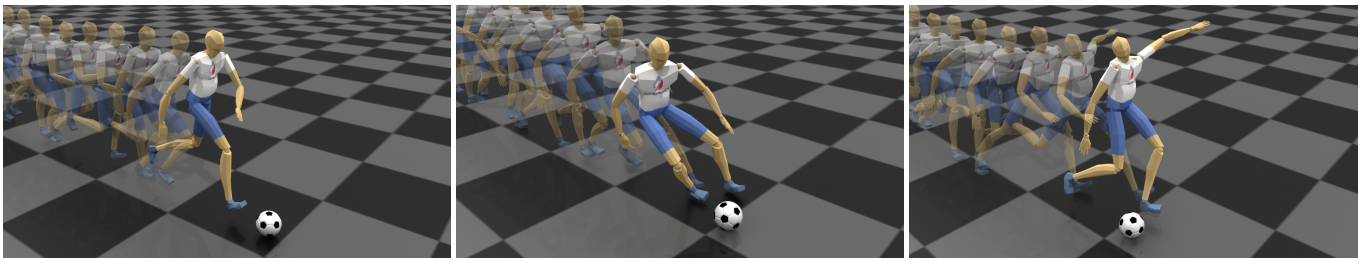
Physics-based Full-body Soccer Motion Control for Dribbling and Shooting

SEOKPYO HONG, Visual Media Lab, KAIST
DASEONG HAN, Handong Global University
KYUNGMIN CHO, Visual Media Lab, KAIST
JOSEPH S. SHIN (formerly SUNG YONG SHIN), Handong Global University and KAIST
JUNYONG NOH*, Visual Media Lab, KAIST



(a) Dribbling forward (b) Dribbling to the side (c) Shooting

Fig. 1. Soccer motions

Playing with a soccer ball is not easy even for a real human because of dynamic foot contacts with the moving ball while chasing and controlling it. The problem of online full-body soccer motion synthesis is challenging and has not been fully solved yet. In this paper, we present a novel motion control system that produces physically-correct full-body soccer motions: dribbling forward, dribbling to the side, and shooting, in response to an online user motion prescription specified by a motion type, a running speed, and a turning angle. This system performs two tightly-coupled tasks: data-driven motion prediction and physics-based motion synthesis. Given example motion data, the former synthesizes a reference motion in accordance with an online user input and further refines the motion to make the character kick the ball at a right time and place. Provided with the reference motion, the latter then adopts a Model Predictive Control (MPC) framework to generate a physically-correct soccer motion, by solving an optimal control problem that is formulated based on dynamics for a full-body character and the moving ball together with their interactions. Our demonstration shows the effectiveness of the proposed system that synthesizes convincing full-body soccer motions in various scenarios such as adjusting the desired running speed of the character, changing the velocity and the mass of the ball, and maintaining balance against external forces.

CCS Concepts: • **Computing methodologies** → **Animation**; **Physical simulation**.

Additional Key Words and Phrases: 3D character animation, physics-based simulation, motion control

ACM Reference Format:
Seokpyo Hong, Daseong Han, Kyungmin Cho, Joseph S. Shin (formerly Sung Yong Shin), and Junyong Noh. 2019. Physics-based Full-body Soccer Motion Control for Dribbling and Shooting. *ACM Trans. Graph.* 38, 4, Article 74 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3322963

1 INTRODUCTION

Playing with a soccer ball is not easy even for a real human. To dribble a soccer ball, for example, the player runs after the ball, following a time-varying trajectory while occasionally touching it with a foot in the desired direction and speed, which involves highly complex, subtle dynamics for full-body human motions and their interactions with the ball. Soccer motions have been studied extensively in robotics and character animation [Barrett et al. 2010; Choi et al. 2015, 2016; Hester et al. 2010; Jain and Liu 2009; Leottau et al. 2014; Sung et al. 2013; Yi et al. 2013]. However, fundamental issues still remain unsolved in the synthesis of even basic soccer motions such as dribbling and shooting: how to adjust the footsteps of the character to make it kick the ball with a foot at a right time and place, how to achieve physics-based online motion control over the character that interacts with a ball, and how to make the character balance against unexpected external forces while controlling the ball. These issues have not been settled yet although there are partial solutions for simple cases: kicking a ball in a stationary stance [Ha

*Corresponding author

Authors' addresses: Seokpyo Hong, Visual Media Lab, KAIST, sphorpin89@gmail.com; Daseong Han, Handong Global University, dshan@handong.edu; Kyungmin Cho, Visual Media Lab, KAIST, cluemaker@kaist.ac.kr; Joseph S. Shin (formerly Sung Yong Shin), Handong Global University and KAIST, josephsyshin@gmail.com; Junyong Noh, Visual Media Lab, KAIST, junyongnoh@kaist.ac.kr.

0730-0301/2019/7-ART74 $15.00
https://doi.org/10.1145/3306346.3322963

and Liu 2016] and dribbling a large sphere at a low speed [Peng et al. 2017].

In order to deal with these issues comprehensively, we present a data-driven, online motion control system that synthesizes a physically-correct full-body character motion based on a Model Predictive Control (MPC) framework, in response to an online user specification. Our system predicts a reference motion in accordance with the ball trajectory and generates the corresponding character motions, to make the character kick the moving ball at a correct time and position. The system consists of two tightly-coupled components: data-driven motion prediction and physics-based motion synthesis. Given the previous character motion, the first component synthesizes a reference motion in an online manner, using a motion library. This component also adjusts the character footsteps so as to touch the ball timely in the next component. Provided with the reference motion, the second component solves an optimal control problem formulated based on dynamics for the full-body character and the moving ball as well as their interactions, to generate a physically-correct soccer motion. Built on the MPC framework with smoothed contact dynamics [Han et al. 2016; Todorov 2014], our system achieves an interactive performance for robust motion tracking while ensuring the balancing of the character against unexpected perturbation forces. Unlike the previous work, our system can synthesize full-body dribbling and shooting motions that controls a dynamically-moving soccer ball in a physically-correct manner.

The contributions of this paper are three-fold: First, our approach provides the physically-based control system for full-body soccer motions that interact with a moving soccer ball, employing predictive motion synthesis and optimal feedback control. The generated motions are physically-correct and cover fundamental soccer motions such as dribbling forward, dribbling to the side, and shooting. To our knowledge, those are the first non-trivial, plausible soccer motions synthesized in character animation. Second, our system is equipped with a feedback loop to synthesize a reference motion with the footsteps automatically adjusted to the ball-kicking pose. The reference motion guides the swing foot to kick the rolling ball at a right moment and position. The use of the feedback loop further enhances the quality of synthesized motions. Finally, based on a differentiable dynamics model with smoothed contacts, we formulate an optimal control problem with novel cost terms for soccer-specific physics-based motion control, which facilitates both timely ball kicking and reference motion tracking. Guided by the reference motion that is generated by data-driven motion prediction, our full-body character performs soccer tasks such as dribbling and shooting successfully while preserving the efficiency and robustness of the baseline MPC method.

The remainder of this paper is organized as follows: Section 2 reviews previous work related to our framework. In Section 3, we present an overview of our system. Section 4 presents how to generate a reference motion, and Section 5 discusses physics-based simulation for soccer motion synthesis. In Section 6, we show results to demonstrate the effectiveness of the proposed system. In Section 7, we conclude this paper with the summary of our research. Finally, Section 8 discusses the limitations and future work.

2 RELATED WORK

We begin with a literature survey on kinematic motion synthesis and physics-based motion control, which are related to data-driven motion prediction and physics-based motion synthesis, respectively. We then review previous work on soccer motion synthesis in robotics and character animation.

Kinematic Locomotion Control: Graph-based methods were popular in the early stage of data-driven motion synthesis research [Arikan and Forsyth 2002; Kovar et al. 2002; Lee et al. 2002]. Allowing motion transitions between example motion clips only at pre-chosen transition points, such a method is difficult to be used for the online synthesis of highly reactive motions to the ever-changing environment, according to a user motion specification. Recently, learning-based approaches have been proposed for real-time motion synthesis [Holden et al. 2017, 2016; Lee et al. 2010b; Levine et al. 2012; Treuille et al. 2007]. However, it is not clear how to extend these approaches to deal with a moving external object such as a soccer ball in an online manner. For our data-driven motion prediction, we propose a variant of the motion matching scheme [Clavet 2016] based on the motion field technique [Lee et al. 2010b]. Unlike learning-based approaches, our scheme enables agile and smooth motion transitions for the synthesis of soccer motions such as dribbling and shooting without heavy preprocessing.

