A Hierarchical Strategy for Learning of Robot Walking Strategies in Natural Terrain Environments

Ayanna M. Howard
Human-Automation Systems (HumAnS) Lab
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA, USA

Lonnie T. Parker
Department of Electrical Engineering
Rochester Institute of Technology
Rochester, NY

Abstract – In this paper, we present a hierarchical methodology that learns new walking gaits autonomously while operating in an uncharted environment, such as on the Mars planetary surface or in the remote Antarctica environment. The focus is to maintain persistent forward locomotion along the body axis, while navigating in natural terrain environments. The hierarchical strategy consists of a finite state machine that models the state of leg orientations coupled with a modified evolutionary algorithm to learn necessary leg movement sequences. Locomotion behavior is assessed by monitoring the robot's progress toward a specified goal location. Details of the methodology are discussed, and experimental results with a six-legged robot are presented.

I. INTRODUCTION AND BACKGROUND

Field mobile robots must traverse long distances on hazardous terrain safely and autonomously using uncertain and imprecise information. Research such as traversability analysis, deliberative path planning with pre-stored terrain maps and embedded reactive behavior [1] have been used to address the problems of navigation in natural terrain, but the process of successfully navigating between two designated points in rough terrain with minimal human interaction is still difficult to achieve in some environments [2]. Legged robots, versus wheeled mobility platforms, offer many advantages due to their ability to traverse a wide variety of terrain, but the control of walking poses special challenges in natural environments. Even *simple* legged-robot platforms have a large degree of coupled interactions and no single walking gait is suitable for all terrain surfaces. Walking surfaces can vary in a number of factors including traction properties, hardness, frictional coefficients, and bearing strength.

To successfully operate within varying terrain environments, an automatic gait adaptation method for field mobile robots is a desirable attribute. In [3], a learning method using a gradient descent algorithm was able to optimize the fastest 4-legged walking gait, as published in the literature. Typical learning time approximated three physical hours, with the process simultaneous learning over three distributed robots. A main limitation was the necessity for initializing the algorithm with a known set of reasonable walking gait parameters. Efforts that use evolutionary approaches, such as proposed by Veloso in [4], have also been used to control walking motions. In this work, walking gaits derived from a random initialization of gait parameters were learned in approximately five hours, with learning spread simultaneously over four robots. Another innovative approach is the utilization of central pattern generators (CPG), which can adapt rhythmic walking patterns in real-time using sensory feedback. In this focus, Fukuoka [5] applied the method for adaptive dynamic walking on irregular terrain, which required a manually intensive design process with regards to the mechanical system and control parameters. On the other hand, Lewis [6] used a staged evolution approach to learn the parameters of the central pattern generator for the control of leg movements of a six-legged walking robot. This approach was designed to accelerate learning such that evaluations could be carried out on real hardware, and required approximately 52 generations to evolve a walking gait. The fitness score was provided by the experimenter, and not by the system, so the actually learning time-period could not be determined. This limits one's ability to compare the approach to other methods.

The limitations with many current methods are the need for either a manually intensive process to design the control system *apriori* or a large number of gait evaluations, which makes it infeasible for learning during run-time. As such, in this paper we discuss a hierarchical methodology that learns new walking gaits autonomously in order to enhance operations in remote planetary environments such as Mars. The focus is to maintain persistent forward locomotion along the body axis, while navigating in natural terrain environments. Section 2 provides detail of the hierarchical approach while Section 3 presents experimental results using a six-legged robot. Conclusions drawn from this work are reported in Section 4.

II. HEIRARCHICAL STRATEGY

Typically, developing gaits for legged robots is a difficult task that requires optimizing parameters in a high-dimensional parameter space. Since our goal is to maintain consistent forward progress in natural terrain environments, we focus on developing a method in which performance can be evaluated in the real-world and a new satisfactory gait can be learned in as few field trials as possible. These objectives are achieved by combining a finite state machine that models the state of leg orientations coupled with a modified evolutionary algorithm to learn necessary leg

movement sequences. Satisfactory forward motion is evaluated by monitoring the robot's progress toward a specified goal location using vision-based terrain assessment methods.

