Policy gradient methods with model predictive control applied to ball bouncing

Paul Kulchenko
Department of Computer Science and Engineering
University of Washington, Seattle, WA
Email: paul@kulchenko.com

Emanuel Todorov
Departments of Applied Mathematics and Computer Science and Engineering
University of Washington, Seattle, WA
Email: todorov@cs.washington.edu

***Abstract*—We propose a policy parameterization well suited for control problems that involve both continuous dynamics and discrete events. The key idea is to parameterize the policy using a scalar function defined on the subset of states corresponding to discrete events. This function approximates the cost-to-go with respect to some master cost. Once the function is given, we define the policy using model-predictive control (MPC) extended to a first-exit setting: instead of optimizing to a predefined horizon, we optimize up to the next discrete event (ball-paddle contact). The proposed parameterization relies on numerical optimization to obtain the actual policy as opposed to evaluating an explicit formula, and has the advantage of being more compact and focusing on the aspects of the task. Once the policy has been defined, we simulate it using "quenched" noise, and improve the parameters of the function via gradient descent on the resulting average master cost. We apply this method to the task of two-ball juggling on the same paddle and analyze its performance using a simulated model of the system.**

I. Introduction

Policy gradient methods in Reinforcement Learning provide a systematic way to improve the performance of a parameterized policy [1]–[4]. The choice of parameterization is often crucial, and yet we have no general way of constructing good parameterizations.

In this paper we propose a policy parameterization that is well-suited for control problems involving both continuous dynamics and discrete events. The specific example we focus on is robotic ball bouncing, although the same methodology could be applied to more complex tasks such as walking or object manipulation. The discrete events correspond to contacts.

The key idea is to parameterize the policy using a scalar function $h(x, w)$ defined on the subset of states x corresponding to discrete events. Note that we will not treat general hybrid systems with both continuous and discrete state variables. Instead we focus on more restricted hybrid systems whose state is fully captured by the continuous variables (e.g. position and velocity of the ball and the paddle). Event states correspond to some sub-manifold in this continuous state space.

Our function $h(x, w)$ will approximate the cost-to-go with respect to some master cost $\ell(x, u)$ which defines an infinite-horizon optimal control problem. Once $h(x, w)$ is given, we will define the policy using model-predictive control (MPC) as in our recent work [5]. The idea behind MPC is as follows: at each time step of the control loop, compute an optimal state-control trajectory starting at the current state and reaching up to some horizon into the future, send the first control to the plant, and then repeat the procedure at the next time step. In our recent work we extended this idea to a first-exit setting: instead of optimizing up to a predefined horizon, we optimize up to the next discrete event (ball-paddle contact). The function $h(x, w)$ is used as a final cost incurred at the event/contact state, while the function $\ell(x, u)$ is the running cost for the MPC trajectory optimization problem. In this way, the function $h(x, w)$ is defined over contact states only, and yet it parameterizes the policy for all states. This parameterization is implicit: it relies on numerical optimization to obtain the actual policy, as opposed to evaluating an explicit formula (as is usually done in policy gradient methods). The advantage of such a policy parameterization is that it is more compact and focuses on the key aspects of the task; indeed in most hybrid systems the discrete dynamics are key.

The algorithm we propose is as follows. Given the current parameters w of the final cost $h(x, w)$, simulate the policy resulting from our first-exit MPC method [5] starting from a number of predefined initial states, and average the master cost $\ell(x, u)$ along the state-control trajectories. This average is the performance $L(w)$ of the current set of parameters. Now improve w using gradient descent on $L(w)$; here we use the BFGS (Broyden–Fletcher–Goldfarb–Shanno) method. While many recent studies have estimated the gradient from a single long stochastic simulation, we have opted for shorter pseudo-deterministic simulation (using the same noise sequence for every w), letting the numerical optimizer compute gradients via finite differences. It is not clear which approach is better, and anyway the issue of how the gradient is computed is orthogonal to the main idea of the paper - which is the specific parameterization via MPC. Note that most policy gradient results assume a linear parameterization of the policy, while the implicit parameterization we use here is generally non-linear.

The choice of "features" for the function $h(x, w)$ is still important and problem-specific. Yet this choice is now simplified because discrete event states tend to be more intuitive. In the ball bouncing example, the features are chosen so as to

encode what is a good way to hit a ball (see below).

While ball-bouncing has not been studied in the context of policy gradient methods, it has nevertheless received considerable attention [6]–[9]. Most of that work has focused on analysis of (usually passive) stability. Indeed one of the more remarkable findings has been that passive stability in the vertical direction requires hitting the ball with negative acceleration, and that humans exploit this strategy [8], in general agreement with the idea that the brain optimizes motor behavior [10]. Our work puts the emphasis on intelligent feedback control rather than simple solutions. What we gain is the ability to recover from a wide range of perturbations, change the task objectives on the fly (e.g. the desired height of the bounce), and perform more complex tasks such as bouncing two balls with a single (hard) paddle.

