Composing Dynamical Systems to Realize Dynamic Robotic Dancing

Shishir Kolathaya, Wen-Loong Ma and Aaron D. Ames

Texas A&M University, College Station, TX 77843
shishirny,wenlongma,aames@tamu.edu

Abstract. This paper presents a methodology for the composition of complex dynamic behaviors in legged robots, and illustrates these concepts to experimentally achieve robotic dancing. Inspired by principles from dynamic locomotion, we begin by constructing controllers that drive a collection of *virtual constraints* to zero; this creates a low-dimensional representation of the bipedal robot. Given any two poses of the robot, we utilize this low-dimensional representation to connect these poses through a dynamic transition. The end result is a *meta-dynamical system* that describes a series of poses (indexed by the vertices of a graph) together with dynamic transitions (indexed by the edges) connecting these poses. These formalisms are illustrated in the case of dynamic dancing; a collection of ten poses are connected through dynamic transitions obtained via virtual constraints, and transitions through the graph are synchronized with music tempo. The resulting meta-dynamical system is realized experimentally on the bipedal robot AMBER 2 yielding dynamic robotic dancing.

1 Introduction

The problem of realizing different motion behaviors (or tasks) and switching between these different behaviors in robots has been well studied [6],[11]. Examples of techniques employed include the elastic strip framework for robot manipulators [6], the decision theoretic approach for mobile robots [5] and Eigen behaviors for generic robots [9]. In particular, the elastic strip framework is used to deviate from original preplanned tasks to reactively avoid obstacles while allowing for smooth transitions; the decision theoretic approach is used for mobile navigation; and Eigen behaviors are used to learn the tasks themselves. The resulting behaviors (or motion primitives) obtained through these methods are important in meeting different task requirements like pick and place, assembly, but have conclusively failed in applications like legged locomotion which require dynamic stability and handling of instantaneous discrete transitions (foot strikes).

While different locomotion behaviors have been obtained in legged robots [10, 20, 19] individually, formally composing these primitives in one single format and realizing them on robots is still a subject in its infancy. In [16], a method for composing stair climbing and flat ground walking behaviors on a bipedal robot in simulation was presented; importantly, in this work a formalism for composing these dynamic behaviors was presented: meta-hybrid systems. Other methodologies have also been considered which show similar characteristics including [15] which utilized state machines to navigate over rough terrain and [18] which applies reinforcement learning techniques to

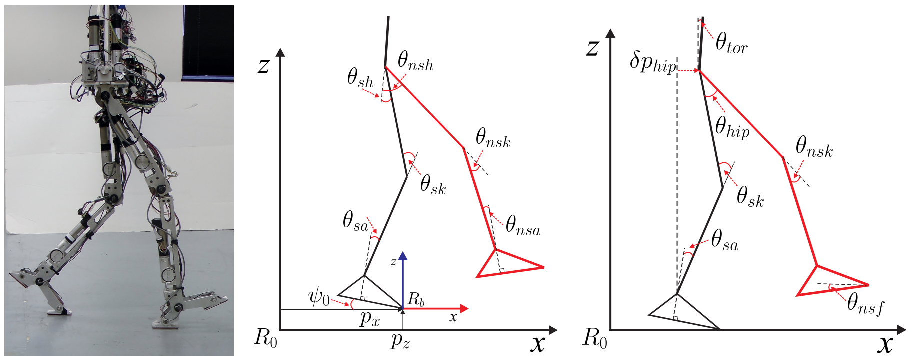

Fig. 1: The bipedal robot AMBER 2 (left), configuration angles (middle), and virtual constraints (right).

switch between behaviors and thus navigate varying ground slopes. Motivated by these constructions and the need to extend them beyond locomotion, this paper explores an approach to achieving dynamically stable advanced locomotion behaviors on bipedal robots by considering the problem of obtaining dynamic dancing on the bipedal robot AMBER 2 (see Fig. 1).

Robotic dancing has been achieved in the past by copying the movements of human motions through their realization as trajectories that ensure static stability [4],[14]. Robotic dancing has also been utilized in the context of social interaction [12], where special emphasis was given to synchronizing the rhythmic movements with the music. But these works were mainly focused on maintaining the stability of the robot while realizing dancing for the purposes of entertainment. With the goal of placing more emphasis on establishing dynamically stable motion behaviors that strictly satisfy time constraints (tempo of a music), this paper presents a methodology of composing dynamical systems to yield meta-dynamical systems. In particular, different poses are represented as the vertices of a directed graph and, according to the edges of this graph, dynamic transitions are created that connect these poses. To achieve this dynamic behavior, an optimization problem is presented for generating dynamic transitions through methods motivated by human-inspired control [2, 3, 20, 21]. In particular, *virtual constraints* are considered that create a low-dimensional representation of the robot through *zero dynamics* [19]. These yield desired trajectories of the robot (parameterized by the phase variable) that can be therefore designed to dynamically transition between robot poses. Creating a low dimensional representation facilitates the ease of constraining the timing of these behaviors with a simple manipulation of a phase variable (position of the hip). The end result is a methodology for dynamically composing behaviors, designed specifically with a view toward robotic dancing.

The paper starts with a discussion of modeling and control of AMBER 2 in Section 2. Two phases, single support (SS) and double support (DS), are considered and described. A sequence of poses is formulated along with corresponding desired transitions between these poses. These are discussed in Section 3 along with dynamic transitions which are designed through virtual constraints, with the end result being

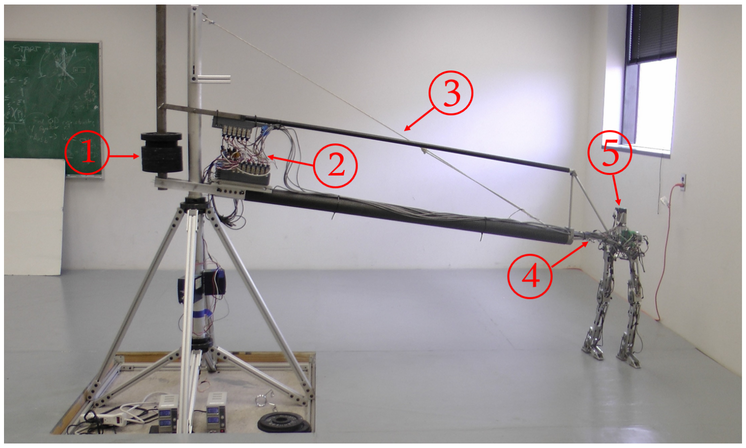

Fig. 2: AMBER 2 with the boom and electronics. The boom restricts motion to the sagittal plane. As shown in the figure: **(1)** Counterweight used to balance the boom around the pivot, **(2)** Controller module where the walking algorithm is running, **(3)** The boom, **(4)** Boom support structure which keeps the torso horizontal by using a parallel four-bar linkage mechanism, **(5)** The bipedal robot AMBER 2.

a meta-dynamical system for dancing. Finally, to practically implement these behaviors on AMBER 2, Section 4 describes how desired angles and angular velocities are reconstructed from the zero dynamics through a novel reconstruction process. The dynamic transitions are synchronized with the music tempo through the parameterization of time used to define the virtual constraints. The end result is the experimental realization of dynamic robotic dancing on AMBER 2 (a video of the dancing can be found at [1]).

