Learning How to Solve "Bubble Ball"

Hotae Lee   HOTAE.LEE@BERKELEY.EDU
Monimoy Bujarbaruah   MONIMOYB@BERKELEY.EDU
Francesco Borrelli   FBORRELLI@BERKELEY.EDU
2169 Etcheverry Hall, Berkeley, CA, USA

Abstract

"Bubble Ball" is a game built on a 2D physics engine, where a finite set of objects can modify the motion of a bubble-like ball. The objective is to choose the set and the initial configuration of the objects, in order to get the ball to reach a target flag. The presence of obstacles, friction, contact forces and combinatorial object choices make the game hard to solve.

In this paper, we propose a hierarchical predictive framework which solves Bubble Ball. Geometric, kinematic and dynamic models are used at different levels of the hierarchy. At each level of the game, data collected during failed iterations are used to update models at all hierarchical level and converge to a feasible solution to the game. The proposed approach successfully solves a large set of Bubble Ball levels within reasonable number of trials (GitHub, Videos). This proposed framework can also be used to solve other physics-based games, especially with limited training data from human demonstrations.

Keywords: MPC, Physics-Based Games, Trajectory Optimization.

1. Introduction

The field of physics-based games has garnered significant attention for the testing and evaluation of control algorithms/reinforcement learning methods Mnih et al. (2015); Heess et al. (2015); Kurach et al. (2019). Most approaches to solve such games are model-free, such as Q-learning or policy gradient methods Mnih et al. (2015); Lillicrap et al. (2015). However, most of these approaches require sizeable training data and are difficult to generalize across multiple game levels with widely varying configurations. The primary reason is the lack of use of intuitive models/motion primitives arising from the notions of physics, such as the effect of contact forces.

On the other hand, using physics driven high fidelity modeling for solving these problems can also be challenging. For example, the contact forces between any two bodies can result in a discontinuous motion evolution. Moreover, the contact forces themselves are the solutions to hard optimization problems, such as a Linear Complementarity problems Todorov et al. (2012); Kuindersma et al. (2016); Hwangbo et al. (2018). Using such system modeling approach for solving a physic-based puzzle in real-time can become computationally intractable.

In this paper, we propose a model-based hierarchical approach to overcome the aforementioned challenges. The presence of physics-based models circumvents the shortcomings of model-free methods, while hierarchies allow for the use of low-complexity models for the right prediction objective. We specifically focus on solving the very popular (millions of downloads) game Bubble Ball Nay (2010). The objective of the game is to decide a configuration of a finite set of objects, in order to get a bubble-like ball to reach a target flag following the laws of physics. Each hierarchy of our approach uses simplified abstractions of the game's physics engine for attempting to solve

a game level. Data of failed trials is used at each hierarchical level to update the models and the goal set, and attempt the next trial. This is motivated by recent works such as Rosolia and Borrelli (2017); Hewing et al. (2020); Vallon and Borrelli (2020) . Our contribution is a disciplined solution approach based on three principles:

- *Exploiting Locality*: At each trial we restrict our focus to only a specific spatial region of the state-space, where a failed solution from a previous trial needs improvement. By doing so, we reduce the computational complexity of solution search.

- *Iterative Learning*: If a trial of the game fails, using the collected game data we improve our approximated solution by updating the local region and the model parameters of the movable objects.

- *Exploiting Model Hierarchy*: We use three hierarchies of models. A geometry-based high-level path of the ball connecting the start and target positions provides us a warm start. Then, using kinematic and dynamic models of the movable objects, we solve a tractable optimization problem to place the objects and track the ball's planned path in any local region.

We demonstrate the performance of our algorithm in Section 6, by using the physics simulator of Bubble Ball (the one used by the actual commercial game). The proposed approach successfully solves a large set of Bubble Ball levels within reasonable number of trials (GitHub, Videos).

Our approach is focused on Bubble Ball and the three principles discussed above are (in one form or another and not necessarily all together) already used by different communities to solve complex problems, e.g., Grymin et al. (2014); Zhou et al. (2019); Rosolia et al. (2020); Wabersich et al. (2021). Nonetheless, we believe that the exercise presented in this paper can be beneficial to the "learning to control" community as it tries to bring some discipline on how to glue the different hierarchical levels together. In particular, the same framework that utilizes models of different complexity for each hierarchy and uses a terminal set to a lower level Model Predictive Control has been already successfully used in manipulators' and race-cars' application Vallon and Borrelli (2020); Shen et al. (2020).

2. Introduction to Bubble Ball

Figure 1: The target position of the ball is marked by the checkered flag. The black objects indicate the environment and the remaining (i.e., wooden and metallic) pieces are the movable objects that the user will place to decide the initial game configuration.