Physics-based Motion Control: Physics-based motion control has been studied intensively for the last decade. It has dealt with not only locomotive motions [Da Silva et al. 2008; De Lasa et al. 2010; Han et al. 2014; Lee et al. 2010a; Mordatch et al. 2010; Muico et al. 2011; Peng et al. 2017; Yu et al. 2018], but also non-locomotive motions such as martial arts kicks, cartwheels, crawling, jumping, and dancing [Al Borno et al. 2013; Han et al. 2016; Liu et al. 2016, 2012, 2010; Peng et al. 2018]. In general, dynamics equations for full-body locomotion control suffer from discontinuities caused by the change of contact points with the environment. This precludes a derivative-based optimization technique from determining contact points and their timings with the environment. Liu et al. [2016] addressed this issue with a control graph that represents rapid contact changes in example motion clips. Han et al. [2016] resolved contact discontinuities by exploiting smoothed contact dynamics [Todorov 2014]. Their methods focused on reference motion tracking and did not consider the interaction of the character with a moving external object. To synthesize soccer motions with interactive performance, we adapt an MPC framework [Han et al. 2016] to our problem setting in two aspects. First, we build the soccer-specific framework that controls the character to timely kick a moving ball while chasing it. Second, we enhance the framework to equip with a feedback control loop between data-driven motion prediction and physics-based motion simulation.

Soccer Motions: Few results have appeared in literature for physics-based full-body soccer motion control. In robotics, there have been attempts for a humanoid robot to play with a soccer ball [Barrett et al. 2010; Hester et al. 2010; Leottau et al. 2014; Sung et al. 2013; Yi et al. 2013]. Due to complex physical mechanisms of humanoid robots, their soccer motions were limited to stationary kicking or walk-kicks. In character animation, various techniques for soccer

motion synthesis have been reported. Jain and Liu [2009] proposed an interactive motion editing method for human interactions with a soccer ball. Given captured human motions, this method automatically created the ballistic path of a ball based on simple rigid body dynamics, considering its contact with a human body part. The contact timing and position between the ball and the body part were manually specified by the user. Focusing on the editing of human motions and ballistic ball paths, their method did not take into account full-body dynamics. Choi et al. [2015; 2016] proposed a data-driven method that can generate 2D juggling motions including hand juggling, soccer juggling, and in-place basketball dribbling, by analyzing kinematic information such as end-effector velocities. In physics-based soccer motion synthesis, Ding et al. [2015] proposed a low-dimensional linear feedback strategy that effectively searches for a control policy to generate a simple ball-kicking motion with the hips fixed. Peng et al. [2017] suggested a scheme that learns a control policy for walking motions using a soccer-specific reward function, in order to make the character dribble a ball toward the desired destination. Unfortunately, the resulting motion looks more like simply pushing a large sphere with a foot, instead of dribbling a soccer ball.

Commercial soccer games such as *Pro Evolution Soccer* series [KONAMI] and *FIFA* series [EASports] have been popular around the world for a long time. Mainly based on example-based motion synthesis, these games show a variety of visually-plausible soccer motions. However, these motions often exhibit visual artifacts in dribbling and shooting due to the lack of physics. To the best of our knowledge, there have been no games that exhibit physically-correct full-body soccer motions such as dribbling and shooting in an online manner.

3 OVERVIEW

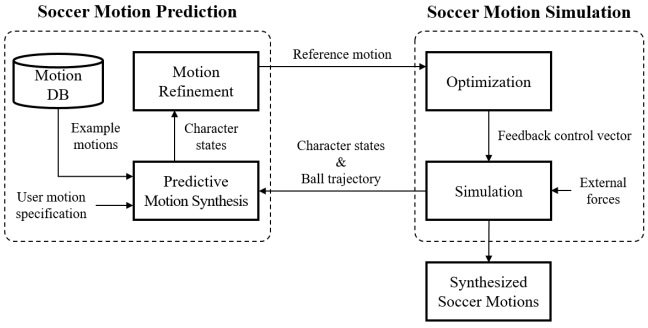


Fig. 2. System overview

Our system consists of two components: data-driven soccer motion prediction and physics-based soccer motion simulation, as depicted in Figure 2. The two components work in tandem with each other to generate soccer motions. Given a motion library, the first component synthesizes a reference motion, which is sufficiently lengthy to contain a kick pose, via predictive motion search in the motion library, given the user prescription. The resulting motion is further refined to place the swing foot at the kick frame at the target ball position that is automatically determined.

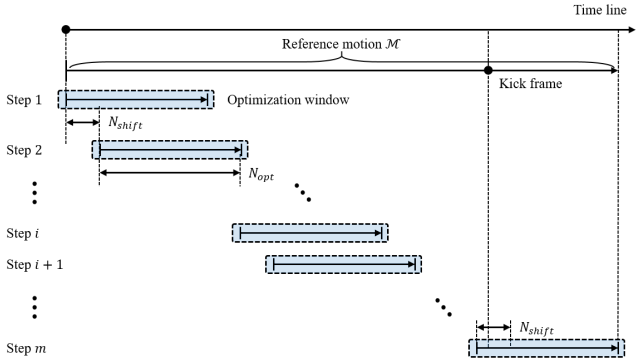


Fig. 3. Shifting window

Given reference motion $\mathcal{M}$, the second component produces a locally-optimal feedback control vector, which is used for physics-based full-body motion simulation in accordance with a simulated ball trajectory. To produce the control vector, this component solves an optimal control problem within the optimization window of N_{opt} frames while repeatedly shifting the window m times along $\mathcal{M}$ by N_{shift} frames per shift, as depicted in Figure 3. At each window, the optimal control is guided by the corresponding portion of $\mathcal{M}$. The first component synthesizes $\mathcal{M}$ such that the last window contains a kick pose in its first N_{shift} frames. The resulting motion from the second component is fed back to the first component for the prediction of the next reference motion.

4 SOCCER MOTION PREDICTION

In this section, we describe how to decide both the ball-kicking time and position while synthesizing a reference motion. We also explain how to refine the reference motion so as to place the swing foot of the kick pose at a simulated ball position. We begin with providing the representation of a reference motion as well as that of the ball trajectory.

4.1 Motion and trajectory representations

Our system performs physics-based motion control for soccer motion synthesis, guided by reference motion $\mathcal{M}$ that is represented as follows:

$$\mathcal{M} = \{C^i | i = 1, 2, ..., N_p\},$$

where each C^i is a character state at frame i, and N_p is the size of $\mathcal{M}$. C^i consists of pose Q^i and displacement D^i that are defined as follows:

$$C^i = (Q^i, D^i), \text{where}$$
$$Q^i = (\mathbf{p}_0^i, \mathbf{q}_0^i, \mathbf{q}_1^i, ..., \mathbf{q}_{N_J-1}^i) \text{ and } D^i = (\Delta\mathbf{p}_0^i, \Delta\mathbf{q}_0^i, \Delta\mathbf{q}_1^i, ..., \Delta\mathbf{q}_{N_J-1}^i).$$

Here, $\mathbf{p}_0^i \in \mathbb{R}^3$ and $\mathbf{q}_0^i \in \mathbb{S}^3$ respectively are the position and the orientation of the root, and $\mathbf{q}_j^i \in \mathbb{S}^3$ is the orientation of joint j. $\Delta\mathbf{p}_0^i \in \mathbb{R}^3$ and $\Delta\mathbf{q}_0^i \in \mathbb{S}^3$ are the positional and the angular displacements of

the root from frame i to frame $i+1$, respectively. Similarly, $\Delta\mathbf{q}_j^i \in \mathbb{S}^3$ is the angular displacement of joint j. N_J is the number of joints of a character. We specify the trajectory $\mathcal{B}$ of a moving ball as follows:

$$\mathcal{B} = \{B^i | i = 1, 2, ..., N_b\},$$

where each $B^i \in \mathbb{R}^3$ is the ball position. N_b is the size of $\mathcal{B}$, that is, the length of the ball trajectory for reference motion synthesis.

4.2 Predictive motion synthesis

Our system synthesizes a new reference motion upon receiving a motion specification, having used up the current reference motion, or experiencing large external forces unexpectedly. As preprocessing, we first annotate the example motions in the motion library to classify motion clips into three groups according to their motion types: dribbling forward, dribbling to the side, and shooting. We then manually mark all kick poses in the motion data. We also mark each short motion sub-clip containing a kick pose in the middle as a pre-specified kick motion. Specifying the turning angle of each motion clip when capturing it, we finally label the angle at each frame in the clip automatically with the same turning angle. Let $G = \{g_{fwd}, g_{side}, g_{shoot}\}$ and $\bar{K} = \{\bar{Q}_k | k = 1, 2, ..., N_K\}$ be a set of all motion groups and that of all kick poses, respectively. Here, g_{fwd}, g_{side}, and g_{shoot} denote the groups containing motion data for dribbling forward, dribbling to the side, and shooting, respectively. $\bar{Q}_k$ and N_K represent a kick pose and the total number of kick poses in the motion data, respectively.

Our system produces a reference motion according to user motion specification $U = \{t, v, \theta\}$, where $t \in \{fwd, side, shoot\}$ is the type of the soccer motion, and v and θ are the desired speed and the turning angle, respectively. Motion type t determines the group of motion clips to work with for reference motion synthesis. Provided with user motion specification U and character state $C^i = (Q^i, D^i)$ at current frame i of the reference motion, our system selects character state $\bar{C}^p = (\bar{Q}^p, \bar{D}^p)$ from the motion clips in the motion group corresponding to motion type t, where superscript p denotes the frame index of the chosen character state in the motion clip. To choose $\bar{C}^p$, the system minimizes our task-dependent cost function (to be given soon) over all character states $\bar{C}^j = (\bar{Q}^j, \bar{D}^j)$ in the motion group.