2 AMBER 2 Model and Control

This section will provide a short description of the bipedal robot used, AMBER 2, to realize dynamic dancing. This section will also show the control law used for tracking the desired angles and velocities. AMBER 2 is a 2D bipedal robot with seven links (two calves, two thighs, two feet and a torso, see Fig. 1. AMBER 2 is the second generation and an expansion upon its predecessor, the non-footed (point feet) bipedal robot, AMBER 1 (see [20]). Each of the joints are actuated by brushless DC (BLDC) motors. In addition, the motion of AMBER 2 is restricted to the sagittal plane via a boom Fig. 2. The boom is fixed rigidly to a rotating mechanism, which allows the biped to walk in a circle with minimum friction. In addition, counterweights are provided to cancel the weight on the robot due to the boom. The controller modules are remotely connected to the stationary power supply with the help of slip rings located below the pivot.

2.1 Robot Dynamics

Due to the changes of contact points on the foot throughout the course of the dance, generalized coordinates are naturally used to characterize the robot. Specifically, the configuration space, $Q \in \mathbb{R}^n$ is represented in coordinates as $\theta = \{\psi_0, \theta_b\}$, where the extended coordinate $\psi_0 \in \mathbb{R}$ represents the rotation angle of the body fixed frame with respect to a fixed inertial frame R_0; here $\theta_b = [\theta_{sa}, \theta_{sk}, \theta_{sh}, \theta_{nsh}, \theta_{nsk}, \theta_{nsa}]^T$, $\theta_b \in \mathbb{R}^b$ denotes the body coordinates of the robot as shown in Fig. 1. Note that the translational coordinates p_x, p_z are also shown in Fig. 1, which are not considered in the dynamics since the stance toe is assumed to be pinned to ground throughout the course of dancing. For AMBER 2, $n = 7$, $b = 6$, i.e., $\theta_b \in \mathbb{R}^6$ and $\theta \in \mathbb{R}^7$.

Continuous Dynamics. The Lagrangian dynamics for this n-DOF robot is obtained as:

$$M(\theta)\ddot{\theta} + H(\theta, \dot{\theta}) = Bu, \tag{1}$$

with the notations $M \in \mathbb{R}^{n\times n}$ is the mass inertia matrix, $H \in \mathbb{R}^n$ is obtained from Coriolis, Centrifugal and gravity forces apparent from the standard EOM for rigid bodies. $u \in \mathbb{R}^k$ is the torque input with k the number of inputs, and $B \in \mathbb{R}^{k\times k}$ is the mapping from torque to joints. For AMBER 2, $k = 6$.

With the multiple foot behaviors that can be realized, we know that the feet cannot go below ground. The dynamics need to be realized through the use of holonomic constraints which constrain both heel and toe of the non-stance foot whenever they are in contact with ground. These holonomic constraints are enforced in the following manner.

$$M(\theta)\ddot{\theta} + H(\theta, \dot{\theta}) = Bu + J_{sh}^T F_{sh} + J_{nst}^T F_{nst} + J_{nsh}^T F_{nsh}, \tag{2}$$

where $J_i(\theta)$ is the Jacobian of specific contact points $i \in \{sh, nst, nsh\}$ corresponding stance heel, non-stance toe and non-stance heel respectively. $F_i(\theta, \dot{\theta})$, which are the reaction forces due to the holonomic constraints, are defined for each domain based on the contact conditions of the heel and toe. Note that $F_i = 0$ if there is no contact with ground. F_i can be explicitly derived from the states x and the controller u by differentiating the holonomic constraints twice. The details are omitted here and can be found in [13]. If $F_{nst} = 0$, $F_{nsh} = 0$ and $F_{sh} > 0$, then a fully actuated condition of the robot is realized.

Relabeling. If the robot takes a step, i.e., after the non-stance leg swings forward and hits the ground, it is convenient to swap the stance and non-stance legs so that the same motion primitive can be realized without having the need to change the controller for the robot. Therefore, at the end of every step relabeling of the angles are done which is considered a discrete transition in a formal model.

2.2 Control

Since the objective is to achieve dancing, a convenient step is to make the joint angles track a set of trajectories. We would like to generalize this by picking a vector of l functions of joint angles, referred to as actual outputs y^a, which we want to track termed the desired outputs y^d. The objective is to drive the error $y = y^a - y^d \to 0$. These outputs are also termed *virtual constraints* in [19]. The outputs are picked such that they are relative degree two outputs. In other words, y^a will be functions of joint angles, and not angular velocities.

We first introduce the actual set of outputs (virtual constraints) which are *independent* (as motivated by [2]): the linearized hip position, i.e., linearization of the horizontal hip position (calculated from calf length L_c and thigh lenghth L_t) w.r.t. the stance toe of the robot:

$$\delta p_{hip} = -(L_c + L_t)(\psi_0 + \theta_{sk} + \theta_{sa}) - L_t \theta_{sk}; \tag{3}$$

the stance ankle angle, θ_{sa}; the stance knee angle, θ_{sk}; the non-stance knee angle, θ_{nsk}; the hip angle, $\theta_{hip} = \theta_{nsh} - \theta_{sh}$; the torso angle, $\theta_{tor} = \psi_0 + \theta_{sa} + \theta_{sk} + \theta_{sh}$; and the non-stance foot angle, $\theta_{nsf} = \psi_0 + \theta_{sa} + \theta_{sk} + \theta_{sh} - \theta_{nsh} - \theta_{nsk} - \theta_{nsa}$.

We now introduce the Canonical Walking Function (CWF) which was first introduced in [2] to realize human-like walking in robots [20]. The CWF is given by:

$$\begin{aligned} y_{cwf}(t,\alpha) = e^{-\alpha_4 t}(\alpha_1 \cos(\alpha_2 t) + \alpha_3 \sin(\alpha_2 t)) + \ldots \\ \alpha_5 \cos(\alpha_6 t) + \frac{\alpha_2 \alpha_4 \alpha_5}{\alpha_2^2 + \alpha_4^2 - \alpha_6^2} \sin(\alpha_6 t) + \alpha_7. \end{aligned} \tag{4}$$

This CWF will be used to formulate our desired outputs with the parameters α dictating the shape of the trajectory, but it is useful to establish a relationship between time and the linearized hip position through a parameterization:

$$\tau(\theta) = (\delta p_{hip}(\theta) - \delta p_{hip}(\theta^+))/v_{hip}, \tag{5}$$

which relates the hip position and time. v_{hip} is the hip velocity. In other words, the robot moving forward can be seen as increasing hip position or an increase in time. Similarly, the robot moving backward can be seen as hip position reducing or the parameterization of time going in reverse. Note that θ^+ represents the robot configuration at the beginning of one step which can be defined such that parameterized time is zero at initial hip position. This parameterization can then be utilized to directly get the initial configuration of the robot from the parameters α which helps in reducing computation of the trajectory optimization parameters (see [2]).

