Simulation-Based Design of Dynamic Controllers for Humanoid Balancing

Jie Tan Zhaoming Xie Byron Boots C. Karen Liu

***Abstract*— Model-based trajectory optimization often fails to find a reference trajectory for under-actuated bipedal robots performing highly-dynamic, contact-rich tasks in the real world due to inaccurate physical models. In this paper, we propose a complete system that automatically designs a reference trajectory that succeeds on tasks in the real world with a very small number of real world experiments. We adopt existing system identification techniques and show that, with appropriate model parameterization and control optimization, an iterative system identification framework can be effective for designing reference trajectories. We focus on a set of tasks that leverage the momentum transfer strategy to rapidly change the whole-body from an initial configuration to a target configuration by generating large accelerations at the center of mass and switching contacts.**

I. Introduction

A classical approach to optimal control problems in robotics is to compute a reference trajectory offline (e.g. via Differential Dynamic Programming) and then derive a local feedback control of perturbations (e.g. Linear-Quadratic-Gaussian control) around the reference trajectory. Unfortunately, when executing the reference trajectory on robotic hardware, this type of model-based approach tends to fail due to modeling assumptions and parameter settings. Various methods have been proposed to mitigate the issues of model inconsistency in order to transfer the model-based reference trajectories to the real world [1]–[4].

However, for an under-actuated biped performing a highly-dynamic task involving contact switches, the problems due to model inconsistency can be greatly amplified. Large body accelerations can magnify inaccurate inertia parameters, leading to an unstable reference trajectory beyond the range of perturbation recoverable by the feedback control. To make matters worse, the under-actuated system imposes Newton-Euler constraints on the optimal control problem while contact switches break the control space into fragmented feasible regions. As a result, a small modeling error can generate bifurcated consequences. For tasks with the above mentioned characteristics, manually designing a reference trajectory that succeeds in the real world is exceedingly challenging.

Classical system identification provides a framework to address the general problem of model inconsistency. Given an initial physical model, an optimal control policy is learned and applied to the hardware. If the results are different from those predicted by the current model, data are collected and used to improve the model. In this paper, we adopt existing system identification techniques and show that, with appropriate choices of model parameterization and control optimization methods, the iterative system identification framework can be effective for designing reference trajectories for highly dynamic, contact-rich tasks performed by under-actuated bipeds. Specifically, we propose a complete system that automatically designs reference trajectories that succeed in real world tasks after a very small number of real world experiments. Our system consists of three main components: a *physical simulator* that simulates the dynamics of the robot and its interaction with the environments, a *trajectory optimization* algorithm that searches for the optimal reference trajectory in the simulation, and a *simulation calibration* process to overcome the issues of model inconsistency.

System identification in last decades has focused on identifying parametric or non-parametric models for relatively simple dynamic systems. Widely used models include state transition functions [5], linear models [5], Gaussian processes [6], [7], and differential equations [8], [9]. However, it is challenging to employ these models to control a high dimensional humanoid robot to perform highly-dynamic tasks in contact-rich environments because classical system identification methods would require prohibitively large amount of data. In contrast, fewer studies have been conducted to identify parameters for modern physical simulators, such as DART [10] and Mujoco [11]. These simulators solve the governing differential equations with the linear-complementarity-program (LCP) contact model to accurately simulate high-dimensional dynamics and discontinuous contact forces. They provide a good model parameterization but leave many (simulation) parameters unidentified. In this paper, we use DART as the simulator and apply our simulation calibration algorithm to identify model parameters.

As a proof of concept, we focus on a set of tasks that rapidly change whole-body from an initial configuration to a target configuration by generating large acceleration at the center of mass and switching contacts. These transition tasks involve a control strategy called the *momentum transfer strategy* [12], in which rapid movements are used to maintain balance even when the center of mass is outside of the support polygon. For example, the stand-to-handstand task takes less than one second for our humanoid to complete.

II. Related Work

Transitioning a bipedal robot from the current pose to the target pose using a momentum strategy involves whole-

The authors are with College of Computing, Georgia Institute of Technology, USA. Email:{jtan34, zxie47}@gatech.edu and {bboots, karenliu}@cc.gatech.edu. This research is funded by NIH grant R01 114149, NSF EFRI-M3C 1137229 and NSF IIS-1064983.

body acceleration and balance during the motion. Some of transition tasks are extensively studied in robotics, including sit-to-stand [13]–[16] and lie-to-stand [17]–[20]. While many of these prior techniques developed control methods for a specific motion, a few research groups proposed more generic solutions to handle a wider range of transition tasks. Jones [21] developed rising motions for both biped and quadruped using pose tracking, orientation correction and virtual force. Lin and Huang [22] used motion planning and dynamic filtering to develop rising up motions from various initial lying poses. Tassa et al. [23] used Model Predictive Control to synthesize complex behaviors, including getting up from an arbitrary pose on the ground. Successful results were demonstrated in simulation, but not on robot hardware. In this paper, we aim to develop a general method for a variety of transition tasks and demonstrate that the control policies learned in the simulation can be transferred to the real world.

A controller that is optimized in simulation may not work in the real environment due to model inconsistency, also known as the Reality Gap. One way to overcome this problem is to apply system identification which calibrates the physical model using the robot's performance data. In practice, system identification is often interleaved with controller optimization to minimize the number of required robot experiments [5], [24]. System identification optimizes the model parameters given a model parameterization. Some widely-used models, such as linear dynamics model and Gaussian processes were proven effective for continuous dynamics and relatively simple control tasks, but it is not clear whether they can be successful for a dynamic system involving discontinuous contact forces.