There are two cases depending on whether or not $\bar{C}^p$ is contained in a pre-specified motion. Suppose that $\bar{C}^p$ is contained in a pre-specified motion. In this case, the system replays the pre-specified kick motion starting from $\bar{C}^p$ to complete the reference motion. Now, suppose that $\bar{C}^p$ is not contained in a pre-specified kick motion. In this case, $\bar{C}^p$ together with next state $\bar{C}^{p+1}$ in the same motion clip is used to generate next character state $C^{i+1} = (Q^{i+1}, D^{i+1})$. To achieve it, our system first computes guiding character state $\hat{C}^{i+1}$ as follows:

$$\hat{C}^{i+1} = (\hat{Q}^{i+1}, \hat{D}^{i+1}) = (Q^i \oplus \bar{D}^p, \bar{D}^{p+1}), \text{ where}$$
$$Q^i \oplus \bar{D}^p = (\mathbf{p}_0^i + \Delta\bar{\mathbf{p}}_0^p, \Delta\bar{\mathbf{q}}_0^p \mathbf{q}_0^i, \Delta\bar{\mathbf{q}}_1^p \mathbf{q}_1^i, ..., \Delta\bar{\mathbf{q}}_{N_J-1}^p \mathbf{q}_{N_J-1}^i).$$

This system then acquires next character pose Q^{i+1} by blending $\hat{Q}^{i+1}$ and $\bar{Q}^{p+1}$ with a larger weight used for the former, to make the resulting motion as smooth as possible. Next character displacement D^{i+1} is obtained by blending $\hat{D}^{i+1}$ and $\bar{D}^{p+1}$ in the similar manner. This process is repeated to generate character states one by one until $\bar{C}^p$ is contained in a pre-specified kick motion. For the details on how to update and blend the character states, we refer the readers to [Lee et al. 2010b].

In the remainder of this section, we first present our task-dependent cost function that guides the predictive motion synthesis. We then describe how to initialize the reference motion at the beginning of the predictive motion synthesis.

Cost function: The cost function guides the synthesis of soccer-specific reference motion $\mathcal{M} = \{C^i | i = 1, 2, ..., N_p\}$, given user motion specification $U = (t, v, \theta)$. Our system chooses character state $\bar{C}^p$ from the motion clips in a given motion group using the cost function that consists of four terms:

$$cost = \alpha_r S_r(v, \bar{C}^j) + \alpha_d S_d(C^i, \bar{C}^j) + \alpha_b S_b(C^i, \bar{C}^j, \mathcal{B}) + \alpha_\theta S_\theta(\bar{C}^j, \theta),$$

where, α_r, α_d, α_b, and α_θ are weights for their respective cost terms. $S_r(v, \bar{C}^j)$ is the cost for achieving desired speed v. $S_d(C^i, \bar{C}^j)$ enhances the inter-frame dependence between consecutive character states while satisfying the motion specification. $S_b(C^i, \bar{C}^j, \mathcal{B})$ guides the character to kick the ball at a right time and place. $S_\theta(\bar{C}^j, \theta)$ enforces the desired turning angle θ. In the remainder of this section, we discuss each of the cost terms in detail.

First cost term $S_r(v, \bar{C}^j)$ is defined as follows:

$$S_r(v, \bar{C}^j) = \sum_{l=0}^{N_s-1} \left(v - \frac{\|\Delta\bar{\mathbf{p}}_0^{j+l}\|_{xy}}{h} \right)^2,$$

where $\bar{C}^j$ is the character state at frame j of a motion clip, $\Delta\bar{\mathbf{p}}_0^{j+l}$ is the root displacement at the l-th frame from frame j, and h is the time step size between two consecutive frames. $\|\Delta\bar{\mathbf{p}}_0^{j+l}\|_{xy}$ is the projected length of $\Delta\bar{\mathbf{p}}_0^{j+l}$ onto the ground plane. This term looks ahead to the future root speed beyond state $\bar{C}^j$ of a motion clip to select character state $\bar{C}^p$ from the motion data, by minimizing the sum of squared root speed differences from desired speed v for N_s states starting from any $\bar{C}^j$ in every motion clip in the chosen motion group.

Second term $S_d(C^i, \bar{C}^j) = \sum_{l=1}^{N_d} d(C^{i-l}, \bar{C}^{j-l})$ looks back on the character states starting from current state C^i in the synthesized reference motion to choose character state $\bar{C}^p$, where N_d is the number of previous states that are considered. For motion distance $d(C^{i-l}, \bar{C}^{j-l})$ between C^{i-l} of the reference motion and $\bar{C}^{j-l}$ of a motion clip, we refer the readers to [Lee et al. 2010b].

Third term $S_b(C^i, \bar{C}^j, \mathcal{B})$ is defined as follows:

$$S_b(C^i, \bar{C}^j, \mathcal{B}) = \begin{cases} \|\bar{\mathbf{f}}^{j+k} - (B^{i+k} - (\bar{\mathbf{p}}_0^j + \mathbf{p}_0^i))\|_{xy}^2 & \text{if } \bar{\mathbf{f}}^{j+k} \text{ exists} \\ \epsilon & \text{otherwise} \end{cases},$$

where $\bar{\mathbf{f}}^{j+k} \in \mathbb{R}^3$ is the kick foot position for state $\bar{C}^{j+k}$ in the motion sub-clip starting from state $\bar{C}^j$. B^{i+k} is the ball position at frame $i+k$ of ball trajectory $\mathcal{B}$. $\bar{\mathbf{p}}_0^j$ is the root displacement of the character at frame j with respect to $\bar{\mathbf{p}}_0^0$ of the motion clip. This term is used to select character state $\bar{C}^p$ with the closest kick foot in $\bar{K}$ to the ball if

there is any kick foot in a motion clip; otherwise, it returns a small positive constant ϵ.

Last term $S_\theta(\bar{C}^j, \theta)$ plays a role of finding character state $\bar{C}^p$ such that the projected root direction onto the ground is close to input turning angle θ. We set $S_\theta(\bar{C}^j, \theta) = (\bar{\theta}^j - \theta)^2$, where $\bar{\theta}^j$ is a pre-specified turning angle of $\bar{C}^j$.

State initialization: In order to facilitate the looking back feature with $S_d(C^i, \bar{C}^j)$, the reference motion should have at least N_d character states. However, no character states are available in the current reference motion at the beginning of predictive motion synthesis. There are two cases depending on whether or not the physics-based motion simulation has ever been completed. We first consider the easier case in which the physics-based motion simulation has been completed. In this case, the latest synthesized motion is available so that our system initializes reference motion $\{C^{1-i} | i = 1, 2, ..., N_d\}$ with the last N_d states of the simulated motion. This facilitates the feedback loop between soccer motion prediction and soccer motion simulation.

The other case occurs at the very beginning of predictive motion synthesis. In this case, our system initializes reference motion $\{C^{1-i} | i = 1, 2, ..., N_d\}$ with a motion sub-clip consisting of N_d states. To do it, the system selects initial character state C^1_{init} corresponding to initial user specification $U_0 = (t_0, v_0, \theta_0)$ from the motion clips in the group corresponding to t_0, by minimizing initial cost function S_{init} over all states $\bar{C}^j$ of the motion data, which is defined in the same manner as first cost term $S_r(v, \bar{C}^j)$:

$$S_{init}(v_0, \bar{C}^j) = \sum_{l=0}^{N_s-1} \left(v_0 - \frac{\|\Delta \bar{\mathbf{p}}_0^{j+l}\|_{xy}}{h} \right)^2 .$$

This cost function finds C^1_{init} by minimizing $S_{init}(v_0, \bar{C}^j)$ over all states $\bar{C}^j$ in the motion group. Upon determining C^1_{init}, the system chooses N_d states starting from C^1_{init} in the chosen motion clip to initialize reference motion $\{C^{1-i} | i = 1, 2, ..., N_d\}$, where $C^{1-N_d} = C^1_{init}$. Root orientation and displacement, $\mathbf{q}_0^{1-i}$ and $\Delta\mathbf{p}_0^{1-i}$ in each state C^{1-i} are further refined to be aligned in a direction from the root of the character to the initial ball position.

4.3 Motion refinement

Through predictive motion synthesis, we obtain the best-fitted reference motion to the cost function. However, the ball and the kick pose may not be aligned well with the reference motion obtained in Section 4.2 due to the lack of motion data in the motion library. In other words, the position of the kick foot of this motion could slightly deviate from the corresponding ball position, which may cause a problem in physics-based simulation discussed later in Section 5.2. The character states are further refined with motion transformation techniques [Gleicher 2001; Park et al. 2002] so as to make the character chase and kick the rolling ball at the right position and time.