Single Support and Double Support. There are two types of phases which will be considered in the paper, single support SS (when one foot is flat on ground) and double support DS (when both the feet are always on ground). There are other phases like underactuation where only the stance toe is on the ground, which also can be modeled but are more complicated to analyze and are therefore omitted from the paper. Depending on the contact conditions being enforced, we get control systems associated with the single support and double support phases, denoted by $(f_{\mathrm{SS}}, g_{\mathrm{SS}})$ and $(f_{\mathrm{DS}}, g_{\mathrm{DS}})$, respectively (see [21]).

Single Support. In the single support phase, the foot angle $\psi_0 = 0$, and the non-stance foot is always above ground. Picking only the base coordinates θ_b, $l = 5$ desired outputs,

$y^d_{\mathrm{SS}} : \mathbb{R}^6 \to \mathbb{R}^5$ and 5 actual outputs, $y^a_{\mathrm{SS}} : \mathbb{R}^6 \to \mathbb{R}^5$ are considered:

$$y^a_{\mathrm{SS}}(\theta_b) = \begin{bmatrix} \theta_{sk} \\ \theta_{nsk} \\ \theta_{hip} \\ \theta_{tor} \\ \theta_{nsf} \end{bmatrix}, \quad y^d_{\mathrm{SS}}(\tau(\theta), \alpha_{\mathrm{SS}}) = \begin{bmatrix} y_{cwf}(\tau(\theta), \alpha_{sk}) \\ y_{cwf}(\tau(\theta), \alpha_{nsk}) \\ y_{cwf}(\tau(\theta), \alpha_{hip}) \\ y_{cwf}(\tau(\theta), \alpha_{tor}) \\ y_{cwf}(\tau(\theta), \alpha_{nsf}) \end{bmatrix}, \tag{6}$$

where τ is a function of the configuration θ, as defined in (5), and the desired outputs are functions of θ and $\alpha_{\mathrm{SS}} = [\alpha_{sk}, \alpha_{nsk}, \alpha_{hip}, \alpha_{tor}, \alpha_{nsf}]^T$. Therefore, the desired trajectories are a function of $(v_{hip}, \alpha_{\mathrm{SS}}) \in \mathbb{R}^{36}$. It is also important to note that the actual output vector y^a_{SS} is a linear function of the angles: $y^a_{\mathrm{SS}} = H_{\mathrm{SS}}\theta$, with $H_{\mathrm{SS}} \in \mathbb{R}^{5\times 6}$ being the transformation matrix. Since, $\psi_0 = 0$, $\theta = [0, \theta_b^T]^T$ for single support phase.

The objective of the controller is to drive the outputs $y_{\mathrm{SS}} = y^a_{\mathrm{SS}} - y^d_{\mathrm{SS}}$ to zero, i.e., $y_{\mathrm{SS}} \to 0$. This can be achieved by using a feedback linearizing controller (see [17]), which involves using the model of the robot. Due to inaccuracies in the model parameters and difficulty in identification, we could instead use a simpler controller, e.g. PD controller, which does not guarantee convergence to zero, but will ensure minimum tracking error provided sufficient gains are used. Firstly, we define the following coordinate[1]:

$$\xi_{\mathrm{SS}} = -(L_c + L_t)(\theta_{sa} + \theta_{sk}) - L_t\theta_{sk} = C_{\mathrm{SS}}\theta_b, \tag{7}$$

where $C_{\mathrm{SS}} \in \mathbb{R}^{1\times 6}$ is the row vector of constants. Note that ξ_{SS} derived here is same as (3) with ψ_0 omitted. If the controller used is expected to achieve zero tracking error, i.e., $y^a_{\mathrm{SS}} - y^d_{\mathrm{SS}} = 0$, then the desired joint angles $\dot{\theta}^d_{\mathrm{SS}} \in \mathbb{R}^6$ and velocities $\theta^d_{\mathrm{SS}} \in \mathbb{R}^6$ of the robot for the single support phase which realize this equality can be obtained as:

$$y^a_{\mathrm{SS}} = y^d_{\mathrm{SS}} \qquad \Longrightarrow \qquad \begin{bmatrix} C_{\mathrm{SS}} \\ H_{\mathrm{SS}} \end{bmatrix} \theta^d_{\mathrm{SS}} = \begin{bmatrix} \xi_{\mathrm{SS}} \\ y^d_{\mathrm{SS}} \end{bmatrix}. \tag{8}$$

Therefore, the desired angle configuration and the angular velocities are:

$$\theta^d_{\mathrm{SS}} = \begin{bmatrix} C_{\mathrm{SS}} \\ H_{\mathrm{SS}} \end{bmatrix}^{-1} \begin{bmatrix} \xi_{\mathrm{SS}} \\ y^d_{\mathrm{SS}} \end{bmatrix}, \quad \dot{\theta}^d_{\mathrm{SS}} = \begin{bmatrix} C_{\mathrm{SS}} \\ H_{\mathrm{SS}} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ \frac{\partial y^d_{\mathrm{SS}}}{\partial \tau} \end{bmatrix} \frac{\dot{\xi}_{\mathrm{SS}}}{v_{hip}}. \tag{9}$$

The PD controller can thus be defined as:

$$u^{pd}_{\mathrm{SS}} = -K^p_{\mathrm{SS}}(\theta_b - \theta^d_{\mathrm{SS}}) - K^d_{\mathrm{SS}}(\dot{\theta}_b - \dot{\theta}^d_{\mathrm{SS}}), \tag{10}$$

where K^p_{SS}, K^d_{SS} are the proportional and derivative gains respectively.