We describe our general control methodology in the next section, and specialize it to ball-bouncing in subsequent sections. The details of first-exit model predictive control are covered in more depth in our earlier work [5], and are also summarized here for convenience.

II. First-exit model predictive control

MPC is normally applied to tasks that continue indefinitely. Thus we formalize the task as an infinite-horizon average-cost stochastic optimal control problem, with dynamics given by the transition probability distribution $p(x'|x,u)$. Here x is the current state, u the current control, and x' the resulting next state. The states and controls can be discrete or continuous. Let $\ell(x,u)$ be the immediate cost for being in state x and choosing control u. It is known that the differential optimal cost-to-go function $\widetilde{v}(x)$ satisfies the Bellman equation

$$c + \widetilde{v}(x) = \min_u \left\{ \ell(x,u) + E_{x' \sim p(\cdot|x,u)} \widetilde{v}(x') \right\} \qquad (1)$$

where c is the average cost per step. The solution is unique under suitable ergodicity assumptions.

Now consider a first-exit stochastic optimal control problem with the same dynamics p, immediate cost $\widehat{\ell}(x,u)$, and final cost $h(x)$ defined on some subset $\mathcal{T}$ of terminal states. In such problems the total cost-to-go v is finite and satisfies the Bellman equation

$$v(x) = \min_u \left\{ \widehat{\ell}(x,u) + E_{x' \sim p(\cdot|x,u)} v(x') \right\} \qquad (2)$$

for $x \notin \mathcal{T}$, and $v(x) = h(x)$ for $x \in \mathcal{T}$.

When $\widehat{\ell}(x,u) = \ell(x,u) - c$ the two Bellman equations are identical and so $v(x) = \widetilde{v}(x)$. Thus we can find the optimal solution to an infinite-horizon average-cost problem by solving a first-exit problem up to some set of terminal states $\mathcal{T}$, and at the terminal states applying a final cost h equal to the differential cost-to-go $\widetilde{v}$ for the infinite-horizon problem. Of course if we knew $\widetilde{v}$ the original problem would already be solved and we would gain nothing from the first-exit reformulation. However, if we only have an approximation to $\widetilde{v}$, choosing greedy actions with respect to $\widetilde{v}$ is likely to be worse than solving the above first-exit problem. While there is no proof that such a procedure will improve the control law, it usually does in practice.

There is an important difference between the approach we used and the way MPC has been used in the past. Traditionally MPC solves (in real time) a finite-horizon problem rather than a first-exit problem. That is, at each time step it computes an optimal trajectory extending N steps into the future, where N is predefined. The final state of such a trajectory can be any state; therefore the final cost h needs to be defined everywhere. In contrast, our method always computes a trajectory terminating at a state in $\mathcal{T}$, and so our final cost only needs to be defined for $x \in \mathcal{T}$. This is advantageous for two reasons: **(1)** guessing/approximating $\widetilde{v}$ is easier if we have to do it at only a subset of all states; **(2)** in the case of contact dynamics, if we define $\mathcal{T}$ as the set of states where contacts occur, then the real-time optimization does not need to deal with contact discontinuities; instead the effects of such discontinuities are incorporated in h.

One way to specify h is to use domain-specific heuristics as we demonstrated in our earlier work [5]. In the case of ball-bouncing, our formulation of MPC makes it particularly easy to come up with obvious heuristics – which basically define what is a good way to hit a ball. In other tasks such as walking, the heuristics may define what is a good way to place a foot on the ground.

Another approach, which is the main focus of this paper, is policy gradient. The vector w defines a function h, which in turn defines an MPC control law, which in turn can be evaluated empirically (through sampling) on the original infinite-horizon problem. In this way we can define the average empirical cost $L(w)$ for every possible w, and perform gradient descent on it.

III. Application to ball-bouncing

We have applied this method to the challenging problem of juggling two table tennis balls on one paddle. The paddle moves in three dimensions inside a workspace defined by a cylinder (with a center of the cylinder positioned at $[0\ 0\ 0]^T$) and has a fixed orientation (always stays in the horizontal plane). Let p_x, p_y, and p_z be positions of the paddle in their respective coordinates; b_x, b_y, and b_z be positions of one ball and o_x, o_y, and o_z be positions of the other ball. The state has 18 dimensions: $\mathbf{x} = \left[p_x\ p_y\ p_z\ \dot{p_x}\ \dot{p_y}\ \dot{p_z}\ b_x\ b_y\ b_z\ \dot{b_x}\ \dot{b_y}\ \dot{b_z}\ o_x\ o_y\ o_z\ \dot{o_x}\ \dot{o_y}\ \dot{o_z}\right]^T$ The dynamics are