Modern physical simulators [25] solve the governing dynamics equation with LCP constraints to accurately model contacts. With the aid of these simulators, model-based approaches can effectively control humanoid robots in contact rich scenarios [26], [27]. However, system identification for these modern physical simulators has not been extensively studied [28]. When applying system identification for tasks with frequent contact changes, the process usually requires specially designed robot experiments independent of the control tasks of interest because the model parameters can be inferred more accurately from a contact-free behavior [28]. As a result, a large amount of data are needed for system identification, in the hope that the collected data cover the important and relevant regions of the control space. In contrast, our system tightly couples simulation calibration and trajectory optimization. The current reference trajectory that is optimized in the simulation is used as the experiment for system identification. Using Covariance Matrix Adaptation (CMA) [29], we are able to estimate parameters with frequent contact changes in the experiments. CMA is a stochastic sampling-based optimization algorithm, which has been successfully applied to search for control parameters when the problem domain is highly discontinuous [30]–[32].

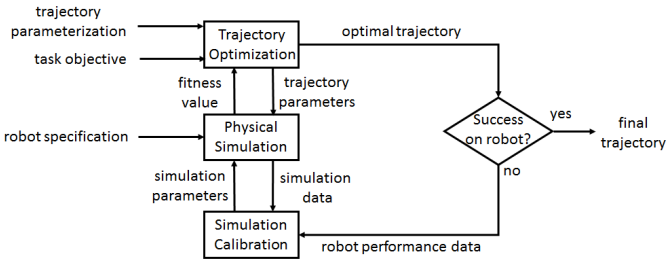

Fig. 1. Overview of our algorithm.

III. Overview

We have developed a system that can automatically design reference trajectories for robots to execute transition motions (Figure 1). Given the specification of the robot, including its body shape, the physical properties of each body, and the types of joints, we build a physical simulation using DART. The trajectory optimization subsystem runs thousands of simulations to search for the optimal trajectory that maximizes the task-related fitness function. We then test this optimal trajectory on the robot. If the robot successfully completes the task, a solution is found and our algorithm terminates. Otherwise, we record the robot performance data and feed it into the simulation calibration subsystem. Simulation calibration runs another optimization, which searches for the optimal simulation parameters to minimize the discrepancy between the performance of the robot in the simulation and in the real world. The loop of trajectory optimization and simulation calibration is performed iteratively until the reference trajectory works successfully on the real robot. In the next three sections, we will present the algorithmic details of these components.

IV. Physical Simulation

A. Dynamics Equation

We model the robot as an articulated rigid body system in our simulator, which satisfies the following dynamics equations, non-penetration and linear complementarity conditions for contact points.

$$\begin{aligned} \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) &= \boldsymbol{\tau} + \mathbf{J}(\mathbf{q})^T\mathbf{f} \\ \mathbf{d}(\mathbf{q}) &\geq \mathbf{0} \\ \mathbf{d}(\mathbf{q})^T\mathbf{f} &= \mathbf{0} \\ \mathbf{f} &\in \mathbf{K} \end{aligned} \tag{1}$$

where $\mathbf{M}(\mathbf{q})$ is the mass matrix and $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ is the Coriolis and Centrifugal force. $\boldsymbol{\tau}$ are joint torques exerted by the actuators. An actuator model that computes $\boldsymbol{\tau}$ based on the current state and a reference pose is presented in the next section. $\mathbf{J}(\mathbf{q})$ is the Jacobian matrix and $\mathbf{f}$ is the contact force. $\mathbf{d}(\mathbf{q})$ is the distance of the contact to the ground and $\mathbf{K}$ is the friction cone. We use DART to solve the above equations to simulate the dynamics of the robot.

B. Actuator Model

A realistic robot simulation relies heavily on an accurate actuator model. A common practice is to set the same PID

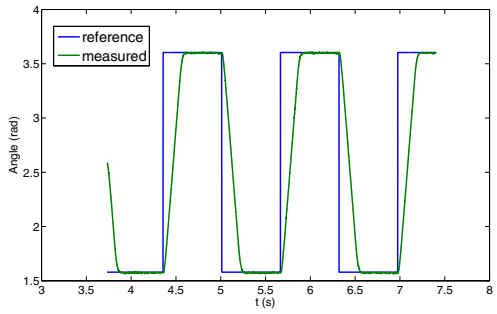

Fig. 2. The trajectories of the reference (blue) and the measured (green) joint angle for acutator identification.

gains in the simulation as those in the servo. However, Dynamixel AX-18 servos, used on our robot (ROBOTIS BIOLOID GP), do not support PID control. Although the servo can track an input reference angle $\bar{q}$, the relation between the joint error $\bar{q} - q$ and the output torque τ is unclear. We derive an actuator model based on the ideal DC motor assumption and the specification of the servo:

$$\tau = -k_p(q - \bar{q}) - k_d\dot{q} - k_c \operatorname{sgn}(\dot{q}) \tag{2}$$

where k_p, k_d and k_c are the *actuator gains*. The detailed derivation of the actuator model (eq. (2)) can be found in the Appendix.

Accurately identifying these gains would require a large number of carefully designed experiments to be run on each of the actuators. In contrast, we choose to run only one simple experiment on one actuator. This simple experiment gives us an initial guess of the gains. We then rely on simulation calibration to refine this estimate.

In this experiment, we clamp the entire robot on a table except for the left foot. We then send a periodic reference joint angle $\bar{q}(t)$ that oscillates between two extreme angles (blue curve in Figure 2) to the servo at the left ankle. We record the trajectory of the actual joint angle $q(t)$ throughout the experiment (green curve in Figure 2).