Double Support. The double support phase adds extra constraints to the robot like friction, pinning conditions (holonomic constraints [13]) and normal forces, which will constrain the dynamics of the robot. The actual and desired outputs for the robot can be

[1]Note that the motivation for this coordinate is given by Partial Zero Dynamics as considered in [3].

defined similar to (6). In the double support phase, the stance ankle angle θ_{sa} is added as:

$$y^a_{\mathrm{DS}} = \begin{bmatrix} \theta_{sa} \\ y^a_{\mathrm{SS}} \end{bmatrix}, \quad y^d_{\mathrm{DS}} = \begin{bmatrix} y_{cwf}(\tau(\theta), \alpha_{sa}) \\ y^d_{\mathrm{SS}} \end{bmatrix}, \tag{11}$$

where $y^a_{\mathrm{DS}} : \mathbb{R}^7 \to \mathbb{R}^6$, $y^d_{\mathrm{DS}} : \mathbb{R}^7 \to \mathbb{R}^6$, $\alpha_{\mathrm{DS}} = [\alpha_{sa}, \alpha^T_{\mathrm{SS}}]^T$, with $\alpha_{\mathrm{DS}} \in \mathbb{R}^{43}$. The actual outputs can also be written as $y^a_{\mathrm{DS}} = H_{\mathrm{DS}}\theta$, with $H_{\mathrm{DS}} \in \mathbb{R}^{6\times 7}$. Since the actuators have the potential to fight each other due to overactuation, a feedback linearizing controller will not necessarily yield exponential convergence:$y^a_{\mathrm{DS}} - y^d_{\mathrm{DS}} \to 0$. However, since the objective of the controller during the double support phase is to achieve dynamic behaviors in the robot to realize a dancing sequence, the convergence of the outputs to zero is ignored. Similar to (3), the following coordinate is defined:

$$\xi_{\mathrm{DS}} = -(L_c + L_t)(\psi_0 + \theta_{sk} + \theta_{sa}) - L_t\theta_{sk} = C_{\mathrm{DS}}\theta, \tag{12}$$

where $C_{\mathrm{DS}} \in \mathbb{R}^{1\times 7}$ is the row vector of constant terms. Having obtained the expression for ξ_{DS}, the desired joint angles and velocities can be defined as:

$$\theta^d_{\mathrm{DS}} = \begin{bmatrix} C_{\mathrm{DS}} \\ H_{\mathrm{DS}} \end{bmatrix}^{-1} \begin{bmatrix} \xi_{\mathrm{DS}} \\ y^d_{\mathrm{DS}} \end{bmatrix}, \quad \dot{\theta}^d_{\mathrm{SS}} = \begin{bmatrix} C_{\mathrm{DS}} \\ H_{\mathrm{DS}} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ \frac{\partial y^d_{\mathrm{SS}}}{\partial \tau} \end{bmatrix} \frac{\dot{\xi}_{\mathrm{DS}}}{v_{hip}}. \tag{13}$$

With K^p_{DS}, K^d_{DS} as the proportional and derivative matrices, the PD controller is:

$$u^{pd}_{\mathrm{DS}} = -K^p_{\mathrm{DS}}(\theta - \theta^d_{\mathrm{DS}}) - K^d_{\mathrm{DS}}(\dot{\theta} - \dot{\theta}^d_{\mathrm{DS}}), \tag{14}$$

Note that, both these matrices are not square since $u^{pd}_{\mathrm{DS}} \in \mathbb{R}^6$. In fact, the first row and column of the gain matrices are zeros.

2.3 Configuration Zero Dynamics

For the single support phase, with the feedback linearizing controller (see [17]) being applied, the outputs y_{SS} are exponentially driven to zero. The control system $(f_{\mathrm{SS}}, g_{\mathrm{SS}})$ will exhibit zero dynamics. In other words, we have the following restriction of the dynamics to the zero dynamics surface given by:

$$\mathbb{Z}_{SS} = \{(\theta, \dot{\theta}) : y_{\mathrm{SS}}(\theta_b) = 0, L_f y_{\mathrm{SS}}(\theta_b, \dot{\theta}_b) = 0, \psi_0 = 0, \dot{\psi}_0 = 0\}. \tag{15}$$

This restriction of the dynamics to a surface enables us to connect different motion primitives of the single support phase in a way such that the transition between domains occurs without change of y_{SS} [2]. In other words, the transition will be smooth. Motivated by the desire to relax the derivative condition in (15), we introduce the notion of *configuration zero dynamics* defined to be:

$$\mathbb{CZ}_{SS} = \{(\theta, \dot{\theta}) : y_{\mathrm{SS}}(\theta_b) = 0, \psi_0 = 0\}. \tag{16}$$

[2]Note: The construction of the PD controller defined in (7) through (10) is based on the notion that the desired angles and velocities: $(\theta^d_{\mathrm{SS}}, \dot{\theta}^d_{\mathrm{SS}}) \in \mathbb{Z}_{\mathrm{SS}}$.

For the double support phase, due to geometric constraints, it is not possible to realize zero dynamics. But, it is possible to connect a motion primitive with the single support phase due to the choice of controller. The concept of configuration zero dynamics plays an important role in the context of dancing, since when switching between a large collection of surfaces, if the configuration zero dynamics constraints are ensured, this allows for a transition without a sudden change in desired angles. If the Zero Dynamics constraints are ensured, then it allows for a smooth transition (jerk free) from one desired trajectory to other. This will be utilized in the next section through the composition of Configuration Zero Dynamic surfaces to allow for minimum jerk transitions between domains. In addition, this constraint will be independent of the speed in which the transition is executed.

3 Meta-Dynamical Systems

To achieve dancing, the primary goal is to connect trajectories, i.e., desired outputs y^d, for each motion primitive; that is, we wish to *compose* dynamical systems. To this end, this section will present the notion of *meta-dynamical* systems which give a formalism to the notion of composition. We begin by considering different poses of the robot that will be connected through dynamic transitions.

Pose. A *pose* of a robot is a configuration θ, which is intended to be realized in the robot. In other words, a pose is just a captured frame of a robot while in motion. For example, a robot with hip forward and low and both feet flat is a crouch, and is considered a pose of the robot. There are several possible poses that the robot can assume. If the stance toe is always on ground (since jumping is not considered), the three remaining points (non-stance toe and heel and stance heel) can be either in contact or not. Therefore, there are eight possible general cases for pose generation. Accordingly, we will consider: front heel lift (FHL), front toe lift (FTL), back heel lift (BHL), all feet flat on ground (FF), swing (S) with stance foot being flat on ground, double heel lift (DHL), front toe and back heel lift (FTBH), and underactuation (UA) with only stance toe in contact with ground. All the eight generic poses are shown in Fig. 3.

It is important to note that there could be more than one type of Back Heel Lift, Front Toe Lift, and other combinations as well. In other words, there are more than eight types of poses. For example, we could have two different kinds of flat footed poses, where the vertical hip position is high for one and low for the other. This will be discussed further in Section 4 where the poses of dancing on AMBER 2 are introduced. If a set of poses $\theta_1, \theta_2, \ldots, \theta_i$ is considered, then dancing is achieved by just executing dynamic transitions between these poses.

Dynamic Transition. Let $x = (\theta^T, \dot{\theta}^T) \in \mathbb{R}^{2n}$, and $\dot{x} = f(x)$ be a dynamical system Let $\Phi(t;x_0)$ be the solution to $\dot{x} = f(x)$ at time $t \in \mathbb{R}$ with initial condition x_0, and let π_θ be the canonical projection $\pi_\theta(x) = \theta$.