Given reference motion $\mathcal{M} = \{C^i | i = 1, ..., N_p\}$, our motion refinement process transforms $\mathcal{M}$ to align the kick foot with the simulated soccer ball position. To describe this process, we start with defining the two displacement vectors from root position $\mathbf{p}_0^1$ for state C^1 at the first frame to kick foot position $\mathbf{f}^\kappa$ for state C^κ and to ball position B^κ at kick frame κ, respectively:

$$\mathbf{v}_0 = \mathbf{f}^\kappa - \mathbf{p}_0^1, \quad \mathbf{v}_1 = B^\kappa - \mathbf{p}_0^1.$$

With these two vectors, we derive scaling factor $\eta \in \mathbb{R}$ and rotation vector $\boldsymbol{\theta}_r \in \mathbb{R}^3$, which are used to transform the whole reference motion to make the kick foot position coincide with the target ball position:

$$\eta = \frac{\|(\mathbf{v}_1)_{xy}\|}{\|(\mathbf{v}_0)_{xy}\|},$$
$$\boldsymbol{\theta}_r = \cos^{-1}((\hat{\mathbf{v}}_0)_{xy} \cdot (\hat{\mathbf{v}}_1)_{xy})((\hat{\mathbf{v}}_0)_{xy} \times (\hat{\mathbf{v}}_1)_{xy}).$$

Here, the hat symbol represents the normalization of a given vector. For each frame $i = 1, 2, ..., N_p$, we transform the position of the root and its linear and angular inter-frame displacements using η and $\boldsymbol{\theta}_r$:

$$(\mathbf{p}_0^i)'_{xy} = \begin{cases} (\mathbf{p}_0^i)_{xy} & \text{if } i = 1 \\ (\mathbf{p}_0^{i-1})'_{xy} + \eta E(\boldsymbol{\theta}_r)(\Delta\mathbf{p}_0^{i-1})_{xy} & \text{otherwise} \end{cases},$$
$$(\Delta\mathbf{p}_0^i)'_{xy} = \eta E(\boldsymbol{\theta}_r)(\Delta\mathbf{p}_0^i)_{xy},$$
$$(\Delta\mathbf{q}_0^i)' = \exp(\boldsymbol{\theta}_r)\,\mathbf{q}_0^i.$$

Here, $E(\boldsymbol{\theta}_r)$ and $\exp(\boldsymbol{\theta}_r)$ maps rotation vector $\boldsymbol{\theta}_r$ to the corresponding rotation matrix and quaternion, respectively.

5 SOCCER MOTION SIMULATION

In Section 5.1, we explain how to transform the reference motion to compensate for the deviation of the simulation result from it. In Section 5.2, we present a system dynamics model to simulate the motion of a full-body character together with a soccer ball in the presence of contacts. Based on this model, we formulate a finite-horizon optimal control problem to generate an optimal control policy for soccer motions in Section 5.3. In Section 5.4, we discuss how to compute the feedback information on the control policy so as to make the character manipulate the soccer ball while robustly maintaining its balance.

5.1 Reference motion transformation

In general, the simulated character could not follow reference motion $\mathcal{M}$ exactly due to subtle and complex motion dynamics required to control the soccer ball. As the simulation is proceeded, the character's current state tends to deviate from the corresponding state of $\mathcal{M}$, which could adversely affect the stability of motion control. Our system compensates for this deviation from window to window through motion transformation. Once the optimization window is shifted, the system first translates the remaining part of $\mathcal{M}$ so that the root position of the character in its initial state coincides with that of the current state and then aligns the swing foot at the kick frame of $\mathcal{M}$ with the target ball position, employing motion refinement scheme discussed in Section 4.3.

5.2 System dynamics model

For the effective ball motion control, it is necessary to predict the motions of both the character and the ball simultaneously. We present

a system dynamics model that deals with the physical interaction between them as well as their respective dynamics. We start with defining state vector $\mathbf{x}^i$ and control vector $\mathbf{u}^i$ at each time step i:

$$\mathbf{x}^i = [(Q_c^i)^\top \ (Q_b^i)^\top \ (\dot{Q}_c^i)^\top \ (\dot{Q}_b^i)^\top]^\top, \ \ \mathbf{u}^i = \boldsymbol{\tau}^i.$$

Here, $Q_c^i \in \mathbb{R}^{n_c}$ and $Q_b^i \in \mathbb{R}^6$ are respectively the character's pose with n_c degrees of freedom and the ball position and orientation at time step i. $\dot{Q}_c^i$ and $\dot{Q}_b^i$ are their respective time derivatives. For computational efficiency, we reduce the degrees of freedom of the state vector by representing the motion of internal ball joints with Euler angles in this section unlike in Section 4, except for that of the shoulder joints to avoid singularity for a wide swing of arms. $\boldsymbol{\tau}^i \in \mathbb{R}^{n_c-6}$ is the actuated joint torques acting on the character's joints except its root at time step i. Our system dynamics model for state evolution is built on semi-implicit Euler integration with integration step size h:

$$\begin{aligned}\mathbf{x}^{i+1} &= \mathbf{g}(\mathbf{x}^i, \mathbf{u}^i) \\ &= [(Q_c^i + h\dot{Q}_c^{i+1})^\top \ (Q_b^i + h\dot{Q}_b^{i+1})^\top \ (\dot{Q}_c^{i+1})^\top \ (\dot{Q}_b^{i+1})^\top]^\top.\end{aligned}$$

Here, character's joint velocities $\dot{Q}_c^{i+1}$ and ball velocity $\dot{Q}_b^{i+1}$ at the next step are derived by combining the dynamics of the character and the ball together with their physical interaction:

$$\begin{aligned}\begin{bmatrix}\dot{Q}_c^{i+1} \\ \dot{Q}_b^{i+1}\end{bmatrix} &= \begin{bmatrix}\dot{Q}_c^{i} \\ \dot{Q}_b^{i}\end{bmatrix} + hM^{-1}\left(\begin{bmatrix}\boldsymbol{\tau}_c^i \\ \mathbf{0}_{6\times1}\end{bmatrix} - \begin{bmatrix}\mathbf{e}(Q_c^i, \dot{Q}_c^i, \mathbf{f}_c^e) \\ \mathbf{e}(Q_b^i, \dot{Q}_b^i, \mathbf{f}_b^e)\end{bmatrix} - k_d\begin{bmatrix}\dot{Q}_c^i \\ \dot{Q}_b^i\end{bmatrix}\right) \\ &\quad + M^{-1}\begin{bmatrix}J(Q_c^i)^\top \\ J(Q_b^i)^\top\end{bmatrix}\mathbf{f}_{ct}^i, \ \text{where}\, M = \begin{bmatrix}M(Q_c^i) & \mathbf{0}_{n_c\times6} \\ \mathbf{0}_{6\times n_c} & M(Q_b^i)\end{bmatrix}.\end{aligned}$$

Here, $\boldsymbol{\tau}_c^i = [\mathbf{0}_{1\times6} \ (\boldsymbol{\tau}^i)^\top]^\top \in \mathbb{R}^{n_c}$, $M(\cdot)$ is the total mass matrix, and $\mathbf{e}(\cdot)$ is the bias force. Mass matrix $M(\cdot)$ is a variant of the plain mass matrix that results from armature inertia and implicit damping, in order to enhance the stability of the numerical integration of the dynamics. For details on the mass matrix, we refer the readers to [Han et al. 2016; Todorov 2014]. Bias force $\mathbf{e}(\cdot)$ represents the resulting torques due to the gravitational, Coriolis, and centrifugal forces together with external forces applied to the character ($\mathbf{f}_c^e$) and to the ball ($\mathbf{f}_b^e$) if there are any. k_d is the damping gain and $J(\cdot)$ is the kinematic Jacobian at contact points on which contact impulses $\mathbf{f}_{ct}^i$ are acting. Instead of a conventional Linear Complementarity Problem (LCP) formulation, we adopt a complementarity-free formulation to find contact impulses $\mathbf{f}_{ct}^i$ based on smoothed contact dynamics, so as to differentiate our system dynamics model for optimal character control. For details on smoothed contact dynamics, we refer the readers to [Han et al. 2016; Todorov 2014]. Note that we used same symbols, $M(\cdot)$, $\mathbf{e}(\cdot)$, and $J(\cdot)$ for both the character and the ball while sacrificing mathematical rigor for simplicity.

5.3 Optimal control problem formulation

Given current state vector $\mathbf{x}$ and reference trajectory $\mathcal{M}$, our objective is to optimize the character and ball motions, so that the character robustly follows the reference motion while interacting timely with the ball. Based on our system dynamics model, we formulate a finite-horizon optimal control problem over a short window consisting of N_{opt} frames to find a sequence of control vectors $\mathbf{U} = \{\mathbf{u}^i | i = 1, 2, ..., N_{opt} - 1\}$ as follows:

$$\begin{aligned}\min_{\mathbf{U}} \quad & \textstyle\sum_{i=1}^{N_{opt}-1} c(\mathbf{x}^i, \mathbf{u}^i) + c^f(\mathbf{x}^{N_{opt}}), \\ \text{subject to} \quad & \mathbf{x}^1 = \mathbf{x}, \ \ \mathbf{x}^{i+1} = \mathbf{g}(\mathbf{x}^i, \mathbf{u}^i) \ \text{ for } i = 1, 2, ..., N_{opt} - 1, \\ & \gamma_j(\mathbf{x}^i) \le 0 \ \text{ for } i = 1, 2, ..., N_{opt}, \ \ j = 1, 2, ..., L.\end{aligned} \tag{1}$$

Here, $c(\mathbf{x}^i, \mathbf{u}^i)$ and $c^f(\mathbf{x}^{N_{opt}})$ are the cost functions at time step $i = 1, 2, ..., N_{opt} - 1$ and at the last time step, respectively. The first two equalities together constrain the system dynamics model for state evolution from $\mathbf{x}^i$. $\gamma_j(\mathbf{x}^i)$ is the j-th inequality constraint at time step i for $j = 1, 2, ..., L$, where L is the number of inequality constraints. In the remainder of this section, we will define the cost functions and inequality constraints.