Given $q(t)$ and $\bar{q}(t)$, we apply regression to estimate the actuator gains:

$$\min_{k_p,k_d,k_c} \int ||I\ddot{q}(t)+k_p(q(t)-\bar{q}(t))+k_d\dot{q}(t)+k_c \operatorname{sgn}(\dot{q}(t))||^2 \mathrm{d}t$$

where I is the moment of inertia of the foot with respect to the rotating axis. The above equation is derived by plugging into $\tau = I\ddot{q} + \dot{I}\dot{q}$ and the fact that $\dot{I}\dot{q} = 0$ because the foot is a rigid body that rotates along a fixed axis. Our experiments and computation show that the actuator gains are $k_p = 9.272(N \cdot m/rad)$, $k_d = 0.3069(N \cdot m \cdot s/rad)$, and $k_c = 0.03(N \cdot m)$.

V. Trajectory Optimization

Given the physical simulator, we can design controllers to enable the robot to achieve various transition tasks in the simulated environment. The four tasks that we use to test our system are: rising from a sitting, leaning or kneeling position and flipping towards a handstand position (Figures 3, 4, 6 and 7). For each task, the initial pose $\bar{\mathbf{q}}(0)$ and the final pose $\bar{\mathbf{q}}(T)$ are given as input. The goal of this optimization is to compute a reference trajectory $\bar{\mathbf{q}}(t)$ so that the robot can move from the initial to the final pose without losing balance. We purposefully choose to control the robot with only open-loop reference trajectories[1], which produces control signal $\bar{\mathbf{q}}(t)$ as a function of time t independent of the states of the robot. Using only the open-loop reference trajectory, we can better evaluate our system during the simulation calibration process, because the accuracy of the simulation becomes more critical if no feedback balance control is involved.

A. Trajectory Parameterization

We parameterize the reference trajectory $\bar{\mathbf{q}}(t)$ with a sparse set of keyframes $\{\bar{\mathbf{q}}_i\}_{i=0\ldots n}$ at time $\{t_i\}_{i=0\ldots n}$, where the first $\bar{\mathbf{q}}_0$ and last keyframes $\bar{\mathbf{q}}_n$ are given, but the final time T, which equals t_n, is optimized. The optimization searches for both the joint configuration and the time of each keyframe and the reference trajectory is the linear interpolation between the keyframes. In our implementation, the number of keyframes $n+1$ is fixed and manually specified by the user, although it is possible to incorporate it in an outer-loop optimization.

B. Fitness Function

The criterion of success for all the tasks is whether the robot remains stably upright (or upside down in the last task) at the end of its motion. We use the following fitness function to reward the trajectories that keep the robot balanced throughout the entire motion.

$$\{\bar{\mathbf{q}}_i\}_{i=1\ldots n-1}, \{t_i\}_{i=1\ldots n} = \operatorname{argmax} \int_0^{T+1} \frac{1}{|\alpha(t)| + \epsilon} \mathrm{d}t \tag{3}$$

where α is the angle between the up direction in the local frame of the robot's torso and the up (or down in the last task) direction in the global frame. It measures how far the robot is from losing its balance. ϵ is a small positive number to prevent the denominator from being zero. We choose $\epsilon = 0.1$ in all of our tasks. T (same as t_n) is the time when the robot reaches the final pose. Note that the upper limit of the integration is $T + 1$. The extra one second is to wait for the robot to settle down. We use the time horizon $T + 1$ because it is still possible for the robot to fall while settling down. The fitness function will penalize if this situation occurs.

C. Optimization with CMA

During the transition motion, discrete contact events can happen frequently. They impose additional challenges for the continuous optimization algorithms that rely on gradient information. To overcome this challenge, we apply a sample-based stochastic optimization algorithm, CMA, to optimize the reference trajectory. CMA does not need gradient information and can explore multiple local minima. This makes it ideal for optimizing open loop trajectories or feedback controllers for transition tasks. For the completeness of the

[1]The internal feedback loop in the actuators still exist but we do not alter this feedback loop in our trajectory optimization.

paper, we will briefly describe the CMA algorithm here. Please refer to the original paper [29] for more details. At each iteration, a population of CMA samples are drawn from an underlying Gaussian distribution. In our case, a CMA sample is a set of keyframes and their associated time. Each CMA sample is used to control the robot in the simulation and evaluated using eq.(3). The samples with lower fitness values are discarded. The underlying Gaussian distribution is updated according to the remaining good samples. With more iterations, the distribution gradually converges to a better region of the search space. After a maximum number of iterations, we choose the best CMA sample and reconstruct the optimal reference trajectory $\bar{\mathbf{q}}(t)$.

VI. Simulation Calibration

Although the optimal reference trajectory $\bar{\mathbf{q}}(t)$ can work effectively in the simulation, it may fail in the real world. One cause of this failure is the erroneous parameter settings in the physical simulation, such as the mass, the moment of inertia, the COM of each body segment, and the gains of the actuators. Usually, simulation parameters are set according to the specification of the robot. However, we find that many of these specifications are inaccurate. For example, there is about 40% of error in the total mass of our robot between the open-source CAD file and our own measurement, which highlights the importance of simulation calibration in our system. We formulate an optimization that searches for the simulation parameters $\boldsymbol{\theta}$ to minimize the discrepancy between the simulated results and the robot performance in the real environment.