$$\begin{aligned}
\dot{\mathbf{x}}_{pp} &= \mathbf{x}_{pv} \\
\dot{\mathbf{x}}_{pv} &= \mathbf{u} + [0\ 0\ \text{-}g]^T \\
\dot{\mathbf{x}}_{bp} &= \mathbf{x}_{bv} \\
\dot{\mathbf{x}}_{bv} &= [0\ 0\ \text{-}g]^T - d\|\mathbf{x}_{bv}\|\mathbf{x}_{bv} \\
\dot{\mathbf{x}}_{op} &= \mathbf{x}_{ov} \\
\dot{\mathbf{x}}_{ov} &= [0\ 0\ \text{-}g]^T - d\|\mathbf{x}_{ov}\|\mathbf{x}_{ov}
\end{aligned}$$

where the state variables are $\mathbf{x}_{pp} = [p_x\ p_y\ p_z]^T$, $\mathbf{x}_{pv} = [\dot{p}_x\ \dot{p}_y\ \dot{p}_z]^T$, $\mathbf{x}_{bp} = [b_x\ b_y\ b_z]^T$, $\mathbf{x}_{bv} = \left[\dot{b}_x\ \dot{b}_y\ \dot{b}_z\right]^T$, $\mathbf{x}_{op} =$

$[o_x\ o_y\ o_z]^T$, $\mathbf{x}_{ov} = [\dot{o}_x\ \dot{o}_y\ \dot{o}_z]^T$, g is the acceleration due to gravity, and d is the coefficient calculated based on the drag coefficient and other parameters. The goal is to juggle two balls given their desired velocity after the contact while keeping the paddle in the workspace (close to the center of the workspace). The control objective is to find the control $\mathbf{u}(t)$ that minimizes the performance index for the MPC solver

$$J_0 = h(x(T)) + \sum_{t=1}^{T-1} \ell(x(t), \mathbf{u}(t)) \tag{3}$$

$$\begin{aligned} \ell(x, \mathbf{u}) &= \|\mathbf{u}\|^2 + w_w \|\mathbf{p}_{xy}\|^2 + w_z (p_z - p_{target})^2 \\ &+ w_p \|\mathbf{p}_{xy} - \mathbf{b}_{xy}\|^2 \end{aligned} \tag{4}$$

$$\begin{aligned} h(x) &= w_v \|\mathbf{x}_{bv}^{contact} - \mathbf{v}_{target}\|^2 + w_w \|\mathbf{p}_{xy}\|^2 \\ &+ w_p \|\mathbf{p}_{xy} - \mathbf{b}_{xy}\|^2 \\ &+ \sum_{i=1}^{18} w_i x_i + \sum_{i=1}^{18} \sum_{j=1}^{18} w_{ij} x_i x_j \end{aligned} \tag{5}$$

where $\mathbf{x}_{bv}^{contact}$ is the ball velocity after the contact, $\mathbf{v}_{target}$ is the target ball velocity after the contact, p_{target} is the target z coordinate for the paddle, w_v is a weight on the velocity error, w_w is the weight on the distance from the middle of the workspace in the xy coordinates, w_z is the weight on the distance from p_{target}, w_p is the weight on the distance from the ball projection on the xy-plane, w_i and w_{ij} are parameters for linear and quadratic terms. All w's are assembled into one parameter vector, which will be optimized later using gradient descent. We used a time step of $10msec$ and set T to $1sec$.

A. Master cost

The master cost $\ell(x, u)$ is defined differently for contact and non-contact states. For the contact states it is set to track the z position of the paddle in the workspace and the position of the ball relative to the paddle in the following way

$$\begin{aligned} \ell(x, \mathbf{u}) &= m_z \left((p_z - p_{target}) / 0.08 \right)^8 \\ &+ m_p \left(\|\mathbf{p}_{xy} - \mathbf{b}_{xy}\| / 0.05 \right)^8 \end{aligned} \tag{6}$$

For all other (non-contact) states, it is defined similarly to the immediate cost

$$\begin{aligned} \ell(x, \mathbf{u}) &= \|\mathbf{u}\|^2 + w_w \|\mathbf{p}_{xy}\|^2 + w_z (p_z - p_{target})^2 \\ &+ w_p \|\mathbf{p}_{xy} - \mathbf{b}_{xy}\|^2 \end{aligned} \tag{7}$$

The performance of the current set of parameters $L(w)$ is calculated as the average of the master cost $\ell(x, u)$ along the state-control trajectories. We also added to $L(w)$ regularization terms of the form $\sum_{i=1}^{18} |w_i| + \sum_{i=1}^{18} \sum_{j=1}^{18} |w_{ij}|$ to encourage parameter values to be sparse. The master cost parameters m_z and m_p have been set both to 100. The parameters for the immediate cost (w_w, w_z, and w_p) have been set to 100, 36000, and 100 respectively and not changed during the optimization.