Let $\bar{\mathbf{X}}_c = \{(\bar{Q}_c^i, \bar{\dot{Q}}_c^i) | i = 1, 2, ..., N_{opt}\}$ be a desired character trajectory over the optimization window, which is extracted from reference motion $\mathcal{M}$. Each cost function $c(\mathbf{x}^i, \mathbf{u}^i)$ at time step $i = 1, 2, ..., N_{opt} - 1$ is composed of three terms as follows:

$$c(\mathbf{x}^i, \mathbf{u}^i) = c^{ref}(\mathbf{x}^i, \mathbf{u}^i) + c^{fpos}(\mathbf{x}^i) + c^{ball}(\mathbf{x}^i).$$

$c^{ref}(\mathbf{x}^i, \mathbf{u}^i)$ represents the cost for tracking desired character trajectory $\bar{\mathbf{X}}_c$ as in [Han et al. 2016]:

$$\begin{aligned}c^{ref}(\mathbf{x}^i, \mathbf{u}^i) = &\frac{w_1}{2}\|\bar{Q}_c^i - Q_c^i\|^2 + \frac{w_2}{2}\|\bar{\dot{Q}}_c^i - \dot{Q}_c^i\|^2 + \frac{w_3}{2}\|\bar{\mathbf{r}}^i - \mathbf{r}^i\|^2 \\ &+ \frac{w_4}{2}\|\boldsymbol{\tau}^i\|^2.\end{aligned} \tag{2}$$

Here, $\mathbf{r}^i \in \mathbb{R}^3$ is the displacement vector from the root to the head at time step i, and $\bar{\mathbf{r}}^i$ is its desired vector computed from $\bar{Q}_c^i$. The first term is for following the desired character pose. The next two terms are for keeping the joint velocities and the upper-body orientation as close to their desired ones as possible. The last term is for minimizing energy consumption. w_i, $i = 1, 2, 3, 4$ are weight constants.

$c^{fpos}(\mathbf{x}^i)$ is the cost for tracking the desired global foot positions:

$$c^{fpos}(\mathbf{x}^i) = \frac{w_5}{2}\left\|\begin{bmatrix}\bar{\mathbf{p}}_{lf}^i \\ \bar{\mathbf{p}}_{rf}^i\end{bmatrix} - \begin{bmatrix}\mathbf{p}_{lf}^i \\ \mathbf{p}_{rf}^i\end{bmatrix}\right\|^2.$$

Here, $\mathbf{p}_{lf}^i$ and $\mathbf{p}_{rf}^i$ are the global positions of the left foot and right foot at time step i, respectively. $\bar{\mathbf{p}}_{lf}^i$ and $\bar{\mathbf{p}}_{rf}^i$ are their respective desired vectors. w_5 is a weight constant. This cost term guides the character to place its supporting foot at the right place while timely kicking the ball with its swing foot.

Finally, $c^{ball}(\mathbf{x}^i)$ represents the ball control cost defined as follows:

$$c^{ball}(\mathbf{x}^i) = \frac{w_6}{2}({}^R b_y^i)^2 + \frac{w_7}{2}\|\dot{\bar{\mathbf{b}}}^i - \dot{\mathbf{b}}^i\|^2,$$

where ${}^R b_y^i \in \mathbb{R}$ is the lateral component of the ball position at time step i with respect to root coordinate system R (See Figure 4). $\dot{\mathbf{b}}^i$ and $\dot{\bar{\mathbf{b}}}^i$ are respectively the time derivative of the ball position $\mathbf{b}^i \in \mathbb{R}^3$ and the desired ball velocity at time step i with respect to the global coordinate system. In the above equation, the left superscript is used to denote a local coordinate system. w_6 and w_7 are weight constants.

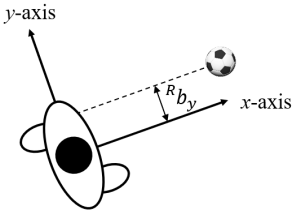
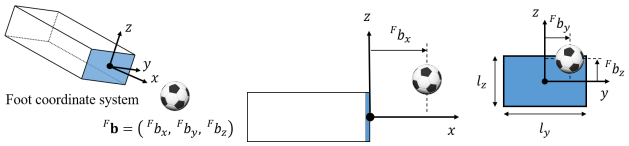


Fig. 4. The lateral component of the ball position with respect to the root coordinate system in the top view



(a) Relative ball position (b) Side view (c) Front view

Fig. 5. Ball coordinates with respect to the local coordinate system at the center of the front side of the kicking foot

The first term is activated only when the character dribbles forward in order to put the ball in front of the character for the ease of kicking it while running. The second term is for the character to kick the ball at a desired ball velocity. Since the ball velocity is controlled through kicking the ball, this term is activated only for a few frames near the kick frame.

Inequality constraints $\gamma_j(\mathbf{x}^i) \leq 0$, $j = 1, 2, ..., L$ at each time step $i = 1, 2, ..., N_{opt}$ are activated for a few frames near the kick frame so as to timely kick the ball with the front side of the kicking foot under the assumption that the foot geometry is a rectangular parallelepiped. Let $^{F}\mathbf{b}^i$ be the ball position at time step i with respect to the kick foot local coordinate system F (See Figure 5). Then, the first two inequality constraints make the front side touch the ball in the depth direction (the x-axis):

$$-{}^{F}b_x^i \leq 0, \quad {}^{F}b_x^i - r_b \leq 0,$$

where r_b is the radius of the ball. The next four inequality constraints enforce the projected ball position onto the y-z plane of the local coordinate system to be in the front side of the kicking foot:

$$-\frac{1}{2}l_y - {}^{F}b_y^i \leq 0, \quad {}^{F}b_y^i - \frac{1}{2}l_y \leq 0,$$
$$-\frac{1}{2}l_z - {}^{F}b_z^i \leq 0, \quad {}^{F}b_z^i - \frac{1}{2}l_z \leq 0.$$

Here, l_y and l_z are the width and height of the front side, respectively. Note that our system dynamics model allows the kicking foot to penetrate the ball slightly in kicking because the model employs smoothed contact dynamics, which softens the contact between colliding bodies. We employ an unconstrained optimizer, iterative Linear Quadratic Gaussian (iLQG) [Tassa et al. 2012], to solve our optimal control problem. Thus, we have to convert the above ball constraints to their corresponding cost terms. For this, we define the following smooth-max and smooth-min functions with threshold α by extending the existing work [Tassa et al. 2012]:

$$\text{smax}(x; \alpha, \beta) = \frac{1}{2}\left(\sqrt{(x-\alpha)^2 + \beta^2} + (x-\alpha)\right),$$
$$\text{smin}(x; \alpha, \beta) = \frac{1}{2}\left(\sqrt{(x-\alpha)^2 + \beta^2} - (x-\alpha)\right).$$

Here, the first function returns a value close to $x - \alpha$ if $x > \alpha$ and a near-zero value otherwise. The second function returns a value close to $\alpha - x$ if $x < \alpha$ and a near-zero value otherwise. β is the smoothness coefficient that adjusts how close the function is to a zero at $x = \alpha$. Using smax$(\cdot)$ and smin$(\cdot)$, we convert our inequality constraints on the kicking foot to a cost function consisting of three terms as follows:

$$c^{kick}(\mathbf{x}^i) = w_8\left(c_x^{kick}(\mathbf{x}^i) + c_y^{kick}(\mathbf{x}^i) + c_z^{kick}(\mathbf{x}^i)\right),$$

where w_8 is a weight constant. Each term of $c^{kick}(\mathbf{x}^i)$ is defined as follows:

$$c_x^{kick}(\mathbf{x}^i) = \text{smin}\left({}^{F}b_x^i; 0, \beta\right) + \text{smax}\left({}^{F}b_x^i; r_b, \beta\right),$$
$$c_y^{kick}(\mathbf{x}^i) = \text{smin}\left({}^{F}b_y^i; -\frac{1}{2}l_y, \beta\right) + \text{smax}\left({}^{F}b_y^i; \frac{1}{2}l_y, \beta\right),$$
$$c_z^{kick}(\mathbf{x}^i) = \text{smin}\left({}^{F}b_z^i; -\frac{1}{2}l_z, \beta\right) + \text{smax}\left({}^{F}b_z^i; \frac{1}{2}l_z, \beta\right).$$

Finally, we conclude this section, stating that cost term $c^f(\mathbf{x}^{N_{opt}})$ at last frame N_{opt} in Equation (1) takes the same form as $c(\mathbf{x}^i, \mathbf{u}^i)$ at other frame i except that $c^{ref}(\mathbf{x}^{N_{opt}}, \mathbf{u}^{N_{opt}})$ in Equation (2) does not have energy minimizing term $\frac{w_4}{2}\|\boldsymbol{\tau}^{N_{opt}}\|^2$, since control vector $\mathbf{u}^{N_{opt}}$ is not required at the last frame of a window.

