Dexterous Mobility with the uBot-5 Mobile Manipulator

Scott R. Kuindersma, Edward Hannigan, Dirk Ruiken, and Roderic A. Grupen

***Abstract*— We present an initial demonstration of *dexterous mobility* using the uBot-5, a dynamically balancing mobile manipulator. Dexterous mobility refers generally to a level of bodily resourcefulness that permits the autonomous reassignment of effectors for the purpose of maintaining mobility in a variety of situations. We begin by describing a set of postural stability controllers in terms of a small number of simple control objectives. We then show how the resulting postures support dexterous mobility by enabling a new "knuckle walking" mobility mode. In a preliminary experiment, we develop this mobility mode by formulating a practical reinforcement learning problem that allows the robot to learn an efficient gait on-line in a single trial.**

I. Introduction

The construction of autonomous mobile manipulators that are capable of operating in unstructured, dynamic environments is an active area of research with many important applications, ranging from planetary exploration to human assistive care [1]. The uBot-5 (Figure 1) is an 11-DOF mobile manipulator designed to perform work in its environment using a whole-body approach to mobility and manipulation. In this paper, we present results demonstrating the potential for dexterous mobility using the uBot-5. Here dexterity takes on the meaning expounded by Bernstein [2], who spoke generally of speed, agility, skillfulness, and resourcefulness of body and mind. In the context of mobile manipulators, we are interested in the autonomous reassignment of effectors to meet many, perhaps competing, objectives relating to stability and mobility. Such capabilities are becoming a central design point for many next generation mobile manipulators (e.g., the NASA ATHLETE rover [3] and RobotCub's iCub [4]).

One of the initial goals of this research is the development of a set of controllers for transitioning between and maintaining a variety of stable postures. We claim that the robot's ability to exploit redundant degrees of freedom from different postural stances allows it to solve a wider variety of control tasks. For example, we expect that many control tasks will require achieving mobility in environments not conducive to an upright posture, e.g., navigating over irregular or slippery terrain. In order for the robot to attempt such tasks, it must implement a gait that allows it to gracefully cope with such environmental irregularities. One solution may be to lean forward and use its endpoints as stabilizing ground contacts as it translates its body forward. In this work, we demonstrate how such a behavior can be learned by exploring a constrained subspace of the postural stability transition graph (described in Section III). Since this behavior resembles the style of locomotion familiar to gorillas and chimpanzees, we refer to it as "knuckle walking."

The authors are affiliated with the Laboratory for Perceptual Robotics in the Department of Computer Science, University of Massachusetts Amherst. Corresponding author email: scottk@cs.umass.edu

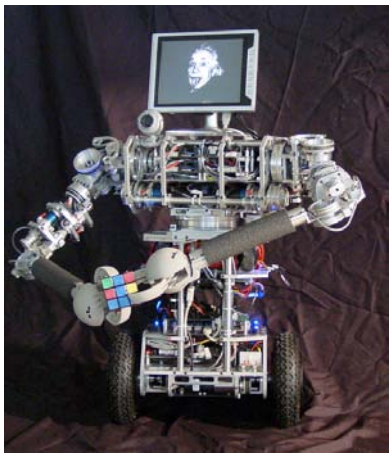

Fig. 1. The uBot-5 mobile manipulator in the balancing posture demonstrating two 2-DOF hand prototypes.

The rest of the paper is organized as follows. In Section II we provide background on the *control basis* [5] approach to developmental robot programming. In Section III we introduce a postural stability graph that illustrates the relationships between the four basic postures studied in this paper. Using the control basis approach, we formulate the principle controllers necessary to achieve robust transitions between these postures. Finally, in Section IV, we present a preliminary demonstration of dexterous mobility by formulating a simple reinforcement learning (RL) [6], [7] problem that allows the robot to learn an efficient knuckle walking gait on-line in a single trial.

II. Control Basis

The control basis is a framework for generating and combining closed-loop controllers to produce integrated behavior from a small set of primitive system objectives [5]. It is formally defined by three sets: potential functions Ω_ϕ, sensor resources Ω_σ, and control resources Ω_τ. An element in Ω_σ is a reference value derived from the robot's sensors, such as the Cartesian position of an endpoint or the angle of the left elbow joint. The control resources contained in the set Ω_τ are generally references for low-level joint position or torque controllers.

Each potential function $\phi_i \in \Omega_\phi$ serves as a description of a primitive subtask in an integrated behavioral program. A potential function takes as an argument a subset of the available sensor resources and returns a positive scalar which may be thought of as a measure of the error in the system for that task. Examples of potential functions include fields that describe kinematic conditioning [8], harmonic functions for

collision-free motion [9], and force closure functions [10]. In each of these cases, ϕ is a scalar navigation function [11] defined to satisfy properties that guarantee asymptotic stability. In this work, we will make extensive use of a simple quadratic function,

$$\phi(\sigma_{ref}, \sigma) = (\sigma_{ref} - \sigma)^T (\sigma_{ref} - \sigma), \tag{1}$$

where $\sigma_{ref}, \sigma \in \Omega_\sigma$.

