Proceedings of 2016 International Conference on Advanced Robotics and Intelligent Systems
Taipei Nangang Exhibition Center, Taipei, Taiwan, 31 August~2 September, 2016

Control of a 2-D Bounding Passive Quadruped Model with Poincaré Map Approximation and Model Predictive Control

Austin Shih-Ping Wang, William Wei-Lun Chen, and Pei-Chun Lin
Department of Mechanical Engineering
National Taiwan University

***Abstract*—In this paper, we simplify the dynamics of a bounding quadrupedal robot by using a planar and conservative model which has a rigid body and two massless virtual spring legs to separately simulate the effects of the front and hind legs. The quantitative formulation of the model is derived by using the Lagrangian method. We proceed to search for stable operation points in state space, and the local system behaviors of which are then approximated by a trained neural network model. Next, a model predictive controller is utilized to stabilize the model. In simulation results, the cont vroller succeeded in balancing the model in a dynamic bounding gait without any form of energy input. This research may serve as a guideline for real quadrupedal robots with actuators to create energy conservative gaits.**

I. Introduction

The majority of quadrupedal robots today rely heavily on the leg motors to execute stable bounding or galloping motion. The Wildcat from Boston Dynamics [1] and the MIT cheetah [2] all utilize multiple leg motors to perform these stable gaits. However, we are curious if similar gaits could be actuated in a completely passive manner.

The Spring-Loaded Inverted Pendulum (SLIP) has been a widely used planar model of the dynamics of the running gait of animals, as in [3] and [4]. A step in the running gait is modeled by compression and extension of the spring, thus theoretically requiring no external energy input. In reality, animals use similar dynamics to conserve the energy needed to run at high speeds. Various research has also been conducted on leg designs improvised upon SLIP, such as [5] and [6].

To analyze the fundamental dynamics of quadrupedal running, we propose a simplified 2-dimensional model without any actuation, with two SLIP-like structures as front and back legs to potentially achieve passive running. Since the bounding gait is symmetrical with respect to the plane of forward motion, a 2-dimensional model will be sufficient to describe the ideal dynamics of this gait, which is the why bounding is chosen as the focus of this research. If a passive bounding gait exists on the model, then ideally a similar robot would be able to run without the need of constant energy supply into the system. The motors would then only have to make up for errors, friction, and other means of energy loss in the actual case, which will require extremely little energy to do so when compared to directly actuating the robot into the desired gait. The purpose of this study is to excite a such a gait on the passive model, with the hope of inspiring energy conservative gaits on real robots.

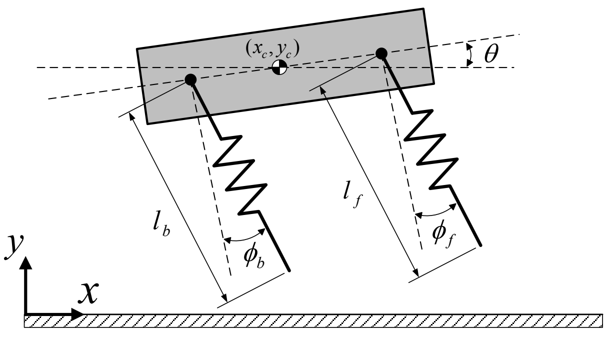


Fig. 1: The mathematical model of the quadrupedal robot. (x_c, y_c): position of COM of body, θ: body pitch, ϕ_f: angle of front leg relative to body, ϕ_b: angle of back leg relative to body, l_f: length of front leg, l_b: length of back leg

II. The mathematical model

A typical SLIP model involves a point mass with a spring attached. A double SLIP model with a point mass and two springs has also been proposed to model the dynamics of a biped, such as in [6]. In order to simulate the gaits of a quadruped (especially the bounding gait), our model consists of a rigid body and two massless springs attached at opposite ends, as shown in Fig 1.[1] All analysis will be done on the plane of forward motion of this model.

A. System parameters and states

In order to further simplify the model, we will only consider the case where the model is symmetrical, e.g. the body has symmetrical mass distribution and both legs have identical properties. Thus the body can be completely defined by its mass m_{body}, length l_{body}, and rotational inertia I_{body}. Since we made the assumption that the legs are massless, the mass

[1]Although several studies emphasize the importance of the waist of quadruped in most bounding gait, we will leave this degree of freedom out of our model since torque is required to maintain a configuration if we add a joint at the waist, thus contradicts with our goal of creating completely passive dynamics.

and inertia of the leg will be ignored, hence $m_{leg} = 0$ and $I_{leg} = 0$. We then define l_0 as the uncompressed length of legs and k as the stiffness of leg springs.