5.4 Feedback control vector generation

Given current state vector $\mathbf{x}^i$ and reference motion $\mathcal{M}$, our optimal control problem is solved with iLQG [Tassa et al. 2012], that is, a variant of differential dynamic programming (DDP) [Jacobson and Mayne 1970]. Avoiding the use of the second-order derivative of the system dynamics model, iLQG is practically faster than DDP. This optimization solver produces an optimal control policy to acquire the optimal feedback control vector for robust soccer motion control. For details on iLQG, we refer the readers to [Tassa et al. 2012].

The resulting optimal control policy is composed of optimal control vectors $\{\mathbf{u}^{*i} | i = 1, 2, ..., N_{opt} - 1\}$, optimal state vectors $\{\mathbf{x}^{*i} | i = 1, 2, ..., N_{opt}\}$, and optimal feedback gain matrices $\{K^{*i} | i = 1, 2, ..., N_{opt} - 1\}$. Provided with the control policy, our system computes optimal feedback control vector $\mathbf{u}_{fb}^{*\,i}$ at each frame i of the window as follows:

$$\mathbf{u}_{fb}^{*\,i} = \mathbf{u}^{*i} + K^{*i}(\mathbf{x}^i - \mathbf{x}^{*i}).$$

Here, the second term compensates for the deviation of the current state from its optimal state vector. Together with external forces if there are any, feedback control vector $\mathbf{u}_{fb}^{*\,i}$ that represents the feedback joint torques of the character is used to update current state $\mathbf{x}^i$ through forward dynamics integration. Our system performs the

physics-based simulation window by window while shifting the optimization window, until the reference motion is used up or a large external perturbation is applied to the character to severely deviate its simulated state from the corresponding state in the reference motion. In either case, our system requests for a new reference motion with the last N_d simulated character states to activate the feedback control loop of our system.

6 RESULTS AND DISCUSSION

In this section, we present our results on soccer motion synthesis. We begin with providing implementation details followed by evaluation and examples in Sections 6.2 and 6.3, respectively.

6.1 Implementation details

Our framework was implemented in C++, and all of the experiments were performed on a PC with Intel Core i7 processor (3.50GHz, 6 cores) and 32GB memory. As stated in Section 3, the proposed motion control system consists of two components: soccer motion prediction and soccer motion simulation. The computation time of the latter was demanding, compared to the former which runs in real time. On average, the total computation time was $6 \sim 9$ times larger than real time (20ms/frame) as shown in Table 2.

For soccer motion prediction, we used 22 example motion clips (1,569 frames in total): 6 clips for dribbling forward with different speeds, 7 clips for dribbling to the side, and 3 clips for shooting. Our motion library also included 6 motion clips for extra soccer skills. $\bar{K}$ was composed of 8 kick poses for dribbling forward, 13 kick poses for dribbling to the side, and 3 kick poses for shooting. To define cost terms in Section 4.2, we fixed $N_s = 20$, $N_d = 5$, and $\epsilon = 1$. We set the weight values of cost terms as shown in the first row of Table 1. For soccer motion simulation, we set respectively the total mass and height of the character to 80kg and 1.8m, and used a soccer ball with the official weight and radius, which are 0.425kg and 0.11m, unlike the previous work [Ha and Liu 2016; Peng et al. 2017]. To define cost terms and constraints in Section 5.3, we fixed h = 20ms, $N_{opt} = 20$, $N_{shift} = 3$, l_y = 0.088m, and l_z = 0.080m. We set the weight values as given in the second row of Table 1. We believe that these weight values could be a good starting point to build a similar system. Table 2 shows the timing data of our system for each motion type.

Table 1. Weight values used for data-driven motion prediction and physics-based motion simulation

Data-driven	α_r	α_d	α_b	α_θ	-	-	-	-
motion prediction	0.05	0.5	2.0	10^{-4}	-	-	-	-
Physics-based	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
motion simulation	1.0	10^{-5}	1.0	5.0×10^{-8}	0.25	2.0	0.25	0.1

Table 2. Average computation time (ms/frame) for soccer motion control

Motion type	Data-driven motion prediction	Physics-based motion simulation		Total
		Optimization	Simulation	
Dribbling forward	8.32	114.64	0.142	123.10
Dribbling to the side	10.65	169.86	0.131	180.64
Shooting	8.65	165.81	0.154	174.62
Step-over	17.20	158.06	0.160	175.42
Feint	17.64	163.83	0.167	181.64

6.2 Validation

In this section, we validate our system in terms of five aspects. We start with testing the controllability of our system with respect to the motion specification, in particular, the desired running speed and turning angle. Varying the number of previous character states (N_d), we then show the effect of looking back with cost term $S_d(C^i, \bar{C}^j)$. We next demonstrate the effectiveness of motion refinement for stable dribbling, followed by comparing the qualities of dribbling motions for different lengths of the reference motion (N_p). We finally examine the strategy of replaying pre-specified motion clips that generate realistic ball-kicking motions.

To validate the controllability of our system, we first performed an experiment on the desired running speed with a motion of dribbling forward. The character's desired running speed (v_d) was varied from 5.0m/s to 7.0m/s with an interval of 0.25m/s. The desired ball speed was set accordingly by interpolating it from 12m/s to 20m/s. We measured the average speed of the simulated motion (v_s) for 300 frames for each desired speed. The results showed that the character was able to run with an accuracy of $95.14 \pm 2.47\%$ on average for the entire range of desired speed, where the accuracy of speed control (a_s) is defined as follows:

$$a_s = \left(1 - \frac{|v_d - v_s|}{v_d}\right) \times 100.$$

The small error in accuracy would arise due to lack of motion clips in the motion library. We then performed an experiment on the controllability for turning angle with a motion of dribbling to the side. For each of three desired running speed values, 6.0m/s, 6.5m/s, and 7.0m/s, we randomly sampled 20 desired turning angles (θ_d); one half of them drawn from a range of -20° to -40° (right turn) and the other half from a range of 20° to 40° (left turn). We measured the turning angles of the simulated character (θ_s) from the moving directions before and after kicking the ball at which the character made a turn. We did not experiment with desired running speed lower than 6.0m/s since our example motion clips for dribbling to the side exhibited running speed of around 7m/s. Despite lack of motion clips, our system successfully made desired turns with high accuracy as shown in Table 3, mainly due to motion refinement in reference motion synthesis. Here, the accuracy of turning angle control (a_θ) is defined as follows:

$$a_\theta = \left(1 - \frac{|\theta_d - \theta_s|}{\theta_d}\right) \times 100.$$

Table 3. Average and standard deviation for accuracy of dribbling to the side

Desired root speed (m/s)	6.0		6.5		7.0	
Turning direction	Right	Left	Right	Left	Right	Left
Average accuracy (%)	95.25	92.26	92.20	92.60	96.01	95.04
Standard deviation (%)	±3.86	±4.82	±5.89	±3.18	±4.28	±3.37

To show the effect of looking back with cost term $S_d(C^i, \bar{C}^j)$, we performed an experiment, varying the number of previous character states (N_d), in the range from 1 to 5 for dribbling forward. We observed that the resulting soccer motions tended to become stable as N_d increases. In our experiment, the use of more than 5 previous frames did not lead to any noticeable improvement in visual quality. We provide the result in the accompanying video (00:01:23 ~ 00:01:41). We also measured the tracking error of the root speed and that of the upper-body displacement vector from the root to the head denoted by $\mathbf{r}^i$ in Equation (2). For each N_d from 1 to 6, we varied the desired running speed from 5.0m/s to 7.0m/s with an increment of 0.4m/s. For each desired running speed, we also varied the desired ball speed as fast as 2.5 to 4.0 times each running speed with an increment of 0.5m/s (a total of 144 simulation results for dribbling forward). Even with small changes of such measures, the dribbling motion of the character was better stabilized visually with $N_d = 5$, as shown in the accompanying video. The tracking of a dribbling motion in physics-based motion synthesis was sensitive to the root speed and the upper-body displacement vector (see the cost terms in Equation (2)). Table 4 provides the numerical data on the average tracking errors of the root speed (e_{spd}) and the upper-body displacement vector (e_{r2h}).