Closed-loop controllers are generated by combining a potential function $\phi \in \Omega_\phi$, with sensor resources $\sigma \subseteq \Omega_\sigma$, and control resources $\tau \subseteq \Omega_\tau$. The notation we use to describe a controller is $c(\phi, \sigma, \tau)$. Controllers provide a time series of reference inputs to the underlying control resources. The sensitivity of the potential function to changes in the values of control resources is captured in the error Jacobian, $J = \partial\phi(\sigma)/\partial\tau$. Reference inputs to control resources can then be computed by $\Delta\tau = J^{\#}\phi(\sigma)$, where $J^{\#}$ is the pseudoinverse of J [12].

Multi-objective control actions are constructed by allowing controllers to execute concurrently in a prioritized manner. Concurrency is managed by projecting lower priority controllers into the nullspace of superior controllers [13],

$$\Delta\tau = J^{\#}_{sup}\phi_{sup} + \left[I - J^{\#}_{sup}J_{sup}\right] J^{\#}_{inf}\phi_{inf}. \tag{2}$$

This prioritized mapping assures that inferior control inputs do not destructively interfere with superior objectives. In the following, we will use the "subject-to" operator "$\triangleleft$" as an abbreviation for the nullspace projection. We can then write $c_{inf} \triangleleft c_{sup}$, which is equivalent to (2). In general, we can cascade n controllers using nullspace projections, $c_n \triangleleft \cdots \triangleleft c_2 \triangleleft c_1$, allowing for a greater degree of multi-objective control [14], [5].

III. Postural Stability Control

Figure 2 illustrates four principle modes of postural stability for the uBot-5. The simplest mode is the prone posture of the robot, where it lies face down on the ground plane. In a sequence of control actions that move the arms into a push-up configuration and then execute the push-up, the robot can transition from the prone posture to a four-point stance or "quadruped" posture. From there, the robot may reconfigure itself so that one endpoint can be safely withdrawn to yield the "tripod" stance. Figure 2 shows contacts between the endpoints and the ground plane exclusively, however, these contacts can, theoretically, provide bracing forces against any surface in the environment including vertical surfaces. From the quadruped stance, the robot can push-up further into a nearly vertical posture from which the robot can transition to a balancing postural mode. In this mode, both hands are free to serve as manipulation devices instead of as stabilizing mechanisms for mobility.

Next we provide detailed descriptions of controllers for achieving the posture transitions described in Figure 2. Although we are interested in the development of postural stability controllers that are transferrable between morphologically-similar robots, the main result of this section is an initial demonstration of whole-body postural control

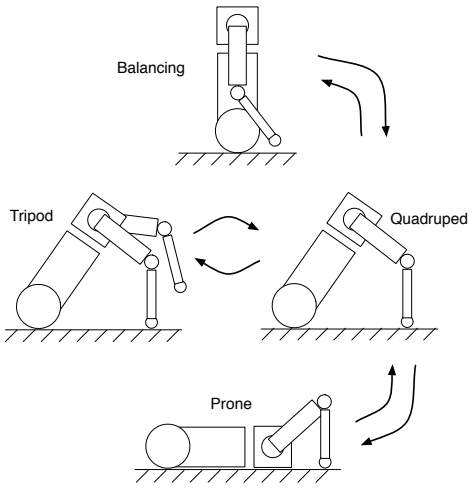

Fig. 2. The uBot-5 postural stability transition graph.

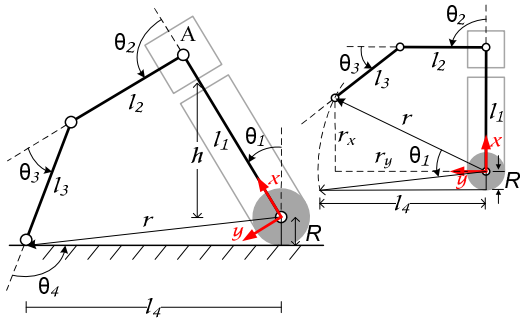

Fig. 3. Geometry of the uBot-5 in the sagittal plane. The inset on the right illustrates the uBot in a vertical body tilt configuration with the same upper body posture.

with the uBot-5. Thus, we leave a more general formulation of postural stability control for future work and present two simple and efficient controllers for achieving the transitions in Figure 2.

A. Push-up

We first consider the "push-up" maneuver, in which the robot transitions from prone to a four-point stance. Figure 3 provides a kinematic description of the robot that is sufficient to design the controllers for this maneuver. In this application, configuration variables include the attitude of the body relative to vertical, θ_1, joint angles, θ_2 and θ_3, the orientation of the forelimb with respect to the ground plane, θ_4, and the distance from the arm endpoint ground contact to the wheel axle, l_4. If we assume that the endpoint is rigidly connected to (and the wheel remains in contact with) the horizontal ground plane, the resulting closed-chain mechanism has only two degrees of freedom, θ_2 and θ_3, the values of which are measured directly using joint encoders on the uBot. We determine the values of the remaining configuration variables (θ_1, θ_4, and l_4) using the kinematic relations of the closed-chain mechanism.

