Proceedings of the ASME 2010 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2010
August 15-18, 2010, Montreal, Quebec, Canada

DETC2010-28857

FUZZY CONTROL OF VERTICAL JUMPING WITH A PLANAR BIPED

Matthew Hester
Moog Inc
Salt Lake City, Utah 84119

Patrick M. Wensing
Electrical and Computer Engineering
The Ohio State University
Columbus, OH 43210
Email: wensing.2@osu.edu

James P. Schmiedeler
Aerospace and Mechanical Engineering
University of Notre Dame
Notre Dame, Indiana 46556
Email: schmiedeler.4@nd.edu

David E. Orin
Electrical and Computer Engineering
The Ohio State University
Columbus, OH 43210
Email: orin.1@osu.edu

ABSTRACT

This paper develops a control strategy to produce vertical jumps in a planar biped robot as a preliminary investigation into dynamic maneuvers. The control strategy was broken into two functional levels to separately solve the problems of coordination and execution of the jump maneuver. A high-level fuzzy controller addresses the complexities that arise from the system's hybrid nonlinear dynamics and series-elastic actuators embedded in the articulated legs. A novel fuzzy training scheme is used because the system is too complex for traditional training methods. A low-level controller is based on a state machine that sequences the legs through the phases of a jump. The modular nature of the control strategy allows quick adaptation to other dynamic maneuvers. Validity is demonstrated through dynamic simulation and testing with the experimental biped KURMET which result in stable successive jumps over a range of heights.

1 INTRODUCTION

Although much research has attempted to harness the power and efficiency of human locomotion in mechanical systems, a solution to this problem is still in its infancy. Human legs and the associated neurocontrol system provide extreme agility in a variety of terrain conditions that is unparalleled by any physical system to date. Much of this agility can be attributed to biological systems' ability to execute truly dynamic movements. If mechanical bipeds are to reach this level of performance, the feasibility of basic dynamic movements such as the run or the jump must be investigated more thoroughly. Control of a jump has been studied on relatively few bipedal systems [1, 2, 3]. Some of the most promising results have been demonstrated by Mowgli, a biped designed specifically to jump. The system has demonstrated controlled takeoff and landing with jump heights of 0.5 m through open-loop motor commands to its pneumatic actuators. Still, the system's large foot size does not reflect the structure of animals such as humans, and prevents performance of a rich set of movements.

Raibert's early investigations into general dynamic movement sought to exploit the natural dynamics of legged systems to improve stability and performance [4]. Three-dimensional hopping for a monopod was produced using a relatively simple control strategy based on the SLIP (spring loaded inverted pendulum) model that exploited the structure of the robot. While this strategy was extended to a bipedal system to produce a run, the system still lacked realism compared to legged animals, which rely on articulated legs.

More recently, the SLIP model has contributed to the development of control strategies for dynamic running on an articulated monopod [5]. The monopod relies on a hybrid zero dynamics (HZD) controller. A biped robot, RABBIT, used the HZD control approach to perform six running steps [6]. While demonstrating impressive results, the approach requires intensive mathematical analysis and modeling and is only applicable for planar systems. The ability of animals to perform complex, unconstrained maneuvers suggests that less analytically intensive control strategies may be viable.

Intelligent control strategies have emerged as an alternative to analytically intensive methods. Antsaklis and Passino [7] describe intelligent control systems as a necessary development to address the increasingly complex nature of dynamic systems. These methods are well suited for the control of biologically realistic bipeds due to complexities introduced by the articulated legs, highly-nonlinear hybrid dynamics, and underactuated nature, most notably during flight. Additionally, intelligent control provides advantages to control systems that employ compliance in their actuation schemes. While both series-elastic and biarticular actuation schemes have been shown to enhance performance and efficiency for legged systems (e.g. [8] and [9] respectively), the associated control problem increases drastically in complexity. Fuzzy control, an intelligent method, offers further advantages through its ability to integrate heuristic knowledge to develop a controller without a complex system model. Previous work has shown positive results with fuzzy control on legged systems [10, 11].

The main objective of this work is to develop a layered control system with high-level fuzzy control and a low-level state machine to perform stable, repetitive jumps. The control strategy will be developed for and tested on an experimental biped, KURMET, shown in Fig. 1. This work shows the usefulness of a biologically inspired motor primitive control approach coupled with a fuzzy controller to control complex biologically inspired systems. A novel approach to the construction of the high-level fuzzy controller is detailed to promote extension of this strategy to other movements and systems.

The remainder of the paper is organized as follows. Section 2 describes the biped hardware and the model of it developed in this work, and Section 3 introduces the layered control approach. Section 4 describes the novel approach taken to train the fuzzy controller. Section 5 presents both simulation and experimental results.