The configuration of the system can be completely defined with the following 7 variables: x_c, y_c, θ, ϕ_f, ϕ_b, l_f, l_b. (as defined in Fig.1) However, the two leg springs are uncompressed when in midair, while the length of compressed springs can be completely determined from the relative positions between the body and ground when in stance. Thus, l_f, l_b are dependent on the other system variables and thus are not included in our definition of system states. Also, since $m_l = 0$ and $I_l = 0$, the angular acceleration of the leg will have no meaning. As a result, we define our system states as:

$$s = \begin{bmatrix} x_c & y_c & \theta & \phi_f & \phi_b & \dot{x_c} & \dot{y_c} & \dot{\theta} \end{bmatrix}^\top \tag{1}$$

B. *Planar dynamics of the model*

As with similar models derived in [7] and [8], the system we defined exhibits both continuous and discrete dynamics behaviors, fulfilling the definition of a hybrid system as defined in [9] and [10]. As in [8], the method of analyzing such systems involves partitioning the state space into discrete "phases", each with it own continuous dynamics. This system has 4 distinct phases, shown in Fig.2. For each of the phases, a set of equations governing the system dynamics in the form

$$\dot{s} = f(s) \tag{2}$$

can be derived using Lagrangian mechanics. A partitioning of the state space is done by calculating the position of the tip of the two legs as if the legs were at uncompressed length l_0, thus:

$$y_b = y_c - \frac{l_{body}}{2} \sin\theta - l_0 \cos(\theta + \phi_b) \tag{3}$$

$$y_f = y_c + \frac{l_{body}}{2} \sin\theta - l_0 \cos(\theta + \phi_f) \tag{4}$$

The dynamical equations of the system can then be expressed in the form

$$\dot{s} = f_{sys}(s) \tag{5}$$

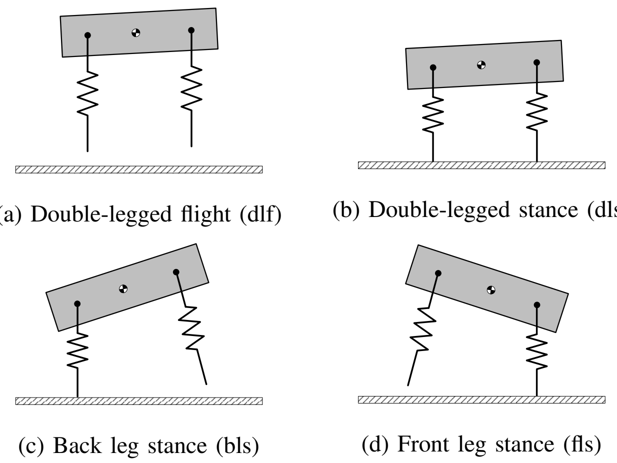
where

$$f_{sys} = \begin{cases} f_{dlf}, & y_b > 0 \text{ and } y_f > 0 \\ f_{bls}, & y_b < 0 \text{ and } y_f > 0 \\ f_{dls}, & y_b < 0 \text{ and } y_f < 0 \\ f_{fls}, & y_b > 0 \text{ and } y_f < 0 \end{cases} \tag{6}$$

Note that from our definition of the model, it is clear that the system does not dissipate energy of any form, since there is no friction and the springs are ideal. The Hamiltonian of the system is

$$H = m_b g y_c + \frac{1}{2} m_b (\dot{x_c}^2 + \dot{y_c}^2) + \frac{1}{2} I_b (\dot{\theta}^2) \tag{7}$$

, which is time independent.


(a) Double-legged flight (dlf) (b) Double-legged stance (dls)

(c) Back leg stance (bls) (d) Front leg stance (fls)

Fig. 2: The 4 phases of the model

C. *Transformation to a discrete system via the Poincaré map*

The "stable" gaits of legged locomotion differ from the usual definitions of system stability in the fact that the system states do not converge to a constant value. Instead, the trajectory of the system in the phase portrait converges to a cyclic path, which is the concept of limit cycle stability [11]. We analyze such periodic systems by defining the Poincaré section: a subspace of interest inside the state space. The system is then observed only when its trajectory passes the Poincaré section. If a stable gait is reached, then the discrete observation of the states will indeed converge to a constant value.