A. Finite State Machine

At the lowest level of the hierarchy, a finite state machine (FSM) is used to model the robotic system by representing achievable leg orientations and the corresponding commands that allow transitions between each.

Given that the legs of the robotic platform can be oriented in a number of different positions, a finite number of states for each leg orientation are first determined using a symbolic construct. The symbolic representation is designed to be generic so that it can be applied to a general class of legged robots, as shown in Figure 1. This representation leads to different leg state combinations for the platform where Left-Leg-Pair (L), Center-Leg-Pair (C), and Right-Leg-Pair (R) act as the variables of the system, each with their own subset of possible values. The *L* and *R* variables are represented by values of Forward (F), Neutral (N), or Back (B) and the *C* variable is represented by Tilt-Right (T_R), Neutral (N), or Tilt-Left (T_L). For example, the state where the left legs are positioned forward, the right legs are placed in the back position and the platform is tilted to the left, is described as FT_LB.

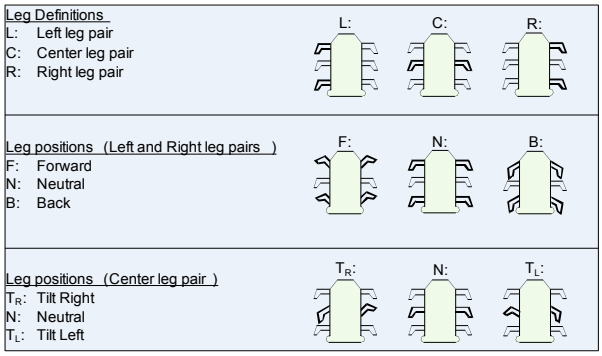

Figure 1: Leg state definitions for a six-legged robot

To describe transitions from one state to another, a function was designed to uniquely characterize both changes in direction and degree of leg movement. The six corresponding commands were modeled as linear functions and were based on servo instructions that control each leg pair, such that:

	Transition Command	Description
1	τ_{LF} for $\theta \geq 0$	Left Legs Forward
2	τ_{LB} for $\theta \leq 0$	Left Legs Backward
3	τ_{RF} for $\theta \leq 0$	Right Legs Forward
4	τ_{RB} for $\theta \geq 0$	Right Legs Backward
5	τ_{CT_R} for $\theta \leq 0$	Tilt Frame Left
6	τ_{CT_L} for $\theta \geq 0$	Tilt Frame Right

Table I. Description of transition commands

where θ is the instruction that provides control of each leg pair and τ is the function that controls the robot's transition from one state to another. In our implementation, θ is a percentage value of servo angle rotation. A negative value suggests counter-clockwise rotation and a positive value suggests clockwise rotation. The output of each command is measured in percentage of angular displacement. For our application, the following equations model each of the six legs of the hexapod robot based on *x* (the percentage of servo position displacement) used in our experimental studies:

$$\xi_{LF}(x) = -0.7891x + 87.5 \quad (1)$$

$$\xi_{LB}(x) = -0.7023x + 85.045 \quad (2)$$

$$\xi_{RF}(x) = -0.6495x + 95.5 \quad (3)$$

$$\xi_{RB}(x) = -0.7764x + 99.163 \quad (4)$$

$$\xi_{CT_R}(x) = -0.0007x + 0.0482 \quad (5)$$

$$\xi_{CT_L}(x) = -0.0007x + 0.0401 \quad (6)$$

where

	Individual Leg Model	Description
1	ξ_{LF}	Left FRONT Leg
2	ξ_{LB}	Left BACK Leg
3	ξ_{RF}	Right FRONT Leg
4	ξ_{RB}	Right BACK Leg
5	ξ_{CR}	Center RIGHT Leg
6	ξ_{CL}	Center LEFT Leg

Note that ζ_{CTR} and ζ_{CTL} were measured in centimeters off the ground, whereas ζ_{LF}, ζ_{LB}, ζ_{RF} and ζ_{RB} were measured in degrees of rotation [0, 180]. As an example, the combination of $\zeta_{LF}(\mathbf{x})$ and $\zeta_{LB}(\mathbf{x})$ leg equations designates that the left legs will jointly rotate between 0 to 50% of their maximum angular displacement for instructions in the range [-50, 0]. These instructions are provided to the servo that commands the left leg pair. Construction of these equations (i.e. associating control signals to leg movement) was implemented using an off-line approach and is discussed further in [7].