Table 4. Average (μ) and standard deviation (σ) for tracking errors of the root speed (e_{spd}) and the upper-body displacement vector (e_{r2h}) according to N_d

N_d		1	2	3	4	5	6
e_{spd} (m/s)	μ	0.796	0.779	0.764	0.738	0.723	0.711
	σ	± 0.103	± 0.094	± 0.104	± 0.096	± 0.108	± 0.117
e_{r2h} (deg.)	μ	3.524	3.399	3.220	3.119	3.065	3.013
	σ	± 0.478	± 0.516	± 0.555	± 0.428	± 0.488	± 0.455

The purpose of motion refinement is to place a kicking foot at the corresponding ball position by transforming the root trajectory. We performed an experiment with dribbling forward to demonstrate the effect of motion refinement in reference motion synthesis. Fixing $N_d = 5$, we used scaling factor η in a range of $[0.85, 1.15]$ to transform the root trajectory of the reference motion. The desired running speed was varied in a range from 5.0m/s to 7.0m/s with an increment of 0.1m/s, and the desired ball speed was as fast as 2.5 to 4.0 times each running speed with an increment of 0.5m/s. With and without employing motion refinement, we repeated the experiment 84 times using different sets of parameters. Without motion refinement, our character kept running 27 times (32.14%) without losing the ball control for 7 seconds of simulation. With motion refinement, the character failed in dribbling only once (1.19%) in the same time. In the latter case, the failure would be due to a limited range of speed in the motion data. We also show the result in the accompanying video (00:00:44 ~ 00:01:06).

Effective ball control requires a relatively long reference motion. MPC-based optimization with a large window, however, incurs a heavy computation load. Instead of using a long window for MPC, our system synthesizes a reference motion long enough to contain a kick pose, and performs the optimization in a short window while shifting the window along the reference motion, as illustrated in Figure 3. We synthesized the dribbling motions, varying the size of the reference motion (N_p). When N_p was equal to N_{opt} as in [Han et al. 2016], the character tended to kick the ball abruptly or to miss the ball. In contrast, our system successfully controlled the ball with a long reference motion containing a kick pose. We provide the results in the accompanying video (00:01:07 ~ 00:01:22).

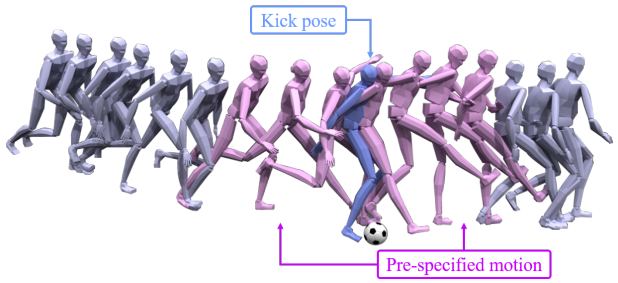


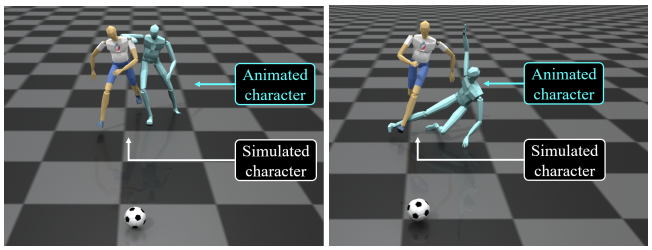
Fig. 6. A reference motion consisting of the predicted character states (light purple), the state with a kick pose (blue), and the pre-specified motion (pink) around the kick pose

When dribbling to the side or shooting a ball, the character should plant a supporting foot at the right position and swing the other foot timely for a ball-kicking motion. In order to generate a realistic kicking motion, our system uses a pre-specified kick motion to acquire the coordinated movements around the kick pose in a reference motion, as shown in Figure 6. Specifically, upon choosing a character state that initiates a pre-specified kick motion, the system simply replays a short pre-specified motion sub-clip containing the kick pose to complete a reference motion. We performed an experiment with shooting motion with and without this strategy to show its effectiveness. Note that the number of frames of the pre-specified motion is not a decision variable but determined from the stride length change of a motion at its frames through visual examination. This strategy enhanced the visual quality of resulting motions significantly by preventing an abrupt transition to a kick pose and also reduced computation time to 0.025ms/frame. We show the result in the accompanying video (00:01:42 ~ 00:01:56).

6.3 Examples

In this section, we show the effectiveness of our system for generating soccer motions: dribbling forward, dribbling to the side, and shooting.

Dribbling forward: We performed an experiment on the character's response to the mass of the ball, which was set to 0.85kg and 2.25kg. With the same desired ball speed, the heavier the ball was, the more the character stretched its kicking leg toward the ball to hit it hard. We finally demonstrated the robustness of the proposed system against external perturbations. In our scenario, the opponent perturbed the dribbling character with different forces applied to different body parts (See Figure 7). Figure 7a shows that the opponent pushed the character with a linear force ($250N$) applied to the character's torso in the lateral direction for 0.3s. Figure 7b exhibits that the opponent tackled the character with an impulse force ($500N$) applied to the foot. The character successfully overcame

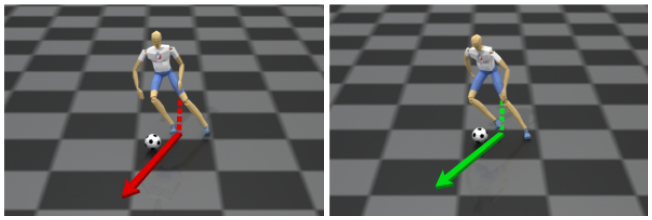


(a) Pushing (b) Tackling

Fig. 7. Two different perturbations: pushing and tackling

the unexpected perturbations while continuing dribbling. Note that the motions of the opponent was replayed with motion clips. We provide the results in the accompanying video (00:02:00 ~ 00:02:45).

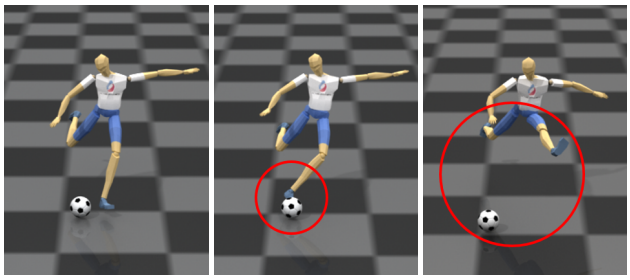
Dribbling to the side: In order to see the ability of our system to make turning motions with a ball, we synthesized a motion of dribbling to the side with different turning angles θ. As shown in Figure 8, the leaning of the character body in the turning direction is apparent according to different values of θ, which were set to 20° and 40°, respectively. We also observed that the character made relatively short footsteps for turning. To support the versatility of our system, we synthesized additional soccer skills such as step-over and feint that change the dribbling direction and deceive an opponent, respectively. The results are available in the accompanying video (00:02:46 ~ 00:03:32).



(a) Turning angle θ = 20° (b) Turning angle θ = 40°

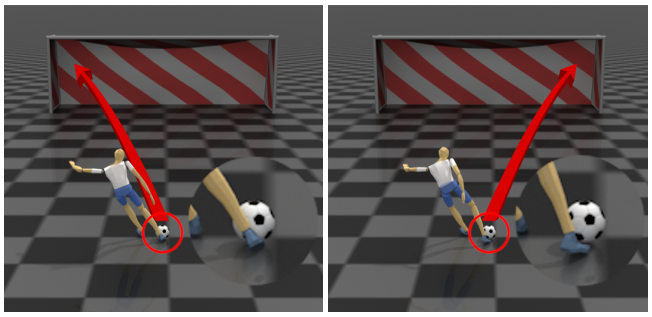
Fig. 8. Motions for dribbling to the side with different turning angles

Shooting: We first produced motions of shooting the ball straight with different ball speeds set to relatively large values, 20m/s, 30m/s, and 50m/s. The character tended to kick the ball hard to move it fast as the desired ball speed increases. We then show the effectiveness of cost term $c^{fpos}(\mathbf{x}^i, \mathbf{u}^i)$ for tracking desired global foot positions, comparing shooting motions synthesized with and without the term. With this cost term, the character planted the supporting foot at a right position on the ground to convincingly kick the ball with the swing foot, as shown in Figure 9a. Without the cost term, however, the character often planted the supporting foot at a wrong position (Figure 9b) or failed to plant it on the ground (Figure 9c). We finally synthesized two penalty kick motions according to different scenarios of shooting the ball to the left and to the right of a goal, respectively. The character was able to control the moving direction of a swing foot according to desired turning angle θ as shown



(a) (b) (c)

Fig. 9. Shooting motions with and without the foot tracking cost term: (a) A kick pose (with the cost term); (b) Planted the supporting foot at a wrong position (without the cost term); (c) Failed to plant the supporting foot (without the cost term)



(a) Shooting to the left (b) Shooting to the right

Fig. 10. Ball-kicking poses in different shooting directions (The goalpost model created by *bigonethegod*, courtesy of *sketchfab.com*)

in Figure 10. We provide the results in the accompanying video (00:03:33 ~ 00:04:25).

7 CONCLUSIONS

In this paper, we presented a novel framework for synthesizing three types of soccer motions: dribbling forward, dribbling to the side, and shooting. This framework gives rise to the first physics-based full-body motion control system for synthesizing basic soccer motions such as dribbling and shooting, to the best of our knowledge. Our system is able to control a human-like character in response to online motion specifications and external perturbations.