Notice that for the two-ball juggling the master cost does not explicitly track whether trajectories of two balls intersect or get too close. This is left to be handled by the return velocity calculation (as described in Section Target identification). We are considering implementing explicit tracking of trajectories as one of the items for future work.

B. Target identification

As noted by Schaal and Atkeson [6], "In order to juggle more than one ball, the balls must travel on distinct trajectories, and they should travel for a rather long time to facilitate the coordination of the other balls." In the one-ball configuration the return velocity is calculated based on the desired height and the target position (set to the center of the workspace), whereas in the two-ball configuration the target position is calculated based on where the other ball is expected to intersect $z = 0$ plane according to the formula: $\mathbf{p}_t = \mathbf{p}_2 - d_{target}\mathbf{p}_2 / \|\mathbf{p}_2\|$, where $\mathbf{p}_2$ is the position of the other ball, $\mathbf{d}_{target}$ is a desired distance between the balls, and $\mathbf{p}_t$ is the target position for the ball at the contact. The target height is calculated in such a way as to have one ball at the apex when the other ball is at contact.

C. System design

The system we simulated has three main components: Delta Haptic robot [13] with three degrees of freedom that has a regular table tennis paddle mounted on its effector; high-speed Vicon Bonita cameras, and table tennis balls covered with a reflective tape to enable tracking. One thing to note is that the size of the workspace is quite small—a cylinder $0.36m$ in diameter and $0.30m$ in height—which contributed to the challenge of finding good solutions. We used this system to run experiments on one- and two-ball bouncing and collect data [5]. In this work we are using these data to construct a good model and doing everything in simulation (leaving the application of gradient descent to the real system for future work).

The simulation engine was implemented in MATLAB and allowed to simulate a sequence of one- and two-ball bounces using the velocity noise model (as described in Section Noise model estimation). All parameters in the simulator were the same as used with the real system. The values of coefficients in the model–the coefficient of restitution and the drag–have been estimated from the experimental data collected on the real system.

Gradient descent was based on the BFGS Quasi-Newton method[1] with a cubic line search procedure and finite-difference gradient estimation (as implemented by the $fminunc$ function in MATLAB).

D. Noise model estimation

As the performance of the system is significantly affected by how the ball bounces off the paddle, to reproduce the same effects in simulation we estimated the noise model of the velocity after the contact based on data from the real robot.

[1]BFGS method is a method for solving nonlinear optimization problems that belongs to a class of hill-climbing optimization techniques that find a stationary point of a twice differentiable function. It is using a generalization of a secant method of finding the root of the first derivative (the cubic line search has been used for this optimization).

As Reist and D'Andrea [11] observed, "the main source of noise in the measured impact times and locations are stochastic deviations of the impact parameters, i.e. ball roundness and CR"; they also specifically called out table tennis balls as generating too much noise in the horizontal degrees of freedom at contact. In our case both of the aspects—ball roundness and the coefficient of restitution—have been affected by the reflective tape we applied to the table tennis balls, which introduced more noise for the system to deal with, even though we took extra care in applying the tape.

To estimate the noise model we applied markers to the paddle attached to the robot and then measured positions and velocity of the paddle and the ball with the highest frequency available from the Vicon system (240Hz) while juggling one ball. Because of the discretization effect, the actual point of contact may happen somewhere between the measurements. We calculated it by extrapolating trajectories using data from before and after the contact measurements and finding the point of their intersection (using z coordinates). We then estimated velocities and calculated how much the velocity after bounce deviated from the velocity predicted by the model. The errors in each dimension are shown in Fig. 1; during the simulation we used samples from these distributions to model noise for each bounce.

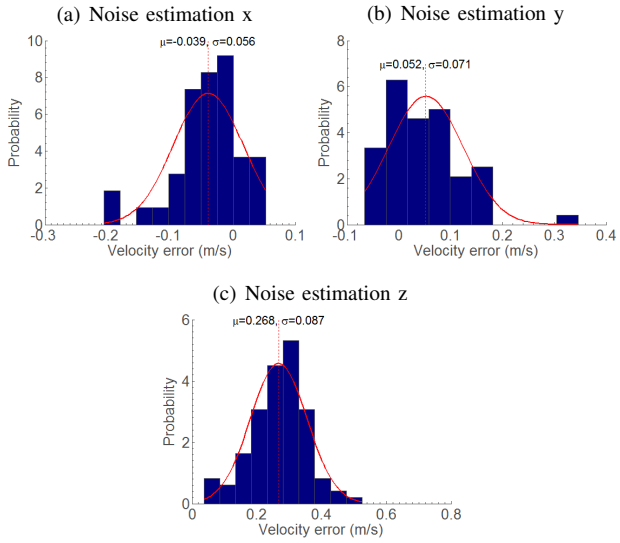

Fig. 1. Ball velocity after contact noise estimation and fit Gaussians for each of the dimensions.