2 BIPED HARDWARE AND MODEL

The experimental biped, KURMET, shown in Fig. 1, is a 5-link planar robot with articulated legs. The hip and knee joints are revolute, actuated in parallel by compact unidirectional series-elastic actuators (USEA) that include brushless DC motors [12]. This design allows direct-drive actuation in the direction of leg return and incorporates a torsion spring between the motor and link in the direction of leg thrust. The USEAs, physically located within the torso to reduce leg mass, are connected to their respective links via steel cables and pulleys. To restrict motion to the sagittal plane, the biped is rigidly mounted on a carbon-fiber boom collinear with the hip axes. The boom provides pitch and yaw degrees-of-freedom at its wall mount and is horizontal with full leg extension and the feet on the ground.

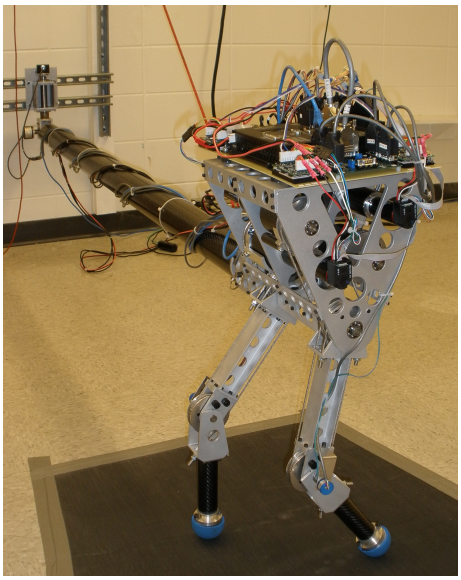

FIGURE 1. Photograph of the experimental biped, KURMET, attached to its boom.

The biped is modeled using five rigid links: the torso, two thighs, and two shanks, seen in Fig. 2. The forward velocity of the system v is measured in the sagittal plane at the center of the torso, while the height h is the distance from the ground to the right hip. The mass and geometric properties of each link, summarized in Table 1, were estimated from solid models of the experimental hardware. Fig. 3 shows the coordinate frames of the full dynamic model, which were established according to the Denavit-Hartenberg convention.

The environmental interactions of the robot are simplified to point contacts at the ends of the feet. The contact with the ground is modeled using a linear spring and damper in the normal and tangential directions. Stiffness, damping, and friction values were chosen to mimic the behavior of rubber on concrete.

Additionally, realistic actuator dynamics are accounted for with a dynamic model of the USEA [13] that includes the effects of amplifier saturation, forward and backward motor efficiencies, and hybrid dynamics from the direct-drive/series-elastic operation. Each of these adds difficulty to control the system through traditional approaches.

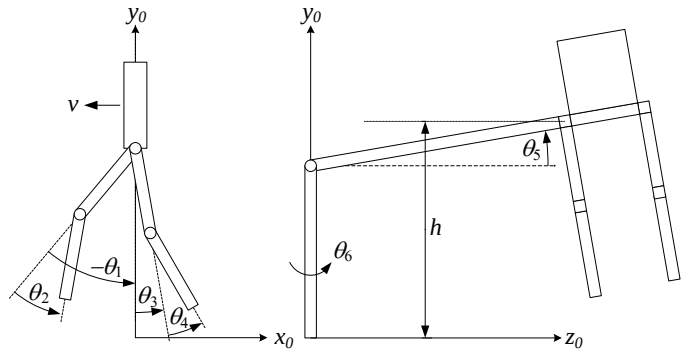

FIGURE 2. Kinematic model of the biped. (a) Side view. (b) Front view.

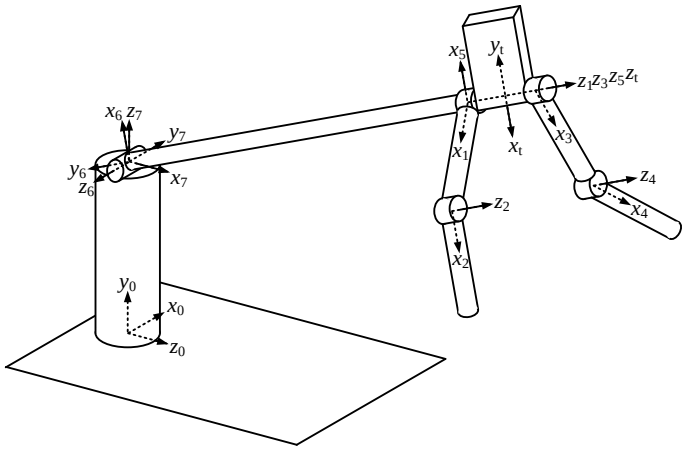

FIGURE 3. Coordinate frame definitions for modeling. Not all axes are included for clarity. The remaining axes follow the right-hand rule.

3 CONTROL APPROACH

The control strategy developed to execute a jump is split into two functional levels. The low-level control maintains continuous control of the actuators during all phases of the motion. High-level fuzzy control is used in this work to coordinate a jump through determination of a set of values that parameterize the low-level control. The fuzzy controller must address the complex nonlinear relationship between the actuation parameters and the resulting motion. The series-elastic actuation scheme and hybrid dynamics of intermittent ground contact and flight phases serve to compound the difficulty of this task. As a result, it is much more tractable to derive these relationships through a learning process than through an analytic treatment of the problem.