Definition 1. *A dynamic transition between two poses, θ_0 and θ_f, is a solution $\Phi(t;x_0)$ to the dynamical system $\dot{x} = f(x)$ such that there exists a point $x_0 \in \mathbb{R}^{2n}$ and a time $t_f \geq 0$ with $\pi_\theta(\Phi(0;x_0)) = \theta_0$ and $\pi_\theta(\Phi(t_f;x_0)) = \theta_f$*

This definition allows us to formally introduce meta-dynamical systems:

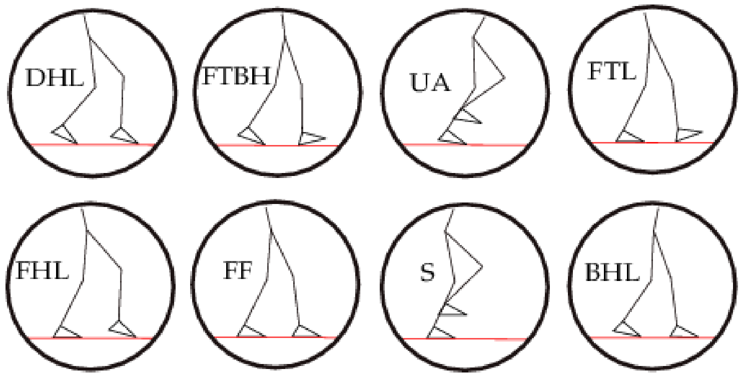

Fig. 3: Eight generic poses of a robot based upon possible contact points.

Definition 2. *The meta-dynamical system is defined as a tuple:*

$$\mathbb{M} = (\Gamma, \mathbb{P}, \mathbb{T}), \tag{17}$$

- Γ *is a directed graph given as:* $\Gamma = (\mathbb{V}, \mathbb{E})$, *where* $\mathbb{V}$ *is the set of vertices describing desired poses realizable on the robot, and* $\mathbb{E}$ *represents transitions between these poses. We denote the source and target of an edge* $e \in \mathbb{E}$ *by* $\text{source}(e) \in \mathbb{V}$ *and* $\text{target}(e) \in \mathbb{V}$.
- $\mathbb{P}$ *is the set of poses given by:* $\mathbb{P} = \{\mathbb{P}_v\}_{v\in\mathbb{V}}$, *where* $\mathbb{P}_v = \theta_v \in \mathbb{R}^n$.
- $\mathbb{T}$ *is the set of dynamic transitions:* $\mathbb{T} = \{\mathbb{T}_e\}_{e\in\mathbb{E}}$, *where* $\mathbb{T}_e = \Phi_e$ *is the dynamic transition between the poses* $\theta_{source(e)}$ *and* $\theta_{target(e)}$.

Creating dynamic transitions. Suppose we want to construct a meta-dynamical system. Assume we are given a directed graph Γ with the set of poses $\mathbb{P}$. Using the constructions given in Section 2.2, we can construct a set of dynamic transitions $\mathbb{T}$. Given that the desired outputs y^d are obtained through canonical walking functions as described in (6) and (11), we propose the following optimization problem for creating a dynamic transition $\mathbb{T}_e$ for a particular edge $e \in \mathbb{E}$:

$$(v^*_{hip}, \alpha^*) = \underset{(v_{hip},\alpha)\in\mathbb{R}^d}{\text{argmin}} \ \text{Cost}_{\text{D}}(v_{hip}, \alpha, v^r_{hip}, \alpha^r) \tag{18}$$

$$\text{s.t.} \begin{bmatrix} y^d_{\text{Phase}}(0,\alpha) \\ y^d_{\text{Phase}}(\tau_{max},\alpha) \end{bmatrix} = \begin{bmatrix} H_{\text{Phase}}\theta_0 \\ H_{\text{Phase}}\theta_f \end{bmatrix}, \tag{CZD}$$

where v^r_{hip}, α^r are the reference parameters, $\text{Phase} \in \{\text{SS}, \text{DS}\}$ denotes whether the robot is in single support or double support phase, and τ_{max} is the time at the end of the step which is computed in the following manner:

$$\tau_{max} = (C_{\text{Phase}}\theta_f - C_{\text{Phase}}\theta_0)/v_{hip}, \tag{19}$$

with v_{hip} being the hip velocity, as introduced in (5). The cost of *dancing* (or objective function), Cost_{D}, is the least squares error relative to reference data:

$$\text{Cost}_{\text{D}} = \sum_i [y_d(t[i],\alpha) - y_d(t[i],\alpha^r)]^T [y_d(t[i],\alpha) - y_d(t[i],\alpha^r)], \tag{20}$$

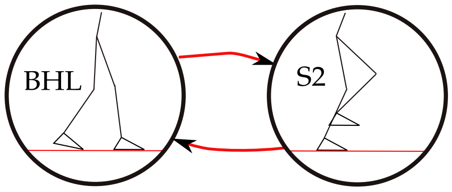

Fig. 4: Figures showing the initial pose (left) and the final pose (right) for crouch respectively. The red arrows are the edges.

where the reference used is either obtained from human data which have discrete heel toe behavior, or obtained from the formerly established walking gaits which were provably stable and experimentally realized on robots (see [20],[21]). Note that in some of the transitions for dancing where there was no reference trajectories, a zero cost will be used. The defining aspect of this paper is using the constraints (CZD), which realizes configuration zero dynamics and is thus instrumental in being able to compose different motion primitives to form a meta-dynamical system. This follows from the fact that the end result of the optimization is a dynamic transition; for example, if Phase $=$ SS, the parameters obtained from the optimization (v^*_{hip}, α^*), utilized in the feedback linearizing controller and applied to the control system $(f_{\mathrm{SS}}, g_{\mathrm{SS}})$, yields a dynamic transition.

Example: dynamic leg swing. To illustrate meta-dynamical systems, we will consider a simple example consisting of two poses: back heel lift (BHL) and swing (S) (see Fig. 4). Due to space constraints, it is not possible to show how the optimization problem was formulated for each and every transition in case of dancing. Therefore, we consider a specific example of transition from pose $\mathbb{P}_{BHL}$ to $\mathbb{P}_{S2}$, i.e., from back heel lift to swing. The two poses with the transition is depicted separately in Fig. 4. We can accordingly define the meta-dynamical system in the following manner:

Discrete structure and poses. The graph is given by:

$$\Gamma = (\mathbb{V}, \mathbb{E}), \quad \mathbb{V} = \{\mathrm{S}, \mathrm{BHL}\}, \quad \mathbb{E} = \{\mathrm{S} \to \mathrm{BHL}, \mathrm{BHL} \to \mathrm{S}\}. \tag{21}$$

The set of poses is given by: $\mathbb{P} = \{\mathbb{P}_v : v \in \mathbb{V}\}$. The set of transitions is given by: $\mathbb{T} = \{\mathbb{T}_e : e \in \mathbb{E}\}$. The edges are depicted by the arrows shown in Fig. 4. Note that the above example can have more than 2 edges depending on the how the transitions between poses are obtained. We will now introduce the optimization problem which realizes the dynamic transitions, $\mathbb{T}_e$, from one pose to the other.