The push-up maneuver is conceived as the combination of two objectives. The first is a kinematic conditioning objective related to protecting the relatively weak elbow joint in uBot-5 from the large forces involved in the behavior. This objective is achieved by constraining the attitude of the forelimb with

respect to the ground plane. In particular, the variable θ_4 in Figure 3 should remain close to $\pi/2$ radians. Note that this objective can be viewed more generally as maximizing the vertical component of the force ellipsoid by controlling the elbow joint angle [15]. The second objective is to raise the shoulder of the robot vertically from the prone position to an upright posture that bears most of the robot's weight on the wheels. The resulting four-point stance is very stable and operates the upper arm motors at nominal currents.

A multi-objective controller for transitioning between the prone and quadruped postures is designed using the control basis approach, where potential fields are used to map sensor resource, $\sigma = [\theta_2 \ \theta_3]^T$, to reference inputs for control resources, $\tau = [\theta_2 \ \theta_3]^T$, in a manner that reflects the independent objectives of the movement. Since l_4 changes throughout the push-up behavior, we must also send reference positions to the wheels consistent with the changes in θ_2 and θ_3 to avoid contention between the arm and wheel motors due to ground friction. All other degrees of freedom are held fixed while these closed-loop systems implement the push-up behavior.

Since the first objective is described by requiring that θ_4 be maintained near $\theta_{4,ref} = \pi/2$ radians, the error with respect to this objective can be expressed simply as

$$\epsilon_1 = \theta_{4,ref} - \theta_4 = \theta_1 + \theta_2 + \theta_3 - \pi.$$

The forelimb orientation constraint can be represented as a quadratic potential function as in (1), $\phi(\theta_{4,ref}, \theta_4) = \epsilon_1^2$. We can then write the error Jacobian, J_1, that describes the sensitivity of the orientation constraint with respect control variables, θ_2 and θ_3,

$$J_1 = \left[\begin{array}{cc} \frac{\partial \epsilon_1^2}{\partial \theta_2} & \frac{\partial \epsilon_1^2}{\partial \theta_3} \end{array} \right],$$

where $\frac{\partial \epsilon_1^2}{\partial \theta_2} = 2\epsilon_1(\frac{\partial \theta_1}{\partial \theta_2} + 1)$ and $\frac{\partial \epsilon_1^2}{\partial \theta_3} = 2\epsilon_1(\frac{\partial \theta_1}{\partial \theta_3} + 1)$.

The second objective of the push-up behavior is to elevate the shoulder joint (labeled "A" in Figure 3) to a vertical height l_1 in the wheel coordinate frame. This objective is stated most succinctly in terms of the postural variable θ_1. That is, by setting the reference value[1] $\theta_{1,ref} = 0$ we can describe the error in the system with respect to this objective as $\epsilon_2 = \theta_{1,ref} - \theta_1 = -\theta_1$. Using ϵ_2^2 as our potential function, we can write the Jacobian, J_2, describing the sensitivity of the shoulder constraint with respect to control variables, θ_2 and θ_3,

$$J_2 = \left[\begin{array}{cc} \frac{\partial \epsilon_2^2}{\partial \theta_2} & \frac{\partial \epsilon_2^2}{\partial \theta_3} \end{array} \right],$$

where $\frac{\partial \epsilon_2^2}{\partial \theta_2} = 2\theta_1 \frac{\partial \theta_1}{\partial \theta_2}$ and $\frac{\partial \epsilon_2^2}{\partial \theta_3} = 2\theta_1 \frac{\partial \theta_1}{\partial \theta_3}$.

Now that we have precisely defined our two control subtasks, we combine them using the approach to multi-objective control described by (2). Namely, we wish to elevate the shoulder joint *subject-to* maintaining a near vertical forelimb attitude. We write this mathematically as

$$\left[\begin{array}{c} \Delta\theta_2 \\ \Delta\theta_3 \end{array} \right] = J_1^{\#}\phi(\pi/2, \theta_4) + (I - J_1^{\#}J_1)J_2^{\#}\phi(0, \theta_1)$$

[1]Notice that the robot can push-up to different body tilt angles simply by changing the value of $\theta_{1,ref}$.

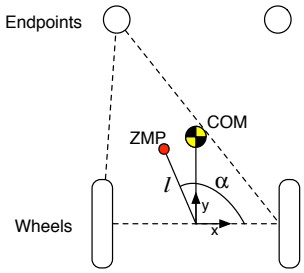

Fig. 4. Geometry of the uBot-5 in the ground plane. The dashed line represents the support polygon for the left tripod posture.

or $c(\epsilon_2^2, \sigma, \tau) \triangleleft c(\epsilon_1^2, \sigma, \tau)$, using the notation described in Section II. Here the forelimb postural control objective is higher priority than the shoulder elevation objective. Note that if the priorities were reversed, the robot might elevate its shoulder more quickly, but while doing so it may violate the constraint on forelimb attitude which could result in instability due to insufficient elbow motor torques.