Bubble Ball is a game built on a physics engine (Box2D), where the objective is to decide a configuration of a finite set of objects, to get a bubble-like ball to reach a target flag. The game is

organized into levels of increasing difficulty. In each level, the player is provided with a different set of movable objects which can affect the ball's trajectory. Each level also has a set of fixed objects which describe the environment. Fig. 1 shows a set of levels of Bubble Ball depicting the aforementioned components.

The player decides the initial configuration of a subset of the given movable objects and has no further inputs after the game starts. The trajectories of the ball and the movable objects then evolve under the laws of physics, affected by gravity and contact forces. In order to solve the game, the player has to predict how the objects affect the ball's trajectory. Such prediction is challenging since contact forces between all the elements of the game during the simulation change discontinuously as a function of the objects' material and the objects' initial state. In addition to the continuous decision variables (the objects' initial configurations), the combinatorial decision on which objects to select adds complexity to the game.

Human players use multiple trials of the same level to converge to a solution and then advance to the next level. Observations of each unsuccessful trial are then used to modify the previous blocks' initial condition, or to try a different combination of blocks.

3. Problem Formulation

We model the game evolution as a discrete time nonlinear system interacting with the environment:

$$s_{k+1} = f(\begin{bmatrix} s_k \\ m_k \end{bmatrix}, z) , \ m_{k+1} = g(\begin{bmatrix} s_k \\ m_k \end{bmatrix}, z), \tag{1}$$

where $s_k \in \mathbb{R}^4$ is the state of the ball, $m_k \in \mathbb{R}^{8 \times n}$ (n objects) is the state of the movable objects, and $z \in \mathbb{R}^{8 \times p}$ (p fixed environment objects) is the state of the immovable objects. Considering the
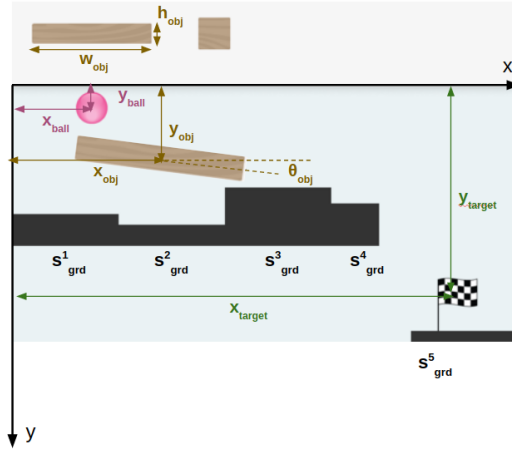

Figure 2: Description of the states and coordinate frame used to formulate the Bubble Ball problem.

coordinate frame as shown in Fig. 2, we have:

$$s_k = [x_{\text{ball}}, y_{\text{ball}}, \dot{x}_{\text{ball}}, \dot{y}_{\text{ball}}]^\top \in \mathbb{R}^4, \ m_k = [(s^1_{\text{obj},k})^\top, \dots, (s^n_{\text{obj},k})^\top]^\top \in \mathbb{R}^{8 \times n}, \tag{2}$$

where the state of the given i^{th} movable object is denoted as s_{obj}^i. The state of each object contains its position, angle with the positive x-axis, velocity, angular velocity, width, and height[1], i.e.,

$$s_{\text{obj}}^i = [x_{\text{obj}}^i, y_{\text{obj}}^i, \theta_{\text{obj}}^i, \dot{x}_{\text{obj}}^i, \dot{y}_{\text{obj}}^i, \dot{\theta}_{\text{obj}}^i, w_{\text{obj}}^i, h_{\text{obj}}^i]^\top, \ \forall i \in \{1, 2, \dots, n\}. \tag{3}$$

The immovable objects' vector z consists of the ground blocks' states s_{grd}, with the same components as s_{obj}, i.e, $z = [(s_{\text{grd}}^1)^\top, \dots, (s_{\text{grd}}^p)^\top]^\top \in \mathbb{R}^{8 \times p}$. In equation (1), $f(\cdot, \cdot)$ is obtained by discretizing the Euler-Lagrangian equation, which includes contact forces between the ball and the objects. We assume that s_0 is fixed. Our goal is to decide an initial state m_0, such that s_k reaches a target set $\mathcal{T}_{\text{tar}}$ at any time step $\tau \leq T$ during evolution, i.e., $\exists \tau \in \{0, 1, \dots, T\} : s_\tau \in \mathcal{T}_{\text{tar}}$, where $T > 0$ is the maximum simulation time. Thus, our goal is formulated as a feasibility problem as:

$$\min_{m_0} \ 1,$$

$$\text{s.t.,} \quad \begin{bmatrix} s_{k+1} \\ m_{k+1} \end{bmatrix} = \begin{bmatrix} f([s_k^\top, m_k^\top]^\top, z) \\ g([s_k^\top, m_k^\top]^\top, z) \end{bmatrix}, \tag{4a}$$

$$s_0 = \bar{s}_0,$$

$$s_k \in \mathcal{S}(z), m_k \in \mathcal{M}(z), \ \forall k \in \{0, 1, \dots, T\}, \tag{4b}$$

$$\exists \tau \in \{0, 1, \dots, T\} : s_\tau \in \mathcal{T}_{\text{tar}}, \tag{4c}$$

where $\mathcal{S}(z), \mathcal{M}(z)$ are the set of constraints on the ball's and the movable objects' states, respectively, imposed by the environment, and $\bar{s}_0$ is a known constant vector. Note that we only consider taking the position of the ball to a small neighborhood of a target position $(x_{\text{tar}}, y_{\text{tar}})$ without any conditions on the target velocity. Therefore the set $\mathcal{T}_{\text{tar}}$ can be defined as:

$$\mathcal{T}_{\text{tar}} = \left\{ s : s \in \mathbb{R}^4, \left\| \begin{bmatrix} s^{(1)} \\ s^{(2)} \end{bmatrix} - \begin{bmatrix} x_{\text{tar}} \\ y_{\text{tar}} \end{bmatrix} \right\|_2 \leq \epsilon \right\}, \text{ for some } \epsilon > 0, \tag{5}$$

where $v^{(i)}$ denotes the ith component of any vector v. We assume the availability of prior knowledge about the size of all objects in the game, such as, the blocks, the ground, the ball, The acceleration due to gravity used in the game's simulation is also known. The availability of such prior knowledge is common in model-based control design.

Assumption 1 (Well Posedness) *We assume that there exists a feasible solution to problem* (4).

There are two main challenges in solving (4), namely: (i) time horizon T can be large, making computations cumbersome, and (ii) the dynamics in (4a) are affected by contact forces and friction, whose magnitude, direction and spatio-temporal characteristics are hard to predict. One way of resolving (ii) is to build high-fidelity models of all the dynamics equations in (4), and then learn its parameters (basically reverse engineering the simulator). This would be very hard and would require solving a Linear Complimentarity Problem at every time step Todorov et al. (2012); Hwangbo et al. (2018). To address these challenges, we propose a hierarchical framework to obtain a feasible solution to (4), focusing on a small problem locally and iteratively updating the local problem using observed trajectories of the ball.

In the next section, we outline the key principles that we use in our proposed approach to find a solution to (4). These principles are then elaborated further in Section 5, where we finally pose our tractable approximation to (4) in local partitions of the state-space.

---

1. The mass and the moment of inertia are calculated from the width and the height, since a constant, known object density is used in this game.

4. Proposed Approach Outline

In order to simplify the formalism of the proposed approach (which can be very notation-heavy when properly formalized) we next outline the three main principles used in our solution.

(i) Locality.
At each trial the solver restricts its focus to a region of the state-space called the *Local Region*, where a candidate solution to (4) requires improvement (as it has failed at the previous iteration). By doing so, we reduce the computational complexity associated with (4).

(ii) Iterative Trials.
A simulation of the game according to dynamics (1) is referred to as an iteration or trial. The data from an iteration is subsequently utilized to improve the candidate solution in the Local Region and decide a next Local Region.

(iii) Hierarchical Decomposition.
Three hierarchies are used in our solution and can be summarized as:

H-1: High-Level Guide Path: In this hierarchy we plan a path of the ball from the starting position of $\bar{s}_0$ to the target set $\mathcal{T}_{\mathrm{tar}}$ using an off-the-shelf planner, such as Probabilistic Road Map, RRT, RRT*. This path is called the Guide Path, and it only uses the geometry of the free space. Based on the deviation of a recorded ball trajectory from the Guide Path, we identify a Local Region, where objects need to be placed.

H-2: Kinematics and the Event Dependent Model: Using simple kinematics of the objects and the environment, we derive a set of prediction models describing the ball's state evolution. Each one of such models (referred to as $\bar{f}^{\mathrm{kin}}(\cdot,\cdot)$ in Section 5.2) is a function of the objects' state or the environment state which are predicted to enter in contact with the ball.

Main and Supporting Blocks. Inside the Local Region, the solver focuses on placing only *one* object which will enter in contact with the ball. We refer to such an object as the *Main Block*. The Main Block states are to be chosen to lower the deviation of the ball's predicted path from the Guide Path. The objects that constrain the position and velocity of the Main Block to the desired ones, are called the *Supporting Blocks*.