The proposed system consists of two tightly-coupled components: data-driven soccer motion prediction and physics-based soccer motion simulation. The former synthesizes a reference motion state by state until a pre-specified kick motion is encountered, replaying the motion to complete the reference motion. Guided by the reference motion, the latter solves a finite-horizon optimal control problem window by window to synthesize physically-valid soccer motions, while shifting the window along the reference motion. This scheme avoids the use of a large optimization window, which results in a heavy computational burden. Together with the updated ball trajectory, the simulated motion from the latter is fed back to the former to facilitate the looking back strategy for reference motion

synthesis. We present soccer-specific cost terms for physics-based motion synthesis so that a full-body character timely kicks the ball while chasing it. We demonstrated the effectiveness of the proposed system through various experiments.

8 LIMITATIONS AND FUTURE WORK

Although our system can synthesize motions of dribbling to the side faithfully in general, it is difficult to make a sharp turn while running because of large contact forces induced at the supporting foot due to sudden momentum change. Our system could create a motion artifact of jerky limb movements when the character tries to balance against such contact forces, as shown in the accompanying video (00:04:29 ~ 00:04:40). A similar artifact could be observed in shooting when the character kicks the ball hard with an extremely large foot speed, due to the large momentum change of the swing leg as in the accompanying video (00:04:41 ~ 00:04:52). We simplified the foot geometry as a rectangular box, assuming that the character kicks a ball only with its front side. In an actual soccer game, however, a human player utilizes various parts of a foot such as the inner part, the outer part, the top part, the heel, and even the sole. As future work, we would like to extend our framework for the robust handling of complex and subtle contacts between a foot and the ball.

The current version of our system does not take into consideration the constraints on contact forces and joint torques, which could result in artifacts due to large contact forces or excessive joint torques, as shown in the accompanying video (00:04:29 ~ 00:04:52). The artifacts can be addressed while scarifying time efficiency by employing a control-limited differential dynamic programming framework [Tassa et al. 2014], which sets joint torque limits as hard constraints in their optimal control formulation. Our system dynamics model does not take into account self-collisions between the character bodies to enhance computational efficiency, which could result in self-penetration artifacts. Such artifacts can also be alleviated by properly specifying joint angle limits as constraints on top of our current formulation.

Our system can deal with three basic types of soccer motions such as dribbling forward, dribbling to the side, and shooting. However, there exist many other types of soccer motions such as trapping, passing, heading, juggling and so on. In each of these types, a human player controls a ball with different body parts and different footsteps. It would be an interesting future research direction to study how to deal with all the soccer motion types with a unified framework. One of promising approaches is employing Deep Reinforcement Learning (DRL) to find a control policy for each soccer skill in an unsupervised fashion, based on the character's interactions with the ball and the environment. DRL has recently made great progress in physics-based motion control, coming up with approaches such as imitation learning [Peng et al. 2018, 2017] and curriculum learning [Heess et al. 2017; Yu et al. 2018] to learn various human-like motion skills under physical interactions with an environment. Our framework could be extended to improve the efficiency of DRL by effectively guiding the policy search for soccer skills as in [Levine and Koltun 2013].

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their invaluable comments; Jaewon Song, Mi you, and Sunjin Jung for modeling the character and rendering the scenes; Haegwang Eom for the helpful discussion about the experiments; Allen Kim for providing the voice-over; Bumki Kim and Dawon Lee for helping the creation of the demonstration video and the figures. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2017R1A2A1A05000979).

REFERENCES

Mazen Al Borno, Martin De Lasa, and Aaron Hertzmann. 2013. Trajectory optimization for full-body movements with complex contacts. *IEEE transactions on visualization and computer graphics* 19, 8 (2013), 1405–1414.

Okan Arikan and David A Forsyth. 2002. Interactive motion generation from examples. In *ACM Transactions on Graphics (TOG)*, Vol. 21. ACM, 483–490.

Samuel Barrett, Katie Genter, Todd Hester, Michael Quinlan, and Peter Stone. 2010. Controlled kicking under uncertainty. In *The Fifth Workshop on Humanoid Soccer Robots at Humanoids*.

Jong-In Choi, Shin-Jin Kang, Chang-Hun Kim, and Jung Lee. 2015. Virtual ball player. *The Visual Computer* 31, 6-8 (2015), 905–914.

Jong-In Choi, Sun-Jeong Kim, Chang-Hun Kim, and Jung Lee. 2016. Let's be a virtual juggler. *Computer Animation and Virtual Worlds* 27, 3-4 (2016), 443–450.

Simon Clavet. 2016. Motion Matching and The Road to Next-Gen Animation. *In Proc. of GDC 2016* (2016).

Marco Da Silva, Yeuhi Abe, and J Popović. 2008. Simulation of Human Motion Data using Short-Horizon Model-Predictive Control. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 371–380.

Martin De Lasa, Igor Mordatch, and Aaron Hertzmann. 2010. Feature-based locomotion controllers. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 131.

Kai Ding, Libin Liu, Michiel Van de Panne, and KangKang Yin. 2015. Learning reduced-order feedback policies for motion skills. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 83–92.

EASports. 1993-2019. FIFA series. www.easports.com/fifa.

Michael Gleicher. 2001. Motion path editing. In *Proceedings of the 2001 symposium on Interactive 3D graphics*. ACM, 195–202.

Sehoon Ha and C Karen Liu. 2016. Evolutionary optimization for parameterized whole-body dynamic motor skills. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 1390–1397.

Daseong Han, Haegwang Eom, Junyong Noh, et al. 2016. Data-guided Model Predictive Control Based on Smoothed Contact Dynamics. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 533–543.

Daseong Han, Junyong Noh, Xiaogang Jin, Joseph S Shin, and Sung Y Shin. 2014. Online real-time physics-based predictive motion control with balance recovery. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 245–254.

Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. 2017. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286* (2017).

Todd Hester, Michael Quinlan, and Peter Stone. 2010. Generalized model learning for reinforcement learning on a humanoid robot. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2369–2374.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 42.

Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 138.

David H Jacobson and David Q Mayne. 1970. Differential dynamic programming. (1970).

Sumit Jain and C Karen Liu. 2009. Interactive synthesis of human-object interaction. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 47–53.

KONAMI. 1995-2019. Pro Evolution Soccer series. www.konami.com.

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002. Motion graphs. In *ACM transactions on graphics (TOG)*, Vol. 21. ACM, 473–482.

Jehee Lee, Jinxiang Chai, Paul SA Reitsma, Jessica K Hodgins, and Nancy S Pollard. 2002. Interactive control of avatars animated with human motion data. In *ACM Transactions on Graphics (TOG)*, Vol. 21. ACM, 491–500.

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010a. Data-driven biped control. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 129.

Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. 2010b. Motion fields for interactive character locomotion. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 138.

Leonardo Leottau, Carlos Celemin, and Javier Ruiz-del Solar. 2014. Ball dribbling for humanoid biped robots: a reinforcement learning and fuzzy control approach. In *Robot Soccer World Cup*. Springer, 549–561.

Sergey Levine and Vladlen Koltun. 2013. Guided policy search. In *International Conference on Machine Learning*. 1–9.

Sergey Levine, Jack M Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. 2012. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 28.

Libin Liu, Michiel Van De Panne, and KangKang Yin. 2016. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics (TOG)* 35, 3 (2016), 29.

Libin Liu, KangKang Yin, Michiel van de Panne, and Baining Guo. 2012. Terrain runner: control, parameterization, composition, and planning for highly dynamic motions. *ACM Trans. Graph.* 31, 6 (2012), 154–1.

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 128.

Igor Mordatch, Martin De Lasa, and Aaron Hertzmann. 2010. Robust physics-based locomotion using low-dimensional planning. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 71.

Uldarico Muico, Jovan Popović, and Zoran Popović. 2011. Composite control of physically simulated characters. *ACM Transactions on Graphics (TOG)* 30, 3 (2011), 16.

Sang Il Park, Hyun Joon Shin, and Sung Yong Shin. 2002. On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 105–111.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills. *ACM Trans. Graph.* 37, 4, Article 143 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201311

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 41.

ChangHyun Sung, Takahiro Kagawa, and Yoji Uno. 2013. Whole-body motion planning for humanoid robots by specifying via-points. *International Journal of Advanced Robotic Systems* 10, 7 (2013), 300.

Yuval Tassa, Tom Erez, and Emanuel Todorov. 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 4906–4913.

Yuval Tassa, Nicolas Mansard, and Emo Todorov. 2014. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1168–1175.

Emanuel Todorov. 2014. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 6054–6061.

Adrien Treuille, Yongjoon Lee, and Zoran Popović. 2007. Near-optimal character animation with continuous control. In *ACM Transactions on Graphics (tog)*, Vol. 26. ACM, 7.

Seung-Joon Yi, Stephen McGill, and Daniel D Lee. 2013. Improved online kick generation method for humanoid soccer robots. In *The 8th Workshop on Humanoid Soccer Robots, Georgia USA. Disponible en http://www. ais. uni-bonn. de/humanoidsoccer/ws13/papers/HSR13_Yi. pdf*.

Wenhao Yu, Greg Turk, and C Karen Liu. 2018. Learning Symmetry and Low-energy Locomotion. *arXiv preprint arXiv:1801.08093* (2018).