$$\boldsymbol{\theta} = \arg\min \frac{1}{n} \sum_{i=1}^{n} \int_{0}^{T+1} ||\tilde{\mathbf{q}}_i(t) - \mathbf{q}_i(t;\boldsymbol{\theta})||^2_{\mathbf{W}} \mathrm{d}t \qquad (4)$$

where $\mathbf{q}(t;\boldsymbol{\theta})$ is the sequence of simulated robot states with simulation parameter $\boldsymbol{\theta}$. The subscript i denotes that this sequence is generated using the optimal reference trajectory found in the ith iteration of the "trajectory optimization-simulation calibration" loop. $\tilde{\mathbf{q}}_i(t)$ is the real robot states[2] by executing the same reference trajectory. Note that $\mathbf{q}$ and $\tilde{\mathbf{q}}$ here include both the joint angles and the global position and orientation of the robot's torso. $\mathbf{W}$ is a weight matrix, which encapsulates the relative importance of each joint. In our case, we use the same $\mathbf{W}$ among all the tasks and choose the $\mathbf{W}$ to heavily penalize the difference in the torso orientation.

Although dozens of simulation parameters $\boldsymbol{\theta}$ can potentially be optimized, we only focus on the most relevant parameters to our tasks: changing postures and maintaining balance. As such, we calibrate only the gains (k_p, k_d and k_c) that are shared by all the actuators and the COM of each body segment on the body frame because the actuator gains are critical to posture change and the COM is vital to balance. Note that although we have a process to identify the actuator gains (Section IV-B), that simple experiment does not give accurate results. Instead, it gives a good initial guess of the actuator gains to be used in simulation calibration. By focusing on actuator gains and COM's, utilizing symmetry of the robot, our simulation calibration needs to optimize 30 simulation parameters.

Once again we use the gradient-free CMA to optimize eq. (4) due to the presence of contact changes in the transition task and the complex interplay between the simulation results and the simulation parameters. In this case, each CMA sample is a candidate set of simulation parameters $\boldsymbol{\theta}$. To evaluate each CMA sample, we set the parameters $\boldsymbol{\theta}$ in the physical simulator, simulate the robot movement, and then evaluate the objective function eq. (4).

VII. Results

We evaluate our system on four tasks. Please watch the accompanying video[3] for the robot performance in the simulation and in the real world.

A. Experiment Setup

We use BIOLOID GP as our robot platform. The hardware includes 22 degrees of freedom, 18 of which are controllable by Dynamixel AX-12/AX-18 servos. The remaining six degrees of freedom, indicating the torso position and orientation in the Cartesian coordinates, are unactuated. To control the robot, a master program on the PC writes the reference pose $\bar{\mathbf{q}}$ to the serial port connected to the robot. A slave program that runs on the robot's onboard microprocessor listens to this port and sends the reference joint angle to each actuator. At the same time, the robot performance data $\tilde{\mathbf{x}}$ is measured and sent back to the computer. BIOLOID GP has limited sensing capabilities. It has one two-axis Gyro sensor that we do not use. Instead, we glue four reflective markers on the robot's torso and use a VICON motion capture system to measure global position and orientation of the torso. We also read out the joint angles from the servos.

Our system is implemented in C++ and runs on a laptop with a 2.6GHz quad-core CPU. We use DART to simulate the physics. We find that all the tasks can be achieved with symmetric lower body motions, which enables us to reduce the dimensionality of the control space in trajectory optimization. For each simulation calibration, we collect three episodes of robot data by running the same reference trajectory three times to average out the noise and other possible perturbations. Each episode is approximately two seconds. We use 32 samples per iteration and at most 50 iterations in CMA, which takes about 15 minutes to find an optimal solution.

B. Rising from a Sitting Position

The first task is to rise from a chair (Figure 3). The initial and the final poses $\mathbf{q}_0$ and $\mathbf{q}_2$ are shown in the leftmost and rightmost images in Figure 3. The trajectory optimization needs to search for an additional inbetween keyframe $\mathbf{q}_1$, as well as the time t_1 and t_2 of these keyframes.

We intentionally choose the initial pose such that the feet are far from the projection of the robot's COM on the ground.

[2]We execute the same reference trajectory three times and average the three sequences as $\tilde{\mathbf{q}}(t)$ in the robot experiments to smooth out the noise and slight perturbations in initial conditions.

[3]https://dl.dropboxusercontent.com/u/36899427/iros2016LowRes.mp4

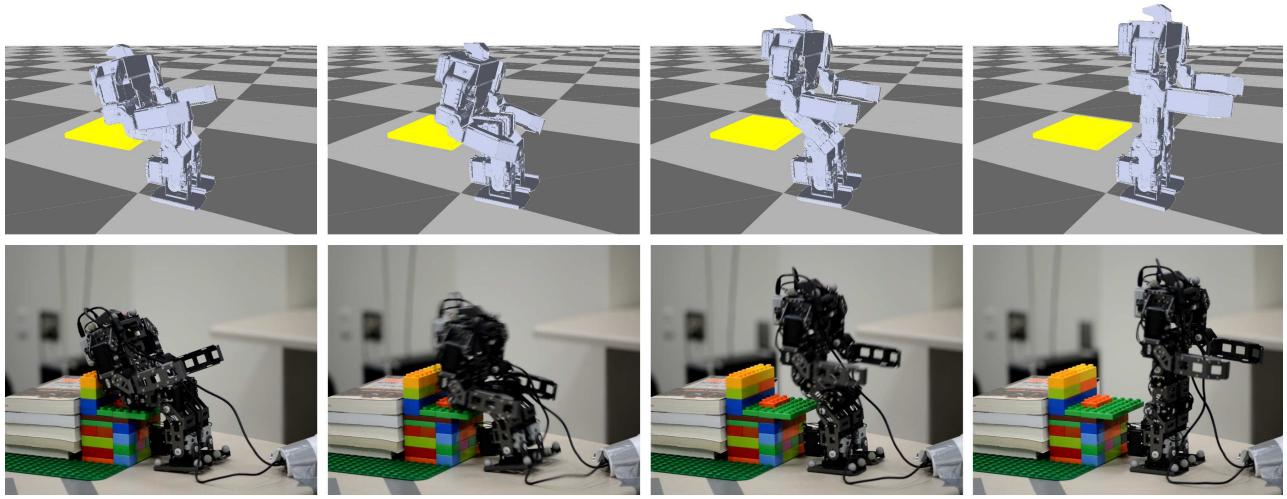

Fig. 3. The results of the sit-to-stand task in the simulation and on the real robot.