3.1 Low-level control

The low-level controller controls the legs in unison to achieve the desired overall jump. A state machine is used to detect the occurrence of specific events and to prescribe the control policy for the current state of movement. This decomposition of complex motion into simpler states, or motor primitives, has its roots in biological motor control theory. Experimental and theoretical work both support the concept that the central nervous systems of animals control the performance of complex behaviors through concatenation of multiple simple motions [14]. Researchers have noted the promise of using sequences of motor primitives to perform imitation learning in humanoids [15]. In addition, the modular nature of the motor primitive control approach offers adaptability of the overall control strategy to further dynamic maneuvers.

TABLE 1. Link parameters for KURMET

Model Parameter	Units	Link	Value
Mass	kg	torso	12.1
		thigh	0.81
		shank	0.63
		boom	2.7
Length	m	torso (height)	0.26
		torso (width)	0.26
		thigh	0.25
		shank	0.25
		boom	2.0
Mass Center[a]	m	torso (height)	0.138
		torso (width)	0.849
		thigh	0.0784
		shank	0.0784
		boom	1.0

[a]The mass center of each link is measured along the link's major axis from the preceding joint.

The state machine developed for a jump, shown in Fig. 4, positions the leg for ground contact, slows the body's vertical motion through compression of the legs, delivers energy to the body through leg extension, and retracts the legs to clear an obstacle. The state machine is cyclic to accommodate repetitive jumping. Although the state machine operates in a cyclic manner, this control strategy should not be associated with a central pattern generator approach. The goal of the low-level controller is not to produce rhythmic patterned outputs, but rather to provide continuous non-periodic control as dictated by the high-level fuzzy controller.

The **position** state drives the hips and knees to their desired positions at ground contact during the descent of the robot in flight. The primitive begins at top-of-flight and is terminated upon touchdown. The hip and knee actuators use independent PD controllers to track coordinated trajectories defined by equal periods of constant acceleration and deceleration, as well as a final position. Since the legs are controlled in unison, the control

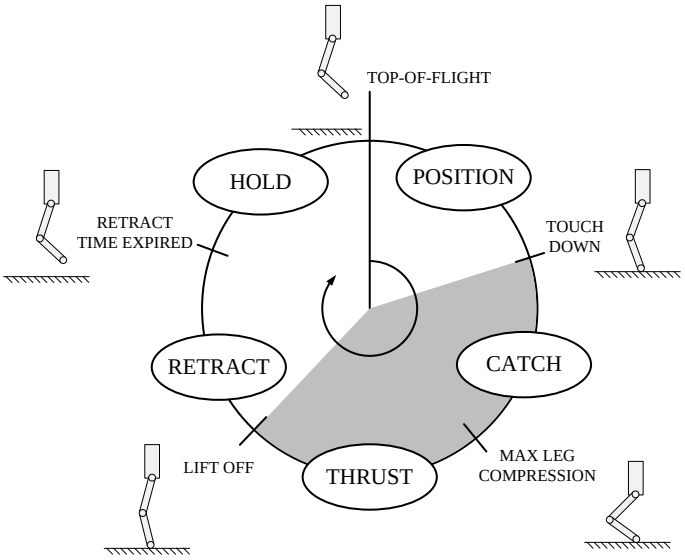

FIGURE 4. Low-level control state machine for a jump. Gray portions indicate states that take place with the robot in contact with the ground.

policy for **position** is parameterized by the hip and knee joint final positions, denoted $\theta_{h,pos}$ and $\theta_{k,pos}$.

The **catch** state begins at ground contact and is terminated at maximum leg compression. The associated control law is simple; open-loop currents $i_{h,cat}$ and $i_{k,cat}$ are commanded for the hip and knee motors to reverse the vertical velocity of the body. The motor current parameters can be interpreted as a metric for the strength applied by the mechanical equivalent of the leg musculature. These currents do not lead directly to torques at the joints due to the series-elastic actuation scheme. The motor currents serve to deflect the compliant elements in the USEAs, which then lead to a joint torque. These actuator dynamics complicate the appropriate selection of motor currents for each jump.

The **thrust** state begins at maximum leg compression and ends when the angle between the thigh and shank becomes less than 1.3 radians. **Thrust** can be viewed as terminating approximately at liftoff. This state uses the actuators to inject energy into the jump movement through rapid deflection of the compliant elements in the USEAs. A control law similar to that of the **catch** state is implemented for **thrust**, with open-loop hip and knee command currents $i_{h,thr}$ and $i_{k,thr}$.