B. Tripod Stability

We next describe a method for transitioning to and maintaining stable tripod stances. First we notice that it is not enough to simply withdraw one endpoint while in a four-point stance, since this would, in most cases, cause the robot become unstable and fall in the direction of the raised endpoint. We must first identify a suitable stability criterion, then develop a means of achieving a stable tripod according to this criterion.

As a first attempt, we make use of the familiar zero moment point (ZMP) [16] to define a measure of quasi-static stability. Given a set of N ground contact points, $\{p_1, p_2, \ldots, p_N\}$ and a set of ground reaction forces, $\{f_1, f_2, \ldots, f_N\}$, we use the z-component of the ground reaction force at each point to calculate the ZMP [17], $\text{ZMP} = \sum_{i=1}^{N} p_i f_{iz} / \sum_{i=1}^{N} f_{iz}$. We say the robot is stable if its center of mass (COM) projection onto the ground plane is within some small distance of the ZMP. Thus, the controller that stabilizes the robot in the left tripod posture must move the robot's COM projection toward the ZMP that is calculated using both wheel ground contacts and the left endpoint ground contact. Similarly, for the right tripod we calculate the ZMP using the right endpoint contact instead of the left, and for the four-point stance we use all four ground contacts. To design such a controller, we use the simplified kinematic description of the robot shown in Figure 4.

This task can also be described using our quadratic potential function,

$$\phi_T = \phi(\sigma_{com}, \sigma_{zmp}) = (\sigma_{com} - \sigma_{zmp})^T(\sigma_{com} - \sigma_{zmp}),$$

where $\sigma_{com} = [x_c \ y_c]^T$ and $\sigma_{zmp} = [x_z \ y_z]^T$ are sensor resource vectors corresponding to the location of the COM and ZMP in the ground plane. To minimize this potential, we use control resources $\tau = [y_z \ \alpha]$ to produce base translations and rotations on the robot. We make the assumption that for small changes in y_z and α, the ZMP remains fixed in the world frame and the COM projection remains fixed in the robot base coordinate frame (shown in Figure 4). In practice

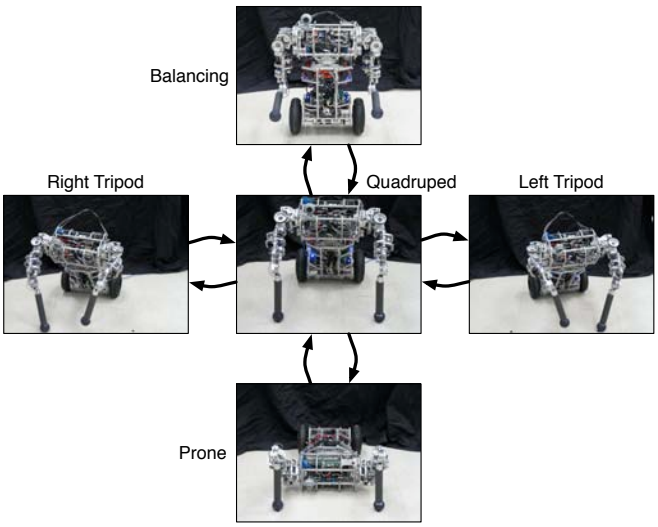

Fig. 5. The uBot-5 executing the postural stability controllers.

this assumption holds since mass of the body is much greater than the mass of the arms and the ground contact forces change very little during small movements.

Note that unlike the control resources in the push-up controller, y_z and α do not correspond directly to motors on the robot. However, since we can easily convert value changes in these parameters into specific motor commands, we can use these higher-level parameters to make our controller specification simpler. For example, a change in y_z can be translated into position changes in each wheel and corresponding changes in each endpoint position to avoid contention between wheel and arm motors.

We write the Jacobian for transitioning to the tripod posture as

$$J_3 = \begin{bmatrix} \dfrac{\partial \phi_T}{\partial y_z} & \dfrac{\partial \phi_T}{\partial \alpha} \end{bmatrix},$$

where $\frac{\partial \phi_T}{\partial y_z} = 2y_z - 2y_c$ and $\frac{\partial \phi_T}{\partial \alpha} = 2lx_c \sin(\alpha) - 2ly_c \cos(\alpha)$.

C. Implementation

Figure 5 shows the behavior of the uBot-5 while executing the postural stability controllers. The robot uses the push-up controller from the prone position with a reference body tilt angle of $\pi/6$ radians to achieve the shown quadruped posture. From there, it can use the left or right tripod stability controller to achieve a stable tripod posture, from which it can raise the non-supporting endpoint. The robot can replace the raised endpoint in the ground plane and transition back to a stable quadruped stance. From there, it can push-up further by setting the reference body tilt angle to 0 radians. After this objective is reached, it can activate a linear quadratic regulator (LQR) for balancing and withdraw its endpoints to transition to the upright mode. The robot can transition back to prone by positioning its endpoints for a four-point brace and deactivating the LQR controller. From the quadruped stance, the robot achieves the prone posture by using the push-up controller with a reference body tilt angle of $\pi/2$ radians, resulting in a set-down behavior.

