoiding unnecessary accelerations and decelerations.

- For a given vehicle speed, the trajectory should minimize the required peak actuator accelerations and peak actuator speeds.

Trajectories can be planned and specified using two basic approaches: joint space and Cartesian space (sometimes called task space). Each has its own basic advantages and disadvantages.

2.6.1.7.1 Joint Space Trajectory Specification

Using the joint space representation, trajectories are specified in terms of the positions of each of the joints as functions of time. As a practical matter, with joint space planning, the beginning and ending positions of the trajectory are initially defined in Cartesian space, and are then converted into their joint space equivalents using inverse kinematics. Then, all the intermediate positions along the trajectory are specified without reference to the robot's environment and without the use of inverse kinematics. Instead, the intermediate joint positions are determined by an algorithm, such as splining cubic polynomials or linear interpolation of each axis of motion, so that the initial joint space solution is smoothly transitioned to the ending joint space solution. Thus, a joint space trajectory is specified as coefficients for the spline curves that define the positions and velocities of the joints as functions of time. For the case of the multibody passive-legged crawling vehicle, these joint positions are the translations of each of the prismatic links of the Stewart-Gough platform mechanisms. During execution, the spline curves for all the joints are

progressively evaluated at the update rate of the actuator controllers. Each new set of the resulting prismatic joint parameters is fed to the corresponding actuator controllers as reference inputs. Finally, the motions of the actuators produce the 6 DOF (*x*, *y*, *z*, roll, pitch, and yaw) trajectory through Cartesian space.

Joint space specification is advantageous in that inverse kinematics is not needed for intermediate positions, it may lend itself to a less centralized control system, and any continuous joint space path within the workspace of the robot can be successfully executed (i.e. singularities are not a problem). Furthermore, joint space trajectories can be rapidly executed because they involve computing relatively simple curve functions, with results that are *directly* fed as reference inputs to the actuator controllers of each corresponding joint.

The major problem with joint space planning and specification is that there is NO guarantee that the resulting leg pair trajectories will be accurate or reasonable in the real world. The resulting trajectories may not stay on the intended path (possibly causing collisions with the terrain), and may produce excessive wasted motion, etc. Thus, they may violate the desired characteristics and even the functional constraints for trajectories described at the beginning of Section 2.6.1.7.

2.6.1.7.2 Cartesian Space Trajectory Specification

Cartesian space trajectories are defined directly in terms of 3-D paths through the working environment of the robot (i.e. Cartesian or task space) as functions of time. Thus, a Cartesian space trajectory is specified as coefficients of the curves that define the 3-D paths. However, in practice, most actuators are controlled using joint space parameters for their reference inputs, feedback signals, and error signals. Thus, during runtime execution, the path curves for each independently moving rigid body are progressively evaluated, and then each new Cartesian coordinate frame is transformed via the inverse kinematics computation into its equivalent joint space

representation. These joint space results are then fed to the actuator controllers, which, in turn, cause the robot to move through Cartesian space according to the proscribed trajectory.

Cartesian space specification is advantageous because the entire trajectory is planned and specified from the measured terrain data, resulting in leg pair trajectories that should be accurate in the real world. Thus, smooth, efficient, well-coordinated, collision-free trajectories should be achievable.

One disadvantage of Cartesian space specification is that it requires that the inverse kinematics computations be performed many times to transform the Cartesian trajectories into the joint space trajectories usable by the low-level controls. Also, for some mechanisms and certain inverse kinematics solutions, a continuous path specified in Cartesian space will become discontinuous when converted into joint space, producing erratic motions during execution. (This happens when the inverse kinematics solution becomes imaginary due to a Jacobian singularity (Craig, 1988).)

2.6.1.7.3 Selecting the Basic Trajectory Specification Method

So far, looking at joint space and Cartesian space trajectory specification in general terms, it is unclear which would be best for the crawling vehicle. There are potential problems with both options. Therefore, their use must be evaluated based upon the particulars of the conceptual design. The logic for choosing the basis for the trajectory specification scheme will be presented as a progressive series of questions and answers.

1. *When the specified trajectories are generated at runtime, what form of output should be sent to the actuator controls?*

So far, it has been assumed that the actuator controls of the robot would follow the more common practice of using joint space control (that is, the actuator controllers

would use joint space parameters (typically joint position and velocity) for the reference inputs, feedback, and error signals to control each actuator). However, in light of this difficult design decision, we now look at the alternative to joint space controls, namely, Cartesian space controls. Using Cartesian-based controls would require that the forward kinematics be evaluated inside the servo control loop (i.e. at the update rate of the controllers) (Craig, 1988). However, as discussed in Section 2.2.1.4, the forward kinematics of the Stewart-Gough platform is an iterative calculation. Thus, Cartesian based control would slow the update rate of the actuator controllers, limiting their effectiveness. Hence, it is probably not appropriate for this robot and we are back to the original assumption of using joint space control.

2. *Must trajectories be specified to avoid obstacles?*

Rodney Brooks (1989) and his colleagues constructed a small walking robot that successfully traveled over moderately rough terrain without the use of trajectory plans that included obstacle avoidance. The inevitable collisions with uneven terrain features were handled by using reflexive (or reactive) controls and aided by the fact that lightweight robots (in this case, 1 kg) cannot easily damage themselves anyway. However, it would be extremely difficult to construct the crawling vehicle on a similarly small scale, especially if it is made to carry significant payloads. Since the impact forces on a vehicle scale with the 4^{th} power of its size (Donner, 1987), unwanted collisions with the terrain could be damaging for a larger vehicle. Furthermore, a major design goal for this conceptual design is to make a robot that can traverse more than just moderately rough terrain. Thus, for the crawling vehicle design, obstacle avoidance is necessary.