Though simple, this experiment is a good test for dynamic balance because the robot cannot simply track the extended positions of knees and hips to rise; it would fall backward immediately. Our method successfully computes a reference trajectory that enables the robot to stand up in the simulation: the robot first produces a forward momentum by quickly leaning its upper body to the front. It then starts to extend the hips and the knees at the moment when the COM is moving towards the supporting feet.

When the computed trajectory is applied to the hardware, the robot is able to rise successfully without the need of simulation calibration. This shows that the "Reality Gap" is not always a problem. Any proposed technique that attempts to address the issue needs to be tested on sufficiently challenging tasks in which the stability region is small and accurate model parameters are critical.

C. Rising from a Leaning Position

In this task, the robot is required to rise from leaning on the wall to a standing position (Figure 4). The hip joints are initially bent and straightened out in the final configuration while all other joints remain the same. The initial and final poses are the only two keyframes for this task.

The goal of trajectory optimization is to find an appropriate time interval T between these two keyframes. This one-dimensional problem largely simplifies the challenge of trajectory optimization. However, without the simulation calibration, the trajectory optimization fails to find any solution that resulted in a successful rising motion. If the time interval is too long ($T > 0.11s$), the virtual robot does not accelerate sufficiently to rise. If the time interval is too short ($T \leq 0.11s$), the virtual robot tends to push off the wall with a large force and fall forward to the ground. When tracking the optimal trajectory with the highest fitness value ($T = 0.11s$) in the real world, the robot also fails to rise but in a different way from the simulation result. That is, the robot fails to rise due to the lack of momentum, while the virtual robot overshoots and falls forward in the simulation. Figure 5 compares the trajectories of the robot's torso orientation in the simulation (blue curve) and in the real world (red curve). After one iteration of simulation calibration, the discrepancy is greatly reduced (Figure 5 green curve), and the optimal reference trajectory leads to successful motion both in the simulation and in the real world. This experiment only requires six seconds of robot data.

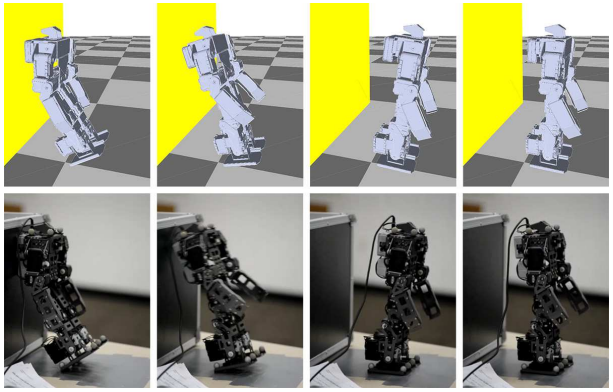

Fig. 4. The results of the lean-to-stand task in the simulation and on the real robot.

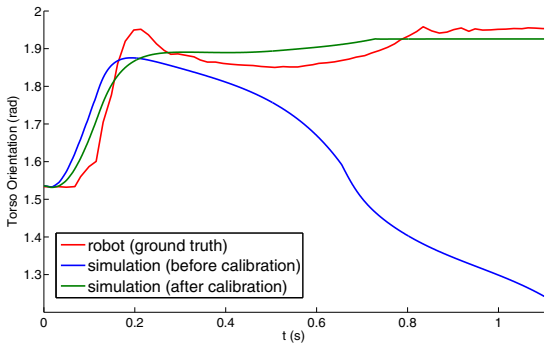

Fig. 5. Comparisons of the robot's global (torso) orientation over time in the simulation (before/after calibration) and in the real environment.