Dynamic transitions. Having obtained the desired angles and angular velocities (9),(13), we can now discuss the transition optimization which yields the motion primitive for the swing action.

The cost for the optimization was evaluated by obtaining the least squares fit with the multi-domain walking trajectory obtained on AMBER2 as found in [21]. The time parameter was picked such that only the swing portion of [21] was considered for the cost. In other words, the value τ was constrained in the optimization to match the reference trajectories. Additional constraints, like sufficient foot clearances on ground, were imposed throughout the step. The knee angle was also constrained to be within a cer-

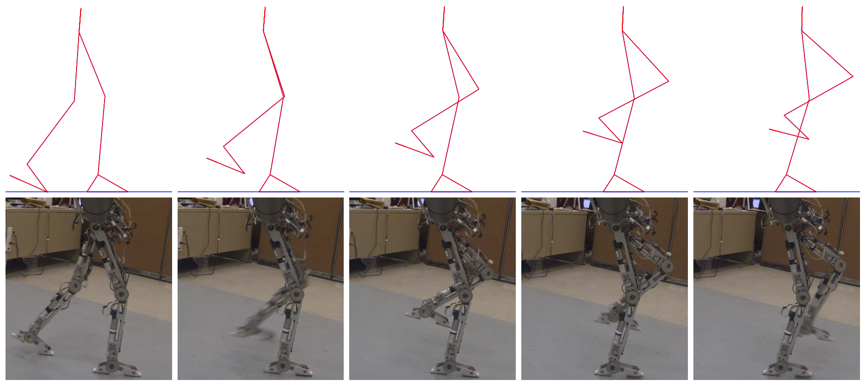
Fig. 5: Tiles of a leg swing behavior consisting of a transition from back heel lift to swing pose. The top tiles illustrate the behavior of the robot achieved in simulation, and the bottom tiles show the same behavior realized experimentally on AMBER 2.

tain limit to ensure low torque is utilized. That is, the final optimization (with physical constraints) is given by:

$$\begin{aligned} (v^*_{hip}, \alpha^*) = \underset{(v_{hip},\alpha)\in\mathbb{R}^{36}}{\operatorname{argmin}} \quad & \mathrm{Cost_D}(v_{hip}, \alpha, v^r_{hip}, \alpha^r) \\ \text{s.t.} \quad & \text{(CZD)} \\ & \tau_{max} < 0.4 \\ & min(h_{nst}) > 0 \\ & min(\theta_{nsk}) > 0, \end{aligned} \tag{22}$$

where the resulting optimal solution yields the parameters for the desired trajectories represented through the outputs. Since the constraints are non-linear, the optimization method used is active-set through MATLAB. If this optimization were to be done online, a learning technique like [8] could have been used instead. But, this does not yield dynamically stable trajectories which are necessary for maintaining balance. When this is applied on the robot through trajectory reconstruction (10), the swing of the non-stance leg is observed as shown in Fig. 5. The same controller was also applied on AMBER 2, both in simulation and experiment, with tiles of the resulting behavior shown in Fig. 5.

4 Dynamic Robotic Dancing on AMBER 2

This section presents the process of realizing dynamic dancing in AMBER 2 by using the methods introduced in this paper. We will not consider the cases UA, DHL, FTBH from Fig. 3 since they require higher torque and are relatively difficult to realize in the robot. Therefore, we will consider the remaining five generic cases of the feet behavior for generating the pose. We will consider three types of front heel lift: FHL1, FHL2,

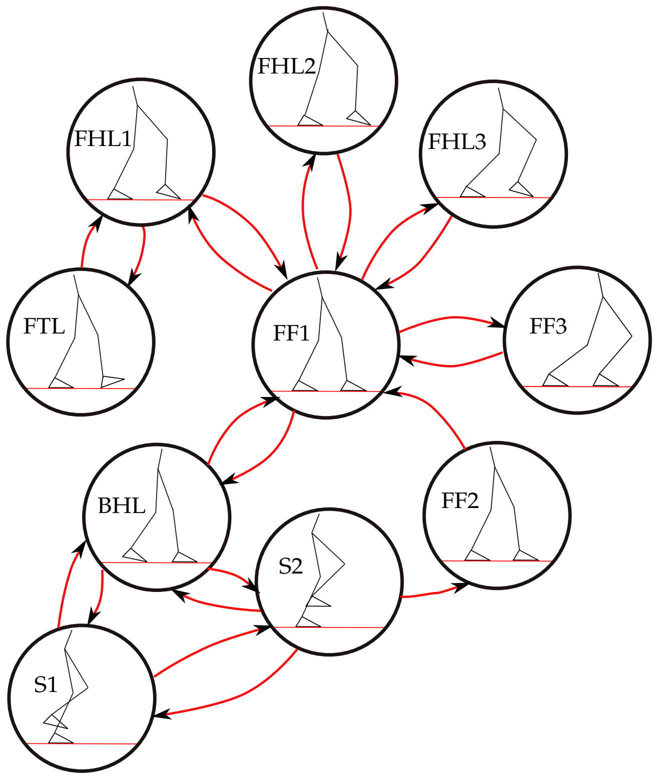

Fig. 6: Oriented graph for the meta-dynamical system considered for AMBER 2 in order to obtain dynamic dancing.

FHL3, one front toe lift: FTL, three types of flat-footed poses: FF1, FF2, FF3, two types of swing poses: S1,S2, and finally one back heel lift pose: BHL. All ten poses are shown in Fig. 6. The end result is an oriented graph $\Gamma = (\mathbb{V}, \mathbb{E})$, where:

$$\mathbb{V} = \{\text{FHL1}, \text{FHL2}, \text{FHL3}, \text{FTL}, \text{FF1}, \text{FF2}, \text{FF3}, \text{S1}, \text{S2}, \text{BHL}\}, \tag{23}$$

and $\mathbb{E}$ is the set of red arrows in Fig. 6.

For generating the dynamic transition between poses, the optimization (18) was accordingly solved. Since it was not necessary to optimize trajectories to transition from every pose to every other pose, we considered 20 edges (or optimized dynamic transitions) which satisfied configuration zero dynamic (CZD) constraints. Therefore, we consider the set of edges as shown in Fig. 6, with the resulting dynamic transition: $\mathbb{T} = \{\mathbb{T}_e : e \in \mathbb{E}\}$, obtained through the optimization in (18). Note, additional constraints were also implemented in the optimization to realize different behaviors varying from constraining the angles, to allowing sufficient foot clearance, to constraining the velocities, to constraining final parameterized time: τ_{max}.