IV. Experimental results

In this section we present and discuss experimental results from the tuning of the cost function, using our MATLAB simulator. The focus of these experiments has been on validating that the policy gradient method can improve performance of the system comparing to the hand-crafted cost function and on evaluating the difference.

The goal was not simply to improve the cost function, but also improve it in a way that produces good results with juggling one and two balls and allows to juggle with a small limit on the value of the control signal. As we observed in experiments on human subjects (not covered in this paper) and in the published research [8] the acceleration applied by human subjects is significantly lower than the acceleration applied by the robot. However, the current values of the parameters used in the cost function didn't allow to simply reduce the value, as it generated not acceptable solutions. So, the second goal for this optimization was to find a set of parameters that supported robust juggling using a low limit on the control signal (paddle acceleration).

For each bounce the velocity after contact was adjusted based on the noise model (as described in Section Noise model estimation). We applied the same sequence of random values to calculate velocity noise to make sure the optimization algorithm is using a common history for all calculated solution costs. A completely different sequence of values (with a different starting value and different duration) was used to test parameter values and to plot and analyze the results. We used 2.8s sequences for optimization (this allowed for about 10 contacts) and 24s for performance assessment and data analysis.

A. Performance with hand-crafted parameters

In the analysis of the results our focus was mostly on two-ball bouncing as it presented a much more difficult optimization problem than one-ball bouncing, which was already demonstrated to be robust enough [5]. For each of the conditions we looked at consistency of the interval between bounces and the height of each bounce as well as the positions of each bounce on the paddle (relative to its center). Fig. 2 shows the results for the initial values of the parameters. As can be seen from the figure, the system demonstrates sufficiently good performance; this assessment is supported by the fact that the real system using the same parameters is indeed capable of bouncing two balls, although only for about 25 bounces.

B. Improving performance using policy gradient

As the first step we used the parameter values that were designed manually and added linear and quadratic terms (18 + 18*18), with the total number of parameters being 345 (including 3 existing parameters). However, it didn't produce expected results as the algorithm we used could not demonstrate any significant improvement. Those improvements that were demonstrated looked like a case of overfitting as they didn't generalize well to other sequences of random noise. We attributed this result to the inherent challenge of using finite difference methods with parameters w having different order of magnitude [12].

To address this issue, we changed the approach and looked at first improving those parameters that were used in the hand-crafted cost function (as described in Section Application to ball-bouncing) and only then adding more parameters. This approach worked much better; Fig 3 shows how a better set of parameters was learned during one of the optimization runs (and a corresponding reduction in the master cost).

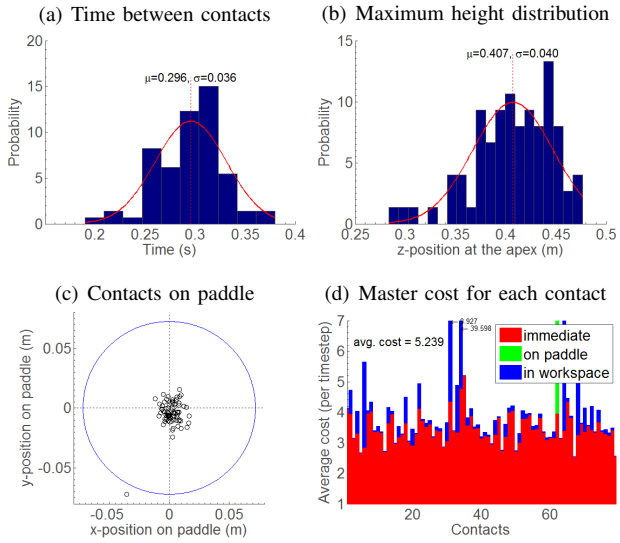

Fig. 2. Results for initial parameters with average cost 5.239; parameters set to $[w_p\ w_v\ w_w] = [10000\ 10000\ 1000]$ with $|u_{max}| \leq 40$.

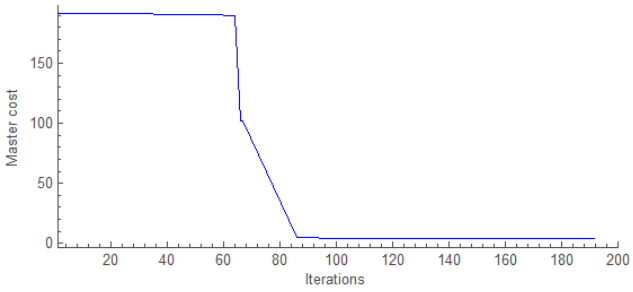

Fig. 3. Improvement in the master cost during optimization starting from initial parameter values $[w_p\ w_v\ w_w] = [10000\ 10000\ 1000]$.