The **retract** state is similar in structure and objective to **position**. The hip and knee motors are driven from their initial positions at the completion of **thrust** to angular positions corresponding to the desired top-of-flight joint angles $\theta_{h,tof}$ and $\theta_{k,tof}$. The primitive is timed to be completed before reaching top-of-flight. The state uses a triangular velocity trajectory as in **position** but is timed with one third acceleration and two thirds deceleration, which provides extra time for deceleration to prevent deflection of the USEA under inertial loads. The control policy for this state requires no parameterization from the high-level control, as the top-of-flight joint angles are fixed for a leg configuration with a moderately shortened virtual length.

The **hold** state follows **retract** and is terminated at top-of-flight. The transition from this state triggers a high-level fuzzy control decision to select parameters for execution of the following jump. The output $\mathbf{y}$ of the fuzzy controller includes all inputs to the low-level controller:

$$\mathbf{y} = (\theta_{h,pos}, \theta_{k,pos}, i_{h,cat}, i_{k,cat}, i_{h,thr}, i_{k,thr}). \quad (1)$$

3.2 High-level fuzzy control

The high-level fuzzy controller operates at discrete instances, once per jump, to define the parameters required by each low-level state. Fuzzy control offers a strategy to create a mapping between the control input $\mathbf{x} = (x_1, ..., x_n)$, typically state variables, and the control output $\mathbf{y} = (y_1, ..., y_m)$ for complex, nonlinear systems. Fuzzy control is particularly well suited to address the difficulties that must be overcome to perform dynamic movements with legged machines [10, 11]. While an analytical formulation of the control strategy may require an oversimplified model that fails to account for important dynamics, fuzzy control can work with a realistic model and can "learn" to take advantage of the essential relationships required to control complex systems. Additionally, this characteristic of fuzzy control eliminates the need for complex modeling of the system. While not implemented in this work, fuzzy controllers can also be augmented with online adaptation mechanisms to improve performance when transitioned from simulation to implementation [16].

The structure of this control system is shown in Fig. 5. As shown, fuzzy control can be divided into three primary processes: fuzzification, inference, and defuzzification. Fuzzification interprets the numeric values of the control inputs $x_1, ..., x_n$. Each input x_i is mapped into one or more membership functions. A membership function describes the certainty μ that a given input value can be represented by one of the discrete values for which the controller contains appropriate outputs. The triangular membership functions used to characterize the input for initial body velocity are shown in Fig. 6. For example, if the initial velocity is 0.055 m/s, then $\mu^{v_o}_{0.00} = \mu^{v_o}_{0.11} = 0.5$, and the certainty for all other membership functions is zero. This indicates that the controller should implement control outputs that equally weight known values for cases where the velocity is 0.00 m/s and 0.11 m/s.

The inference process of fuzzy control uses fuzzy rules to match appropriate control outputs to the given inputs. A fuzzy rule is created for every combination of the control input membership functions. These rules and their control outputs are collectively stored in a table known as the fuzzy rule-base, and the

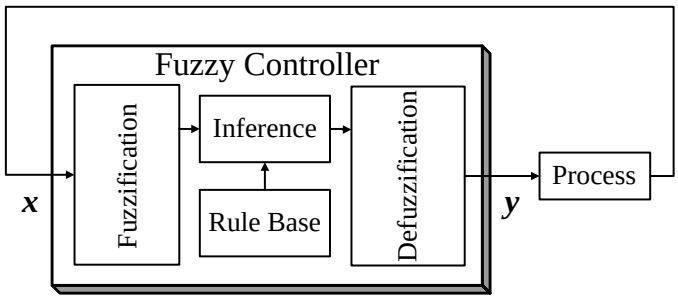

FIGURE 5. Fuzzy control block diagram. In this work, the 'Process' is a complete jump cycle of the biped.

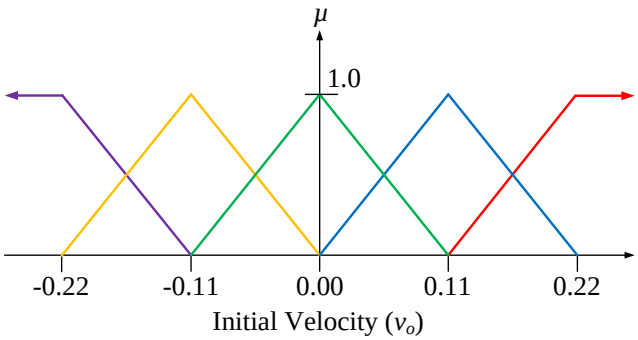

FIGURE 6. Triangular fuzzy membership functions for initial velocity with end saturation.

total number of rules N is equal to the product of the number of membership functions for the inputs. The inference mechanism determines the applicability of each rule to the inputs. The product is used to determine the certainty μ_i that the premise of rule $i = 1,...,N$ and its corresponding output is applicable. The certainty of rule i whose premise is *"If initial velocity is 0.0 m/s and initial height is 0.575 m and desired height is 0.600 m"* would be:

$$\mu_i = \mu_{0.0}^{v_o} \times \mu_{0.575}^{h_o} \times \mu_{0.6}^{h_d}. \tag{2}$$

Associated with each fuzzy rule i are appropriate control outputs, the consequents, represented by the control output vector $\mathbf{u_i} = (u_{i,1},...,u_{i,m})$ for the m outputs. The discrete points in the rule-base described by each individual rule and their consequents approximate the nonlinear, multi-dimensional surface that relates the inputs and ideal outputs.

