# An Omnidirectional Gait Control using a Graph Search Method for a Quadruped Walking Robot

Daniel J. Pack and HoSeok Kang
Robot Vision Laboratory
1285 EE Building
Purdue University
West Lafayette, IN 47907-1285

## Abstract

*Two major approaches exist for devising gait control for a legged machine: 1) finding a sequence of leg and body movements of a robot on the basis of kinematic and possibly dynamic considerations assuming all foot placements are valid; and 2) selecting leg and body movements first on the basis of feasibility of foot placements and then accepting those that also satisfy kinematic and possibly dynamic constraints. Consider now the problem of a legged robot pursuing a quarry. With the first approach, the robot will be limited to operating over a relatively flat terrain. If the terrain is uneven, however, the robot will have no choice but to take recourse to the second approach even though it is computationally more demanding. The work done so far in the second approach is limited by either the constraint that the overall direction of ambulation of the robot is known in advance, or by the constraint that the foot placements are limited to the points of a grid superimposed on the topographic map of the terrain. In this paper, we discuss how a free gait can be generated for following a quarry without invoking such constraints.*

## 1 Introduction

Gait controls for legged robots have been the focus of many researchers over the past two decades. Some of the contributions in the kinematic control of quadruped walking robots, listed here in historical precedence, are [3, 2, 7, 6, 5].

Our main interest in walking robots is the development of sensory intelligence for such machines, especially the development of vision-based intelligence. We believe that as a first step in that direction, it is prudent at this time to use only kinematically-based strategies for the gait control and to ignore the system dynamics.

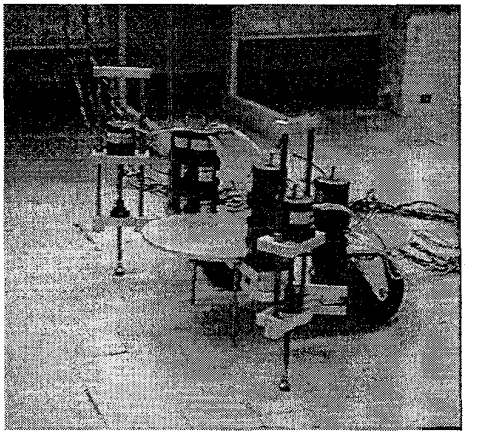


Figure 1: *Quadruped walking robot*

Song and Chen have reported an algorithm to generate a free gait for a quadruped walking chair[7] where a wave-crab gait along with a free gait is used to traverse the four legged chair on an artificial terrain. Pal and Jayarajan[6] have used the heuristic graph search method $A^*$ to generate a straight line free gait for a quadruped. In this paper, we have extended the heuristic search method in [6] to an omnidirectional gait algorithm while removing the following two constraints: (1) the overall direction of ambulation of the robot is known in advance; and (2) the foot placements are limited to the points of a grid superimposed on the topographic map of the terrain.

The paper is organized as follows. Next section describes the design of the quadruped robot being studied, followed by a presentation of the proposed free gait algorithm in section 3. Simulation and experimental results are used to illustrate the proposed free gait in section 4. In section 5, we sumarize the result.

IEEE International Conference
on Robotics and Automation
0-7803-1965-6/95 $4.00 ©1995 IEEE

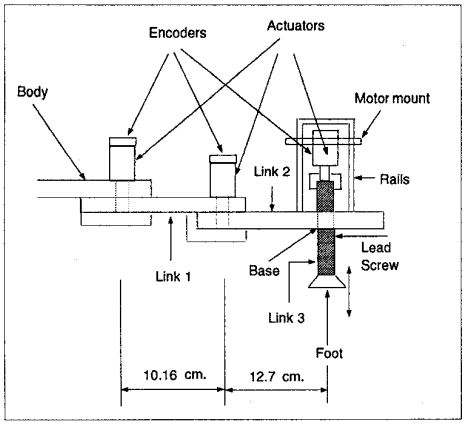
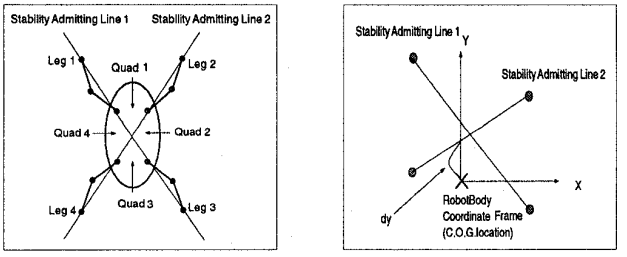