Event Dependent Optimization. The prediction model of the ball changes depending on the contact status, i.e., if the ball has entered in contact with some objects or the environment. For instance, the ball can enter in contact with an object and then roll on it or can bounce off to another object. We call each instance of such change an *event*. We model the ball's transition between any two such events in a Local Region, using model $\bar{f}^{\mathrm{kin}}(\cdot,\cdot)$. This enables fast predictions as a function of the Main Block's state.

H-3: Model Learning from Iteration Data: After the Main and Supporting Block states are chosen in a Local Region, we simulate the game according to (1) and record the ball's trajectory. This trajectory data is used to update the prediction models $\bar{f}^{\mathrm{kin}}(\cdot,\cdot)$ and refine the solution. The next Local Region is chosen as per H-1 using this trajectory and H-2 is repeated.

5. Proposed Approach Details

In this section we outline the optimization problem solved at each hierarchical level discussed in the previous section.

5.1. Hierarchy 1: Guide Path and Local Region

Next we formalize the algorithm at Hierarchy 1 (H-1). Denote by Γ_{in} a recorded trajectory of the ball $\Gamma_{\text{in}} = \{[x_{\text{ball},0}, y_{\text{ball},0}]^\top, [x_{\text{ball},1}, y_{\text{ball},1}]^\top, \ldots, [x_{\text{ball},\tau}, y_{\text{ball},\tau}]^\top\}$ at some trial j of the game evolving according to (1), with a given configuration of movable objects (the selection of the movable objects will be discussed in H-2 and H-3 in Section 5.2 and Section 5.3 later). At the first trial $j = 0$ the Γ_{in} is obtained by recording the ball motion with no moving objects placed in the environment. This is shown in Fig. 3a with the red circles.

Assume Γ_{in} does not solve the game (4). The goal is to solve the game (4) at the next trial $j+1$. An approximation of the desired trajectory of the ball to reach the target flag (see Fig. 1) is denoted by Γ_g and is computed by using an off-the-shelf path planner which uses no information about the game kinematics and dynamics. The deviation between Γ_g and Γ_{in} is used to identify a region $\mathcal{R}^{\text{loc}}(\Gamma_{\text{in}}, \Gamma_g)$ in the (x, y) space where movable objects are to be placed/replaced or moved. The planned path Γ_g is called *Guide Path*, as shown with the blue circles in Fig. 3a, and the region $\mathcal{R}^{\text{loc}}(\Gamma_{\text{in}}, \Gamma_g)$ is called a *Local Region*, as shown with the red rectangle in Fig. 3a.

Let the Guide Path be represented as an ordered set of vectors:

$$\Gamma_g = \{[x_0^g, y_0^g]^\top, \ldots, [x_\tau^g, y_\tau^g]^\top\},$$

where the planner trajectory states $\{[x_0^g, y_0^g]^\top, \ldots, [x_\tau^g, y_\tau^g]^\top\}$ satisfy the static environment constraints (4b)-(4c), but *not* the dynamics (4a). That is, the Guide Path only uses the geometry of the free space. Next we formally define the Local Region $\mathcal{R}^{\text{loc}}(\Gamma_{\text{in}}, \Gamma_g)$.

Definition 1 (Local Region) *For any $\epsilon_2 > \epsilon_1 > 0$, a Local region $\mathcal{R}^{\text{loc}}(\Gamma_{\text{in}}, \Gamma_g)$ is the smallest rectangle containing all v_k, u_k pair with $v_k \in \Gamma_{\text{in}}$ and $u_k \in \Gamma_g$ where $\|(v_k - u_k)\|_2 \in [\epsilon_1, \epsilon_2]$.*

We define $\Gamma_{\text{in}}^{\text{loc}} = \Gamma_{\text{in}} \cap \mathcal{R}^{\text{loc}}(\Gamma_{\text{in}}, \Gamma_g)$ and $\Gamma_g^{\text{loc}} = \Gamma_g \cap \mathcal{R}^{\text{loc}}(\Gamma_{\text{in}}, \Gamma_g)$.

At this hierarchical level, the problem of solving (4) is converted into solving a simpler optimization problem (over the initial state of the movable block) in the Local Region, trying to place the ball in a Local Target Set. A Local Target set is a neighbourhood of the last element of the guide path $\Gamma_g^{\text{loc}}(\text{end})$ in the Local Region, as shown in with the blue disc in Fig. 3a. Here, for a vector (ordered set of vectors) v, $v(\text{end})$ refers to the last component (vector) of v.

