Babadi, Amin; Naderi, Kourosh; Hämäläinen, Perttu

Intelligent Middle-Level Game Control

Published in:
Proceedings of the 2018 IEEE Conference on Computational Intelligence and Games (CIG'18)

DOI:
10.1109/CIG.2018.8490407

Published: 13/08/2018

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Babadi, A., Naderi, K., & Hämäläinen, P. (2018). Intelligent Middle-Level Game Control. In Proceedings of the 2018 IEEE Conference on Computational Intelligence and Games (CIG'18) (pp. 25-32). Article 8490407 IEEE. https://doi.org/10.1109/CIG.2018.8490407

Intelligent Middle-Level Game Control

Amin Babadi
Department of Computer Science
Aalto University
Helsinki, Finland
amin.babadi@aalto.fi

Kourosh Naderi
Department of Computer Science
Aalto University
Helsinki, Finland
kourosh.naderi@aalto.fi

Perttu Hämäläinen
Department of Computer Science
Aalto University
Helsinki, Finland
perttu.hamalainen@aalto.fi

***Abstract*—We propose the concept of *intelligent middle-level game control*, which lies on a continuum of control abstraction levels between the following two dual opposites: 1) high-level control that translates player's simple commands into complex actions (such as pressing Space key for jumping), and 2) low-level control which simulates real-life complexities by directly manipulating, e.g., joint rotations of the character as it is done in the runner game QWOP. We posit that various novel control abstractions can be explored using recent advances in movement intelligence of game characters. We demonstrate this through design and evaluation of a novel 2-player martial arts game prototype. In this game, each player guides a simulated humanoid character by clicking and dragging body parts. This defines the cost function for an online continuous control algorithm that executes the requested movement. Our control algorithm uses Covariance Matrix Adaptation Evolution Strategy (CMA-ES) in a rolling horizon manner with custom population seeding techniques. Our playtesting data indicates that intelligent middle-level control results in producing novel and innovative gameplay without frustrating interface complexities.**

***Index Terms*—game control, physically-based simulation, multi-agent systems, online optimization, continuous control**

I. Introduction

Game control in present video games can be divided into *high-level* and *low-level* control [1]. In high-level control, the player is able to trigger actions such as punch or kick with a simple keypress. In contrast, there are low-level control approaches where the player directly manipulates the game system simulation. For example, using this approach in a martial arts game could mean that the player has to determine the torques applied to each joint of the character's body to produce a punch. The fighting game Toribash [2] and the runner game QWOP [3] are two of the games that have successfully used this kind of low-level control.

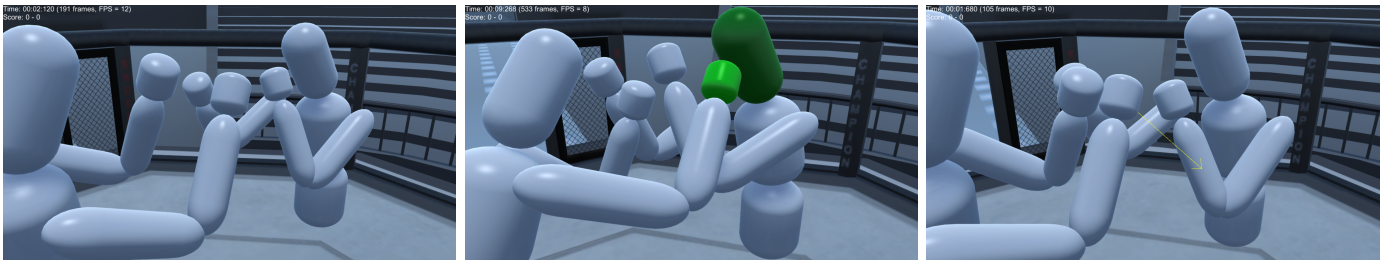
Low-level control allows maximal expressiveness, diversity/complexity, and control in game animations; it can also remove the cost of animation production in game projects. However, it usually makes the character control extremely difficult since it requires the player to manipulate several degrees of freedom with high precision, often under time pressure. On the other hand, games with high-level control usually come with a set of pre-defined smooth and natural animations. The downside is that the animations are costly to produce, and they do not allow for interesting novel movements to emerge [1].