The system performed well with the parameters optimized, which allowed us to reduce the control limit to $25m/s^2$. Fig. 4 shows the trajectories generated in one of the test runs and Fig. 5 shows the results from the same run. While it may not look like a significant improvement, it needs to be considered in the context of the system not being able to previously juggle at all with $25m/s^2$ limit and struggling with $30m/s^2$ limit.

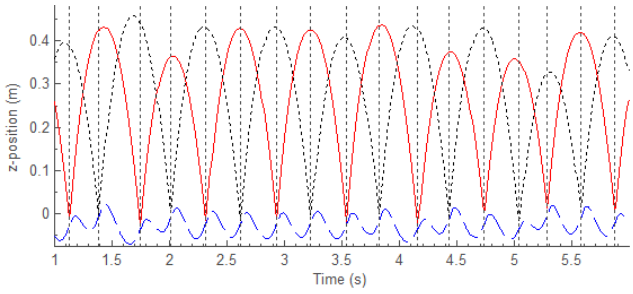

Fig. 4. Trajectories (z-coordinate) of the paddle (blue dashed) and the balls (red solid and black dotted) for $[w_p\ w_v\ w_w] = [9331.227812\ 336.372829\ 5758.953800]$ with $|u_{max}| \leq 25$.

After we added linear and quadratic parameters and ran

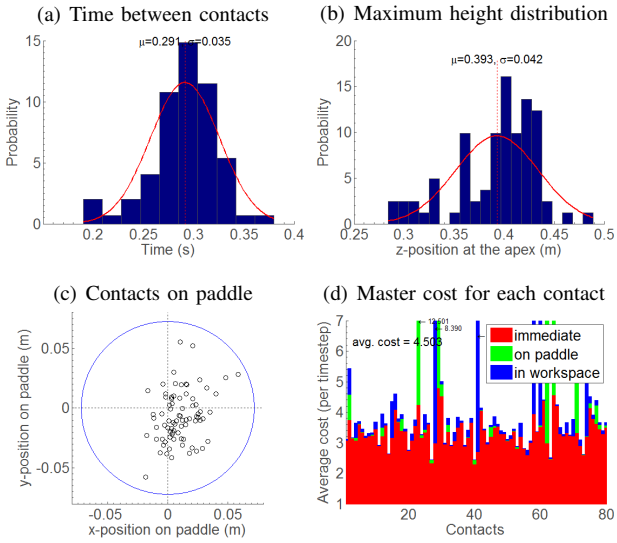

Fig. 5. Results for 3 parameters with average cost 4.503; parameters set to $[w_p\ w_v\ w_w] = [9331.227812\ 336.372829\ 5758.953800]$ with $|u_{max}| \leq 25$.

TABLE I
PARAMETER VALUES FOR LINEAR AND QUADRATIC TERMS

	p_x	p_y	p_z	$\dot{p_x}$	$\dot{p_y}$	$\dot{p_z}$	b_x	b_y	b_z	$\dot{b_x}$	$\dot{b_y}$	$\dot{b_z}$
linear	0	0	+	0	0	0	0	0	0	0	0	0
p_x	−	−	−	+	−	−	0	0	0	0	0	0
p_y	−	−	+	+	+	−	0	0	0	0	0	0
p_z	+	−	−	+	−	+	+	+	−	+	+	−
$\dot{p_x}$	−	−	−	+	−	−	0	0	0	0	0	0
$\dot{p_y}$	−	−	+	+	+	+	0	0	0	0	0	0
$\dot{p_z}$	+	−	+	+	−	+	0	0	0	0	0	0
b_x	0	0	0	0	0	0	0	0	0	0	0	0
b_y	0	0	0	0	0	0	0	0	0	0	0	0
b_z	0	0	0	0	0	0	0	0	0	0	0	0
$\dot{b_x}$	0	0	0	0	0	0	0	0	0	0	0	0
$\dot{b_y}$	0	0	0	0	0	0	0	0	0	0	0	0
$\dot{b_z}$	0	0	0	0	0	0	0	0	0	0	0	0

them through the optimization process (using the $fminunc$ function with default options), it turned out that only some of the terms have non-zero values. As described in Section Master cost, we added the absolute values of the weights to the performance criterion, which encourages sparse solutions. Table I shows that only paddle-related (cross-)terms had impact on the master cost (+ and − indicate non-zero values; only showing first 12 out of total 18 as the rest are zeros). As the result of this analysis, we only used first three linear terms and 36 quadratic terms for paddle position and velocity values.