Figure 2: *Each leg consists of three links. The two joints on the left are revolute joints while the third joint is a linear joint.*




(a) (b)

Figure 3: *The quadrants generated by Stability Admitting Lines. The quadrants are labeled Quad 1, Quad 2, Quad 3, and Quad 4. In (a) is shown a situation where all four quadrants exist while (b) shows how Stability Admitting Lines are used.*

## 2 Robot Design

Fig. 1 shows a photo of the quadruped robot under construction. The main body of the robot is an oval-shaped and attached to the body are four legs in the manner shown (two rear legs will be attached in a similar fashion as the front legs). Currently, we are in the process of building rear legs and have placed two wheels temporarily in their places.

Fig. 2 shows the leg design. This design was inspired by the desire to control each joint of our robot independently. The first two links of the leg, link 1 and link 2, rotate horizontally in a manner that they can be used to position the foot on a two dimensional Cartesian space within a region whose shape and size are dictated by kinematic and physical considerations. Link 3 is a ball-screw joint which translates the rotations of a motor shaft into linear motions of the foot for raising and lowering the foot.

## 3 Free Gait Generation

Gait generation using a search scheme possess couple of advantages over other methods: 1) a unified control structure; and 2) an ability to forecast consequences of candidate actions.

### 3.1 Preliminaries

First a few definitions are in order. A leg will be called a *supporting leg* if its foot is in contact with the ground and exerts pressure on the ground. A leg that is lifted clear off the ground will be called a *free leg.* We now introduce the notion of a *Stability Admitting Line (SAL)* that we have found useful for analyzing the static stability of the robot during a gait, Fig. 3(a).

**Definition Stability Admitting Lines (SALs)** *Lines in the ground plane that either connect the supporting feet 1 and 3 or the supporting feet 2 and 4 are called stability admitting lines.*

The importance of such lines for quadruped machines was first observed by Hirose [2] and was formalized by Pack and Kak[5]. Fig. 3(a) shows a typical robot configuration with the SALs labeled. We will now make the reasonable assumption that at all times the positions of the four feet are known with respect to the robot body coordinate frame centered at the c.o.g. of robot. This means that the equations for the SALs can be easily constructed for those feet that are in contact with the ground. Once the SALs are found, the robot can readily determine which leg can be selected as the next free leg.

For example, if the robot c.o.g. is in Quad 3 as shown in Fig. 3(b), and if leg 3 needs to be selected as the next free leg, the robot can easily compute the minimum distance, denoted $d_y$, which should be exceeded by the motion of the body. We will call the difference between the distance the robot body moves and $d_y$ as the *stability margin.*

### 3.2 A* Search Method

We now briefly present a graph search method called $A^*$. The particular graph search method has been shown to produce an optimal solution to various

problems[4, 1]. However the usage of this particular search method in the robotics community has been sparse at best due to its high computational cost. Usage of a suitable heuristic function, however, can significantly increase the efficiency of the search.

To explain the basics of the $A^*$ search method, we first need the concept of nodes in a graph. Each node in a graph represents a state of a system at time t. The crux of the search method is to expand a solution path only on nodes that are promising in some sense. (A path in a graph is a sequence of nodes that are connected from the root node to the current node.) To determine whether a particular node should be included in a solution path, each node is associated with a real-valued function, called the evaluating function. Typically, an evaluation function $f(n)$ has the following form: $f(n) = g(n) + h(n)$ where the argument $n$ represents a particular node of interest, $g(n)$ is the cost from the root node to node n, and $h(n)$ is an estimate cost from the current node $n$ to a goal node. It is clear that there exists some optimal heuristic function $f^*(n) = g^*(n) + h^*(n)$. It is known that if the evaluation function $f(n)$ is chosen such that $g(n) > g^*(n)$ and $h(n) \leq h^*(n)$ for all n, and $f(n)$ is a monotonically increasing function, it is guaranteed that the search will produce an optimal path.[4].