Definition 2 (Local Target Set) *For a given $\epsilon > 0$ we define a Local Target Set as:*

$$\mathcal{T}_{\text{tar}}^{\text{loc}} = \{s \in \mathbb{R}^4 : \|\Gamma_g^{\text{loc}}(\text{end}) - [s^{(1)}, s^{(2)}]^\top\|_2 < \epsilon\}. \tag{6}$$

In summary, at this hierarchical level (H-1): (i) a Γ_g, called the Guide Path is generated, (ii) given Γ_{in}, i.e., the recorded ball trajectory from last trial, and Γ_g, the Guide Path, we compute a Local Region and a Local Target Set, and (iii) problem (4) is then transformed into the following problem which attempts to place the ball in the Local Target Set while following the Guide Path:

$$\min_{k^{\text{loc}}, m_{k^{\text{loc}}}} \quad \ell(s_{T^{\text{loc}}}, \Gamma_g^{\text{loc}}(\text{end}))$$

$$\text{s.t.,} \quad s_{k+1} = f(\begin{bmatrix} s_k \\ m_k \end{bmatrix}, z),$$

$$m_{k+1} = g(\begin{bmatrix} s_k \\ m_k \end{bmatrix}, z), \tag{7}$$

$$s_{k^{\text{loc}}} = \Gamma_{\text{in}}^{\text{loc}}(k^{\text{loc}}),$$

$$s_k \in \mathcal{S}(z), \ m_k \in \mathcal{M}(z), \ \forall k \in \{k^{\text{loc}}, k^{\text{loc}}+1, \ldots, T^{\text{loc}}\},$$

where $\ell(\cdot, \cdot)$ is a positive definite cost function, $T^{\mathrm{loc}} \ll T$ and $k = k^{\mathrm{loc}}$ is the time instant when the ball's trajectory Γ_{in} requires a change of course through impact with an object. Here the optimization is formulated over k^{loc} and the initial states of the selected objects at time step k^{loc}. Spatio-temporal locality helps focusing on the small problem (7) instead of (4). Since the original problem (4) is to find the objects' state at time step $k = 0$, we reverse calculate m_0 from $m_{k^{\mathrm{loc}}}$ in (7), and assume this is always possible.

(a) Guide Path Γ_g (blue circles), recorded trajectory Γ_{in} (red circles), Local Region $\mathcal{R}^{\mathrm{loc}}(\Gamma_{\mathrm{in}}, \Gamma_g)$ (red rectangle), and the Local Target set $\mathcal{T}_{\mathrm{tar}}^{\mathrm{loc}}$ (blue disc)

(b) Main Block and Supporting Blocks inside the Local Region $\mathcal{R}^{\mathrm{loc}}(\Gamma_{\mathrm{in}}, \Gamma_g)$
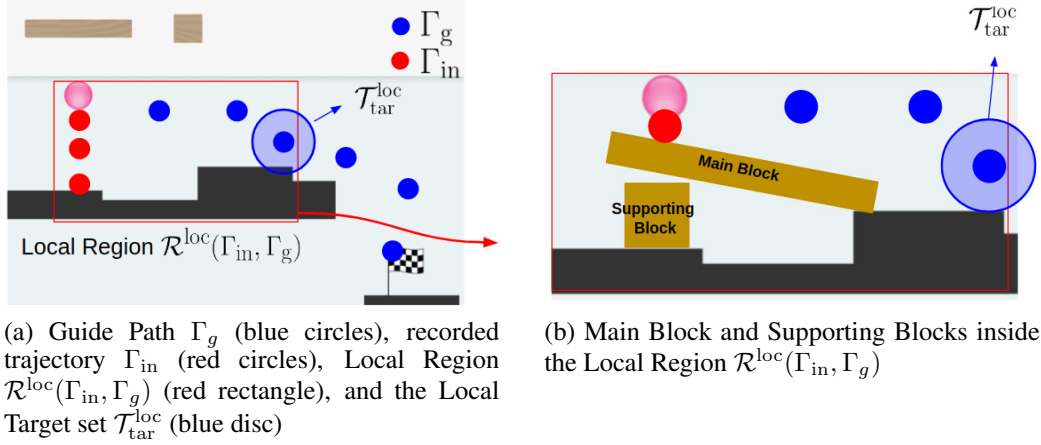
Figure 3

Even though we reduced computational complexity by focusing inside the Local Region, still the solution to (7) is complex due to the unknown/complex $f(\cdot, \cdot)$ and $g(\cdot, \cdot)$. Therefore, problem (7) is not solved at this hierarchy. The proposed algorithm instead builds a simple approximation to model $f(\cdot, \cdot)$ in the next hierarchy. This is done by focusing on only one object affecting the ball and using kinematic modeling. At this hierarchy H-2 the algorithm finds a feasible solution to (7) using this simple model approximation.