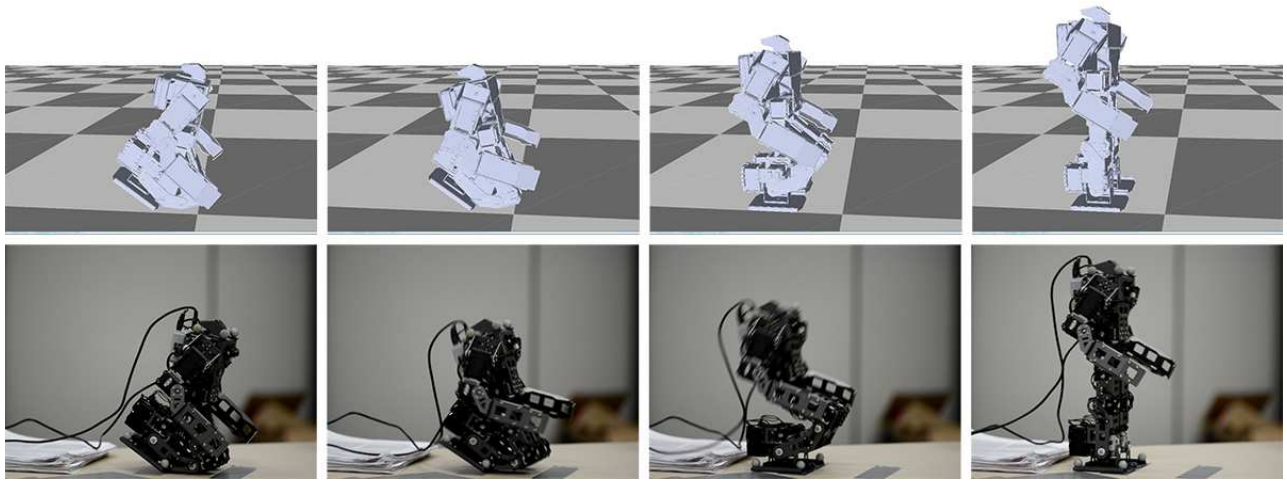

Fig. 6. The results of the kneel-to-stand task in the simulation and on the real robot.

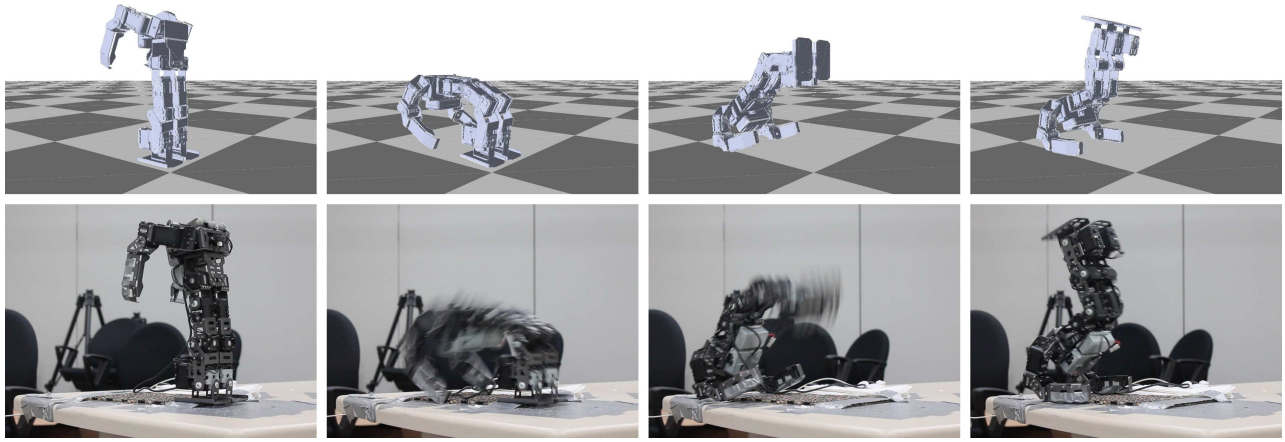

Fig. 7. The results of the stand-to-handstand task in the simulation and on the real robot.

D. Rising from a Kneeling Position

Figure 6 shows that the robot stands up from a kneeling pose. Between the user-specified initial and final poses, the optimal trajectory consists of two additional keyframes. The robot first leans its upper-body backwards. As its COM is moving to the back, it quickly bends the hip, flexes its ankles and stands up without using hands for support. This task requires precise timing and balance control and is a good example for exposing the issue of model inconsistency. Our initial simulator finds an optimal trajectory which succeeds in the simulation, but is not able to transfer the success to the real world. After one iteration and with six seconds of data, the robot efficiently learns to rise from a kneeing position.

E. Flipping to a Handstand Position

We test our system with a challenging gymnastic action: flipping to a handstand position from a standing pose (Figure 7). There are two unique challenges in this task. First, the speed and the curvature of the initial arching motion is crucial and only a narrow range of such speed and curvature can lead to a balanced handstand. Second, the USB cable that connects the robot to the computer will inevitably hit the ground during the backflip and inject a strong perturbation that is not modeled in our simulation.

With two iterations of simulation calibration and trajectory optimization, our system finds a successful trajectory that works both in the simulation and in the real world: The robot arches back rapidly and lifts its feet after the arms touch the ground. This example shows that our system can automatically design open loop reference trajectories for challenging tasks, even with significant unmodeled perturbations.

VIII. Discussion

One important component of our system is the simulation calibration. The results show that it is an effective strategy to narrow down the Reality Gap, with a minimal number of robot experiments. In all the examples, our method only requires at most two iterations of calibration (or approximately 12 seconds of robot data) to transfer the reference trajectory to the real robot. This amount of robot data is far less than those needed in typical system identification methods.

Similar to other system identification methods, the parameters optimized in the simulation calibration process may not be the true physical parameters. For example, we observe that

TABLE I

GENERALIZABILITY OF THE CALIBRATED SIMULATION

Tasks	I	II	III
I	Succeed	Fail	Fail
II	Fail	Succeed	Fail
III	Succeed	Fail	Succeed
I&II	Succeed	Succeed	Succeed
I&III	Succeed	Fail	Succeed
II&III	Succeed	Succeed	Succeed
I&II&III	Succeed	Succeed	Succeed

the simulation parameters optimized for the task of lean-to-stand are different from those optimized for kneel-to-stand. This could be caused by the errors from other simulation parameters that we do not calibrate. This implies that the calibrated simulator is fit to the current task and may not be useful for a different task. Although this could be a problem if the goal is to estimate the true physical parameters of the system, it is not a problem in our case because our goal is to find a reference trajectory for a specific task. In summary, tightly coupling simulation calibration with a specific control task makes it possible to efficiently use only a small amount of robot data.