By taking the apex of the flight phase ($\dot{y} = 0$) as the Poincaré section, the system states under discrete observations should look like:

$$s = \begin{bmatrix} x_c & y_c & \theta & \phi_f & \phi_b & \dot{x_c} & 0 & \dot{\theta} \end{bmatrix}^\top \tag{8}$$

Since the two legs are massless, ϕ_f and ϕ_b can actually be arbitrarily assigned by our controller. Also the horizontal position does not affect the dynamics in any way, so x_c can be discarded from our list of interested states.

According to (7), the Hamiltonian of the system remains constant throughout a single run. Since the Hamiltonian only depends on state variables y_c, $\dot{x_c}$, and $\dot{\theta}$, only two degrees of freedom exists between the three variables, e.g. only two of the three variables are required to fully determine the state of the system.

Regarding the above considerations, we end up with the following independent states of interest on the Poincaré section:

$$s_p = \begin{bmatrix} y_c & \theta & \dot{x_c} \end{bmatrix}^\top \tag{9}$$

Our hybrid nonlinear system defined in (5) can then be transformed to a discrete nonlinear system:

$$s_p[n+1] = P(s_p[n], \phi_f[n], \phi_b[n]) \tag{10}$$

where P is the Poincaré mapping which "maps" the current states $s_p[n]$ and inputs $\phi_f[n], \phi_b[n]$ to the system states at the next cycle $s_p[n+1]$.

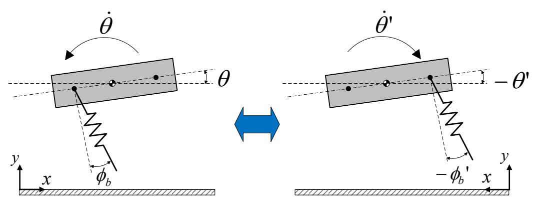


Fig. 3: The symmetrical property of the bounding gait

Due to the symmetrical nature of our model and our desired bounding gait as shown in Fig. 3, we can take advantage of the fact that the dynamics stance phase of the front and back leg are also symmetrical, and further simplify our system by mirroring the system every time the apex of the flight phase is reached using the following transition mapping:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{11}$$

Let $P_{half}(s_p, \phi_b) = TP(s_p, 0, \phi_b)$, our system becomes

$$s_p[n+1] = P_{half}(s_p[n], \phi_b[n]) \tag{12}$$

and thus the half-step Poincaré map P_{half} alone can fully define the dynamics of the system under a bounding gait.

After we obtained P_{half}, a full step Poincaré map P_{full} can be constructed from P_{half} by:

$$P_{full}(s_p, \phi) = P_{half}(P_{half}(s_p, \phi_{b1}), \phi_{b2}) \tag{13}$$

where

$$\phi = \begin{bmatrix} \phi_{b1} \\ \phi_{b2} \end{bmatrix} \tag{14}$$

If we consider a full step to be a single discrete step of our system, the system can then be modeled as

$$s_p[n+1] = P_{full}(s_p[n], \phi) \tag{15}$$

D. Control strategy

Although the idea of controlling the model contradicts with the fact the model is passive, we can exploit the fact that the two legs are massless, and thus would require no energy to rotate in midair. The method of controlling the model in this paper involves ”snapping” the legs to an optimal angle at the apex of flight, with the objective of stabilizing the model at a bounding gait.

III. Initial condition search

To simulate the passive model, we release it from various initial states and an assigned leg angle ϕ_b.[2]

[2]Previous research utilizing similar initial condition searches can be found in [12]

A. An efficient downhill search algorithm

The initial condition space is a five dimensional space, including coordinates y_{c0}, θ_0, $\dot{x}_{c0}$, $\dot{\theta}_0$, and ϕ_b. (For the case of constant Hamiltonian, it is a four-dimensional space with coordinates y_{c0}, θ_0, $\dot{x}_{c0}$, and ϕ_b.)

If the model were to execute a stable bounding gait, the states should return to approximately the same position every step. To efficiently search through initial conditions that satisfy this property, we propose the an algorithm involving the following steps: First, we discretize our state space and input. Then, a cost function is conceived as a measure of how well the result from an initial condition satisfies our assumptions. Our objective is to find all initial condition coordinates with a resulting cost function below a certain threshold, which we will refer to as ”valid” coordinates. In order to do that, we assign an initial starting coordinate, then proceed to do a downhill search until the cost drops below the threshold. At this point, all valid coordinates are searched and recorded. For further details, an explicit description of this algorithm is included in the appendix.