5.2. Hierarchy 2: Kinematics and the Event Dependent Model

In the Local Region, we allow only one object at a time (including the immovable objects) to affect the ball's motion. We call the movable object which will affect the ball's motion in the Local Regions as the *Main Block*. This is shown in Fig. 3b, where the rectangular movable block is the Main Block inside the chosen Local Region.

Consider the Main Block as the lth object out of the n objects, and denote its state at time k as m_k^l, which is a vector stacking $8(l-1)^{\mathrm{th}}$ to the $8l^{\mathrm{th}}$ element of m_k. Additionally, the objects used to constrain the states of the Main Block to the desired one are called the *Supporting Block*s. In Fig. 3b the square movable block is the Supporting Block, which ensures that the Main Block stays pivoted from the left, and maintains its chosen angle. Then, we can rewrite the dynamics from (1):

$$s_{k+1} = \bar{f}(s_k, m_k^l), \tag{8}$$

and since we focus on only m_k^l, we can also take only m_k^l from $g(\cdot, \cdot)$. We can write this as:

$$m_{k+1}^l = \bar{g}(\begin{bmatrix} s_k \\ m_k \end{bmatrix}, z) \tag{9}$$

Remark 3 *Argument m_k^l in (8) is substituted by z^q when the qth immovable object affects the ball.*

Remark 4 *When no object contacts the ball, $f(\cdot, \cdot)$ can be $\bar{f}_g(s_k)$, evolving as a free-fall.*

KINEMATICS BASED APPROXIMATION

Although by using (8) and (9), we reduce the complexity of the system evolution function in (7), we still need to solve the contact dynamics for the ball and the object in order to predict the ball's motion. Instead, we use kinematics based approximation of $\bar{f}(\cdot, \cdot)$, $\bar{g}(\cdot, \cdot)$. This allows us to predict the ball's trajectory across the Local Region to solve a tractable approximation to problem (7).

In order to do so, we classify C different possible types of contacts between the ball and the object, i.e., contacts involving ball and a line, ball and a circle, and so on. For each such type of interaction, we construct a kinematic model for the corresponding trajectories such as the ball moving along the line, changing the direction after a collision, etc. This is similar to the notion of building motion primitives, except that we consider contacts.

We notice that while the lth object, i.e, the Main Block, is in contact with the ball, the variation of it's state m^l is negligible in the Bubble Ball game. With this observation, model (9) is not needed to compute the evolution of (8) while the ball is in contact with the Main Block. In summary, we can write a kinematic approximation to (8) for a type j interaction with the Main Block l as:

$$s_{k+1} = \bar{f}^{\text{kin},j}(s_k, m_k^l), \text{ for some } j \in \{1, 2, \ldots, C\}. \tag{10}$$

By using this approach we obtain a tractable approximation to the optimization problem (7) as:

$$
\begin{aligned}
\min_{k^{\text{loc}}, m_{k^{\text{loc}}|k^{\text{loc}}}} \quad & \ell(s_{T^{\text{loc}}|k^{\text{loc}}}, \Gamma_g^{\text{loc}}(\text{end})) \\
\text{s.t.,} \quad & s_{k+1|k^{\text{loc}}} = \begin{cases} \bar{f}^{\text{kin},j_l}(s_{k|k^{\text{loc}}}, m_{k|k^{\text{loc}}}^l), & \text{while } m_{k|k^{\text{loc}}}^l \text{ is active} \\ \bar{f}^{\text{kin},j_q}(s_{k|k^{\text{loc}}}, z^q), & \text{while } z^q \text{ is active} \\ \bar{f}_g(s_{k|k^{\text{loc}}}), & \text{otherwise} \end{cases}, \\
& s_{k^{\text{loc}}|k^{\text{loc}}} = \Gamma_{\text{in}}^{\text{loc}}(k^{\text{loc}}), \ m_{k+1|k^{\text{loc}}}^l = m_{k|k^{\text{loc}}}^l, \text{ while } m_{k|k^{\text{loc}}}^l \text{ is active}, \\
& s_{k|k^{\text{loc}}} \in \mathcal{S}(z), m_{k|k^{\text{loc}}} \in \mathcal{M}(z), \ \forall k \in \{k^{\text{loc}}, k^{\text{loc}}+1, \ldots, T^{\text{loc}}\}, \\
& l \in \{1, 2, \ldots, n\},
\end{aligned}
\tag{11}
$$

where $s_{k|k^{\text{loc}}}$ is the predicted ball state at time step k according to the model $\bar{f}^{\text{kin},j}(\cdot, \cdot)$, and $m_{k|k^{\text{loc}}}^l$ is the predicted Main Block's state at time step k, when the predictions are made given the ball's recorded position at time step $k^{\text{loc}}, \Gamma_{\text{in}}^{\text{loc}}(k^{\text{loc}})$.

