Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2008
August 3-6, 2008, Brooklyn, New York, USA

DETC2008-49616

DEVELOPMENT AND COMPARISON OF GAIT GENERATION ALGORITHMS FOR HEXAPEDAL ROBOTS BASED ON KINEMATICS WITH CONSIDERATIONS FOR WORKSPACE

Mark Showalter
Robotics and Mechanisms Laboratory
Department of Mechanical Engineering
Virginia Tech
Blacksburg, Virginia 24061
Email: mshowalt@vt.edu

Dennis Hong
Robotics and Mechanisms Laboratory
Department of Mechanical Engineering
Virginia Tech
Blacksburg, Virginia 24061
Email: dhong@vt.edu

Daniel Larimer
Open Tech Inc.
Blacksburg, Virginia 24060
Email: dlarimer@opentechinc.com

ABSTRACT

This paper explores the interdependance of walking algorithm and limb workspace for *the Multi-Appendage Robotic System (MARS). While MARS is a hexapedal robot, the tasks of defining the workspace and walking agorthm for all six limbs can be abstracted to a single limb using the constraint of a tripedal statically stable gait. Thus, by understanding the behavior of an individual limb, two walking algorithms have been developed which allow MARS to walk on level terain. Both algorithms are adaptive in that they continously update based on control inputs. The differences between the two algorithms is that they were developed for different limb workspaces. The simpler algorithm developed for a 2D workspace was implemented, resulting in smooth gait generation with near instantaneous response to control input. This accomplishment demonstrates the feasibility of implementing a more sophisticated algorithem which allows for inputs of: x and y velocity, walking height, yaw, pitch and roll. This algorithm uses a 3D workspace developed to afford near maximum step length.*

INTRODUCTION

This paper presents walking algorithms developed for a mobile robot, MARS. As this paper is the second in a sequence of two papers, discussion of the walking algorithms is based heavily on the robot design, kenimatics, workspace definitions, and nomenclature intoroduced in the first paper [1]. The focus of this paper is to develop walking algorithms applicable to robots kinematically similar to LEMUR IIb (Legged Excursion Mechanical Utility Rover). The algorithms discussed in this paper are mostly limited to walking on a flat terrain.

Considerable work has been done in generating walking algorithms for hexapedal robots. In 1983 Raibert and Southerland developed a hexapedal walking machine capable of navigating rough terrain using onboard computer controlled gaits [2]. In the same year, the machine ODEX 1 demonstrated the maneuverability of a hexapedal platform by using pre-programed movements to climb into the back of a pickup truck [2]. A few years later, Rodney A. Brooks, et al. developed distributed networks which used layered control for robot control as well as for hexapedal walking algorithm control [3, 4]. Since the late 1980's, Roger D. Quinn, et al. have been controlling hexapedal robots using neural networks based on the cockroach [5-10]. This method of control uses interconnected neurons which exercise excitatory or inhibitory control over each other. The arrangement of these neurons produces various predictable gait patterns in the robot legs depending on the speed required. This neural network control also proved very robust and was able to produce working gaits even with damage to the robot.

While many walking algorithms [2, 7] would be suitable for such planar hexapedal locomotion, developing one sufficiently general enough to handle all navigable terrain and to utilize the kinematic structure of the robot adds to the problem complexity. The adaptable gait-planning algorithms under development are basic in the sense that they are currently only capable of planar locomotion, but general in that they could be used as the foundations for a more sophisticated algorithm capable of navigating complex terrain such as the surface of a spacecraft which may have complex surface features. It is also desirable that the basic elements of a walking algorithm be applicable in using the limbs to manipulate tools. For these reasons suitable base walking algorithms, while currently only capable of planer locomotion, must be capable of precise, pre-determined limb tip positioning. Also, the kinematic structure of the robot allows for body translation in any 3-

space direction, as well as pitch, yaw, and roll, while walking. Therefore, in order not to exclude mechanical capabilities, the base algorithm will be capable of instantaneously and simultaneously executing any combination of translations and change of orientation of the body while walking.