The success of this algorithm highly depends on the assumption that the behavior of physical model within a single cycle is continuous and non-chaotic, and thus similar initial conditions will lead to similar states in the next cycle. The above assumption is observed to be true for our model through some experimentation. The advantage of using this method over regular iterative search is incredibly significant in high dimensions, since the program will always find a one-dimensional path to the desired region provided that a suitable starting coordinate is chosen.[3]

The dataset used in the following sections was generated by this search method, which in that case had a search space of 9,228,168 discrete coordinates in the search space. By using the algorithm, only 551,162 were evaluated in order to obtain results of all coordinates within the entire desired region, which is less than 6% of the search space.

B. Gathering data from the desired region

In order to obtain an approximate region of where our desired region is located, we first assign the following cost function with the objective of minimizing the difference between the initial state vector $s_p[0]$ and the states after a limit cycle $s_p[1]$:

$$s_{diff} = s_p[1] - s_p[0] \tag{16}$$

$$C_1 = s_{diff}^{\top} N s_{diff} \tag{17}$$

$$N = \begin{bmatrix} \sigma_y & 0 & 0 \\ 0 & \sigma_\theta & 0 \\ 0 & 0 & \sigma_{\dot{x}} \end{bmatrix} \tag{18}$$

where σ_y, σ_θ, $\sigma_{\dot{x}}$ are normalization coefficients taken from the variance of data in our desired region.

[3]If there is difficulty assigning a starting coordinate that guarantees a path to the desired region, or if several separate ”basins” with valid cost functions exist, there is also the option to choose multiple starting coordinates by adding them all to set C.

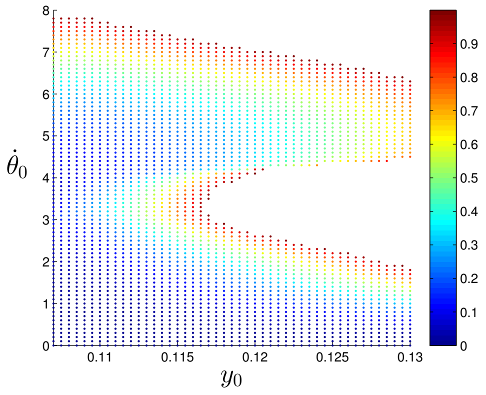


Fig. 4: Results from a iterative initial condition search with varying y_0 and $\dot{\theta}_0$. The low-cost region with lower values of $\dot{\theta}_0$ corresponds to the model bouncing in place, while the low-cost region with higher values of $\dot{\theta}_0$ corresponds to a bounding step where s_{diff} is small, which is our desired reaction.

Due to the fact that we are not optimizing the model in a single run, but observing dynamics for all cases, different system energies have to be considered. Thus in this analysis, we discard the constant Hamiltonian constraint and simulate the model under different energies. As in Fig.4, we do a simple analysis by adding the constraints $\theta_0 = 0$ and $\dot{x}_{c0} = 0$.

After we have located the general region where the states return approximately to the same values, we can now specify a good starting coordinate for the downhill search algorithm. A suitable initial system energy is chosen, and the initial condition space is once more simplified by 1 dimension.

The objective now is to find all initial conditions close to our operating point, then use the data to approximate the Poincaré map in this region. In order to do this, we define a new cost function C_2:

$$C_2 = \begin{cases} 0, & \|x_{diff}\| < x_{thres} \\ C_1, & \text{otherwise} \end{cases} \tag{19}$$

where C_1 is the cost function we defined earlier and x_{thres} is a threshold where only x_{diff} below that value will be accepted. Note that x_{thres} is different from the cost threshold in the description of the algorithm, which is a threshold for C'. If we set the cost threshold to 0, we will ideally obtain the results all initial condition coordinates with $x_{diff} < x_{thres}$, while coordinates outside this region will still converge towards this region.