B. Learning Algorithm

The FSM allows us to construct a representative model of the robotic system, which also characterizes the commands used to transition between states. At the highest level of the control hierarchy, an evolutionary learning algorithm is coupled with the finite state machine. The primary goal of the learning algorithm is to optimize the sequence of states and/or transitions found in the FSM to allow the robotic platform to persistently progress in a specific direction.

Genetic algorithms [8] are a methodology for searching through the space of solution possibilities using the concept of evolution. By constructing individual chromosomes that

represent possible solutions in the search space, genetic algorithms determine the fitness of an individual based on an objective function. The goal of the learning algorithm is to find a solution that correlates to maximizing the fitness value. The genetic algorithm process consists of the following steps:

1) Create initial population

To ensure the learning algorithm is not dependant on platform specifications, the individual genes of each encoded chromosome are represented by the transition commands from the FSM. This representation places a significant amount of restrictions on the genetic algorithm, which is desirable for limiting the population generation process. In essence, when in certain states, only $N \in [1,6]$ of the possible commands are available (Table I), and when transitioning to another state, a different set of N commands become available. For example, when starting from state *NNN* (Neutral-Neutral-Neutral) all six commands are available for selection. If command 4 is chosen, as shown in Figure 2, state *NNB* would then become the current state and only commands 1, 2, 5, and 6 would be available when selecting the next transition.

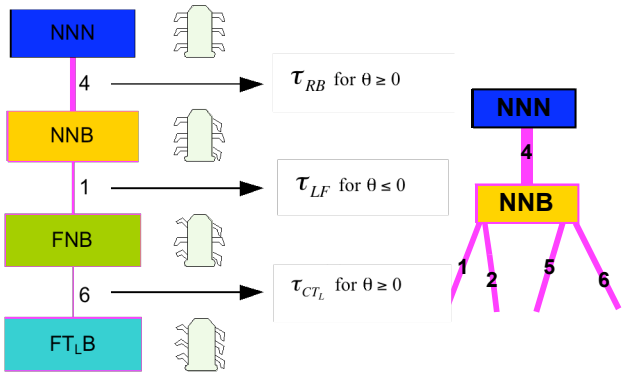

Figure 2: Example of chromosome encoding and transitioning from one state to another

2) Evaluate fitness

To evaluate the fitness value of each chromosome (i.e. measure how well a set of transition commands allow forward progress), we wish to evaluate both robot velocity and position while operating in the field. We assume, as does our platform, that the robot has a sensor suite that includes a color camera capable of image retrieval at a minimum rate of approximately 20-30 Hz. During the navigation cycle, the robot is commanded to locate the largest landmark in the camera field-of-view and record its location before and after executing the commands specified by a chromosome. Forward locomotion along the body axis implies movement in a straight line, while not varying to the left or right significantly. The fitness function therefore combines landmark pixel count as well as movement of the landmark center-of-mass, such that:

$$Fitness = -\left|0.1 * \Delta x_{center-of-mass}\right| + (p_{t+1} - p_t) \quad (7)$$

where $\Delta x_{center\text{-}of\text{-}mass}$ represents the left or right shift in the landmark center-of-mass, p_t represents the pixel count before execution of the chromosome commands, and p_{t+1} represents the pixel count after execution. Significant weight is given to the difference in pixel count since forward velocity has higher priority than left or right movement. Screen shots of example chromosome transitions are shown in Figure 3.