3. *Is there a viable alternative to using Cartesian space planning and specification to accomplish obstacle avoidance?*

To implement obstacle avoidance, either the trajectories must be specified in Cartesian space, where they can use the Local Terrain Map to avoid collisions, or the terrain obstacles must be converted into some type of joint space representation so that collision-free path plans can be synthesized entirely in joint space. (Such joint space mapping methods have been proposed by Tsai and his colleagues (1993) and Selensky (2001 [ONLINE])). However, because this application (unlike manipulators) involves the movement of a vehicle relative to the terrain, the vehicle will, in effect, be continuously confronted by new environments as it travels. Thus, it seems cumbersome to transform the ever-changing landscape ahead of it into a joint space representation. Hence, the traditional approach of handling obstacle avoidance by using Cartesian space trajectory specification seems to be the better choice here. Also, as will be explained in Section 2.6.1.11, accurate real-world position control will be required for coordinating the motions of consecutive Stewart-Gough platforms.

4. *Will singularities cause significant problems when using Cartesian space trajectory specification?*

The primary drawbacks of using Cartesian space trajectory specification are the possibilities of singularities and of long computation times for the inverse kinematics solutions. However, both of these difficulties are highly dependent upon the mechanisms for which the trajectories are to be specified.

According to Ma and Angeles (1991), the singularities of parallel manipulators can be classified into three categories: architectural, configuration, and formulation. In the case of the Stewart-Gough platform mechanisms, architectural singularities can be avoided at the design stage by proper selection of the spheric joint locations (Ma and Angeles, 1991). The configuration singularities of the Stewart-Gough platform described by Fichter (1986) would all involve large angular motions of one leg pair relative to the next. Such extreme motions are not necessary for locomotion, even

on rough terrain, and specifications of such angles can be avoided at the trajectory planning stage. Finally, to the best of the author's knowledge, the inverse kinematics algorithm to be described in Ch. 4 does not exhibit any formulation singularities. Thus, singularities are not expected to cause significant difficulties for this robot.

5. *Will the inverse kinematics be too computationally intensive?*

As will be shown in Chapter 4, the inverse kinematics algorithm for the Stewart-Gough platform mechanism is remarkably simple and efficient. Thus, using Cartesian space trajectory specification with joint space control seems particularly well suited for the crawling vehicle concept because it uses Stewart-Gough platforms for its actuation units. This should provide the best of both options: trajectory specification with effective obstacle avoidance in the real-world, and high-frequency control loops for the actuators, which will enable the accurate tracking required for coordinating the multiple actuators.

Having decided upon Cartesian space trajectory planning and specification, other questions concerning the approach and the mathematics to use to specify motion trajectories still remain. Recall from the in-depth discussion in Section 1.8.3 that a stepwise planning and specification method will be used in this work rather than the more serpentine shape control, or backbone, methods proposed by Salerno, Chirkjian, Burdick, Tavakkoli, and Dowling. Note that this stepwise method of trajectory specification must be performed for each moving leg pair, and is therefore often a parallel task. (Section 2.6.3.4.4 discusses the use of parallel computing to address these tasks.)

Further description of the trajectory specification method will be curtailed here, because it is a primary topic of Ch. 5 – Motion Programming.

2.6.1.8 Trajectory Generation

Based upon the trajectory plans specified at the previous level of the control tasks hierarchy, trajectory generation entails computing the desired Cartesian location and orientation of each leg pair. These leg pair trajectories are generated at run time.

Rapid execution at this level depends upon whether the trajectory specification allows for an efficient algorithm (such an algorithm will be presented in Ch. 5). Like the preceding level, it is composed of parallel tasks since it is performed separately for each leg pair.

2.6.1.9 Inverse Kinematics

The inverse kinematics computation converts the Cartesian space representation of discrete positions along the trajectory into their corresponding joint space representations. Specifically, given the relative positions of the leg pairs, the task of the inverse kinematics level of the controls hierarchy is to determine the link lengths and velocities of all the linear actuators that comprise the intervening actuation unit mechanisms. As discussed in Section 2.2.1.4, this is a simple calculation for the Stewart-Gough platform mechanism. (An algorithm for computing this will be presented in Ch. 4.) The inverse kinematics level of the control tasks hierarchy is composed of parallel tasks, because it is performed for each actuation unit.

2.6.1.10 Actuator Controls

From the discussions so far in Sections 2.6, it is clear that the control functions of the crawling vehicle are inherently both hierarchical and concurrent. It is hierarchical in that each functional level is dependent upon the results of its "parent" level. It exhibits concurrency in two forms: inter-level and intra-level.

Inter-level concurrency occurs because the different levels of the hierarchy work together as a pipelined process. While a “child” level in the hierarchy is using the most recent solution produced by its parent, the parent level is busy computing its next solution. Hence, all the levels of the hierarchy are acting simultaneously. Each level of the hierarchy has a control update rate; generally, the further down in the hierarchy, the faster the update rate must be.

In contrast, intra-level concurrency occurs at some stages of the hierarchy because there are parallel tasks to be performed. The most obvious case of fully parallel concurrency is here with the control of the actuators. Each Stewart-Gough mechanism requires the simultaneous control of 6 actuators. Most movements associated with locomotion would require the simultaneous action of 2 or more of these actuation units, and hence 12 actuators. Thus, like most walking robots, the crawling vehicle would require controlling several actuators simultaneously.

The actuator controls are at the lowest level of the controls hierarchy (shown at the bottom of the diagram in Fig. 2.20). Given the desired link lengths and velocities of all the robot linear actuators, the function of the actuator controls is to cause the linear actuators to track these positions as accurately as possible. That is, the joint space positions and velocities output by the inverse kinematics are the reference inputs to the actuator controls. These controlled actuators then move the prismatic joints of the Stewart-Gough platforms, and hence, collectively, cause the robot to move.

The position control of the individual actuators can be accomplished by employing the control laws used in industrial motion controls, such as Proportional-Derivative (PD) or Proportional-Integral-Derivative (PID). Thus, to achieve smooth, real-time performance, and coordinated motion, it is expected that the update frequency of the actuator's controllers must be on the order of hundreds of updates per second, which is typical for many industrial motion control tasks.

Controlling many actuators simultaneously will provide some engineering challenges because of the multiple-input, multiple-output nature of the problem, the complexity of wiring the controls hardware, and the existence of actuator redundancy (the subject of the next section).