### 3.3 Node Representation

We now list components that form a node in our search graph. Each node contains: locations of the feet and the knees in a robot body frame; the joint angles; elements of the evaluation function; the distance each robot feet traversed; the displacement and algle of the robot body with respect to a world coordinate; an ID number; a selected motion at the node, and the parent node ID number; a current level in the search graph; and a sequence of movements which led to the current node. This node description completely describes the current robot configuration, its position in the world, its relationships with other nodes, and the sequence of past movements that resulted in the current state of the robot.

### 3.4 Generation of Child Nodes

After an expanding node in a search graph is selected, we must make a decision on how to generate its child nodes. To lessen the computational burden of the search, for each node, we limit the number of child nodes to three separate foot and body locations, where each placements must lie on one of the *level circles* shown in Fig. 4. The radius of the level circles

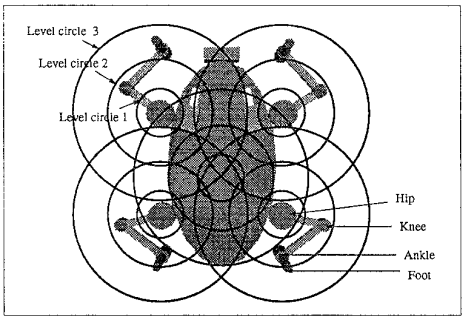



Figure 4: *Generation of child nodes.*

are not fixed and can be changed on-line depending on the current terrain conditions, thus providing flexible foot and body placements that adapt to a local environment. An expanding node can have up to 15 child nodes; three nodes from each leg movement and the body movement. However, kinematic constraints, stability, and the terrain conditions can prevent the generation of desired body or foot locations.

### 3.5 Direction of Movements

To find the new locations of a foot or the body c.o.g, an appropriate direction toward a quarry is computed at the end of each search cycle. The direction for new body locations are computed by first applying a coordinate transformation. The direction is found by simply applying the inverse tangent of x and y values of the quarry in the robot body coordinate frame: angle $\theta = \arctan \frac{y}{x}$. For the movement of legs, the directional angle determined for the robot body frame is converted to each hip coordinate frame.

### 3.6 Search Strategy

If any of the foothold or the body locations on the level circles are not available due to the mechanical, the physical, or the terrain constraints, the radius of a particular level circle is increased in small size. This increment of a level circle continues until either the next level circle is reached, or a valid foothold or a body location is found. This guarantees the robot to find an alternative foothold or a body location within a specified segment along the line toward the quarry if one exists. This process is repeated for all foothold and body placements. We now present the graph generation and search strategy.

[1] Put the root node in the empty set S.
[2] Find a node with minimum $f$ value.

[3] Take the node off from the set S.
[4] Expand the selected node.
[5] Include newly created nodes into the set S.
[6] If the current level of expanding node does not exceeds predetermined level in the search tree goto step 2.
[7] Collect the nodes on the founded path and terminate.

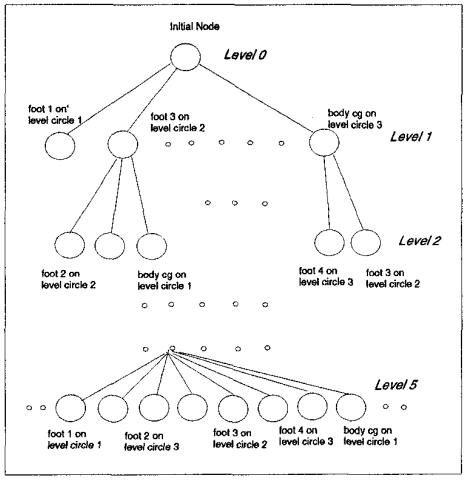



Figure 5: *Typical Graph Generation.*

In step 6, the specified level is empirically determined to be 5, assuming the initial level as 0, see Fig. 5. When a path reaches an intermediate goal (leaf) node, the robot executes the movements found in step 7, and the gait generating search cycle continues with the last intermediate goal node as an initial (root) node for the next cycle of the search. The process continues until the quarry location is within a specified Eucledian distance from the robot.