When learning alternative directions of movement (moving backwards, turning left/right), the data from the vision sensor is manipulated differently. However, only slight variations in the fitness equation are required (i.e. greater weight given to $\Delta x_{center\text{-}of\text{-}mass}$ for determining left/right directions). This allows a robust method for assessing omni-directional motion in the field.

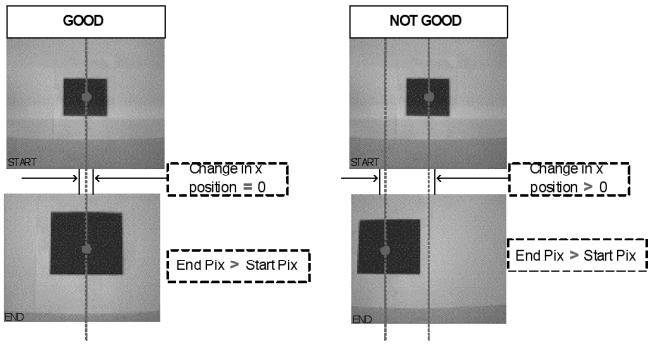

Figure 3: Successful transition (LEFT); Less successful transition (RIGHT)

3) Reproduce

As a variation on the standard five step procedure common in a genetic algorithm (i.e. encoding, fitness evaluation, crossover or combination, mutation, and replacement), a modified version of the evolutionary algorithm is implemented to allow for real-time, fast implementation in the field, which also minimizes undue wear and tear on the hardware. Due to the nature of the FSM and its architecture, two features of the learning algorithm required attention during design: crossover and mutation.

Based on the representation used for encoding the chromosomes, there is no guarantee that randomly combining elements of two of the chromosomes would result in a valid sequence of transitions unless unique attention was paid to the crossover point as well as to each chromosome selected. In addition, the mutation process in genetic algorithms commonly involves independently changing one gene in a given chromosome in order to imitate nature's random influence on evolution. Since transition commands and corresponding states carry a certain degree of dependence on each other in our approach, the random, independent, actions found in the crossover and mutation stage need to be modified. As such, reproduction of

a new chromosome involves randomly selecting a crossover/mutation point and then reproducing the remainder of the chromosome using the same process as in the generation of the initial population. In addition, instead of selecting the fittest member of the population for each crossover operation, the least fit chromosome is selected during the process in order to decrease the number of generations required for in-the-field operation.

4) Create next generation

The primary computational bottleneck in evolutionary algorithms is the number of evaluations required to obtain an optimal solution. For implementation on the physical hardware, three primary factors effect convergence time – 1) the number of genes that comprise a chromosome, which defines the walking pattern, 2) the number of chromosomes that comprise a generation, which equates to the number of times a potential walking gait must be implemented, and 3) the number of generations that must be evaluated before an optimal solution can be found. As such, to minimize the time required for learning, we institute a process that adaptively expands the number of genes found within a population, while keeping the original set of genes constant. In this way, the fitness of a newly created chromosome needs only to be reassessed and remain in the population until the specified number of evolutions has occurred. This duplication and continued evolution process tests the robustness of the first half of the chromosome while independently developing a second half as a supplementary gait. Producing a new generation thus involves the following process:

1) Generate an initial population of α chromosomes with β genes per chromosome.
2) Evolve over γ generations, selecting and mutating the two least fit chromosomes during each evolution and creating a new generation with the most fit chromosomes.
3) Expand the population to α chromosomes with $2*\beta$ genes.
4) Repeat Step 2-4 until convergence.

This process using different values of α, β, and γ was tested multiple times and the results are presented in the next section.

C. Real-time adaptation after learning

After an optimal solution is achieved and a walking strategy that maintains persistent forward progress is learned, the fitness value $Fitness_{max}$ is used to monitor and compare forward progress during navigation. When fitness falls below a certain threshold such that:

$$\left| Fitness_{\max} - \frac{\sum_{k=1}^{N} -\left|0.1 * \Delta x_{center-of-mass}\right| + (p_{t+k} - p_{t+(k-1)})}{N} \right| < \varepsilon$$

the learning algorithm is activated until forward progress is again maintained. In this way, the process used for learning is integrated into real-time operation such that it can be used to re-learn during navigation within varying terrain environments.