2.6.1.11 The Contention Problem

Figure 2.16 shows an example in which actuator redundancy occurs. In the figure, looking at either the leg step of the 2nd or 3rd leg pairs, we see that the striding leg pair structure is supported and moved by the actuation units on either side, and that these actuation units are attached to the adjacent stationary leg pairs, which are on the ground in support phase. The striding leg pair has 6 DOF because it is a rigid body. However, its position must be controlled by the Stewart-Gough mechanisms in front of and behind it. Thus, 6 + 6 = 12 actuators are controlling 6 DOF, and six of the actuators can be said to be *redundant*. Later in this work it will be shown that it is often desirable to have two adjacent leg pairs striding forward simultaneously. This would result in having 12 DOF (2 leg pairs) being controlled by 18 actuators (3 actuation units). Thus, the multibody passive-legged crawling vehicle is redundantly actuated. (Note that this *actuator redundancy* is not the same as the *kinematic redundancy* that is often desired with manipulators (Colbaugh and Glass, 1992). Instead of providing the flexibility of multiple kinematics assembly solutions for a specified position, actuator redundancy demands precise cooperation between actuators to achieve the specified position.)

The need to control redundant actuation systems is not unique to the multibody passive-legged crawling vehicle. It is common to statically stable legged vehicles, because their legs form closed kinematic chains with the ground (Gardner, 1992; Kumar and Waldron, 1988; Klein and Chung, 1987). For example, a single-body robot with 4 legs that each has 3 DOF would result in a total of 12 actuators controlling the 6 DOF of the robot chassis. Even when one of the legs strides

forward and its foot is off of the ground, there will still be 9 actuators controlling the 6 DOF of the robot.

Whenever the number of actuators in a linkage exceeds the number of degrees of freedom that they collectively control, coupling between two or more of the actuators occurs, creating a situation called *antagonism*.

Antagonism can be useful because it facilitates controllable compliance. In fact, vertebrate animals and humans have many muscles that act in antagonism to one another. For example, in the case of human arms, antagonism enables us to move stiffly so as to have high precision paths and high disturbance rejection, or alternatively, we can choose to move with compliance, so as to slide around obstacles or bounce off obstructions.

A number of researchers have recognized the potential of using antagonistic systems to achieve improved performance by providing variable stiffness, removing backlash and dead zones (thus enabling high precision position control), and reducing peak torques by allocating torque between actuators (Zeng and Hemami, 1997; Nakamura, 1991; Hogan, 1985). However, antagonism between actuators is only useful when the control system is sophisticated enough to make use of it.

In his Master's work at Virginia Tech, Patrick Brennan coined the term "contention" to describe a special subclass of antagonism. By contention it is meant antagonistic actuators where the actuators are not backdrivable (i.e. they are self-locking). In this case, when coupled redundant actuators do not cooperate to match positions at any point in time, high stresses and high friction will result, wasting energy and possibly locking-up or even damaging one or more actuators.

Brennan discussed four possible solution strategies for the problem of actuator contention:

1. Precision kinematics solutions (no force feedback)

2. Building passive compliance into the system
3. Passive mechanisms (backdriving)
4. Force feedback control of the actuators

By itself, obtaining precision kinematics solutions may not be sufficient to solve the contention problem. But clearly we should try to minimize Cartesian space position errors by every feasible means, including having both inverse kinematics and joint space controls that are accurate and rapidly updated. This should significantly reduce contention-induced stresses.

Some passive compliance already exists in the form of backlash in the spheric joints, U-joints, and linear actuators, as well as in the elasticity of the prismatic links and leg pair structures. However, it is unlikely that this would provide enough range of compliant deflection. Therefore, more compliance could be added to absorb any relative position errors between coupled actuators. This could take the form of spring-dampers mounted in-line with the prismatic joints or to the bases of the spheric joints. However, there is a limit to the application of this solution because too much compliance would prevent accurate position control. Specifically, the robot would sag whenever leg pairs were held off of the ground (especially during cantilever maneuvers). Also, too much compliance might allow undesirable vibrations. However, it may be useful to add a small amount of compliance as a safety factor.

The next possible solution involves the utilization of backdrivable mechanisms, that is, avoiding the use of self-locking linear actuators. For example, recall that in Section 2.2.1.4, that leadscrews and ball screws (among others) were proposed as possible actuators for the prismatic joints. The self-locking feature of most leadscrews is attractive because no power would be required to hold the actuators (and hence the robot) in desired positions. However, this self-locking feature would also result in high stresses in contention. In contrast, most ball screw drives are backdrivable, and

many commercially available ball screws include brakes that enable locking when required. Furthermore, because ball screws generally have much lower starting torque, higher peak speed, and higher efficiency than leadscrews, they are the better choice.

The last proposed solution method was to use active force feedback control. Even if backdrivable actuators are adopted, it may be desirable to use force feedback control methods in order to minimize the energy losses that result from backdriving.

In response to a related redundant controls problem, Klein and Chung (1987) proposed partitioning the controls so that some actuators operate in position control while some operate in force control. Similarly, Brennan advocated an arrangement where the actuators of a Stewart-Gough platform on one side of a leg pair would operate in position control and the actuators of the Stewart-Gough platform on the other side of the leg pair would operate in force control. With this arrangement, the position-controlled actuators operate as the "master side" of the leg pair and simply perform their planned trajectories, while the actuators on the "slave side" of the leg pair use force control to stay out of their way and remove any contention stresses.

Low-level control of these redundant actuators will require some form of force control. The possible algorithms include: active stiffness control, impedance control, admittance control, hybrid position/force control, hybrid impedance control, implicit force control, or explicit force control (Zeng and Hemami, 1997).

Regardless of which force control algorithm is used, it seems likely that feedback sensors in the form of strain gages mounted to each linear actuator would be required.