TABLE 2. Control Input Membership Function Centers

Symbol	Input	Membership Function Centers
v_o	Initial Velocity (m/s)	-0.22 , -0.11 , 0.00, 0.11 , 0.22
h_o	Initial Height (m)	0.550 , 0.5625 , 0.575 , 0.5875 , 0.600
h_d	Desired Height (m)	0.550 , 0.5625 , 0.575 , 0.5875 , 0.600

Defuzzification combines these consequents into a crisp output $\mathbf{y}$ based upon rule certainties. Center average defuzzification is used, and the output $\mathbf{y}$ is:

$$\mathbf{y} = \frac{\sum_i \mu_i \mathbf{u_i}}{\sum_i \mu_i}. \tag{3}$$

Each rule output is multiplied by its corresponding certainty, which weights the controller output towards the recommendation of the most applicable rules. The particular overlapping triangular membership functions used here ensure that for each input, a maximum of two membership functions will have a non-zero certainty. This property dictates that a maximum of 2^n rules will be applicable for any combination of inputs.

For this work, as shown in Table 2, the fuzzy inputs identified to characterize the system at top-of-flight are the initial velocity v_o and initial height h_o, while the desired behavior of the next jump is characterized through the desired height h_d with zero final velocity. A minimal characterization is important since the fuzzy rule base increases in size exponentially with the number of inputs. In this case, the five triangular membership functions for each of the three fuzzy inputs require a rule base with 5^3=125 rules.

4 FUZZY TRAINING

Determining the consequents $\mathbf{u_i}$ that will produce the desired maneuver for all combinations of control input membership functions is accomplished through a training algorithm with a novel structure. The algorithm locates satisfactory combinations of control parameters through an iterative, error-based feedback learning scheme that is targeted to the specific failure mode at each iteration. This is used because the complexity of the system prevents the application of universal training laws. The algorithm begins by performing a single jump starting at a system state defined by the fuzzy control inputs and membership function centers $v_o = 0.00$ m/s and $h_o = 0.575$ m, with the desired top-of-flight state defined by $h_d = 0.575$ m and $v_d = 0.0$ m/s.

The values of the control parameters $(y_1,...,y_6)$ initially used are manually tuned to produce a jump with a small error from the

desired end state. This initialization point, or seed, improves convergence of the training algorithm. The use of a seed as a starting point for the training algorithm supports Schaal's conception of imitation learning [15]. The actual state of the system at the end of the jump cycle is used to define the error ε from the desired state:

$$\varepsilon_1 = h - h_d \quad \text{and} \quad \varepsilon_2 = v_d - v.$$

The jump is considered acceptable if these errors fall within a specified range, $|\varepsilon_i| \leq \delta_i$ for $i = 1,2$, where δ_i is the error tolerance on the training and reflects the distribution of membership function centers.

The errors are used to determine the necessary changes to the control parameters to improve the jump performance. The errors are assigned to one of three ranges: positive (+), negative (-), and acceptable (x). There exist a total of 8 combinations of these error ranges, each of which represents a specific failure mode. The failure modes each have associated training laws, described in Table 3, which describe the parameters to change as well as the gains used to modify the values. This structure offers more flexibility than universal training laws applied at every iterative jump. Selectively modifying parameters based on the mode of failure improves convergence by first addressing errors that would prevent reduction of remaining errors.

The training laws are used to modify the parameters and the jump is performed again. The process is repeated until all errors fall within the acceptable range. The training algorithm then proceeds on to the next fuzzy rule. The previously trained parameters from an adjacent fuzzy rule are used as the initial seed for all rules subsequent to the first.

A non-minimal selection of fuzzy outputs complicates the approach described. While outputs are needed to provide the training algorithm with flexibility to converge to the desired behavior, too many outputs can allow convergence to different modes of operation. If multiple modes of operation exist in the fuzzy rule base, the controller may select poor outputs when inferring between rules. The outputs in Eq. 1 were selected to provide flexibility and did not underconstrain the training problem.

Traditional fuzzy control techniques rely on the use of heuristic information to describe the relationships between inputs and outputs. In contrast to this, the heuristic knowledge present in this controller is captured in the training laws. The training laws specified in the eight cases described in Table 3 each represent a heuristic statement defining modifications to the control outputs to improve the performance of a jump. The training algorithm presented here demonstrates the ability to locate the 725 control parameters required for the complete rule base. Extension of this method to more complex maneuvers and systems requires the ability to formulate similar heuristics that address any mode of failure unique to the system/maneuver.