III. FIELD TEST STUDIES

For assessment of the learning methodology, we implement the hierarchical strategy on a legged platform that consists of a mechanical body frame, high torque servos, vision sensor and a controller (Figure 4). The skeletal body of the platform is equipped with six legs and has dimensions of [7.5 x 26.5 x 24] cm. Movement of each unit is accomplished using three HI-Tec HS-645MG high torque servos to control movement of three sets of legs. One servo is used to control a set of left legs (front and back) while the other controls a set of right legs (front and back). The third servo is used to control a center set of legs (left and right). When oriented correctly, this center set allows a vertical tilt of the unit from one side to the other. Each servo is orientated in-between its respective set of legs, such that each set is coupled using a push-pull scheme. Thus, as a servo pushes one leg, it simultaneously pulls the other in the same direction. An example of this can be seen with the left set of legs in Figure 5.

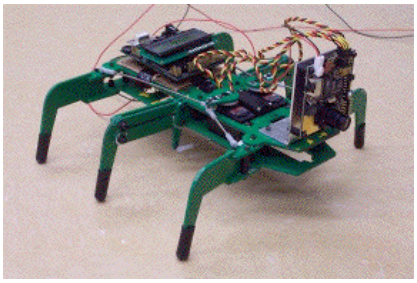

Figure 4: Legged robotic platform

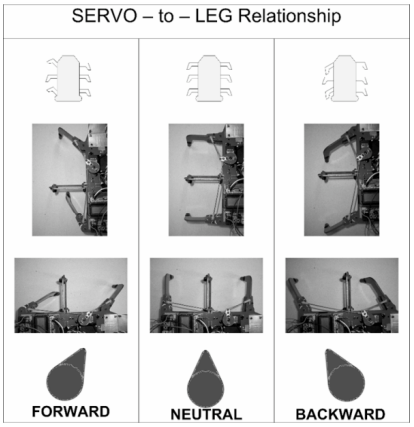

Figure 5: Servo-to-leg relationship

The CMUCam2 vision sensor, with multiple ports for servo control, provides the primary sensor data. Finally, a RidgeSoft Intellibrain™ robotics controller is used to implement the learning algorithm and provide the control signals for commanding robot movement.

Experimental tests were conducted using the robotic platform. Twenty separate trial runs were tested with the robot positioned at varying starting locations. The trials were used to test the effectiveness of our hierarchical strategy in learning a new robot walking strategy, without the traditional time complexity associated with implementation of genetic algorithms on physical hardware. The learned walking strategy was compared with a hand-tuned set of commands as shown in Table II.

The average fitness value associated with the hand-tuned chromosome commands was computed at 68.8 (based on 6 different trial iterations). In Table III, we show a snapshot of an experimental run based on our approach, using chromosomes of varying gene lengths. The fitness values depicted in each row of Table III documents the best fitness values associated with both the initial and evolved population. In our experimental runs, our initial populations consist of 10 chromosomes each. The average number of evaluations until convergence was approximately 400 evaluations, resulting in an implementation time of 17 minutes.

Transition Command	Description
6	Tilt body frame to right
1	Left Leg set forward
5	Tilt body frame to left
3	Right Leg set forward
6	Tilt body frame to right
4	Right Leg set backward
5	Tilt body frame to left
3	Right Leg set forward
2	Left Leg set backward
6	Tilt body frame to right

Table II: Hand-tuned chromosome commands

Initial Population		Expansion (convergence occurred after ~5 evolutions)	
Number of Genes (β)	Best Fitness Value	Number of Genes (β)	Best Fitness Value
3 genes	3.2	6 genes	66.6
	7.1		66.8
4 genes	5.5	8 genes	32.8
	-6.7		65.0
5 genes	-5.0	10 genes	50.1
	18		59.0
6 genes	22	12 genes	-7.3
	16		37.9