2.6.2 Control Behaviors

The hierarchical breakdown shown in Fig. 2.20 is a traditional decomposition of the crawling vehicle control system into functional modules. It is a useful way of looking at the problems of controlling this complex, layered, multiple-input multiple-output system. Similar functional module hierarchies have been used as the bases for the control systems of several previous mobile robot projects (Hirose, 1984; Todd, 1985; Bares and Whittaker, 1989).

A working control system for the multibody crawler can be constructed based on the functional modules illustrated in Fig. 2.20. However, basing a control system, particularly the operating system software of a mobile robot, upon the functional module decomposition is not necessarily the best approach.

An alternative approach, termed “behavior-based control”, was introduced in three papers published in the 1986-87 time frame, the best known being a paper by Rodney Brooks (1986). In this paper, he describes the "subsumption architecture" or SSA. Briefly, instead of decomposing the problem of controlling a mobile robot on the basis of the "internal workings of the solution", the SSA decomposes the problem on the basis of the "desired external manifestations of the robot control system".

As an example of the SSA, Brooks (1986) lists several "layers of control", each of which controls an increasing "level of competence", for a wheeled mobile robot. Specifically, he lists the following (going from the most fundamental competencies to the more sophisticated):

0) Avoid contact with objects (whether objects move or are stationary).

1) Wander aimlessly around without hitting things.

2) "Explore" the world by seeing places in the distance that look reachable and heading for them.

3) Build a map of the environment and plan routes from one place to another.

4) Notice changes in the "static" environment.

5) Reason about the world in terms of identifiable objects and perform tasks related to certain objects.

6) Formulate and execute plans that involve changing the state of the world in some desirable way.

7) Reason about the behavior of objects in the world and modify plans accordingly.

In this list, "each level of competence includes as a subset each earlier level of competence". Each layer of control deals with an individual competency, and is developed and tested, starting from the lowest level and proceeding to the highest. At runtime, the higher-level control layers can "subsume" the inputs to the lower levels when necessary. Higher-level control behaviors can thus be added without modifying the lower level ones. The SSA approach is said to yield a more modular control system, offering advantages of improved "robustness, buildability, and testability" (Brooks, 1986). Thus, these advantages are, in many ways, similar to those that structured and object-oriented software programming offered over previous paradigms.

In their overview of behavior control, Gat and his colleagues (1994) explain that modules for task-achieving behaviors are "connected directly to sensors and actuators and operate in parallel". This results in "two significant advantages to behavior control over traditional functional decomposition". First, fast behaviors do

not have to wait for slow ones, thus improving real time response. And second, because behaviors are task-specific, designers can take advantage of the structures of tasks to make behavior modules surprisingly simple (Gat et. al., 1994).

At the system level, task-achieving behaviors dealing with world perception, route planning, and navigation can be used for essentially all mobile robots. Thus, while the task-achieving behaviors listed above were developed for a three-wheeled mobile robot used for exploration, they could also be used with the multibody passive-legged vehicle. However, due to the additional kinematics complexity of the multibody passive-legged crawler relative to the three-wheeled robot studied by Brooks, it may be useful to add several lower levels of competency.

Therefore, with the crawling vehicle, the following lower level competencies (-7 to -1) are suggested for use along with the higher level ones (0 to 7) previously listed:

-7) Coordinate the prismatic links that are in contention so that no lock-ups or assembly failures occur. (A control law to accomplish this may be to have some actuators move to minimize stress and strain in the links. One implementation of this would involve having the actuators on one side of a moving leg pair act as the "master side", with the actuators on the opposite side of the leg pair being the "slave side". The master side would move the leg pair while the slave side would move to stay "out of the way" according to the control law mentioned above.)

-6) Coordinate redundant actuators so that they work cooperatively to bring about leg pair motions. (The control law may be to *equalize* the compressive stress/strain on the pushing side of a moving leg pair with the tensile stress/strain on the pulling side. Thus, rather than limiting motion performance to just that which could be produced by the actuation unit on the "master side" of the leg pair (while having the "slave side" simply move to stay out of the way

without contributing forces to help move the leg pair), this method will bring about higher performance through cooperative motion.)

-5) Maintain ground contact with as many feet as possible. That is, control the stance of the robot to compensate for the collapse of foot supports, such as by sinking soils, shifting rocks, or broken legs.

-4) Maintain the leg pair pitch orientation to be normal to the nominal surface of the terrain directly underfoot without exceeding a threshold bending load on the legs.

-3) Maintain the approximate longitudinal spacing of leg pairs (accomplished by subsuming position control over some leg pairs when necessary).

-2) Plan and execute a locomotion cycle.

-1) Select an appropriate gait for the given terrain conditions.

The work of Brooks and other pioneers of this paradigm set in motion a great deal of research concerning subsumption architectures, reactive controls, and other instances of behavior-based controls. Numerous research efforts have been completed (Maes, 1989, Mataric, 1992, Gat et. al., 1994; Crowley, 1994, Fernandez et. al., 1994; Arkin and MacKenzie, 1994, Brooks, 1996) and others are underway exploring the permutations of (as Brooks characterizes it) "good old fashioned artificial intelligence (GOFAI)", "behavior-based robotics", and the range of hybrid possibilities between them. In general, the trend of recent years has been away from pure "GOFAI" and increasingly towards behavior-based controls (Inoue, 1996).

The control system functional modules and low-level behaviors described for the crawling vehicle in this section should be helpful for future researchers and designers, but the determination of the ideal control system for the multibody passive-legged crawling vehicle is well beyond the scope of this research.

2.6.3 Control Hardware Architecture

Having discussed the functions to be performed and the behaviors to emerge, we now come to the conceptual design of the basic control hardware architecture. As the prior discussions have shown, both the motion programming and control functions are inherently hierarchical and concurrent. Furthermore, for the vehicle to operate continuously, these tasks must be performed in real-time. These functional interdependencies and constraints on control latency make the design choice of control hardware architectures difficult.

2.6.3.1 Centralized Control Versus Distributed Control

There seem to be two basic philosophies for the control of complex systems, regardless of the system type. Whether the complex system is a nation's economy, a government, an organism, a factory, or a mobile robot, the components of the system can either be orchestrated by a centralized control system or a distributed control system.