First several runs of the optimization algorithm on the initial set of parameters provided important insight that, counter-intuitively, reducing the weight on the velocity after contact error not only didn't degrade the performance, but actually noticeably improved it. Unfortunately, this set of parameters didn't allow for much of further improvement, but using this insight and our domain knowledge, we adjusted the set of the initial values and used the simulator to learn a better set. Fig. 6 shows the improvement of the master cost during the optimization for one set of initial values and Fig. 7 demonstrates results after this optimization.

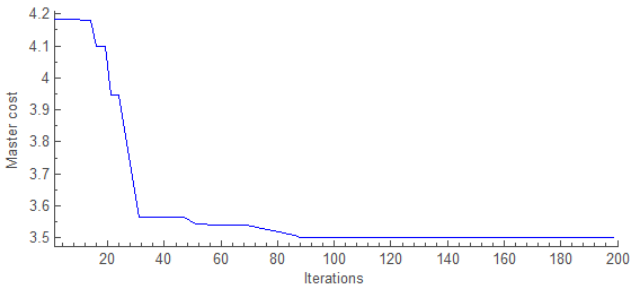

Fig. 6. Improvement in the master cost during optimization starting from initial parameter values $[w_p\ w_v\ w_w] = [10000\ 100\ 1000]$.

It is worth noting that these results compare well with the earlier results in terms of consistency of contact timing and position on the paddle even though we lowered the limit to $21m/s^2$ and the original results (Fig. 2) were taken at $40m/s^2$, as the system was struggling even at $30m/s^2$, which is why we were interested in varying the limit in this optimization.

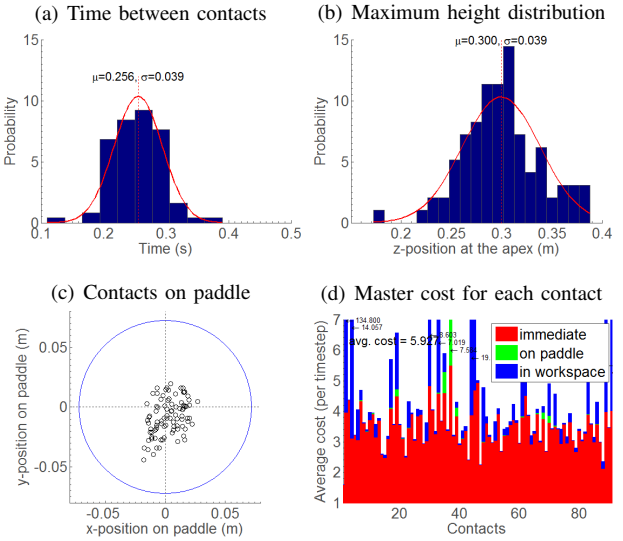

Fig. 7. Results for 3 parameters with average cost 5.927; parameters set to $[w_p\ w_v\ w_w] = [9999.980910\ 74.581537\ 1000.174549]$ with $|u_{max}| \leq 21$.

After getting to the desired range of $20 - 22m/s^2$ (from $40m/s^2$ it was originally set to) we added other parameters (linear and quadratic terms) and varied parameters of the gradient estimator looking for further improvements. We used the result of optimization shown in Fig. 6 and extended the number of parameters to 42 (by adding 3 linear and 36 quadratic terms). As the result of this optimization, we were able to get a slightly better master cost, while keeping the the limit on the control signal set to $21m/s^2$, which is the best result we have been able to demonstrate so far. Fig. 8 shows the results from the test run for these parameter values. Fig. 9 shows the trajectories of the paddle and two balls. While the results shown in Fig. 7 and Fig. 8 look similar, the average cost with the extended set of parameters has been reduced by almost 8% (from 5.927 to 5.470).

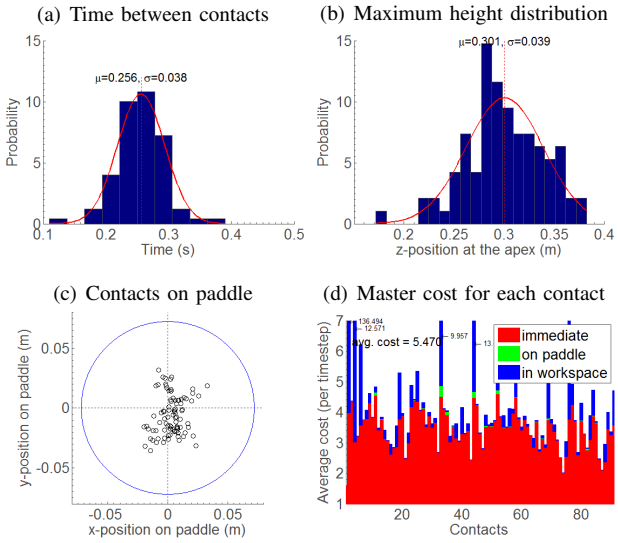

Fig. 8. Results for 42 parameters with average cost 5.470; parameters set to $[w_p\ w_v\ w_w\ ...] = [9999.981038\ 74.557687\ 1000.174682\ ...]$ with $|u_{max}| \leq 21$.