Table III. Data from example experimental scenario

Based on our experiments, a number of observations are made. For a 3-gene and 4-gene initial population set, comparable results were achieved with respect to the hand-tuned walking pattern. Since the system was not biased with any prior knowledge on a suitable walking gait, the developing of a pattern based on a methodology that can quickly learn in the field satisfies our original objectives. We do note that our hierarchy approach did not improve over the hand-tuned walking pattern, primarily due to the fact that our convergence criteria was based on using this value as our stored maximum fitness value, as discussed in Section IIIc. Our primary focus though was to maintain persistent forward locomotion along the body axis, while navigating in natural terrain environments, which the system was able to achieve.

In addition, we notice that although expanding from an initial gene size of 4 (versus 3), shows results comparable to the hand-tuned chromosome, the longer chromosome length combined with the lower fitness values show that less accurate forward movement was achieved. This was viewed to be due to the random selection of state transition commands resulting in excessive servo movement. Examining the 5-gene initial set shows that, although the number of genes per chromosome increases, the average fitness values for the best performing chromosomes remains lower than those of lower chromosome length. This further suggests the occurrence of unnecessary leg executions hindering the overall performance of the unit. Additionally, the fitness values of the least fit chromosomes are noticeably lower than those for the previous two gene cases. Finally, the 6-gene set demonstrates the worst performance of lengthened chromosomes. More qualitative observations include undesired movement to the left and right of the target in excess of 30 degrees with respect to the origin-to-target line of sight. This contributed to the largely negative fitness values shown above. These observations have us conclude that starting with a smaller gene-size provides better results with respect to our forward locomotion criteria used for success.

IV. CONCLUSION

The contribution of this work includes a detailed process for providing a legged mobility platform with the ability to learn new walking gaits autonomously while operating in an un-chartered environment. The primary benefit is that the methodology can be applied to a large class of multi-legged robotic platform, provided that the number of states can be defined for a FSM and the optimal chromosome length is established when implementing the modified learning algorithm. Future work will involve testing in more variable terrain environments, as well as exchanging learned walking gait parameters with other robots that are operating simultaneously within the field.

V. ACKNOWLEDGEMENT

The authors would also like to express their gratitude to the National Science Foundation (Grant Nos. EEC-0453295 and EEC-9402723) for their financial support of the SURE program.

VI. REFERENCES

1. E. Tunstel, A. Howard, T. Huntsberger, A. Trebi-Ollenu, J. Dolan, “Applied Soft Computing Strategies for Autonomous Field Robotics,” Fusion of Soft Computing and Hard Computing for Autonomous Robotic Systems, Physica-Verlag, 2003.
2. Committee on Army Unmanned Ground Vehicle Technology, Technology Development for Army Unmanned Ground Vehicles, National Academy of Sciences, 2002.
3. N. Kohl and P. Stone, “Policy gradient reinforcement learning for fast quadrupedal locomotion,” Proc. IEEE Int. Conf. on Robotics and Automation, May 2004.
4. S. Chernova and M. Veloso, “An Evolutionary Approach to Gait Learning For Four-Legged. Robots,” Proceedings of IROS'04, Sendai, Japan, Sept. 2004.
5. Y. Fukuoka, H. Kimura and A.H. Cohen, "Adaptive Dynamic Walking of a Quadruped Robot on Irregular Terrain based on Biological Concepts", Int. Journal of Robotics Research, Vol.22, No.3-4, pp.187-202, 2003.
6. M.A. Lewis, A.H. Fagg, and G.A. Bekey, “Genetic algorithms for gait synthesis in a hexapod robot,” In Y.F. Zheng, editor, Recent trends in mobile robots. World Scientific, 1993.
7. A. M. Howard, L. Parker, B. Smith, “A Learning Approach to Enable Locomotion of Multiple Robotic Agents Operating in Natural Terrain Environments”, Int. Journal of Intelligent Automation and Soft Computing, to appear 2007.
8. D.E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, 1989.