In this paper, we propose *intelligent middle-level game control* by combining usability and flexibility of high-level and low-level controls, respectively. Using this approach, the player's commands are more abstract and simple than low-level control, but more detailed and expressive than high-level control. To make this possible, we utilize recent movement artificial intelligence (AI) techniques for physically-simulated characters. To clarify this definition, consider the martial arts game example again. Suppose the player can produce the command *"use your left hand to push the opponent's right hand away"* by left-clicking on the opponent's right hand. Then, the game automatically computes and then applies the required torques for producing the requested animation, adapting to the current physical state of the characters. In other words, player commands cause the animations to be generated on the fly and no pre-recorded animations are used. The main advantages of intelligent middle-level control are as follows:

1) The synthesized movements are novel and emergent similar to low-level control, but the player can focus on more strategic planning instead of micro-managing the simulation.
2) The complexity of a simulated human body's dynamics can create interesting challenges [1]. Both novelty and complexity are desirable from the point of view of inducing a feeling of curious interest in the player [4].
3) Since movement is not limited to pre-defined animations, more expressive and precise game controls can be designed and implemented.

Middle-level control has been explored before in a few games such as Octodad [5] and the original PC version of Rag Doll Kung Fu [6], albeit with less control intelligence. The games typically use some form of inverse kinematics which limits the behaviors that can be created. We demonstrate that intelligent middle-level control with online trajectory optimization allows realistic handling of physical constraints such as joint limits and non-penetration of colliding bodies.

We believe that intelligent middle-level control has the potential for introducing various novel gameplay. To support this claim, we developed a novel 2-player martial arts game prototype in which the players are able to control their physically-based humanoid characters through giving middle-level commands. We also developed an online continuous control trajectory optimization algorithm that computes the simulation control parameters needed to produce the requested movements. Screenshots of the game are shown in Fig. 1



(a) Task = *Null* (b) Task = *Punch in head using right hand* (c) Task = *Move right hand to specified position*

Fig. 1: Screenshots of our 2-player martial arts game prototype with 3 different tasks. The left character is controlled by the local player.

and a gameplay video is available online[1]. We evaluated this prototype by running a user study with 12 participants (6 human-vs-human pairs). The players reported that the interface enabled them to use various martial arts strategies, and that low-level controller was able to produce their commands with high precision.

The rest of the paper is organized as follows: A brief overview of literature is given in Section II. Section III explains the details of intelligent middle-level control and our martial arts game prototype. Section IV describes the details and results of the user study that was run for evaluating intelligent middle-level control in our game. Finally, Section V gives conclusions and Section VI analyzes the limitations and future lines of research in this work.

II. Related Works

In this section, we first give an overview of recent methods for synthesis and control of physically-based character animation, followed by game control interface research relevant to this work.

A. Character Movement Synthesis

Traditional animation technology has limited movement expressiveness and emergence, except for simple low-level simulation control (e.g., Toribash [2] and QWOP [3]). However, this is changing due to deep reinforcement learning and novel real-time movement optimization methods, which can endow game characters with expressive movement intelligence not limited to pre-defined animations. We refer the reader to [7] for a thorough introduction to character animation and physically-based simulation along with a survey on common techniques.

Simulation-Based Methods: In physical environments, behavior of objects and their interactions is usually difficult to model and predict. One of the most common approaches for character control in these environments is to use simulation-based methods. The basic idea behind these methods is simple: generate a number of action sequences, evaluate them using forward simulation and computing some cost function, and finally, choose the action sequence that minimizes the cost function.

If the simulation has differentiable dynamics, one can use dynamic programming version of gradient-based optimization [8] to control a variety of systems ranging from an inverted pendulum to a full humanoid. With black-box simulation, similar results were obtained by Sequential Monte Carlo sampling of control trajectories encoded as cubic splines [9]. Instead of a spline parameterization, Control Particle Belief Propagation (C-PBP) uses a Markov Random Field factorization for both sampling and smoothing trajectories [10]. Rajamäki and Hämäläinen [11] have recently shown that adding supervised learning on top of Monte Carlo tree search (MCTS) methods [12] can yield both robust control and low movement noise.