IV. Function approximation of the Poincaré map

The search results from the previous section yields sets of input-output relations of the Poincaré map P_{half} in (12). An approximation model of P_{half} in the vicinity of our operating point is then conceived, which will be used for optimization in the next section. Since $s_p[n+1]$ completely depends on $s_p[n]$ and ϕ_b, we can define the input into our approximation model as

$$\begin{bmatrix} y_{c0} & \theta_0 & \dot{x}_{c0} & \phi_b \end{bmatrix}^\top \tag{20}$$

and the target as

$$\begin{bmatrix} y_{c1} & \theta_1 & \dot{x}_{c1} \end{bmatrix}^\top \tag{21}$$

A feedforward neural network model a single hidden layer and a $\tan h$ activation function is then trained using the data to approximate the Poincaré map, with the objective of minimizing the mean squared error (MSE) between the output produced by the model and the target.

With all inputs and outputs normalized to zero mean and unit variance, the final model has a MSE of 1.75%. The trained model can then be used in model predictive control similar to the methods used in [14] and [15], as presented in the next section.

The full-step Poincaré map P_{full} can then be constructed from the approximated model of P_{half} by (13).

V. Model predictive control

Model predictive control (MPC) is a control method which involves using a dynamical model of the system to perform optimal control across a finite-time horizon [16]. An MPC on our system with a prediction horizon of N can be formulated as the follows:

$$\begin{aligned} &\arg\min_{\phi_b[\cdot]} \sum_{i=0}^{N} \alpha J(s_p[i+1]) \\ &s.t.\ s_p[n+1] = P_{full}(s_p[n], \phi) \end{aligned} \tag{22}$$

where J is the cost function, which is a function of the state vector. α is a weighting coefficient controlling the relative importance of the near future and far future states.

At every step, the controller solves (22) and applies the first input $\phi[0]$ to the system. Predicting a few steps into the future has the advantage of being able to find a policy that converges in the long run, instead of forcibly trying to get close to the desired states by the next step.

A. The fixed point

The objective now is to create an optimal policy with MPC, under which the system will converge to a fixed point s_p^*. However, in contrast to conventional optimal control, the control goal s_p^* is not initially known, since our initial goal is to stabilize the model at a stable gait with a desired constant horizontal speed which we will denote as $\dot{x}_d$. The fixed point in state space s_p^* that allows this gait can be defined as:

$$s_p^* = \begin{bmatrix} y_c^* \\ \theta^* \\ \dot{x}_d \end{bmatrix} \tag{23}$$

$$\exists y_c^*, \theta^*, \phi^* \mid s_p^* = P_{full}(s_p^*, \phi^*) \tag{24}$$

The following problem is first solved to find y^* and θ^*, and therefore s_p^*:

$$s_{diff} = s_p[1] - s_p[0] \tag{25}$$

$$y_c^*, \theta^*, \phi^* = \arg\min_{y,\theta,\phi} s_{diff}^\top N s_{diff}$$
$$s.t.\ s_p[1] = P_{full}(s_p[0], \phi) \tag{26}$$

where N is the diagonal normalization matrix defined in (18).

Solution using the DIRECT optimization algorithm [17] yields a point in the state space which is a plausible fixed point under the control policy.

B. Finding the optimal solution

After our desired state s_p^* is found, we can define the error as:

$$e = s_p - s_p^* \tag{27}$$

The cost function

$$J(s_p) = e^\top N e \tag{28}$$

can then be used in (22), which is also solved using the DIRECT optimization algorithm [17], to perform an optimal control in hope of letting the system converge to a stable gait.

VI. Results

The optimization in (26) yields many points in the state space that has a solution for ϕ which return the model to almost exactly the same point after one limit cycle, although the existence of a stable fixed point on this model has not yet been found.

The control methods we proposed yielded bounding gaits for a few steps which were impossible to perform before. Each time the model reaches the apex of its flight phase, the controller will make obvious leg positioning in order to return to the desired state. After tuning, simulation results showed that the robot can fixate itself in a bounding gait for an indefinite amount of time, with an average horizontal velocity close to the $\dot{x}_d$ we assigned, as shown in Fig. 5.

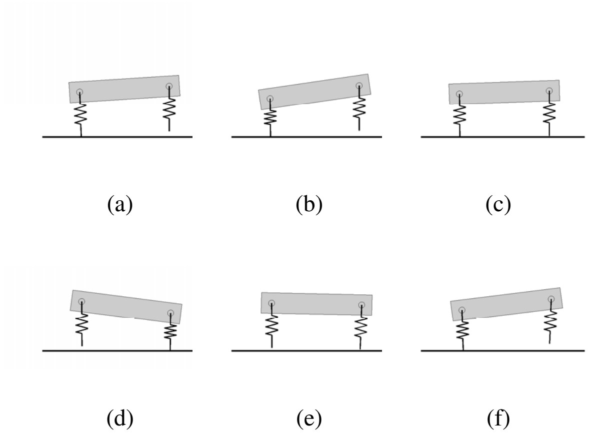


Fig. 5: The bounding gait stabilized with MPC

Due to the fact that our model of Poincaré map is an approximation, the prediction errors will be amplified if the horizon of MPC extends too far into the future. The dimensions of the robot also plays an important role in the stability of the system. Having too short of a body length will oftentimes cause the model to overshoot the desired state and spinning out of control.