TABLE 3. Training Case Structure

Case Number	ε_1	ε_2	Training Law
1	x	+	$\Delta\theta_{h,pos} = -k_{h,a} \cdot \varepsilon_2$
2	x	-	$\Delta\theta_{h,pos} = -k_{h,a} \cdot \varepsilon_2$
3	+	x	$\Delta i_{h,cat} = -k_{h,c} \cdot \varepsilon_1$ $\Delta i_{k,cat} = k_{k,c} \cdot \varepsilon_1$
4	+	+	$\Delta\theta_{h,pos} = -k_{h,a} \cdot \varepsilon_2$
5	+	-	$\Delta\theta_{h,pos} = -k_{h,a} \cdot \varepsilon_2$
6	-	x	$\Delta i_{h,thr} = -k_{h,t} \cdot \varepsilon_1$ if $i_{h,thr} < 11.75$
			$\Delta i_{k,thr} = k_{k,t} \cdot \varepsilon_1$ if $i_{h,thr} < 11.75$
			$\Delta i_{h,cat} = -k_{h,c} \cdot \varepsilon_1$ if $i_{h,thr} > 11.75$ and $i_{h,cat} < 11.75$
			$\Delta i_{k,thr} = k_{k,t} \cdot \varepsilon_1$ if $i_{h,thr} > 11.75$ and $i_{h,cat} < 11.75$
			$\Delta i_{k,thr} = k_{k,t} \cdot \varepsilon_1$ otherwise
7	-	+	$\Delta\theta_{h,pos} = -k_{h,a} \cdot \varepsilon_2$
8	-	-	$\Delta\theta_{h,pos} = -k_{h,a} \cdot \varepsilon_2$

The ability of the control system to learn parameter combinations that produce the desired maneuver avoids the problem of finding a general set of equations that works at all possible system states. The use of error-based feedback as a biologically sound principal is supported by O'Reilly [17].

5 RESULTS

5.1 Simulation Results

The training process previously described was executed in RobotBuilder [18], a robotic simulation environment based upon the DynaMechs software package [19]. Fig. 7 shows an example of the learned fuzzy surfaces for one of the fuzzy outputs, the knee thrust current. While this figure shows the general heuristic that higher knee thrust currents are required for higher heights, it also highlights the complex nonlinear relationships between the inputs and the outputs.

The simulator was used to verify the effectiveness of the constructed fuzzy controller over a range of jump heights. Fig. 8 shows the height over a series of nine jumps. The sets of jumps have desired heights of 0.555 m, 0.595 m, and 0.57 m, respectively, which causes the fuzzy controller to use multiple rules for each jump. Still, the controller demonstrates impressive performance, achieving an average absolute height error of 0.0006 m.

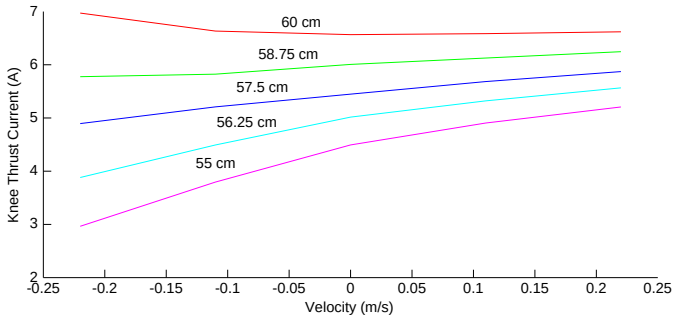

FIGURE 7. Learned fuzzy surfaces for knee thrust current over a variety of desired heights. All use a starting height of 0.55m.

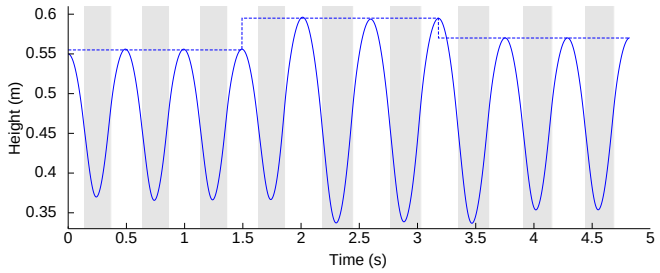

FIGURE 8. Simulated jump height versus desired height over nine jumps. Gray areas represent periods of ground contact.