We now transform problem (11) into an *Event-Dependent* optimization by exploiting the simple nature of the contacts in the game. This lowers the computational complexity of solving (11) and also allows us to pinpoint where the switches in the ball's prediction model occur.

EVENT BASED SIMPLIFICATION

An *event* is the moment when the contact constraint on the ball changes. That is, an event occurs when the ball's motion model in (11) switches. For instance, in case of a free-falling ball hitting a block and rolling thereafter, an event is the collision with the block.

We define the *Event-Dependent* system as the state transition function which links the ball state from one event to the next event. More formally, let e_i be the time step when an event i involving $m_{e_i}^l$ happens. Then the corresponding Event-Dependent model $\bar{f}_{e_i}^{\text{kin},j}$ is denoted by:

$$s_{e_{i+1}} = \bar{f}_{e_i}^{\text{kin},j}(s_{e_i}, m_{e_i}^l), \text{ for some } j \in \{1, 2, \ldots, C\}. \tag{12}$$

Remark 5 *To obtain $\bar{f}_{e_i}^{\text{kin},j}(\cdot, \cdot)$ in* (12) *from $\bar{f}^{\text{kin},j}(\cdot, \cdot)$ in* (11)*, we simply forward integrate f $\bar{f}^{\text{kin},j}(\cdot, \cdot)$ over time in advance. In case of rolling, the integration stops when the motion reaches the object m^l boundary. In case of bouncing at time step e_i, the instantaneous value of $\bar{f}^{\text{kin},j}(\cdot, \cdot)$ at $k = e_i$ is used as $\bar{f}_{e_i}^{\text{kin},j}(\cdot, \cdot)$. The bouncing then switches to free-fall at $k = e_i + 1$. The same principle of forward integration as in rolling is also used for a free-falling motion using $\bar{f}_g(\cdot)$ and it is denoted as $\bar{f}_{g,e_i}(s_{e_i}, d)$ with x-distance d to the next event.*

Using (12) we transform the time-dependent problem (11) into an event-based optimization over a single event for one Main Block in each Local Region. In each Local Region, we limit the search over at most three events, namely (i) event 1: the ball enters in contact with the Main Block at time step e_1, (ii) event 2: the ball leaves the Main Block and goes in free-falling at time step e_2 (iii) event 3: the ball enters in contact with the environment at time step e_3, until e_4. Now, for event 1, Main Block is the only one affecting the ball, and after event 1 ends, only the environment z^q plays a role in the ball's evolution. With these notations, the reformulated optimization problem becomes:

$$
\begin{aligned}
\min_{e_1, l, m_{e_1}^l} \quad & \ell(s_{e_4|e_1}, \Gamma_g^{\text{loc}}(\text{end})) \\
\text{s.t.,} \quad & s_{e_1|e_1} = \Gamma_{\text{in}}^{\text{loc}}(e_1), \text{ (note that } e_1 = k^{\text{loc}}) \\
& s_{e_2|e_1} = \bar{f}_{e_1}^{\text{kin},j_l}(s_{e_1|e_1}, m_{e_1}^l), \ (j_l \text{ is known for } l) \\
& s_{e_3|e_1} = \bar{f}_{g,e_2}(s_{e_2|e_1}, d), \\
& s_{e_4|e_1} = \bar{f}_{e_3}^{\text{kin},j_q}(s_{e_3|e_1}, z^q), \ (j_q \text{ is known for } q) \\
& s_{e_i|e_1} \in \mathcal{S}(z), \ \forall i \in \{1,2,3,4\}, \ m_{e_1|e_1} \in \mathcal{M}(z),
\end{aligned}
\tag{13}
$$

where $s_{e_i|e_1}$ is the predicted ball state at event i using $\bar{f}_{e_i}^{\text{kin},j}(\cdot, \cdot)$, given e_1, and $m_{e_i|e_1}$ is the predicted Main Block's state at event i. This means solving (13) in the Bubble Ball is equivalent to (i) finding the best Main Blocks l, and (ii) finding $m_{e_1}^l$ with the corresponding Supporting Blocks states. where z^q is the given ground's state and d is the distance to the given ground. The decision variables are the starting time of event 1 (i.e., time of contact with the Main Block) (e_1), choice of the Main Block (l), the Main Block's state at contact $(m_{e_1}^l)$. For the numerical strategy used in solving (13), see (Lee et al., 2020, Remark 4).