There are advantages and disadvantages for each of these methods depending upon the system to be controlled and its complexity, task uniformity, task interdependencies, and method of communicating sensor and control signals (which determines the transmission time lags and error rates).

2.6.3.1.1 Advantages of Centralized Control Relative to Distributed Control

Assuming insignificant time delays in processing and communication, centralized control systems have some advantages over distributed control systems. First, if the system is fully understood, it is possible to control it better with a centralized control system because its control laws can simultaneously consider all of the system inputs and outputs. Second, because the central controller has the "big picture", it is good

for arbitrating between the subgoals of its subsystems. Finally, a centralized system is conceptually simpler, because decisions will always be made by the central computer. In contrast, a distributed system requires careful design to make sure that the subsystems share enough information and coordinate with each other so that they work together instead of against each other.

2.6.3.1.2 Disadvantages of Centralized Control Relative to Distributed Control

Centralized control systems also have disadvantages compared to distributed control systems. First, for very complex systems, it is extremely difficult to understand the problem sufficiently to design a central controller that can satisfy the various subgoals of a multiple-input, multiple-output system. Therefore, multiple subgoals can often be satisfied better with multiple single-input, single-output controllers. Second, a large central computer is often more expensive than several less powerful computers used in a distributed control system. Third, for many problems, a central processor is slower computationally than a parallel processor arrangement. Fourth, if the central computer fails, the system will be essentially inoperable. However, if one of the computers of a distributed system fails, the others will probably be able to continue, and (with proper design) may even be able to assume the functions of the malfunctioning computer. Fifth, and perhaps most importantly, centralized control systems usually require longer communication lines from the sensors to the computer, and from the computer back to the actuators. These longer I/O lines cause a number of detrimental effects on the control system:

- Time lags, which can greatly reduce the effectiveness and output stability of the control system, even when the problem is well understood.

- Signal attenuation caused by the longer transmission distances mean that weak sensor signals, such as those from strain gages, must be amplified before being

transmitted back to the central computer. This additional signal conditioning introduces errors in the signals and increases the cost of the control hardware.

- Longer I/O lines have a greater tendency to pick up electromagnetic interference (EMI), especially in the radio frequency spectrum (RFI), which can produce errors in the sensor readings. These misconceptions of the true situation cause the central controller to output erroneous control signals.

- Longer I/O lines also mean more lines to fail. Cabling and connectors are probably the most failure-prone parts of robotic systems. Fatigue failures of cabling are the bane of robotics; non-ferrous cables wear-out because they are flexed when robots move.

2.6.3.2 Control Hardware Architecture Examples

Because of the considerations explained in the previous section, factory process control systems moved away from using Direct Digital Control (DDC) towards using Distributed Computer Control (DCC), where a supervisory computer interacts with numerous individual controllers (Kilian, 2001).

With vertebrate animals, while vision, memory, and planning functions are centralized in the brain, the control of reflexes and simple locomotion can be done using neurons and ganglia distributed throughout the body. This fact is evidenced by experiments with headless chickens and the spinal cat experiments performed by Grillner and Zangger (Donner, 1987). In these experiments, the animals were able to walk despite, in one case, having completely severed heads, and, in the other case, having the spinal cord severed at the neck.

While the hierarchical processes necessary for a mobile robot have been successfully run on time-shared single processor computers (Todd, 1985), more often, legged vehicle researchers have sought to make the form of the computing

hardware resemble the hierarchy of control functions. Todd (1985) discusses a variety of different multiprocessor connection schemes such as tree, bus, and ring configurations. Some functioning examples include: a hexapod controlled by two Motorola 68000 processors, (Donner, 1987); the OSU Hexapod, which used five processors (Todd, 1985); the Adaptive Suspension Vehicle's three-layer hierarchical organization of seventeen single board computers, including one for each of its six legs (Waldron and McGhee, 1986); the Ambler project, which used three CPU boards and nine motion control boards connected together in a VME backplane and communicating via shared memory (Simmons and Krotkov, 1991); and the Dante II (Bares and Wettergreen, 1999), which used a Sparc2 CPU board for perception processing, plus three Motorola 68030 CPU cards, and several custom boards for I/O, all connected in a VME bus backplane.

A strong case for distributed control hardware is made from observing the experiences of the Honda R&D Company. Specifically, when they redesigned the P2 humanoid robot to create the P3 robot, they switched from using a single computer to using a distributed control system made up of several smaller computers. These small computers were located at the joints of the robot and were connected by a LAN (Local Area Network). As a result of this redesign, the number of wires was reduced from 650 down to 30, and the number of connectors and contacts were reduced from 2000 down to 500, thus greatly improving the “practical reliability” of the P3 robot (Hirai, 1999).

On the other hand, legged robot projects such as the Ambler (Simmons and Krotkov, 1991) and Dante II (Bares and Wettergreen, 1999) have found it beneficial to have some centralization in the form of a “task controller”. The task controller orchestrates the other concurrently executing processes by coordinating “interprocess communications, task synchronization, and resource management” (Simmons and Krotkov, 1991).

Based upon the preceding reasoning and examples, the control hardware for the multibody passive-legged crawling vehicle should be designed to be as distributed as possible, but centralized where necessary to ensure that sub-processes do not conflict with one another.

2.6.3.3 Proposed Control Hardware Architecture

The capabilities of commercially available controls components and even their underlying technologies are rapidly moving targets. Therefore, specific processors, busses, and other components will not be selected at this stage. Component specification is a task for the detail design stage. Here at the conceptual design stage, we will address *architectural* issues concerning three important, interrelated questions:

- Where should the major components of the control system hardware be mounted?
- How many separate processing packages should be used to control the robot?
- How shall the tasks of the controls hierarchy (Fig. 2.20) be allocated between these separate processing units?