### 3.7 Evaluation Function

First, we believe it is important that the movement of the robot body should receive a precedence over an individual leg movement. Thus, we have given movements of four legs together an equal amount of importance as the movement of a single robot body motion. Secondly, since there exist various configurations of the robot with its c.o.g. at a same location, the evaluation function includes a measure to examine desirability of a robot configuration to pursue a quarry.

Thirdly, we want to include the stability margin value in the evaluation function to differentiate cases with a strong stability from a weak one. Finally, to facilitate reasonable motions of the free gait, that of not repeating the same type of a motion within a walking gait cycle, a penalty value is included.

The evaluation function $f(n)$ is defined as:

$$f(n) = g(n) + h(n) \tag{1}$$

where

$$g(n) = \frac{1}{sm} \tag{2}$$

and

$$h(n) = p + \frac{1}{(\sqrt{x_b^2 + y_b^2} + D_l)} \tag{3}$$

In the above equations, $sm$ stands for the stability margin of the current robot configuration, p is the penalty value for repeating a same motion, $x_b$ and $y_b$ denote the location of the robot body frame with respect to the world coordinate frame, and $D_l$ represents an average values of four feet locations: $1/4 * (\sqrt{x_1^2 + y_1^2} + \sqrt{x_2^2 + y_2^2} + \sqrt{x_3^2 + y_3^2} + \sqrt{x_4^2 + y_4^2})$, where $x_i$ and $y_i$ are the feet positions of leg i specified in Fig. 3(a).

### 3.8 Quarry, Trajectory Generation, and Error Recovery

As we have mentioned at the start, the goal of the robot is to pursue a moving quarry: a stationary quarry is a special case. We assume that the robot can not predict but can monitor the movement of a quarry. The other important assumption for the success of our method is that the summed distance a quarry can move is less than the summed distance the robot can traverse over a given period of time.

Trajectory planning is currently done after a path in the search graph is selected. For each arc connecting two consecutive robot states, a trajectory module is called to plan the route of which a foot and a knee for each leg should follow. The robot also has an errory recovery capability to handle deadlock situations. The search method is designed to eliminate any deadlock situations.[1] However, since the robot pursues a quarry using a piecewise-connected graph search method, deadlock conditions are encountered if the terrain conditions are severely adverse. We use an error recovery module with two separate layers. The first layer of the recovery module tries to move away from a deadlock condition by removing the robot body locally. When the situation can not be resolved locally, the second layer of the error recovery module is invoked to generate motions that may force the robot

[1] In the search method, a deadlock situation is equivalent to having no child nodes for all possible expanding nodes before reaching an intermediate goal node.

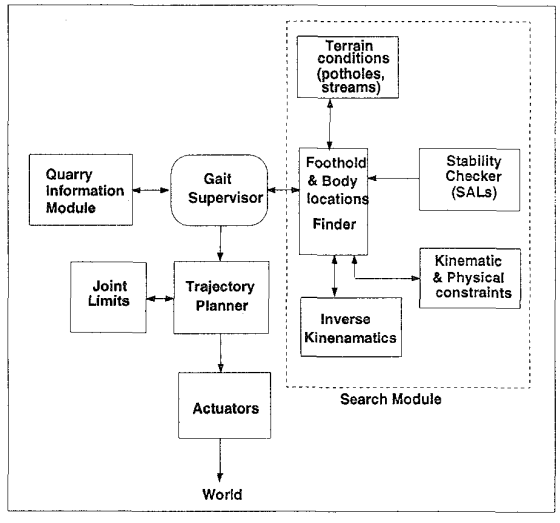



Figure 6: *Flow of control for the gait generation.*

to move away from the quarry temporarily. Fig. 6 shows an overall flow of control for the robot.