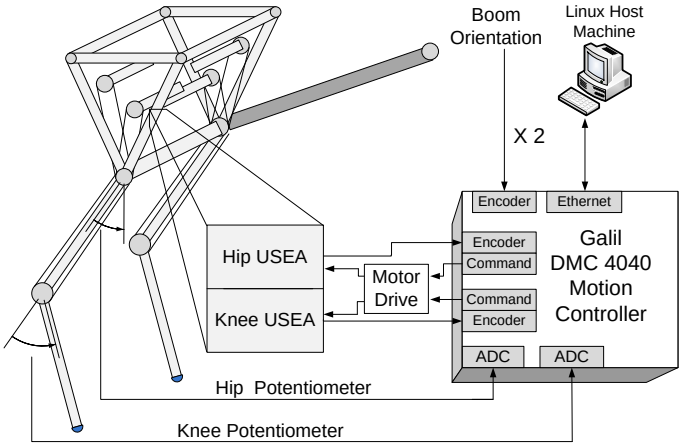

FIGURE 9. Real-time distributed control system diagram. USEA denotes unidirectional series-elastic actuators.

5.2 Real-time control system

Real-time control for the experimental system was provided by a distributed control system, as shown in Fig. 9. The structure of this distributed control system is a natural fit for use with the control strategies developed, as high-level and low-level con-

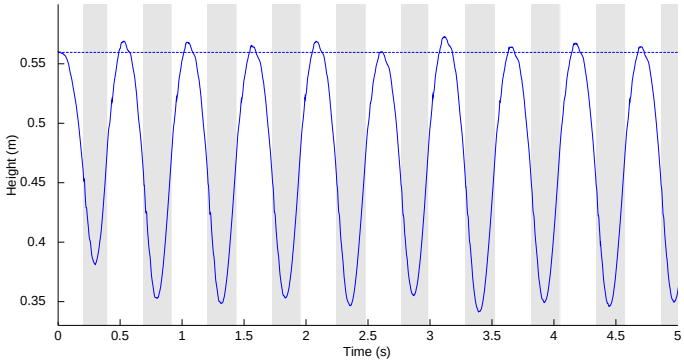

FIGURE 10. Experimental jump height versus desired height over ten jumps. Gray areas represent periods of ground contact. Desired height is 0.56 m.

trols were handled respectively by a Linux host machine and a Galil DMC-4040 motion controller. The controller provided PD servo loop rates of 1 kHz on each of the biped's four axes. The command language for the motion controller was used to implement the control policies for each of the states discussed in Section 2. Each axis employed an Advanced Motion Controls (AZBDC12A8) torque-mode amplifier to control a brushless DC Maxon (EC-powermax 30) motor.

The motion controller also informs the Linux host machine of the sensed system state, including joint and boom angles. Digital filters derived from finite differences and Butterworth filters are used to augment the sensed system state with approximations of angular rates. This full system state is then used to make high-level decisions regarding low-level state transitions as well as perform the fuzzy control function.

5.3 Experimental results

Experimental tests for a single jump height were used to demonstrate the validity of the control approach for the experimental hardware. Simulation parameters were adjusted to reduce discrepancies in the system behavior between the simulation and experimental hardware. All fuzzy rules trained in simulation were then used without modification on the experimental system.

Fig. 10 shows the performance of the real-time distributed control system over ten jumps with a desired height of 0.56 m. The progression of a single jump is shown in Fig. 11. The controller demonstrates a tendency to consistently overshoot the desired jump height, yet the average absolute height error remains small at 0.006 m.

6 SUMMARY

This work has provided a foundation for further studies on the performance of dynamic movements. There are a number

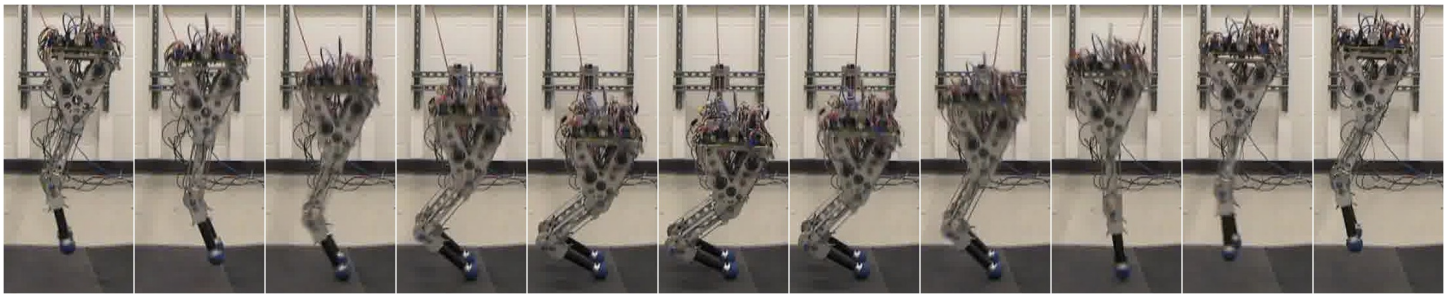

FIGURE 11. Evolution of a single jump. The sequence displays one cycle of the state-based control strategy, from top-of-flight to top-of-flight.

of extensions of this work that could broaden its applicability to other movements and systems. Current work aims to incorporate adaptation schemes that modify the fuzzy rule base online to improve experimental performance and to further decrease the need for simulation parameter accuracy. The speed of the training approach could be further improved through examination of proper gain selection techniques for each of the training laws. In addition, the selection of fuzzy outputs employed represents one of many sets of outputs that could be used to parameterize a jump. While this selection was shown to be effective, it presents the opportunity for further work to examine the effectiveness of other jump parameterizations.