2.6.3.3.1 Control System Hardware Requirements

The first step in answering these questions is to recall the requirements of the control system hardware design. Specifically, it must support the reliable execution of the functions described in Section 2.6.1 and illustrated in Fig. 2.20. To accomplish this, the controls hardware must interact with all the sensors and actuators of the robot. However, the exact components to be used with the robot have not been specified at this stage. Therefore, we must base our conceptual design decisions

upon assumptions of which sensors and actuators are most likely be used in the final design. Hence, we shall assume the following requirements:

- The controls hardware must interface with a scanning laser range finder, two video cameras (for stereo), as well as sensors for foot force/contact, prismatic link position, velocity, and force. With the exception of the first two types, the sensors will be distributed evenly throughout the robot structure.

- The controls must also interface with actuators in the form of 6 motors or control valves (depending upon the types of actuators used) for each Stewart-Gough platform mechanism.

- Additionally, the control system will need to report its activities to and receive instructions from its human teleoperators. Thus, it will likely require communications via digital radio or satellite.

2.6.3.3.2 Selecting the Number of Controls Processing Packages and Their Mounting Locations

Considering the initial question of where to mount the control system hardware components, we must first choose between mounting them outside of the robot, onto the robot, or inside of the robot. While it is possible to connect external controls hardware to the robot via a tether, this is not a practical long-term solution because the numerous wires would make the wire bundles large, unwieldy, and unreliable. Concerning the remaining to options, it is clear that internal mounting would provide better protection for the electronic components than mounting them to the exterior of the vehicle.

Having chosen internal mounting, many alternatives still remain. It is important to note that suitable enclosures for the controls hardware can be found in the payload boxes that are a part of each leg pair assembly. Thus, all the processing power could

be safely mounted within one of the leg pairs' payload boxes, distributed in equal proportions to all of the leg pairs, or any combination between these two extremes. The choice of processing hardware location is governed by the desire to maximize reliability by adhering to four primary design goals:

1. Minimize the I/O line length
2. Minimize the number of data line connectors
3. Minimize the required data transmission bandwidth (don't send data if it can better be used in-situ)
4. Keep things simple

The method applied here was to narrow the design space by recognizing the locations of the sensors and actuators and then applying the short I/O line length constraint.

Looking first at the scanning laser rangefinder and stereo video cameras, we note that in order to get an unobstructed view, they must of necessity be located at the front of the robot. Because the speed of safe locomotion is closely tied to speed of processing terrain perception data and because of the large amount of data involved, the terrain perception processors should be located near the "vision" sensors within the payload box of the front leg pair.

The various sensor fusion, global and local path selection, and other locomotion planning functions all require high speed access to the Local Terrain Map data. A local terrain map with sufficient resolution for locomotion planning will require a large amount of computer memory storage for its data structures. Furthermore, the map is almost continuously being updated with new information. Thus, a large amount of bandwidth would be required to repeatedly transmit the map data over a network to other processors. Therefore, because of this difficulty, it seems prudent

to avoid transmitting the Local Terrain Map, and instead, to let it sit in the block of memory where it was initially stored.

This leads to the conclusion that it would be best for the processors responsible for terrain perception processing, sensor fusion, mapping, and locomotion planning functions to all be combined into a shared memory architecture. Such a scheme would enable all of the processors devoted to these tasks to have access to the memory that stores the Local Terrain Map. Hence, these processors would be packaged so that they share the same high-speed bus. Furthermore, it is recommended that these processors be consolidated into a single card cage mounted in a shockproof, waterproof, temperature-controlled enclosure. According to Bares and Wettergreen (1999), such an arrangement provides effective protection and is convenient for both development and maintenance. For simplicity, this cluster of controls processors will be referred to as the "head" processors.

External communications with the robot would generally concern high-level operations such as teleoperator instructions about destination selection and video feedback of what the robot was "seeing". Therefore, if space permits, the logical place for the communication processing hardware is near the planning processors and video sensors within the "head" leg pair. This would shorten the line lengths between the transmitters/receivers and the planning processors.

Having looked at the controls hardware pertaining to the terrain perception sensors, planning-related functions, and communications equipment, we now focus on the remaining sensors of the robot. First, note that the actuator feedback sensors are all located on the Stewart-Gough platforms, and that two leg pair assemblies flank each of these platform mechanisms. Furthermore, the foot contact and leg force sensors are attached to the lower part of each leg pair.

In order to avoid all the noise and lag problems elucidated in Section 2.6.3.1.2, we strongly desire to minimize the length of I/O lines between the sensors and their

A/D hardware. In light of this, and by virtue of the leg pair payload boxes' close proximity to both the linear actuator feedback sensors and the foot contact/force sensors, there should be, at a minimum, an I/O processor located inside of each leg pair payload box.

Recognizing that all the leg pairs and all the Stewart-Gough platforms have the same basic structures and same basic functions to perform, it is therefore possible to maintain uniformity and simplify the controls design by making all of the leg pairs' controls hardware conform to the same design. Control functions that are not general to all the leg pairs would be performed on additional processors that might also be mounted inside one or more of the leg pairs, where appropriate. (For example, the front leg pair would contain both a generic leg pair processing package and the "head" processors.)

2.6.3.3.3 The Robot Control Processor Network

The network connection diagram in Fig. 2.21 illustrates the architecture concept that results from the foregoing discussions. Note that some connections depicted as a single "wire" in this illustration may, in fact, be bundles of several wires depending on the requirements of the motors and sensors.

With this design, each leg pair processor would send control outputs and receive feedback inputs the Stewart-Gough platform *ahead* of it. The rationale for this design stems from the recognition that there are $n-1$ Stewart-Gough platform mechanisms and n leg pairs. While it would also be possible for each leg pair to control the platform mechanism *behind* it, because the front leg pair already contains the head processors and sensors, it is advantageous to reduce some of its leg pair control responsibilities by using the method chosen.