Many walking algorithms have been developed for hexapedal robots with limbs arranged symmetrically on either side of a longitudinal body axis, similar to an insect. Gaits for hexapods have been defined by Song and Waldron [2], as:

Periodic
- Wave gait
- Equal phase gait
- Backward wave gait
- Backward equal phase gait
- Dexterous periodic gait
- Continuous follow-the-leader gait

Non-Periodic
- Discontinuous follow-the-leader gait
- Large obstacle gait
- Precision footing gait
- Free gait

Various periodic wave gaits have been used for hexapedal robots, combined with biologically inspired coordination mechanisms found in stick insects [7]. However, MARS' limbs are axi-symmetrically arranged around the body, not arranged in rows on either side of the body. This, combined with the kinematic design of the limbs, invites the possibility of omni-directional motion. For this reason a walking algorithm which builds on the maximized omni-directional step length described by Schmiedeler [11] was chosen. This algorithm would be a combination of a tripedal wave gait and a precision footing gait. That is, the algorithm would be based on an alternating tripod gait, but with the capability of precisely positioning each limb tip within the workspace.

WORKSPACE AND WALKING ALGORITHM INTERDEPENDENCE

A range of workspaces are possible with MARS limbs. However, for walking, there is a balance between the size of the workspace and the simplicity of the adaptive walking algorithm. Specifically, a workspace with very simple geometry results in a computationally simpler walking algorithm. Conversely, a workspace with very complex geometry requires more computation to define a stride line across the workspace. Walking algorithms are presented in this paper which use both 2D and 3D workspaces. Generally speaking the larger workspaces are geometrically complex; however, the larger workspaces can be geometrically simplified by only using a simple geometry within the large workspace. A more thorough development of these workspaces can be found in the companion paper to this paper [1].

THE ABSTRACTED WALKING ALGORITHM

The complexity of understanding how six limbs move together to form a walking gait can be reduced by abstracting the problem to one limb. As each limb performs the same role while walking, namely taking a step, a walking algorithm for any number of limbs can be constructed by simply applying the same algorithm to each limb. Walking is achieved by making some of the inputs to such an algorithm dependent on the limb location. For example: from the standpoint of the limb coordinate system, the direction that the robot is walking could be the positive y direction for one limb and the negative y direction for a limb on the opposite side of the body. Also, for statically stable gaits, the limbs are separated into two groups which act together. One group is in contact with the walking surface while the other is not. As a walking algorithm can be developed for any number of limbs by abstracting the problem to one limb, this paper explains the developed algorithms from this abstracted approach. However, all walking algorithms discussed in this paper use the tripedal statically stable gait, much like an insect. Given the two possible walking limb conditions of contact and non-contact, any limb is in the state different from the limbs next to it.

WALKING ALGORITHMS

The walking algorithms presented in this paper are adaptive in that at any point during walking, the robot can respond to input. Further they are limited to walking on level terrain. Because the algorithms operate iteratively, at every iteration new input is transformed to new output which is sent the limb actuators. As the iterations cycle at approximately 10 Hz or more, the algorithms appear to provide seamless instantaneous response to changes in direction, speed, and any other allowed input.

The General Walking Algorithm

The general walking algorithm is based on the general workspace. This algorithem can be used for any robot with six or more limbs kinematically similar to MARS limbs. The limbs can be attached to the robot body at any point and orientation, provided all the limbs can reach the walking surface simultaneously. The limbs can be dimentionaly dissimilar from each other and the robot body can be of any shape, provided that a statically stable gait is possible. Further this algorithem can except as inputs:

- direction
- speed
- walking height
- roll
- pitch
- yaw

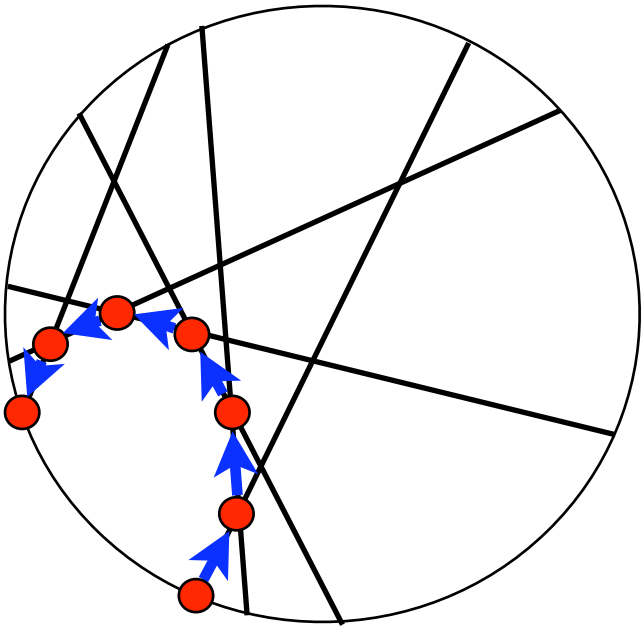

Figure 1: ITERATIVELY GENERATED STRIDE LINES CAN FORM CURVED PATHS.

In this algorithm an entire step is not planned at once. Rather, only a section of the step is planned for each iteration. In this manner, the direction of the robot can be continuously adjusted throughout the step motion, resulting in a curved step path. For each iteration, each limb tip is moved from its current point in space to a new point. Over the course of several iterations the limb tip moves along the path of the step. However, for each iteration, the walking algorithm defines a straight path, or "stride line," across the workspace. To do this, the walking algorithm must compute intersection points of the stride line with the shells which make up the workspace boundary. It should be noted that finding the equation of the stride line is only necessary in order to solve for intersection points with the workspace boundary shells. Otherwise, it would only be necessary to define the next limb tip position by adding the limb tip direction vector to the current limb tip position. Fig. 1 shows the stride lines and limb tip positions associated with a curved step path through a circular 2D workspace.

Defining the Next Limb Tip Location. The basic components of the general walking algorithm are shown in Fig. 2. For each iteration, a new stride line, dependent on the direction and orientation of the robot body, must be found. The next limb tip location will be on this stride line. However, the method used to find the stride line will depend on whether the limb tip is in contact or not. The method for defining each follows:

For the contact limb, the stride line passes through the current limb tip position and is parallel to the velocity vector of the robot body. The direction of the velocity vector of the limb tip is opposite the direction of the velocity vector of the robot body. The next limb tip position is found by adding the limb tip velocity vector to the current limb tip position.

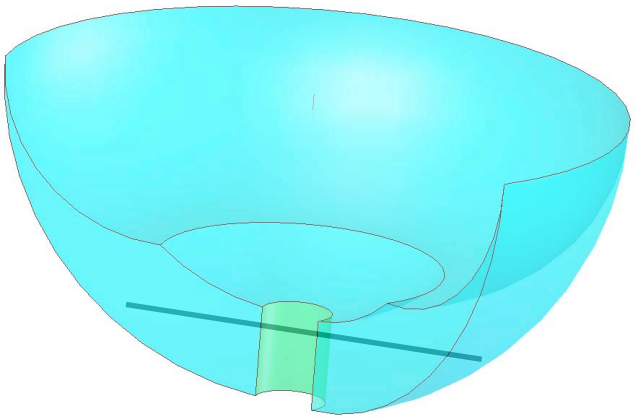

Figure 3: THE IDEAL STRIDE LINES MAKE A LONG PATH THROUGH THE WORKSPACE AVOIDING THE BUFFER CYLINDER.