IV. Knuckle Walking

Next we present preliminary experimental results in which a simulated uBot-5 is able to exploit the ability to achieve multiple postures for learning a new knuckle walking gait. The central result is an early demonstration of dexterous mobility using a combination of the control basis and reinforcement learning (RL) [6], [7]. Our decision to work with the uBot's simulated counterpart is strictly practical, since experiments on the real robot revealed that a subset of the positions encountered during a knuckle walking gait caused two of the upper arm motors to approach their thermal limits. In addition to providing a proof-of-concept, one of the benefits of programming the simulated robot is our ability to do a static analysis of the joint torques generated in the knuckle walking behavior and use this information in the design of new hardware for the uBot.

A. Q-Learning

In our learning experiments, we use single-step tabular Q-learning [18]. Q-learning is an off-policy temporal difference control algorithm used to learn approximations to the optimal action-value function for a Markov decision process (MDP). In this framework, we typically describe a task by a reward function and the goal of the learning agent is to discover a control policy that maximizes the expected sum of future rewards.

Q-learning is defined by the update,

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \right],$$

where s_t and a_t are the observed state and action taken at time step t, and r_{t+1} is the reward received after executing action a_t in state s_t. The parameters α and γ represent the learning rate and discount factor, respectively, the latter describing the current value assigned to future rewards. The actions available to the robot correspond to activations of closed-loop controllers constructed from the control basis. For example, running the tripod stability controller (Section III-B) until convergence is represented as a single discrete action. In our experiments, the states descriptions are derived directly from the set of available controllers, thus removing the need to engineer a task-specific state representation.

B. States and Actions

In previous work [5], we have seen that a natural state description can be derived from the set of controllers available to the robot. We can represent the state of the system as a tuple of bits, with the i^{th} bit being 1 if controller i is converged, and 0 otherwise. Hart *et al.* extended this to a four-valued predicate to describe the state of each controller [19] and Coelho explored states derived from a set of continuous dynamic models in the context of grasping [20].

For this task we use a simple state description derived from the set of control actions listed in Table I. For simplicity, we restrict the set of control actions those that would seem relevant to the knuckle walking behavior. However, it is important to note that the control actions were not specifically engineered for this task. Each action is a controller that is

constructible from the uBot's control basis. Thus, each action is a parameterization of a potential function describing a basic system objective such as improving stability or kinematic conditioning. For example, the action "raise right endpoint" is implemented by maximizing a measure of manipulability for the right endpoint [13] and the action "place left endpoint" repositions the left endpoint on the ground plane so that the resulting left tripod satisfies COM = ZMP. We proceed without providing a more detailed description of the controllers in Table I since their specific form is not central to the results in this section.

The state is described by an 11-tuple of bits, where each bit corresponds to one of the available controllers and is equal to 1 if the controller is converged, and 0 otherwise. For example, the state $(0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ corresponds to a stable left tripod with both endpoints in contact with the ground. We know that both endpoints are in contact with the ground since the bits corresponding to the controllers that raise either endpoint (8,9) are both 0.

Label	Controller Description
1	Stabilize quadruped stance
2	Stabilize left tripod stance
3	Stabilize right tripod stance
4	Forward
5	Reverse
6	Rotate clockwise
7	Rotate counterclockwise
8	Raise left endpoint
9	Raise right endpoint
10	Place left endpoint
11	Place right endpoint

TABLE I

PRIMITIVE CONTROL ACTIONS AVAILABLE TO THE ROBOT IN THE KNUCKLE WALKING TASK.

The total number of possible states given our 11-variable representation is $2^{11} = 2048$, but many of these states can be eliminated by logical constraints derived from stability criterion. It is important to impose restrictions on the actions available to the robot since it might otherwise operate in an unsafe way during the learning phase. This also adds structure to the learning problem, making it possible to find efficient policies on-line through stochastic exploration.

For the knuckle walking problem, we can encode logical constraints on actions succinctly and discover the reachable states using depth-first search. We define two boolean matrices that capture the stability requirements of the system during the knuckle walking task. The first matrix, A, is represented in Table II. The matrix A describes which actions can be executed concurrently, where each row and column correspond to a controller in Table I. For example, $A(4, 6) \equiv T$ means that concurrent execution of the forward translation and clockwise rotation controllers is valid, but since $A(4, 5) \equiv F$, moving forward and backward concurrently is not valid.