Synchronizing with music. The particular method employed to synchronize the behaviors of the robot with the music is to utilize the parameterization of time (5) to change

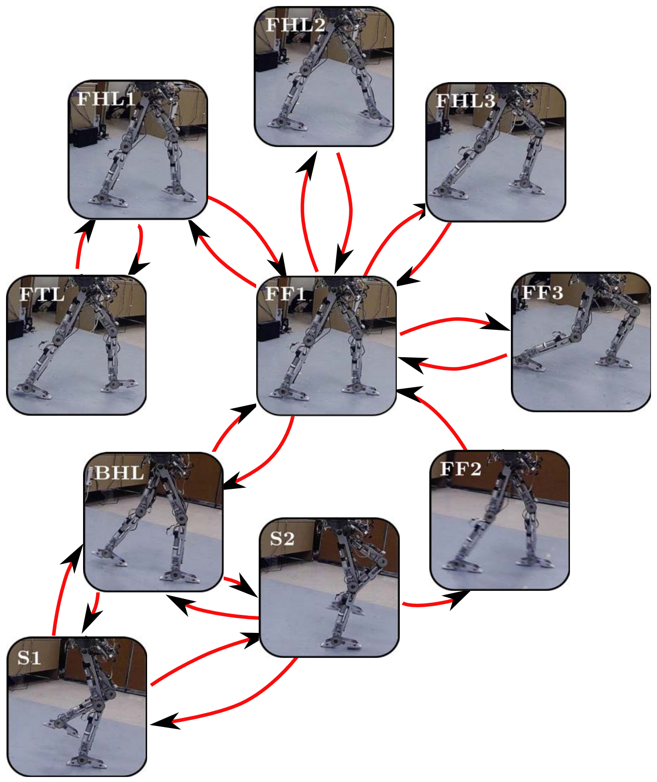

Fig. 7: Experimental realization of the meta-dynamical system on AMBER 2, resulting in dynamic robotic dancing (which can be viewed at [1]).

the hip position of the robot within a prespecified tempo period. Since τ is a direct function of the hip position, a change in τ causes a corresponding change in desired trajectories of the robot (as represented by the outputs parameterized by τ) resulting in synchronization between the beats of the music and the dynamic transitions. Dynamic programming methods as described in [7] are used to generate music tempo speed for a given song.

Sequence Design. To design a proper dancing sequence to the chosen music, the tempo period ΔT, which was obtained from the beats, is utilized as the fundamental period. For AMBER 2 dancing, each sequence input which executes the transition from one pose to the other is given in the following format:

$$\mathbb{S} = \{\alpha, \tau_{max}, n_{dance}, m_0, m_1, n_{freeze}, \text{Phase}, \text{Leg}\}, \tag{24}$$

where α is the set of parameters specifying the desired trajectory. The starting pose is specified by the time parameter, $m_0\tau_{max}$, and the ending pose is specified by $m_1\tau_{max}$. Note that τ_{max} is the maximum time parameter for current gait. n_{dance} is the tempo number specifying for how long AMBER 2 will transition for the current primitive,

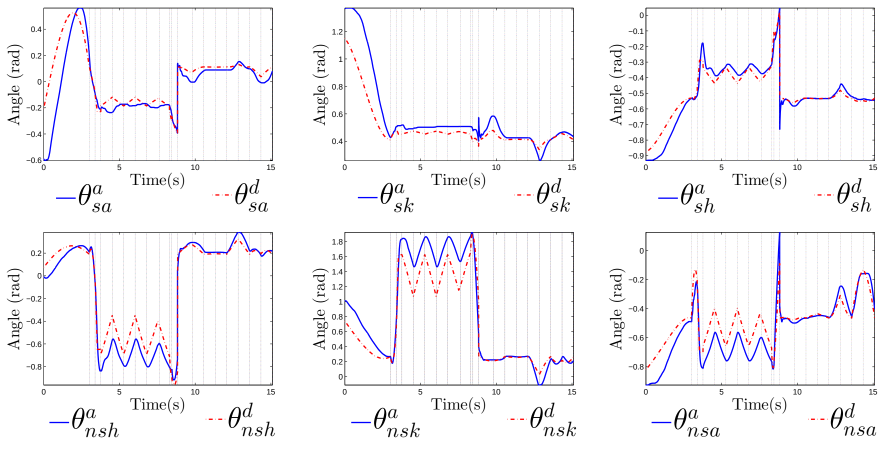

Fig. 8: Experimental data comparing the actual and desired angles for a sequence of steps extracted from a part of the dance sequence as realized on AMBER 2. The vertical dashed lines indicate end points of the transitions.

while n_{freeze} denotes the tempo number specifying for how long the robot will freeze at the end of the transition. Phase $\in \{\mathrm{SS}, \mathrm{DS}\}$ indicates the current phase of the robot, and Leg $\in \{\mathrm{Left}, \mathrm{Right}\}$ determines which leg is the stance leg.

The desired angles and velocities are obtained from (9) for SS, and (13) for DS with the time parameter τ being manually varied from $m_0\tau_{max}$ to $m_1\tau_{max}$ during the period $T_{dance} = n_{dance}\Delta T$ seconds. It is important to note here, since the dancing duration T_{dance} is specified differently than the original time duration, the transition speed changes. Accordingly, the hip velocity should also be scaled as

$$v_{dance} = T_{dance} v_{hip}/((m_1 - m_0)\tau_{max}), \tag{25}$$

with v_{hip} is the designed hip velocity encoded in the motion primitive α. To obtain the torque controller used in the robot, v_{hip} in (9),(13) is replaced with v_{dance} and the desired angles, velocities are accordingly computed. Having the desired angle and velocity of the robot, the torque controller is obtained from (10) or (14) based on the value of Phase given by the sequence S. At the end of each sequence, the robot can also be freezed for the time $T_{freeze} = n_{freeze}\Delta T$ seconds. More details about the algorithm used is shown in Algorithm 1.

Control Implementation and Results. On the hardware level, the controller for AMBER 2 is implemented on two levels: a high level controller, which is realized in Real-Time (RT) with the pseudocode running in RT as shown in Algorithm 1; and a low level controller, which is realized by FPGA for interfacing with the hardware modules. Implementing the proposed algorithm in the robot resulted in dynamically stable dancing accurately synchronized with the tempo of the music. Fig. 8 shows the comparison between desired and joint angle trajectories, Fig. 7 shows the configuration of the robot at different instances of time during the dance sequence. The video of AMBER 2 dancing is shown in [1].