## 4 Results and Discussion

We created two different types of objects for the simulated terrain. The first type of objects are streams whose widths are selected manually. The second type of objects are circular potholes with varying sizes. The robot is to cross streams and avoid potholes while pursuing a quarry. We intentionally put multiple streams and potholes in between the initial position of the robot and the location of a quarry, see Figs. 7. For the experiment, we constructed square potholes ranging from 8 cm x 8 cm to 15 cm x 15 cm in the robot world. Three layers of potholes are placed in between the robot and a quarry (a toy ball), as shown in Fig. 8(a).

We now show both simulation and experimental results of the quadruped pursuing a quarry where the quarry is moved only at the end of each search cycle. Fig. 7 shows a simulation result for a case where the target is moved by a distance (21.0 cm, 10.5 cm). Frame (a) is an initial robot configuration. Frame (b) shows the robot legs 1,2 and 3 across the first stream whereas frame (c) shows that leg 2 over the third stream finding a foothold placement next to a pothole, and the leg 1 being ready to cross the third stream. Frame (d), (e), (f), (g), and (h) show intermediate steps toward the goal. Finally, frame (i) depicts the robot reaching the quarry within a specified distance.

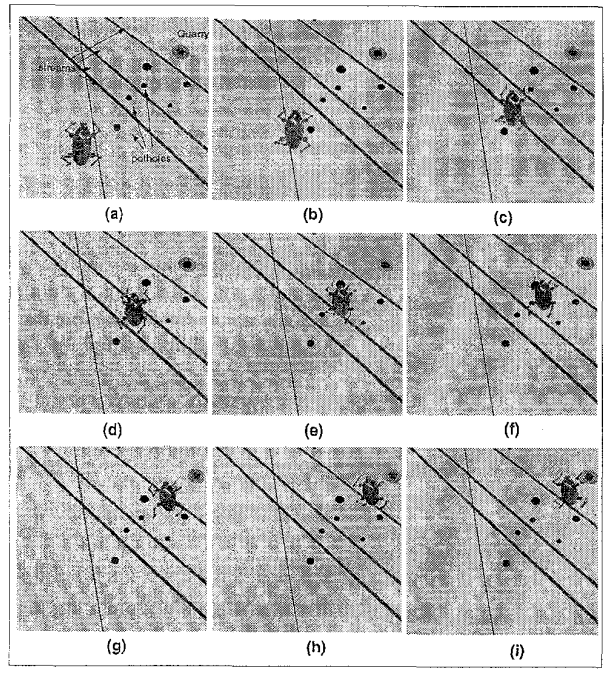



Figure 7: *The sequence of frames shows the robot reaching the moving target position.*

Corresponding experimental results are shown in Fig. 8. In this experiment, the quarry is moved along a line parallel to the robot x axis by a step size of 5 cm at the end of each search cycle. Since we only have two front legs, the search was simpler than the simulation. However, we were able to demonstrate the principle of the proposed method experimentally. Frame (a) shows the robot facing rectangular potholes to reach its quarry. Frame (b) shows the robot moving toward the quarry with its front left foot away from the potholes. Frames (c), (d) and (e) show intermediate stages where the robot is pursuing the ball while avoiding potholes along the way. In frame (f), the robot has crossed over three layers of potholes and is ready to move toward the quarry.

To determine the effectiveness of the search method, we use the *penetrance* measure[4]: *penetrance* $= L/T$ where L is the length of the path found and T is the number of total nodes generated during the search. The optimal search therefore will generate a number 1.0 which indicates that the search was so focused that only a single node was generated at each level of the search graph.