We perform two additional experiments to investigate the generalizability of the calibrated simulator. In the first experiment, we calibrate the simulation for one task (e.g. lean-to-stand) and use it to optimize the reference trajectory for a different task (e.g. sit-to-stand). The results can be found in Table I. We refer Task I to Task III as lean-to-stand, sit-to-stand, and kneel-to-stand respectively. The first three rows show that the simulator optimized for a single task cannot be generalized to other tasks in most cases.

In the second experiment, we calibrate the simulator with multiple tasks (e.g. lean-to-stand and sit-to-stand). The last four rows of Table I show that the generalizability across Task I, II and III is greatly improved. Specifically, using multiple training tasks to calibrate the simulator will improve the generalizability, but only if the tasks are similar. For example, Task IV (stand-to-handstand) is drastically different from the first three tasks and can compromise the calibration process if included in the training task set.

These experiments indicate that if we need to design trajectories for a group of similar tasks, we may not need to perform simulation calibration for each task independently. It is possible that after calibrating the simulator for a small set of tasks, the reference trajectories of the remaining tasks can be optimized in simulation without further calibration.

IX. CONCLUSION

We have presented a complete pipeline to automatically design open loop reference trajectories for robots. The solution consists of a set of powerful computational tools: physical simulation, trajectory optimization and simulation calibration. Our method enables efficient reference trajectory design for a humanoid robot to achieve four different tasks: lean-to-stand, sit-to-stand, kneel-to-stand and stand-to-handstand.

There are two avenues for future work. First, we will include more simulation parameters in simulation calibration.

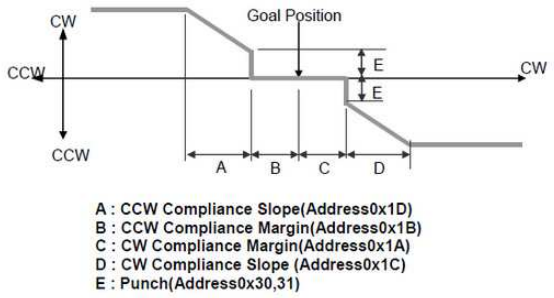

Fig. 8. The mapping between $q - \bar{q}$ and U for an AX-18 actuator [33]. The x-axis is $q - \bar{q}$ while the y-axis is U.

In this paper, we have shown that adjusting the COM and the actuator gains are enough for our needs, but other simulation parameters might also be important for other tasks. A potential issue of including more simulation parameters is the increased risk of overfitting. Performing some automatic prioritizing and selection on candidate parameters would be a promising future research. Second, we believe that our system can be generalized to control other types of motions, such as walking, biking or more challenging gymnastic stunts. In these tasks, feedback control is necessary. In the future, we plan to extend our system to include feedback control and test it on a wider range of tasks.

APPENDIX

In this Appendix, we derive the actuator model (eq.(2)) from the specifications of the Dynamixel AX-18 servo (Figure 8). The servo maps the difference between the desired and the actual joint angle $q - \bar{q}$ to the power level U. The intervals A and D determine the proportion gain for counter-clockwise and clockwise motions respectively. B and C are the compliance margins, which are thresholds below which the servo stops outputting any torque. E, the punch, is the minimum power level before the servo shuts down. In practice, we set A and D the same. In addition, since B, C and E are much smaller than A or D, we ignore their effects and approximate the mapping as linear within the intervals $q - \bar{q} \in A \bigcup B \bigcup C \bigcup D$ with the slope k_e:

$$U = k_e(q - \bar{q})$$

To derive the relation between the power level U and the output torque τ, we adopt a model for the ideal DC motor [34]. It is a valid assumption because the AX-18 servos use high-quality DC motors. Considering the power balance in the motor at a constant voltage U:

$$P_{electric} = P_{mechanic} + P_{heat} \tag{5}$$

The electrical power $P_{electric}$ can be decomposed into the mechanical power $P_{mechanic}$ and the heat P_{heat}. From eq.(5), we can get the following relation:

$$UI = \dot{q}\tau_{motor} + RI^2 \tag{6}$$

where I is the current and R is the motor winding resistance. In an ideal DC motor, the torque is linearly proportional

to the current $\tau_{motor} = k_\tau I$. Plugging it into the above equation, we have:

$$U = k_\tau \dot{q} + \frac{R}{k_\tau}\tau_{motor} \tag{7}$$

where k_τ is the torque constant. The total torque τ_{motor} is the sum of the output torque τ that drives the motor shaft and the frictional torque τ_f inside the motor:

$$\tau_{motor} = \tau + \tau_f \tag{8}$$

The friction torque can be further divided into viscous friction and Coulomb friction [34]:

$$\tau_f = k_v \dot{q} + k_c \operatorname{sgn}(\dot{q}) \tag{9}$$

where k_v and k_c are friction coefficients for the viscous and Coulomb friction respectively. $\operatorname{sgn}(x)$ is the sign function that equals 1 if x is positive, -1 if negative and 0 otherwise.

Combining eq.(7), (8) and (9), we get the actuator model:

$$\begin{aligned}\tau &= \frac{k_\tau k_e}{R}(q - \bar{q}) + (-k_v - \frac{k_\tau^2}{R})\dot{q} - k_c \operatorname{sgn}(\dot{q}) \\ &= -k_p(q - \bar{q}) - k_d \dot{q} - k_c \operatorname{sgn}(\dot{q})\end{aligned}$$

where $k_p = -\frac{k_\tau k_e}{R}$ and $k_d = k_v + \frac{k_\tau^2}{R}$.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their helpful comments. We want to thank Greg Turk, Frank Dellaert, James O'Brien, Jarek Rossignac, Sehoon Ha, Yufei Bai and Yuting Gu for their help on this research.