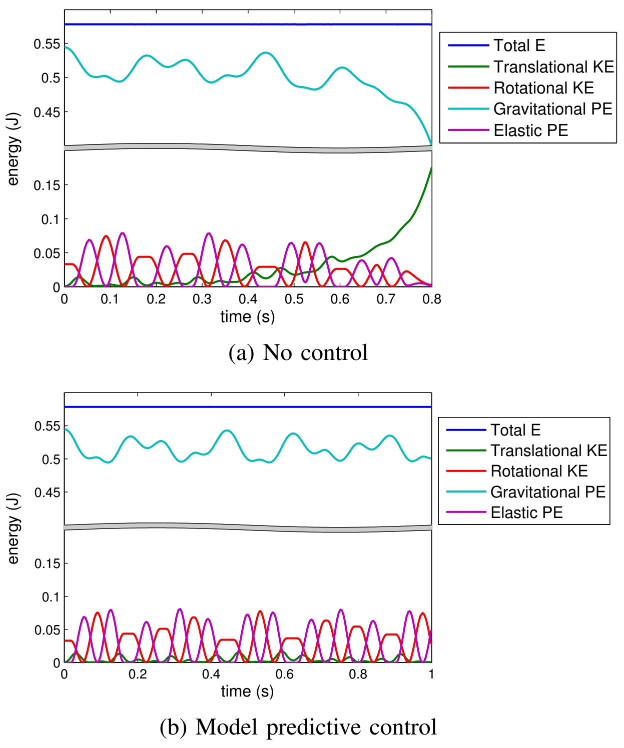


Fig. 6: Energy distribution throughout the passive bounding. (KE: Kinetic energy, PE: Potential energy)

The distribution of mechanical energy, as shown in Fig. 6, gives us insight into the stability of the resulting gaits. In a stable gait, the energy distribution should be a periodic function. Without control, the behavior of the model deviates quickly from the first period, crashing to the ground within 5 periods. Though the model remains in a bounding gait after implementing our approximated model and MPC, the behavior of the model still did not converge to a stable limit cycle. Energy distributions throughout every period of the gait is not identical, as observed in Fig. 6b.

VII. Conclusion and Future Work

Controlling hybrid systems are quite challenging due to its complex system dynamics. The passive controller of our design tackles an even harder problem as it does not have the power to directly actuate the system, and have no control over the total energy of the system. Using limit cycle analysis, we transformed the continuous hybrid nonlinear system into a discrete nonlinear system, enabling prediction of the next step via the Poincaré map, which we approximated with a neural network. The prediction model as then used in a MPC controller to stabilize the model in a bounding gait. With careful implementation of the controller, the model was able to balance itself in a bounding gait and move forward consistently at a desired horizontal velocity.

Though the resulting gait fulfills the objective of moving forward without crashing, it does not converge to a steady limit cycle. Improvement on the approximation model is needed in order to precisely predict the state vector after a limit cycle. Our future work will also include designing more sophisticated MPC controllers with cost functions specific to our objective, and transition between different desired states (acceleration and deceleration of the running gait).

A completely passive gait might not be feasible in actual hardware, but completely actuated limbs may require much less energy if they execute motions that are natural to the system dynamics. Limbs designed for passive running also has the advantage of smaller number of actuators and lower complexity of mechanical structure, and thus lower cost. We hope that this research of completely passive dynamics will be a valuable guideline for controlling similar quadrupedal robots, inspiring low-cost, low-energy robots able of traversing uneven terrain.

APPENDIX

The downhill search algorithm proposed in section III helps us to efficiently find initial conditions that result in desired responses. Note that an initial condition coordinate x includes initial states and initial leg angles.