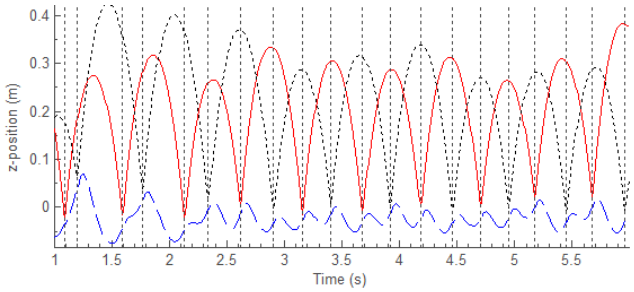

Fig. 9. Trajectories (z-coordinate) of the paddle (blue dashed) and the balls (red solid and black dotted).

V. Conclusion

In this work we implemented and tested tuning of the final cost for a first-exit MPC problem of juggling two balls using policy gradient with a parametric function approximator. We simulated the system that mimics a real robot with significant physical constraints (a small workspace and no tilt control) and were able to robustly bounce two balls on the same paddle in simulation using a noise model inferred from real data.

It should be noted that the successful application of policy gradient methods here required some domain insight, and was interleaved with improving the manual design rather than being applied once "off-the-shelf". We believe this is because we are solving a very hard problem, involving an inherently unstable system where the smallest error leads to failure. The new method is likely to work off-the-shelf on simpler problems, and provide significant benefits by combining the advantages of model-predictive control and policy gradient methods.

We are pursuing extending the work that has been done so far in several directions: **(1)** testing the performance on the real robot and **(2)** tuning the final cost of the solution

using approximate policy iteration. This approach relies on the fact that at the optimal solution, the final cost h coincides with the optimal differential cost-to-go $\widetilde{v}$ for the infinite-horizon problem. Thus h can be improved in the following way. For a set of starting states in $\mathcal{T}$, run the MPC control law corresponding to the current $h(x;w)$, and obtain learning targets for $\widetilde{v}(x)$ on $x \in \mathcal{T}$, using Temporal Difference learning for example. Then adapt the parameters w so that $h(x;w)$ gets closer to these learning targets, and iterate.

We also plan to do a comparison with human subjects to explore how the behavior of the robot is different from humans on similar tasks and to bring the cost model closer to the models that humans may be using. We expect that the behavior of the system running with the adjusted parameters will more closely resemble the behavior of human subjects than the current behavior and we plan to confirm this in future experiments. These comparisons with human subjects will hopefully suggest additional improvements in our control algorithm.

ACKNOWLEDGMENT

This work was supported by the US National Science Foundation.

REFERENCES

[1] J. Bagnell and J. Schneider. Covariant policy search. In *International Joint Conference on Artificial Intelligence*, 2003.

[2] S. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, 2002.

[3] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71:11801190, 2008.

[4] R. Sutton, D. Mcallester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000.

[5] P. Kulchenko and E. Todorov (2011). First-exit model predictive control of fast discontinuous dynamics: Application to ball bouncing. To appear in *2011 IEEE International Conference on Robotics and Automation*

[6] S. Schaal and C. G. Atkeson (1993). Open Loop Stable Control Strategies for Robot Juggling. In proceedings of the *1993 IEEE international conference on Robotics and Automation*

[7] A. Rizzi and D. Koditschek (1994). Further progress in robot juggling: Solvable mirror laws. In proceedings of the *1994 IEEE international conference on Robotics and Automation*

[8] D. Sternad, M. Duarte, H. Katsumata and S. Schaal (2000). Dynamics of bouncing ball in human performance. *Physical Review E*, vol 63

[9] R. Ronsse, K. Wei and D. Sternad (2010). Optimal control of a hybrid rhythmic-discrete task: The bouncing ball revisited. *Journal of Neurophysiology* 104: 2484-2493

[10] E. Todorov (2004). Optimality principles in sensorimotor control. *Nature Neuroscience* 7: 907-915

[11] P. Reist and R. D'Andrea (2009). Bouncing an Unconstrained Ball in Three Dimensions with a Blind Juggling Robot. In proceedings of the *2009 IEEE international conference on Robotics and Automation*

[12] J. Peters and S. Schaal (2006). Policy Gradient Methods for Robotics. In proceedings of the *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*

[13] S. Grange, F. Conti, P. Helmer, P. Rouiller, C. Baur. (2001) The Delta Haptic Device. In proceedings of the *Eurohaptics'01*