Suppose we consider concurrent actions which include up to three controllers. The total number of possible control actions is then $\binom{11}{1} + \binom{11}{2} + \binom{11}{3} = 231$. The validity of a combination of controllers i, j, k is defined by the logical expressions,

$$\text{Valid}(i, j) \Leftrightarrow A(i, j), \text{ and}$$

Controller ID	1	2	3	4	5	6	7	8	9	10	11
1	F	F	F	F	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F	F	F
3	F	F	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	T	T	T	T	F	F
5	F	F	F	F	F	T	T	T	T	F	F
6	F	F	F	T	T	F	F	F	T	F	F
7	F	F	F	T	T	F	F	T	F	F	F
8	F	F	F	T	T	F	T	F	F	F	F
9	F	F	F	T	T	T	F	F	F	F	F
10	F	F	F	F	F	F	F	F	F	F	F
11	F	F	F	F	F	F	F	F	F	F	F

TABLE II

A: VALID CONTROLLER COMBINATIONS.

$$\text{Valid}(i, j, k) \Leftrightarrow A(i, j) \wedge A(i, k) \wedge A(j, k).$$

Thus, by systematically checking all pairs and triples of controllers in this way, we can determine the complete set of allowable actions, the size of which in this case is 25.

The second matrix, C, is represented in Table III. This matrix describes the constraints imposed on actions given the current state. Here each row corresponds to a state bit and each column corresponds to a controller index. For example, if the robot is in a state with the right tripod stability controller converged (i.e., the 3^{rd} state bit is 1), it cannot safely raise the right endpoint since that is a supporting ground contact. This is reflected by $C(3, 9) \equiv F$.

State Bit	1	2	3	4	5	6	7	8	9	10	11
1	T	T	T	T	T	F	F	F	F	F	F
2	T	T	T	T	T	T	F	F	T	F	T
3	T	T	T	T	T	F	T	T	F	T	F
4	T	T	T	F	F	F	F	T	T	T	T
5	T	T	T	F	F	F	F	T	T	T	T
6	T	T	T	F	F	F	F	T	T	T	T
7	T	T	T	F	F	F	F	T	T	T	T
8	F	F	T	T	T	F	T	F	F	T	F
9	F	T	F	T	T	T	F	F	F	F	T
10	T	T	T	T	T	T	T	T	T	F	T
11	T	T	T	T	T	T	T	T	T	T	F

TABLE III

C: EXECUTABLE CONTROLLERS FROM EACH STATE.

Let us represent an action a as an 11-tuple, in the same way as a state. For example, the action $(0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0)$ corresponds to concurrently translating forward, rotating clockwise, and raising the right endpoint. To determine the validity of an action a taken from a state s using the matrix C, we use the following logical equivalence,

$$\text{Valid}(s, a) \Leftrightarrow \forall_{1 \leq i,j \leq 11} \left(s[i] \wedge a[j] \Rightarrow C(i, j) \right),$$

where $s[i]$ and $a[i]$ refers to the i^{th} bit of the state and action vector, respectively. This way an action is only valid from a particular state if for every state bit i equal to 1, every controller j involved in an action satisfies $C(i, j) \equiv T$.

We also make the restriction that *at least one stability controller must be converged at all times*. This means that all control actions must run subject to a stability controller (1–3). For example, when in a quadruped posture and executing the "forward" action, the robot will roll forward until its COM projection strays too far from the ZMP, at which time the

forward controller stops executing and becomes converged. The allowable distance between the COM and ZMP is a parameter that we set to 3 cm in our experiments. Since the "forward" action is a wheel velocity controller with a fixed reference velocity, this stability requirement provides a necessary stopping condition. The other translation and rotation actions (5–7) behave similarly.

Combining these constraints with the set of allowable actions, we can discover the set of reachable states by a simple depth-first search. We start by assuming that the robot is in a stable quadruped state and use the boolean matrix C to determine which of the possible 25 actions are executable from that state. Since the robot transitions from one state to the next in a predictable way, we can generate the next states reached by each allowable action and incrementally build a set of reachable states. We proceed recursively for each new state in turn until no actions remain for all states. In our experiments this procedure discovers a set of 144 reachable states, which is a significant reduction from the original 2048.

C. Learning Results

We conducted two learning experiments, one for the forward gait and another for rotation. In both experiments we used Q-learning with $\alpha = 0.1$ and $\gamma = 0.9$. Throughout learning, the robot used ε-greedy action selection with $\varepsilon = 0.1$. That is, at each time step, with probability ε the robot chooses a random action, otherwise it takes the best action based on its current estimate of the value function.

For the forward task, the reward function produced a small negative reward after each action plus reward proportional to the net body translation. We assume the robot begins each episode in the stable quadruped state. The robot learned online (i.e. by taking actions and observing rewards in real time) and converged to a fixed greedy policy in an average of 282.1 time steps (or actions) over 10 experiments. We consider a policy to be fixed if it remains unchanged for 50 time steps while learning. The learned forward knuckle walking policy is shown in Figure 6.

The rotation task is described by a reward function which yields a small negative reward after each action plus a reward proportional to the amount of clockwise rotation. For the rotate behavior to be effective in a wide range of situations, we would like to minimize its footprint over 2π radians. We therefore add a small negative component to the reward function proportional to the absolute body translation. This results in a rotate-in-place behavior shown in Figure 7. The robot converged to a stable policy in an average of 292.2 time steps in 10 experiments. Note that policies for counterclockwise rotation and backward translation can also be learned by straightforward modification of the reward function.