Algorithm 1 The search algorithm

```
assign a cost function threshold $thres$
assign a starting coordinate $x_0$ from experience
$C \leftarrow \{x_0\}$
while $C$ is not empty do
    $x\_center \leftarrow$ first element in $C$
    remove first element in $C$
    $visit(x\_center, C, thres)$
end while
```

Algorithm 2 $visit(x\_center, C, thres)$

```
if $x\_center$ has not been simulated before then
    simulate $x$ for 1 limit cycle
end if
if $x\_center.cost < threshold$ then
    record simulation result of $x\_center$
end if
for $x$ = each neighbor of $x\_center$ do
    if $x$ has not been simulated before then
        simulate $x$ for 1 limit cycle
    end if
    if $s.cost < thres$ and $x$ has not been visited before then
        add $x$ to $C$
    end if
end for
if none of the neighbors of $x\_center$ are valid then
    add the one with lowest cost to $C$
end if
```

ACKNOWLEDGMENT

The authors would like to thank Chung-En Tsai and Yung-Hsiu Chen for their work on the hardware model and the constant support they gave us. The authors would also like to thank Ruei-Fa Tseng for his guidance on the design of our embedded system, and for his helpful advice throughout this entire project.

This work is supported by the Ministry of Science and Technology (MoST), Taiwan, under the contract MOST 103-2221-E-002-091-MY3.

REFERENCES

[1] Boston Dynamics, “Wildcat,” online:http://www.extremetech.com/extreme/168008-meet-darpas-wildcat-a-free-running-quadruped-robot-that-will-soon-reach-50-mph-over-rough-terrain

[2] Sangok Seok, A. Wang, Meng Yee Chuah, D. Otten, J. Lang, and Sangbae Kim, “Design principles for highly efficient quadrupeds and implementation on the MIT Cheetah robot,” in IEEE IROS, pages 33073312, May 2013.

[3] Alexander, R. M., “Three Uses for Springs in Legged Locomotion”, in The Int. J. of Robotics Research, Vol. 9, No. 2, 1990.

[4] Blickhan R., “The Spring-Mass Model for Running and Hopping”, in J. of Biomechanics, Vol. 22, pp. 1217 1227, 1989.

[5] C. K. Huang, C. L. Chen, C. J. Hu, and P. C Lin*, ”Model-based Bounding on a Quadruped Robot”, in Proc. IEEE International Conference on Robotics and Automation (ICRA), May 2016, Stockholm, Sweden, pp3576-3581

[6] Ketelaar, J.G.; Visser, L.C.; Stramigioli, S.; Carloni, R. “Controller design for a bipedal walking robot using variable stiffness actuators,” in 2013 IEEE International Conference on Robotics and Automation (ICRA)

[7] P., Murali Krishna and Prasanth Kumar R.. “Energetics of constant height level bounding in quadruped robots.” Robotica 34 (2016): 403-422.

[8] Ioannis Poulakakis, “On the Passive Dynamics of Quadrupedal Running,” in the International Journal of Robotics Research July 2006 vol. 25 no. 7 669-687

[9] Arjan van der Schaft and Hans Schumacher, “An Introduction to Hybrid Dynamical Systems”

[10] ‘’ Russell Tedrake. 6.832 Underactuated Robotics, Spring 2009. (Massachusetts Institute of Technology: MIT OpenCourseWare), http://ocw.mit.edu. License: Creative Commons BY-NC-SA

[11] Ian A. Hiskens, “Stability of Limit Cycles in Hybrid Systems,” in Proceedings of the 34th Hawaii International Conference on System Sciences - 2001

[12] K. J. Huang, C. K. Huang, and P. C. Lin*, A Simple Model for Running Behavior with Rolling Contact and its Role as a Template for Dynamic Locomotion on a Hexapod Robot, Bioinspiration and Biomimetics

[13] Hae-Won Park, Meng Yee Chuah, and Sangbae Kim, “Quadruped Bounding Control with Variable Duty Cycle via Vertical Impulse Scaling,” in IEEE/RSJ IROS, pages 32453252, Sept 2014.

[14] A. Draeger, S. Engell, and H. Ranke, “Model Predictive Control Using Neural Networks”

[15] S. Piché, J. Keeler, G. Martin, G. Boe, D. Johnson, and M. Gerules, “Neural Network Based Model Predictive Control”

[16] Manfred Morari and Jay H. Lee, “Model Predictive Control: Past, Present and Future,” in Computers & Chemical Engineering, Volume 23, Issues 45, 1 May 1999, Pages 667682

[17] Daniel E. Finkel, “DIRECT Optimization Algorithm User Guide”