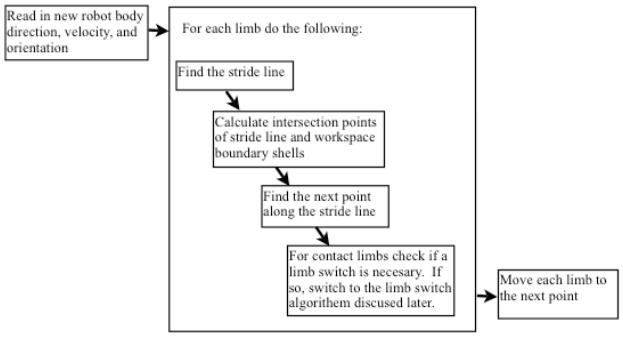

Figure 2: THE BASIC WALKING ALGORITHM.

For non-contact limbs, the process for finding the stride line is more involved. If the robot were directed to walk with the current velocity, the stride lines could be optimized for long strides. This somewhat arbitrary long stride optimization does not necessarily improve the walking ability, but does provide for a more evenly timed alternating gate and reduces the number of contact/non-contact limb switches. Optimization cannot be effected mid-stride for the contact limbs, because they are fully constrained by the robot body velocity and their current contact point. However, in this algorithm both contact and non-contact stride lines would ideally meet the form in Fig. 3. The depicted stride line is tangent to the buffer cylinder. This stride line situation ensures a long stride line and reasonable body stability. Most importantly, however, it is easy to define by the constraints:

- parallel to the body velocity vector
- tangent to the z-axis buffer cylinder at a given height

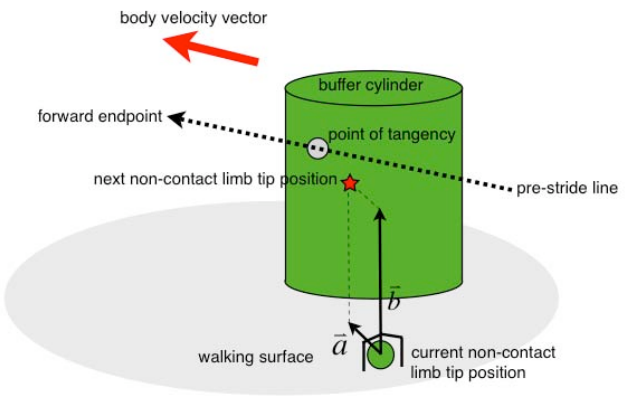

Figure 4: THE NEXT NON-CONTACT LIMB TIP LOCATION IS SELECTED SO AS TO OPTIMIZE THE GAIT FOR THE CURRENT OPERATOR INPUT.

This walking algorithm is not designed to predict the future robot direction, speed, height, and orientation specified by the operator. Rather, the algorithm assumes that the current inputs will remain constant, and the algorithm attempts to optimize for them by moving the non-contact limbs to the starting point of their future contact stride lines. In practice, this method works well because the non-contact limbs position faster than the contact limbs and because of the high iteration rate of the algorithm. However, for the non-contact limbs to move to an optimal position in anticipation of the limbs switching, first a pre-stride line (a line used to find the stride line for non-contact limb tips in a walking algorithm which uses the general workspace) and then a stride line must be found. The pre-stride line is essentially the predicted future contact stride line elevated from the walking surface to the non-contact height. The pre-stride line is defined using the following constraints:

- parallel to the body velocity vector
- tangent to the z-axis buffer cylinder
- a set distance above the walking surface

The immediate purpose of defining the pre-stride line is to identify the forward point where the pre-stride line intersects the workspace boundary -- the forward endpoint. Having defined this pre-stride line, its forward endpoint is then used to help define the non-contact stride line -- the actual line used to help specify the next non-contact limb tip position. The geometry of finding the next non-contact limb tip location is shown in Fig. 4. A vector, let us call it $\vec{a}$, is constructed originating at the current non-contact limb position and pointing toward the pre-stride-line forward endpoint. The magnitude of this vector is twice the magnitude of the body velocity vector. The doubled magnitude allows the non-contact limbs to reach the pre-stride-line endpoint before a contact/non-contact limb switch is necessary. However, if the distance from the current non-contact limb position to the pre-stride-line forward endpoint is less than $\vec{a}$, the remaining steps are skipped, and the next non-contact limb tip position is defined as the pre-stride-line forward endpoint. A second vector, let us call it $\vec{b}$, is formed originating at the current non-contact limb tip position and pointing normal to the walking surface. The magnitude of $\vec{b}$ is equal to a percentage (Ex. 90%) of the difference between *H* and *h*, where:

- *H* is the length of a vector normal to the walking surface which stretches from the walking surface to any point on the pre-stride line
- *h* is the length of a vector normal to the walking surface which stretches from the walking surface to the current non-contact limb tip position

The sum of $\vec{a}$ and $\vec{b}$ added to the current non-contact limb tip position gives the next non-contact limb tip position.

Contact/Non-Contact Limb Switch. If the distance from the current contact limb tip to the contact limb stride line rear endpoint is less than twice the magnitude of the robot body velocity vector, the limbs must switch. The contact/non-contact limb-switch requires two iterations:

Iteration 1

a. The next point for the non-contact limbs is defined as the forward endpoint of a contact stride line passing through the point directly below the current non-contact limb tip position.
b. The contact limbs and non-contact limbs are moved to the next point. At this instant all limbs are in contact.

Iteration 2

a. The original contact limbs become non-contact limbs and the next limb tip position is found using the non-contact limb tip position algorithm.
b. The original contact limbs are moved to their new non-contact position, and the original non-contact limbs (now in contact) are moved to their new contact limb position

Stride-Line Workspace Intersection. Finding where the stride line, or pre-stride line, intersects the workspace gives the endpoints of the stride line. The workspace is constructed of sections of different toruses, spheres, cylinders or plains. First, the stride line intersections with a given geometry are found. Then it is determined if the intersection point or points lie within the section of the geometry which constitutes the shell. After instances of intersection for all shells and plains are found, the two intersection points closest to the "given point" are designated as endpoints. The given point for contact limbs is the current limb tip position. The given point for non-contact limb pre-stride lines is the point tangent to the buffer cylinder. After the end points are found, the points

are screened to determine if they lie on the section of the base geometry which is used in the knee-up workspace. If the points pass the screening, their directional relation to the given point is used to designate the forward and rear endpoints in accordance with the body direction vector.

2D Workspace Walking Algorithm

A simplified version of the general walking algorithm was developed and implemented on MARS by Open Tech Inc. This adaptive iterative walking algorithm uses the 2D circular workspace. Therefore this walking algorithm is not fully capable of changing robot body roll and pitch while walking. However, the algorithm is computationally less intensive and operates at 10 to 60 Hz.

There are two inputs to this algorithm:

- The translational vector in 3-space
- the angular velocity about the z_0-axis

The inputs are translated from body coordinates to limb coordinates. This translation allows for the limbs to be attached to the body at any position and orientation. The next limb tip position is found and the inverse kinematics are used to generate the actuator positions.

The contact limb tips move with each iteration within the circular workspace. However, if the next limb tip position is found to be outside the workspace the limbs switch.

The non-contact limb tips move at a height above the walking surface. The height of the non-contact limb tips is specified by Eqn. (1):

$$Hieght = \sqrt{1-(r_T-r_c)} \tag{1}$$

where r_T is the distance from the limb tip to the center of the circular workspace and r_c is the radius of the circular workspace. The non-contact limb tips must move in the opposite direction of the contact limb tips in order to reach the forward edge of the workspace before the limb switch. Further the non-contact limbs must move at a greater velocity than the contact limbs as the algorithm inputs may change in the middle of a stride. If the non-contact limb tips did not move faster than the contact limb tips constant change in direction could result in a situation where the limbs would need to continuously switch and not be able to achieve the required velocity of the robot body.