The novel training algorithm coupled with the biologically inspired low-level control and high-level fuzzy control offers a framework that is extendable to other movements and systems. This approach was verified through dynamic simulation and experimental testing with an experimental biped. The work's success encourages further studies to realize the full potential of the experimental system.

7 ACKNOWLEDGMENTS

The authors gratefully acknowledge Brian Knox for his sound work on the design and construction of the experimental biped. Support for this work was provided by grant no. IIS-0535098 from the National Science Foundation to The Ohio State University.

REFERENCES

[1] Hirano, T., Sueyoshi, T., and Kawamura, A., 2000. "Development of ROCOS (robot control simulator)-jump of human-type biped robot by the adaptive impedance control". *International Workshop on Advanced Motion Control*, pp. 606–611.

[2] Hosoda, K., Takuma, T., Nakamoto, A., and Hayashi, S., 2008. "Biped robot design powered by antagonistic pneumatic actuators for multi-modal locomotion". *Robotics and Autonomous Systems,* ***56***(1), January, pp. 46–53.

[3] Niiyama, R., Nagakubo, A., and Kuniyoshi, Y., 2007. "Mowgli: A bipedal jumping and landing robot with an artificial musculoskeletal system". *IEEE International Conference on Robotics and Automation*, April, pp. 2546–2551.

[4] Raibert, M., 1986. *Legged robots that balance*. MIT Press.

[5] Poulakakis, I., and Grizzle, J., 2007. "Monopedal running control: SLIP embedding and virtual constraint controllers". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 323–330.

[6] Morris, B., Westervelt, E. R., Chevallereau, C., Buche, G., and Grizzle, J. W., 2006. "Achieving bipedal running with rabbit: Six steps toward infinity". In *Fast Motions in Biomechanics and Robotics*. Springer Berlin / Heidelberg, pp. 277–297.

[7] Antsaklis, P., and Passino, K., eds., 1993. *An Introduction to Intelligent and Autonomous Control*. Kluwer Academic Publishers.

[8] Pratt, G. A., and Williamson, M. M., 1995. "Series elastic actuators". In Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 1, pp. 399–406.

[9] Babic, J., Lim, B., Omrcen, D., Lenarcic, J., and Park, F. C., 2009. "A biarticulated robotic leg for jumping movements: theory and experiments". *ASME Journal of Mechanisms and Robotics,* ***1***(1), pp. 011013:1–9.

[10] Marhefka, D., Orin, D., Schmiedeler, J., and Waldron, K., 2003. "Intelligent control of quadruped gallops". *IEEE/ASME Transactions on Mechatronics,* ***8***(4), pp. 446–456.

[11] Palmer, L. R., and Orin, D. E., 2010. "Intelligent control of high-speed turning in a quadruped". *Journal of Intelligent and Robotic Systems,* ***58***(1), pp. 47–68.

[12] Knox, B., and Schmiedeler, J., 2009. "A unidirectional series-elastic actuator using a spiral torsion spring". *ASME Journal of Mechanical Design,* ***131***(12), pp. 125001:1–5.

[13] Curran, S., Knox, B. T., Schmiedeler, J. P., and Orin, D. E., 2008. "Design optimization of series-elastic actuators for dynamic robots with articulated legs". *ASME Journal of Mechanisms and Robotics,* ***1***(1), pp. 011006:1–9.

[14] Bizzi, E., Cheung, V. C. K., d'Avella, A., Saltiel, P., and Tresch, M., 2008. "Combining modules for movement". *Brain Research Reviews,* ***57***(1), pp. 125–133.

[15] Schaal, S., 1999. "Is imitiation learning the route to humanoid robots?". *Trends in Cognitive Sciences,* ***3***(6), pp. 233–242.

[16] Palmer, L., Orin, D., Marhefka, D., Schmiedeler, J., and Waldron, K., 2003. "Intelligent control of an experimental articulated leg for a galloping machine". In Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 3, pp. 3821–3827.

[17] O'Reilly, R., 1998. "Six principles for biologically based computational models of cortical cognition". *Trends in Cognitive Sciences,* ***2***(11), Nov, pp. 455–462.

[18] Rodenbaugh, S., 2003. "RobotBuilder: a graphical software tool for the rapid development of robotic dynamic simulations". Master's thesis, Ohio State University, Department of Electrical & Computer Engineering.

[19] McMillan, S., Orin, D., and McGhee, R., 1995. "DynaMechs: An object oriented software package for efficient dynamic simulation of underwater robotic vehicles". In Underwater Robotic Vehicles: Design and Control, TSI Press, pp. 73–98.