In Figure 8 we compare the effectiveness of the above policies with those learned when the robot is not allowed to execute controllers concurrently. It is clear that for both tasks, the ability to combine controllers to create concurrent actions leads to more efficient gaits. This result is a simple demonstration of the advantages of stochastically exploring multi-objective control actions using the control basis.

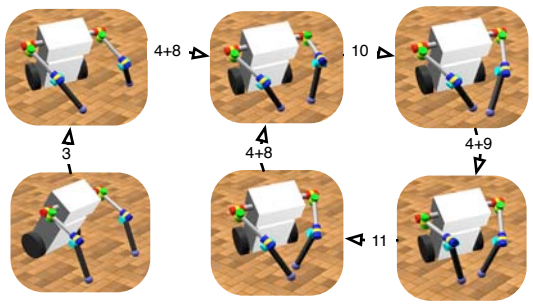

Fig. 6. The learned policy for the forward knuckle walking task. The numbers on the edges correspond to control actions in Table I.

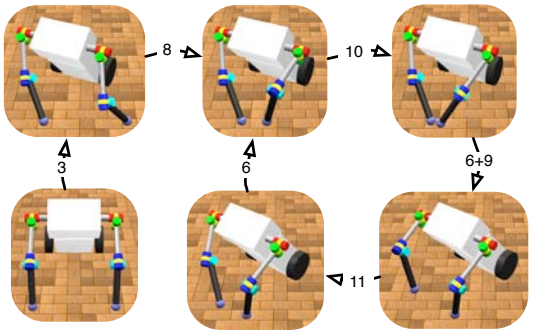

Fig. 7. The learned policy for the rotate-in-place task.

To verify that the learned knuckle walking policy is applicable to the real uBot-5, we allowed the robot to execute the forward gait with the help of an inertial reel to offset gravitational loads. This apparatus relieved enough joint stress to permit successful execution of the gait (Figure 9). Following our general approach to co-evolving behavior and hardware, we have used the knuckle walking behavior as a design specification for future versions of the robot. We are currently assembling a uBot-6 which includes more powerful motors in each arm joint.

V. Conclusions and Future Work

A. Conclusions

We have introduced two postural stability controllers for the uBot-5 mobile manipulator and showed that the ability to achieve new quadruped and tripod postures leads to an alternative knuckle walking mobility mode. Our primary contribution is a preliminary demonstration of dexterous mobility with the uBot-5; specifically, the ability of the robot

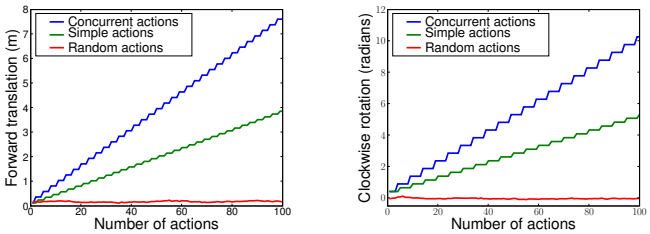

Fig. 8. Comparison of learned optimal policies with and without concurrent actions. (a) Cumulative forward translation as a function of number of actions. (b) Cumulative rotation as a function of number of actions. The random policies are averaged over 10 trials and include concurrent actions.

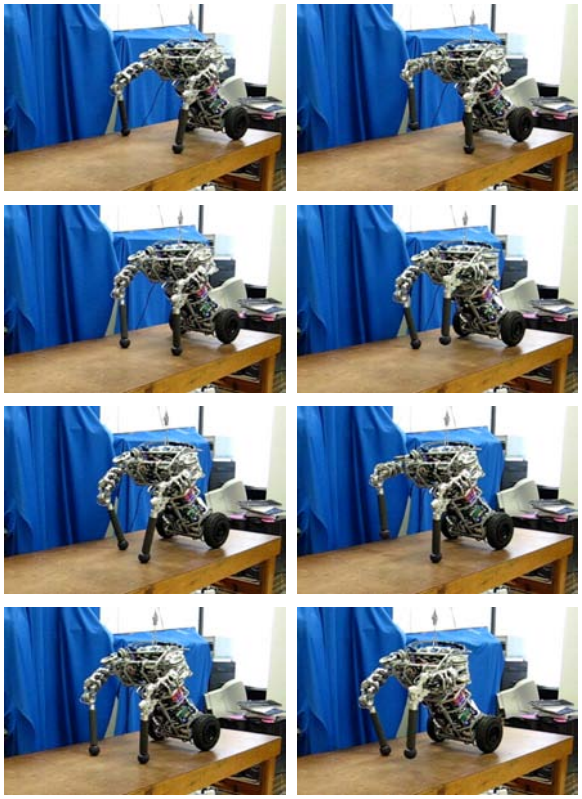

Fig. 9. The uBot-5 executing the learning knuckle walking gait aided by an inertial reel. The inertial reel offsets gravitational loads effectively reducing the robot's mass by 5 kilograms.

to reassign effectors typically used for manipulation to satisfy mobility objectives. Using a combination of the control basis and RL, the robot was able to learn an efficient knuckle walking gait by combining and sequencing control actions derived from task-independent system objectives related to stability, mobility, and kinematic conditioning. We believe that dexterous mobility will become an important design consideration for robotic platforms intended to be competent in navigation and manipulation tasks in a wide range of environments.