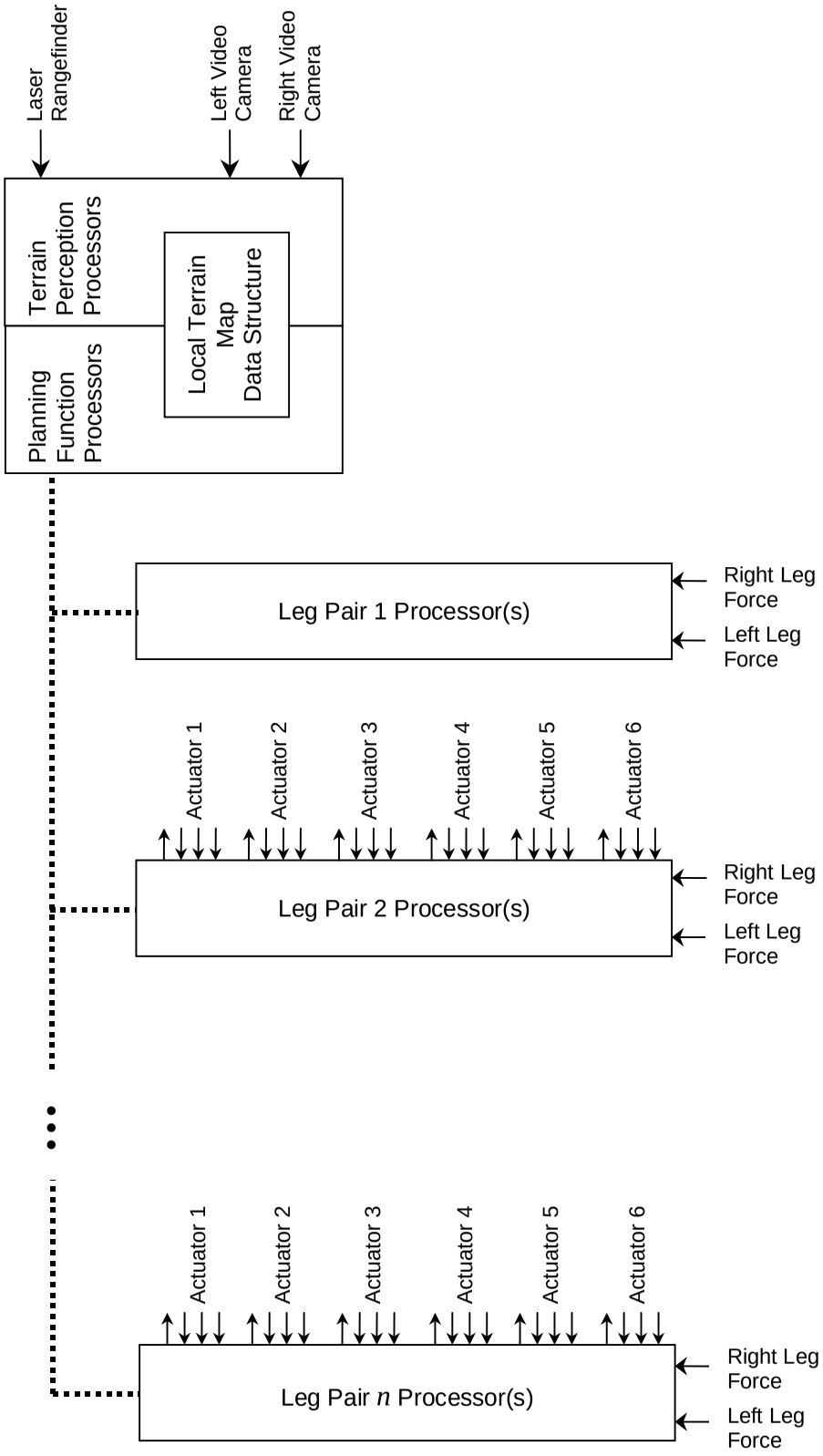

Figure 2.21 — Connection Diagram of the Robot LAN

The use of controls processors distributed throughout the robot structure requires some sort of local area network (LAN) to convey the commands and data of interprocess communications. In Fig. 2.21 these digital connections are denoted by dashed lines.

Some candidate busses/protocols for the robot control processor network include (among others): RS-422, IEEE-488, USB, IEEE 1394, Ethernet, RS-485, and various industrial "fieldbusses". In addition to the numerous wired LAN alternatives, it is possible to transmit network data without physical wires. One way to accomplish this is via infrared light transmissions from one leg pair to the next. Infrared transmitters and receivers mounted on the back side of a leg pair could point at their counterparts mounted to the front side of the leg pair behind. One way to ensure the continuing optical alignment of the transmitters and receivers would be to attach them at opposite ends of one of the intervening prismatic links, aligned parallel to its lengthwise axis. Thus, when one leg pair turns sharply relative to another leg pair, the optical transmission would continue unabated.

A second wireless method of transmitting network data for the robot is via radio waves, for example, by using standards such as Bluetooth, HiperLAN2, or one of the IEEE 802.11 standards. Wireless data transmission could be an important enabling technology for robots because it combines the wiring simplicity of networks with the mechanical robustness of having no flexing wires to break.

The specific busses and protocols are not chosen here because of the extreme progression of these designs in recent years, where a significant new communication protocol seems to be invented every few months. Furthermore, at this conceptual stage of the research, a definitive network bandwidth requirement is not known.

2.6.3.3.4 Allocating Functions to the Processors

Having determined the number of separate processing packages to be used and their mounting locations, we come to the final architectural issue of how the controls function responsibilities will be allocated between the processor packages.

Most of the controls function allocation decisions can be inferred from discussion of the design logic to this point. However, looking at the controls hierarchy in Fig. 2.20, it is unclear whether it would be best to perform leg pair trajectory specification on the head processors or on the leg pair processors. On the one hand, it is a parallel task that must be performed for each leg pair, with results that are then used by each leg pair to generate its leg step trajectories. Hence, it is intriguing to consider performing these parallel tasks on the individual leg pair processors, computing in parallel, and then having the results immediately available for the next operation.