REFERENCES

[1] L. Ljung, Ed., *System Identification (2Nd Ed.): Theory for the User*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[2] J. Morimoto, G. Zeglin, and C. G. Atkeson, "Minimax differential dynamic programming: Application to a biped walking robot," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2003, pp. 1927–1932.

[3] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, 2006, pp. 1–8.

[4] D. Mitrovic, S. Klanke, and S. Vijayakumar, *Adaptive Optimal Feedback Control with Learned Internal Dynamics Models*. Springer-Verlag, 2010.

[5] P. Abbeel and A. Y. Ng, "Exploration and apprenticeship learning in reinforcement learning," in *Proceedings of the 22Nd International Conference on Machine Learning*, ser. ICML '05. New York, NY, USA: ACM, 2005, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/1102351.1102352

[6] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[7] S. Ha and K. Yamane, "Reducing Hardware Experiments for Model Learning and Policy Optimization," *IROS*, 2015.

[8] J. C. Zagal, J. Ruiz-del-Solar, and P. Vallejos, "Back-to-Reality: Crossing the reality gap in evolutionary robotics," in *IAV 2004: Proceedings 5th IFAC Symposium on Intelligent Autonomous Vehicles*. Elsevier Science Publishers B.V., 2004.

[9] P. Abbeel, M. Quigley, and A. Y. Ng, "Using inaccurate models in reinforcement learning," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06, 2006, pp. 1–8.

[10] C. K. Liu and S. Jain, "A short tutorial on multibody dynamics," Georgia Institute of Technology, School of Interactive Computing, Tech. Rep. GIT-GVU-15-01-1, 08 2012.

[11] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control (under review), 2011a. url http://www.cs.washington.edu/homes/todorov/papers/mujoco. pdf."

[12] D. M. Scarborough, C. A. McGibbon, and D. E. Krebs, "Chair rise strategies in older adults with functional limitations," *Journal of rehabilitation research and development*, vol. 44, no. 1, p. 33, 2007.

[13] P. Faloutsos, M. van de Panne, and D. Terzopoulos, "Autonomous reactive control for simulated humanoids." in *ICRA*. IEEE, 2003, pp. 917–924.

[14] S. Iida, M. Kanoh, S. Kato, and H. Itoh, "Reinforcement learning for motion control of humanoid robots." in *IROS*. IEEE, 2004, pp. 3153–3157.

[15] S. Pchelkin, A. S. Shiriaev, L. B. Freidovich, U. Mettin, S. V. Gusev, and W. Kwon, "Natural sit-down and chair-rise motions for a humanoid robot." in *CDC*. IEEE, 2010, pp. 1136–1141.

[16] M. Mistry, A. Murai, K. Yamane, and J. K. Hodgins, "Sit-to-stand task on a humanoid robot from human demonstration." in *Humanoids*. IEEE, 2010, pp. 218–223.

[17] J. Morimoto and K. Doya, "Reinforcement learning of dynamic motor sequence: Learning to stand up," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1998, pp. 1721–1726.

[18] P. Faloutsos, M. van de Panne, and D. Terzopoulos, "Composable controllers for physics-based character animation," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 251–260. [Online]. Available: http://doi.acm.org/10.1145/383259.383287

[19] H. Hirukawa, S. Kajita, F. Kanehiro, K. Kaneko, and T. Isozumi, "The human-size humanoid robot that can walk, lie down and get up," *Int. J. Rob. Res.*, vol. 24, no. 9, pp. 755–769, Sept. 2005.

[20] F. Kanehiro, K. Fujiwara, H. Hirukawa, S. Nakaoka, and M. Morisawa, "Getting up motion planning using mahalanobis distance." in *ICRA*. IEEE, 2007, pp. 2540–2545.

[21] B. J. Jones, "Rising motion controllers for physically simulated characters," Master's thesis, The University Of British Columbia, 2011.

[22] W.-C. Lin and Y.-J. Huang, "Animating rising up from various lying postures and environments," *The Visual Computer*, vol. 28, no. 4, pp. 413–424, 2012.

[23] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IROS*. IEEE, 2012, pp. 4906–4913.

[24] M. Gevers *et al.*, "System identification without lennart ljung: what would have been different?"

[25] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4397–4404.

[26] I. Mordatch, K. Lowrey, and E. Todorov, "Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids," *IROS*, 2015.

[27] S. Ha and C. K. Liu, "Multiple contact planning for minimizing damage of humanoid falls," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 2761–2767.

[28] S. Kolev and E. Todorov, "Physically consistent state estimation and system identification for contacts," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 1036–1043.

[29] N. Hansen, *The CMA Evolution Strategy: A Tutorial*, 2009.

[30] J.-c. Wu and Z. Popović, "Terrain-adaptive bipedal locomotion control," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 72:1–72:10, July 2010. [Online]. Available: http://doi.acm.org/10.1145/1778765.1778809

[31] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Optimizing walking controllers for uncertain inputs and environments," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 73:1–73:8, July 2010.

[32] J. Tan, Y. Gu, C. K. Liu, and G. Turk, "Learning bicycle stunts," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 50:1–50:12, July 2014.

[33] *AX-18F/Ax-18A e-Manual*, 1st ed., ROBOTIS.

[34] M. Schwarz and S. Behnke, "Compliant robot behavior using servo actuator models identified by iterative learning control," in *RoboCup 2013: Robot World Cup XVII*, 2013, pp. 207–218.