For the results shown in Fig. 7, the robot required 21 search cycles to reach the moving quarry from the starting location. For each search cycle the average

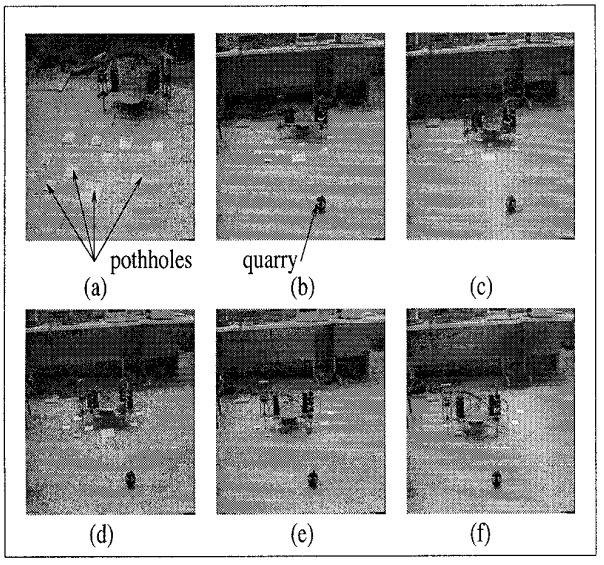



Figure 8: *The sequence of frames shows the robot pursuing a moving ball as its target while avoiding rectangular potholes.*

number of nodes generated is 809.67. This results in the penetrance value 0.093 with an assumption that each level generates at least 15 child nodes. The detail on the performance of the search method in [6] is sketch, and it is difficult to compare the two search methods. However, considering that they have used a size of 1000 nodes for each search cycle in a simple robot world with one stream, and that there are cases where more than 1000 nodes are necessary to reach an intermediate goal, we believe that our omnidirectional gait generation algorithm is at least as efficient as the search method presented in [6]. Furthermore, considering the complexity of the terrain shown in Fig. 7 compared to the one used in [6] and that our search has to incorporate a moving target, in our opinion, our evaluation function is superior for an efficient search of a sequence of walking motions for quadrupeds.

One final issue remains to be addressed. In subsection where we introduced the $A^*$ algorithm, it was stated that $h(n)$ must be less than $h^*(n)$ for all $n$ for the path selected to be optimal. It is, however, difficult to prove this point for the heuristic function we selected. Fortunately, even if the inequality does not hold for some $n$, the search still produces a sub-optimal path. We believe that a free gait for a quadruped machine must contain capabilities proposed by the gait control algorithm in this paper.

## 5 Conclusion

In this paper, we have proposed a graph search method to generate an omnidirectional gait for a quadruped robot. We have shown that the search method is computationally feasible and effective for the free gait generation on a terrain with adverse conditions. It has been shown that by selecting a suitable evaluation function which incorporates the robot configuration, the distance a robot traveled, the stability margin, and a repetition of same type of movement, the robot is able to adapt itself on a varying terrain environment while pursuing a moving quarry. We have demonstrated the effectiveness of the proposed method on a fairly complex terrain which includes multiple streams and varying size of potholes. The generated gait is a periodic gait when the the terrain is flat without any obstacles and a free gait when obstacles exist in the robot world. Validation of the proposed method is presented by simulation and experimental results.

## References

[1] D. Gelprin, "On the Optimality of A*," *Artificial Intelligence*, vol. 8, pp. 69-76, 1977.

[2] S. Hirose, "A study of design and control of a quadruped walking vehicle," In *Int. J. Robotics Research*, vol. 3(2), 1984.

[3] R.B. McGhee and A.A. Frank, "On the Stability Properties of Quadruped Creeping Gaits," In *Mathematical Biosciences.* vol. 3 pp. 331-351, 1968.

[4] N.J. Nilsson, *Principles of Artificial Intelligence*, New York:Springer-Verlag, 1982.

[5] D. Pack and A.C. Kak, "A Simplified Forward Gait Control for a Quadruped Walking Robot," In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, München, Germany, 1994.

[6] P.K. Pal and K. Jayarajan, "Generation of Free Gait-A Graph Search Approach," *IEEE Transactions on Robotics and Automation* No. 3, Vol. 7, June 1991.

[7] S. Song and Y. Chen, "A free gait algorithm for quadrupedal walking machines," *J. Terramechanics*, vol. 28(1), pp. 33-48, 1991.