On the other hand, the use of parallel processing is limited by the need for the leg step trajectory specification process to access the Local Terrain Map data. This data is sizeable and would probably be time prohibitive to transmit to all of the leg pair processors. Even so, it is possible to envision a scheme where the leg pair processors each receive a subset of the Local Terrain Map that represents only the terrain environment for their next stride. Once they have planned using it, they could pass the terrain patch back to the next leg pair processor behind them so it could use the data for its next stride. However, even transferring only part of the terrain map to each leg pair processor would still likely overburden the leg pair processors. This is because, unlike some of the planning processors that may have the luxury to "stop and think", the leg pair processors must function with very low latency, (i.e. in real-time). Since the iterative nature of search algorithms makes their computation times unpredictable, it is likely that executing them would interfere with the leg pair processors' more urgent real-time functions.

If leg step specification was done on the leg pair processors, it is certain that a large amount of map data would need to be transmitted. In contrast, if the leg step specification was done with a head processor, the resulting trajectory data could be sent to the leg pairs with relative ease. Specifically, the motion programming method developed in Ch. 5 will enable the transmission of a smooth leg step trajectory specification using as few as 26 numbers. Thus, after evaluating the two alternatives based upon both the computational latency and network bandwidth issues, it is clear that leg pair trajectory specification should be performed on one of the head processors. Figure 2.22 shows the allocation of the controls hierarchy functions between the "head" processors and the leg pair processors.

Note that, as with a number of prior mobile robot projects discussed in Section 2.6.3.2, the "head processor" will most likely be several separate CPU's sharing the same backplane. For example, it will probably be desirable to have a separate CPU devoted exclusively to vision processing and mapping functions. Likewise, it may also be desirable to partition the planning levels of the controls hierarchy onto separate processors to ensure real-time performance for the low latency operations required for task control and generating the gaits.

With the partial exception of the leg pair processor in the front leg pair, each of the generic leg pair processors is responsible for receiving data, performing operations, and sending output as follows:

- Each leg pair processor receives the leg step trajectory specification for its next stride and clock synchronization signals from the head processor, leg pair position data from the leg pair ahead of it, linear actuator feedback signals for position, velocity, and force for each of the 6 actuators that make up the Stewart-Gough platform ahead of it, and sensor data regarding the contact status of its two feet and the axial force in each leg.

Figure 2.22 - Allocation of Controls Functions Between the Processing Packages Located in the "Head" and in the Leg Pairs

- In addition, each leg processor generates the leg step trajectory for the leg pair in which it is mounted and computes the inverse kinematics for the 6 actuators.

- Finally, each leg processor sends control signals (joint length and velocity inputs) to the link length controllers for the 6 actuators, its own leg pair position to the leg pair behind it, and its leg pair position, foot contact status changes (and perhaps leg force values as well), and error flag signals to the "head" processor. In view of these many tasks, it seems likely that sub-processors for the individual actuator controllers will be required to achieve the desired real-time performance.

Note that, although the leg pair processor in the front leg pair will not be required to control a Stewart-Gough platform ahead of it, it is still needed to generate its trajectory, share its position with the second leg pair and the gait controller/leg step planner, and to handle all of the functions related to its foot and leg sensors.

2.6.3.3.5 Controls Hardware Architecture Summary

So to summarize, the “head” processors, mounted inside the front leg pair of the robot, would control the functions in the controls hierarchy down to the leg step trajectory specification; the leg pair processors would control the lower level functions from the trajectory generation on downwards to the actuator controls. (The shading in Fig. 2.22 indicates this division of responsibilities.)

Based upon this overall hardware architecture, each of the controls packages of the robot should be designed in view of the thermal, vibration, shock, humidity, power, EMI/RFI, and electrostatic discharge (ESD) characteristics of the operating environment. For example, if the operational environment will expose the vehicle to liquids and dust, then well sealed enclosures should be used. (Note that utilizing

high quality enclosures can enable the use of lower-cost, off-the-shelf controls electronics components in a mobile robot (Bares and Wettergreen, 1999)).

2.7 Chapter Summary

Section 1.9.1 described a basic three-stage model of design. This chapter has addressed the first of these stages, conceptual design, for the proposed crawling vehicle. Specifically, the objective of this chapter has been to describe the major components that would make up a multibody passive-legged crawling vehicle, to discuss the alternative designs that are possible for each of these components, and to explain the rationale for selecting between these alternatives. Certain fundamental building blocks of the proposed vehicle were selected, while the final selections of other aspects of the design were not possible at the conceptual design stage, because either further analysis was needed or other prerequisite design features had yet to be determined. In such cases, the field of alternatives was narrowed as much as possible.

The chapter began by defining the physical structure of the vehicle as an alternating sequence of leg pair assemblies and actuation units. A parallel mechanism known as the Stewart-Gough platform was selected for the actuation units because of its stiffness, strength, minimal singularities, simple inverse kinematics, and similar workspace to that of caterpillar segments. After discussing the robot structural configuration, the basic types of motions that the robot would perform were introduced. Next, design alternatives for power sources and sensors were examined, including a discussion of sensor fusion and terrain mapping. Finally, the chapter presented an extensive look at navigation and controls, starting with a description of the hierarchy of functions that must run concurrently to control the crawling vehicle. Further discussions encompassed each level of the hierarchy, the possibility of using

behavioral controls to accomplish lower level functions, and a description of the controls hardware architecture design. Among the results of this section are the decisions to use Cartesian space trajectory planning with joint space actuator controls and to divide the controls processing functions so that functions from trajectory generation and below would be executed in parallel on identical processors located within the leg pairs, while higher level planning functions and terrain perception processing would be performed on a cluster of processors located inside the front leg pair of the robot.

Thus, a basic conceptual design of the robot vehicle has been presented with an emphasis on the physical structure and mobility, and most of the groundwork has been laid for the configuration design stage. However, the conceptual design presented in this chapter has been based upon a rudimentary knowledge of the caterpillar. Therefore, before proceeding to develop synthesis and analysis tools for the configuration design stage, we shall first, in the next chapter, investigate a species of caterpillar in order to confirm and refine the conceptual design, and thus aid in accomplishing the goal of producing a vehicle capable of emulating the excellent mobility of caterpillars.