Algorithm 1 Real Time Module

```
Input: Encoder/motor status; AMBER 2 parameters: calf length $L_c$, thigh length $L_t$;
Input: Optimization parameters: $\delta p_{\mathrm{hip}}(\theta^+)$, $v_{hip}$, $\alpha$;
Input: Joint angles and angular velocities $\theta_a, \dot{\theta}_a$; PD controller gains: $K^p_{\mathrm{Phase}}$, $K^d_{\mathrm{Phase}}$;
Input: Dance sequence (see Fig. 6) in ($\alpha$, $\tau_{max}$, $n_{dance}$, $m_0$, $m_1$, $n_{freeze}$, Phase, Leg);
Output: Torque commands for FOC;
 1: Enable Motor Drives;
 2: repeat
 3:     Wait till all motor drives are Enabled
 4: until ( Drive-Status == Enable )
 5: while ( $\neg$ Stop-RT ) do
 6:     Based on specified stance foot, reform $\theta_a, \dot{\theta}_a$ from Left/Right to Stance/nonStance;
 7:     Read absolute real time $t$ and sequenceIndex;
 8:     if $0 \leq t \leq T_{dance} n_{dance}$ then
 9:         if $m_1 > m_0$ then
10:             $\tau_d = m_0 \tau_{max} + \tau_{max} \frac{t}{T_{dance} n_{dance}}$;
11:         else
12:             $\tau_d = m_0 \tau_{max} - \tau_{max} \frac{t}{T_{dance} n_{dance}}$;
13:         end if
14:     else
15:         $\tau_d = m_1 \tau_{max}$;
16:     end if
17:     Based on $\tau_d$, calculate $(\xi_{\mathrm{Phase}})$ using one of (7) or (12);
18:     Calculate $y^d_{\mathrm{Phase}}(\tau_d, \alpha)$, $\dot{y}^d_{\mathrm{Phase}}(\tau_d, \alpha)$ based on Canonical Walking Function (4);
19:     Calculate $v_{dance}$ based on the time duration $T_{dance}$;
20:     Based on Phase, apply trajectory reconstruction to get $(\theta_d,\ \dot{\theta}_d)$ with updated $v_{dance}$;
21:     Based on Phase, compute torque by choosing one of (10) or (14);
22:     Reform torque $u$ from Stance/nonStance to Left/Right and send it to FPGA;
23:     if $t \geq T_{walk} + T_{freeze}$ then
24:         sequenceIndex $+1$;
25:         Reset time clock;
26:     end if
27:     Log Data into Remote Desktop;
28: end while
29: Disable motor drives; Report errors and stop the Real Time VI;
```

Conclusions

This paper successfully showed how to achieve dynamically stable dancing in the bipedal robot AMBER 2 which is accurately synchronized with the music. The dance sequence is seen as a composition of motion behaviors with different poses and transitions tied together in the form of a meta-dynamical system. The dance was about 1.5 minutes long showing 10 poses and 20 transitions from one to the other. Tracking results also verified the method used. Future work involves implementing more complex behaviors like the underactuation, different surfaces and terrains, and also music with varying tempo.

References

1. Dynamic Robotic Dancing on AMBER 2. `http://youtu.be/IwR9XvojXWo`.
2. A. D. Ames. First Steps Toward Automatically Generating Bipedal Robotic Walking from Human Data. In *Robotic Motion and Control 2011*, volume 422. Springer, 2012.
3. A. D. Ames, E. A. Cousineau, and M. J. Powell. Dynamically stable robotic walking with NAO via human-inspired hybrid zero dynamics. In *Hybrid Systems: Computation and Control, 2012*, Beijing, China, 2012.
4. J-J Aucouturier. Cheek to Chip: Dancing Robots and AI's Future. *Intelligent Systems, IEEE*, 23(2):74–84, 2008.
5. M. Bennewitz, J. Pastrana, and W. Burgard. Active Localization of People with a Mobile Robot Based on Learned Motion Behaviors.
6. O. Brock, O. Khatib, and S. Viji. Task-Consistent Obstacle Avoidance and Motion Behavior for Mobile Manipulation. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 388–393. IEEE, 2002.
7. D. P. W. Ellis. Beat Tracking by Dynamic Programming. *Journal of New Music Research*, 36(1):51–60, 2007.
8. A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in NIPS*, pages 1523–1530. MIT Press, 2003.
9. X. Jiang and Y. Motai. Learning by Observation of Robotic Tasks using On-line PCA-based Eigen Behavior. In *Computational Intelligence in Robotics and Automation, 2005. CIRA 2005. Proceedings. 2005 IEEE International Symposium on*, pages 391–396. IEEE, 2005.
10. A. M. Johnson and D. E. Koditschek. Legged self-manipulation. *IEEE Access*, 1:310–334, May 2013.
11. O. Khatib, L. Sentis, J. Park, and J. Warren. Whole-Body Dynamic Behavior and Control of Human-Like Robots. *International Journal of Humanoid Robotics*, 1(01):29–43, 2004.
12. M. P. Michalowski, S. Sabanovic, and H. Kozima. A dancing robot for rhythmic social interaction. In *Proceedings of the ACM/IEEE International Conference on Human-robot Interaction*, HRI '07, pages 89–96, New York, NY, USA, 2007. ACM.
13. R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, 1994.
14. S. Nakaoka, A. Nakazawa, K. Yokoi, and K. Ikeuchi. Leg Motion Primitives for a Dancing Humanoid Robot. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 610–615. IEEE, 2004.
15. H. Park, A. Ramezani, and J. W. Grizzle. A Finite-State Machine for Accommodating Unexpected Large Ground-Height Variations in Bipedal Robot Walking. *Robotics, IEEE Transactions on*, 29(2):331–345, 2013.
16. M. J. Powell, H. Zhao, and A. D. Ames. Motion Primitives for Human-Inspired Bipedal Robotic Locomotion: Walking and Stair Climbing. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 543–549. IEEE, 2012.
17. S. S. Sastry. *Nonlinear Systems: Analysis, Stability and Control*. Springer, New York, 1999.
18. K. Sreenath, H. Park, I. Poulakakis, and J. W. Grizzle. A compliant Hybrid Zero Dynamics Controller for Stable, Efficient and Fast Bipedal Walking on MABEL. *The International Journal of Robotics Research*, 30(9):1170–1193, 2011.
19. E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris. *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, Boca Raton, 2007.
20. S. Nadubettu Yadukumar, M. Pasupuleti, and A. D. Ames. From Formal Methods to Algorithmic Implementation of Human Inspired Control on Bipedal Robots. In *Tenth International Workshop on the Algorithmic Foundations of Robotics*, Cambridge, MA, 2012.
21. H. Zhao, W. Ma, M. B. Zeagler, and A. D. Ames. Human-Inspired Multi-Contact Locomotion with AMBER2. In *International Conference on Cyber Physical Systems*, 2014.