DISCUSSION

The 2D workspace walking algorithm demonstrates the feasibility of using an iterative approach in developing an adaptive walking algorithms for limbed robots kinematically similar to MARS. The 2D workspace walking algorithm could be expanded to use the 3D spherical workspace. Use of more geometrically complicated workspaces such as the general workspace would require the computationally demanding use of defined shell boundaries. Implementation of the general walking algorithm would require an as of yet undeveloped limb collision avoidance algorithm. However, use of the 3D common workspace ensures that walking limbs do not intersect each other's workspace. For this reason it is recommended that future work on an algorithm which incorporates roll and pitch be based on the 3D common workspace. Such an algorithm would parallel the general walking algorithm by using stride line intersection with workspace boundary shells.

CONCLUSION

An iterative walking algorithm was demonstrated on a hexapedal platform. Steps are not planned ahead, but rather steps are planned continously. The implemented algorithm is capable of updating the control inputs to the robot and direct the robot acordingly 10 to 60 times a second. Operation of the robot with such an algorithm results in smooth seemingly instantaneous response to inputs.

A more sophisticated algorithm has also been developed which expands control inputs from:

- x and y velocity
- z location
- yaw, or z axis rotation

to include roll and pitch, or x and y axis rotation. Future work will include implementation of this algorithm for the 3D common workspace. Implementation of this more general algorithm will be an inportant milestone toward an adaptive iterative algorithm for uneven terrain.

REFERENCES

[1] Showalter, M., Hong, D., "Workspace Analysis for Walking Algorithm Development with Hexapedal Robots", DETC2008

[2] Song, S. M., and Waldron, K. J., 1989, *Machines That Walk: The Adaptive Suspension Vehicle*, MIT Press, Cambridge.

[3] Rodney A. Brooks, "A Robust Layered Control System for a Mobile Robot," IEEE Journal of Robotics and Automation, March 1986, Volume: 2 Issue: 1, pp. 14-23.

[4] Rodney A. Brooks. A Robot That Walks; Emergent Behaviors from a Carefully Evolved Network. *In* N. I. Badler, B. A. Barksy, and D. Zelter (eds.) *Making Them Move: Mechanics, Control, and Animation of Articulated Figures*. Morgan Kaufmann.

[5] Quinn, Roger D., Nelson, Gabriel M., Bachmann, Richard J., Kingsley, Daniel A., Offi, John, and Ritzmann, Roy, E., 2001, "Insect Designs for Improved Robot Mobility," Proceeding of the 4th International Conference on Climbing and Walking Robots, pp. 69-76.

[6] Chiel, H. J., Beer, R. D., Quinn, R. D., and Espenschied, K. S., 1992, "Robustness of a Distributed Neural Network Controller for Locomotion in a Hexapod Robot" IEEE Transaction on Robotics and Automation, Vol. 8, No. 3.

[7] Espenchied, K. S., Quinn. R. D., Chiel, H. J., Beer, R. D., 1993,"Leg Coordination Mechanisms in the Stick Insect Applied to Hexapod Robot Locomotion," The Massachusetts Institute of Technology, Adaptive Behavior Vol. 1. No. 4. pp. 455-468.

[8] Espenchied, K. S., Quinn. R. D., Beer, R. D., Chiel, H. J., 1996, "Biologically based distributed control and local reflexes improve rough terrain locomotion in a hexapod robot," Robotics and Autonomous Systems 18, pp. 59-64.

[9] Quinn, R. D. and Ritzmann, R. E., ""Construction of a Hexapod Robot with Cockroach Kinematics Benefits both Robotics and Biology," Connection Science, 10(3-4), December 1998.

[10] Allen, T.J, Quinn, R.D., Bachmann, R.J., and Ritzmann, R.E., 2003. "Abstracted Biological Principles Applied with Reduced Actuation Improve Mobility of Legged Vehicles," IEEE International

Conference on Intelligent Robots and Systems (IROS 2003), Las Vegas.

[11] Schmiedleler, James P., Bradley, Nathan J., and Kennedy, Brett, 2004. "Maximizing Walking Step Length for a Near Omni-Directional Hexapod Robot," Proceeding of the 2004 AMSE International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Salt Lake City, Utah, September 28 – October 2.