B. Future Work

There are several important directions for future work. First, we would like to generalize the postural stability controllers in a way that would permit transfer to other, morphologically similar, robots. For example, we can redefine the push-up behavior in terms of a more general kinematic conditioning objective that optimizes arm degrees of freedom for generating vertically oriented forces. We also intend to generalize the quasi-static ZMP-based controller to a moment residual controller [5], allowing the robot to achieve stable quadruped and tripod postures when the surface contacts are non-coplanar (e.g. leaning on a wall or walking up small stairs).

We would also like the robot to explore new mobility modes and identify the manipulation affordances present in different postures. This line of research may lead to new methods for managing multiple mobility and manipulation modes. For example, the robot must learn what mobility mode is most appropriate when faced with irregular terrain, and from what postures can it reach under a bed to retrieve an object. Using the level of abstraction the control basis provides, we hope to show that uBot can develop a large control action repertoire and learn to adaptively manage its resources to satisfy a variety of mobility and manipulation objectives.

VI. Acknowledgments

The authors would like to thank Patrick Deegan and Bryan Thibodeau for their early contributions to postural stability on the uBot. This research is supported under the NASA-STTR-NNX08CD41P, ARO-W911NF-05-1-0396, and ONR-5710002229 grants. Scott Kuindersma was also supported by a MSGC Fellowship.

References

[1] O. Brock and R. Grupen, "NSF/NASA workshop on autonomous mobile manipulation (AMM), final report," tech. rep., 2005.

[2] N. Bernstein, "On dexterity and its development," in *Dexterity and its Development* (M. Latash and M. Turvey, eds.), pp. 1–244, Mahwah, NJ: Lawrence Erlbaum Associates, 1996.

[3] B. H. Wilcox, T. Litwin, J. Biesiadecki, J. Matthews, M. Heverly, J. Morrison, J. Townsend, N. Ahmed, A. Sirota, and B. Cooper, "Athlete: A cargo handling and manipulation robot for the moon," *Journal of Field Robotics*, vol. 24, no. 5, pp. 421–434, 2007.

[4] S. Degallier, L. Righetti, L. Natale, N. Nori, G. Metta, and A. Ijspeert, "A modular, bio-inspired architecture for movement generation for the infant-like robot iCub," in *Proceedings of the IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob) 2008*, pp. 795–800, IEEE, 2008.

[5] M. Huber, *A hybrid architecture for adaptive robot control*. PhD thesis, University of Massachusetts Amherst, 2000.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[7] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[8] S. Hart and R. A. Grupen, "Natural task decomposition with intrinisic potential fields," in *International Conference on Robotics and Systems (IROS 2007)*, pp. 2507–2512, 2007.

[9] C. I. Connolly and R. A. Grupen, "The applications of harmonic functions to robotics," *Journal of Robotic Systems*, vol. 10, pp. 931–946, Oct. 1993.

[10] R. P. Jr., A. H. Fagg, and R. A. Grupen, "Manipulation gaits: Sequences of grasp control tasks," in *International Conference on Robotics and Automation (ICRA)*, pp. 801–806, IEEE, 2004.

[11] D. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, no. 4, pp. 412–442, 1990.

[12] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. pub-AW:adr: Addison-Wesley, 1991.

[13] T. Yoshikawa, "Manipulability of robotic mechanisms," *Int. J. Robotics Research*, vol. 4, pp. 3–9, Summer 1985.

[14] M. Huber and R. A. Grupen, "A feedback control structure for on-line learning tasks," *Robotics and Autonomous Systems*, vol. 22, no. 3-4, pp. 303–315, 1997.

[15] S. Chiu, "Control of redundant manipulators for task compatibility," in *International Conference on Robotics and Automation (ICRA)*, vol. 4, pp. 1718–1724, IEEE, 1987.

[16] M. Vukobratovic and Y. Stepanenko, "On the stability of anthropomorphic systems," *Mathematical Biosciences*, vol. 15, pp. 1–37, October 1972.

[17] B. Siciliano and O. Khatib, eds., *Springer Handbook of Robotics*, ch. 16, pp. 361–389. Springer, 2008.

[18] C. J. C. H. Watkins, *Learning from Delayed Rewards*. PhD thesis, King's College, 1989.

[19] S. Hart, S. Sen, and R. A. Grupen, "Intrinsically motivated hierarchical manipulation," in *International Conference on Robotics and Automation (ICRA)*, pp. 801–806, IEEE, 2008.

[20] J. A. Coelho, *Multifingered grasping: Grasp reflexes and control context*. PhD thesis, University of Massachusetts Amherst, Jan. 01 2001.