Remark 6 (From Kinematic to Dynamic Modeling) *In our algorithm, we also need dynamic modeling including moment equilibrium and rotational dynamics to predict $\bar{g}(\cdot, \cdot)$ in* (9)*. We use such information in order to choose the Supporting Blocks' location, corresponding to $m_{e_1}^l$ from the solution in* (13)*. See Lee et al. (2020) for further details.*

5.3. Hierarchy 3: Model Learning From Iteration Data

In Hierarchy 3, we focus on data-driven iterative improvement of the kinematic models (12) derived in Hierarchy 2. Once (13) is solved and the Supporting Blocks are chosen, we simulate the game

engine according to (1). We obtain a set of transition data from this trial, denoted as $[s_{e_i}, m_{e_i}^l]_{i=1}^{i=N_e}$ from N_e Events along the entire collected trajectory. The models $\bar{f}_{e_i}^{\text{kin},j}(\cdot, \cdot)$ in (13) are parametrized with a vector of parameters β for each model. Then, we update β as:

$$\beta \leftarrow \arg\min_{\beta} \sum_{i=1}^{N_e} |Y_i - \bar{f}_{e_i}^{\text{kin},j}(s_{e_i}, m_{e_i}^l, \beta)|^2, \tag{14}$$

where Y_i is data of $s_{e_{i+1}}$ from observations along the collected ball trajectory.

Remark 7 *The process explained in Hierarchy 1 is repeated after we follow the algorithms outlined in Hierarchy 2 and 3 and find a solution trajectory $\Gamma^\star$ in the current Local Region. Recall Γ_{in} from Section 5.1. We reset $\Gamma_{\text{in}} \leftarrow \Gamma^\star$ and repeat the process until the game is solved.*

6. Analysis of the Results

We used our proposed approach to successfully solve a set of Bubble Ball levels. For many of the levels we solved, the simple kinematics based models from Hierarchy 2 captured the featured motions, and only a few of iterations in Hierarchy 3 were enough to refine the solution. Our BubbleBall Website gives a visual explanation of this proposed hierarchical approach and shows how it works under one or multiple Local Regions, and even a dynamic Main Block (e.g, a catapult in level 21). In Table 1, we show the number of trials needed to solve a set of levels, along with the average computation times of (13) in each Local Region. For the solved levels, up to 5 iterations were sufficient, implying that the hierarchy and the kinematics approximation is able to capture the complex game dynamics. For a detailed analysis of more levels, see Lee et al. (2020).

Level	Time to Solve (13) (s)	No. of Trials	No. of Local Regions	No. of Movable Blocks
1	0.116	1	1	1
5	1.211	2	2	2
6	3.736	1	1	2
8	1.138	5	3	6
21	5.732	4	1	5

Table 1: Empirical results from a set of solved levels

7. Conclusion

We proposed a hierarchical framework to solve the physics-based game Bubble Ball. Our approach exploits model hierarchy and model learning from iteration data. Hierarchy 1 uses a geometry-based high-level planner for planning a Guide Path of the ball. In Hierarchy 2, using low-level kinematic and dynamic modeling of the blocks, we solved an *Event-Dependent* optimization problem which decides how the game configuration is chosen for the ball to track the Guide Path. In Hierarchy 3, using the collected game data we improve our solution of Hierarchy 2. Our method successfully solved a large set of levels within reasonable number of trials.

Acknowledgments

References

David J Grymin, Charles B Neas, and Mazen Farhood. A hierarchical approach for primitive-based motion planning and control of autonomous vehicles. *Robotics and Autonomous Systems*, 62(2): 214–228, 2014.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pages 2944–2952, 2015.

Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

Jemin Hwangbo, Joonho Lee, and Marco Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018.

Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40(3):429–455, 2016.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. *arXiv preprint arXiv:1907.11180*, 2019.

Hotae Lee, Monimoy Bujarbaruah, and Francesco Borrelli. Learning how to solve bubble ball. *arXiv preprint arXiv:2011.10668*, 2020.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Robert Nay. *Bubble Ball Game*, 2010. URL https://www.naygames.com/bubbleball/.

Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks: a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017.

Ugo Rosolia, Andrew Singletary, and Aaron D Ames. Unified multi-rate control: from low level actuation to high level planning. *arXiv preprint arXiv:2012.06558*, 2020.

Xu Shen, Edward L Zhu, Yvonne R Stürz, and Francesco Borrelli. Collision avoidance in tightly-constrained environments without coordination: a hierarchical control approach. *arXiv preprint arXiv:2011.00413*, 2020.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Charlott Vallon and Francesco Borrelli. Data-driven hierarchical predictive learning in unknown environments. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 104–109. IEEE, 2020.

Kim Peter Wabersich, Lukas Hewing, Andrea Carron, and Melanie N Zeilinger. Probabilistic model predictive safety certification for learning-based control. *IEEE Transactions on Automatic Control*, 2021.

Boyu Zhou, Fei Gao, Luqi Wang, Chuhao Liu, and Shaojie Shen. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4): 3529–3536, 2019.