Several simulation-based methods have been developed using evolutionary computation. A recent study has used graph search along with Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [13] to develop an offline controller for solving humanoid wall climbing problems [14]. It has been shown that CMA-ES can be used in a rolling horizon manner in single-agent control problems with continuous states and actions [15] and 2-player games with discrete actions [16]. Another study has shown that performance of rolling horizon evolutionary algorithms can be improved significantly using simple population seeding techniques [17]. We are investigating this approach in the context of a more complex 2-player simulation with continuous actions and complex contact dynamics.

Reinforcement Learning: Reinforcement learning (RL) is a field of machine learning that studies how an agent should take actions in an environment in order to maximize rewards. In the past few years, RL has received a lot more attention due to remarkable results of Deep Reinforcement Learning (DRL) in Atari games [18] and the game of Go [19], [20]. These advances have inspired several breakthroughs in RL for continuous control. Mixture of actor-critic experts (MACE) accelerates learning by developing separate pairs of actors and critics such that each pair learns some part of the movement [21]. Schulman *et al.* [22] introduced a method called Trust Region Policy Optimization (TRPO) that uses a surrogate objective function and is able to learn several complex tasks such

[1]https://youtu.be/rnsSWY7HZJA

as swimming and walking. Another method called Proximal Policy Optimization (PPO) has managed to outperform TRPO by clipping the surrogate objective function [23]. The most important limitation of DRL methods is that they need a lot of simulation data and training time to learn. This can be a problem especially when iterating the reward function design.

Data-Driven Methods: Studies have shown that data-driven methods can be effective for generating robust and smooth movements. One of the studies has shown that kinematic controllers can be constructed by learning a low-dimensional space from motion capture data and interpolating in that space [24]. Motion matching is a similar kinematic method that uses a dataset of pre-recorded animations and in each frame finds the closest pose to the character's current pose such that desired future movement is produced [25]. Holden *et al.* [26] use convolutional autoencoders on a large motion capture data set to re-produce and interpolate recorded motions. Another study breaks control problem into short time fragments (0.1s in length), and learns a linear feedback control strategy for each fragment [27]. A more recent method, called DeepLoco, uses a combination of high-level and low-level controllers and is able to produce stable gaits given some reference motions [28]. Phase-functioned neural network is a recent neural network architecture that uses cyclic functions for computing the weights and is trained using a large dataset of pre-recorded animations [29]. Although data-driven methods are able to produce high-quality movements, it can be difficult to obtain the training data, and the resulting movement is limited by the data.

B. Game Control Interface

There is a lot of research on alternative input devices and interfaces, e.g., for controlling games with body movements [30]–[32]. On the other hand, some studies have proposed using traditional input devices but novel input-avatar mappings [1], [33].

A recent study has used predictive simulation for developing 6 novel game prototypes using low-level control of physically-based simulated characters [1]. Our work is close to this work as we solve the same problem of enabling expressive control of fully physically-simulated characters without excessive control difficulty. They propose predictive visualizations as the solution, whereas we explore the possibility of offloading some complexity onto the movement optimizer.

It can be argued that middle-level control is not an entirely new concept. For example, games like Octodad [5], and the original PC version of Rag Doll Kung Fu [6] have used inverse kinematics style control. Compared to Toribash [2] and QWOP [3], they provide the player with a slightly higher-level interface of directly controlling hand and feet locations instead of joint rotations. Our contribution is in demonstrating how modern physically-based optimization methods for online continuous control provide novel tools for exploring game control interfaces and abstractions.

III. Method

In this section we describe our 2-player martial arts game prototype demonstrating the intelligent middle-level control concept. At Section III-A, the details of reference model are explained. Then, at Section III-B we explain the design of intelligent middle-level control in our martial arts game prototype. At Section III-C, the structure of low-level controller is introduced. Finally, the network architecture of our martial arts game is explained in Section III-D. The values of all parameters introduced in this section are reported at Appendix.

A. Character Model

Characters in this game are upper-body humanoid characters with 16 actuated degrees of freedom (DOF) as shown in Fig. 2. Each character has 9 bones that are connected using 3-DOF ball-and-socket or 1-DOF hinge joints. We use Open Dynamics Engine (ODE) [34] for physics simulations.

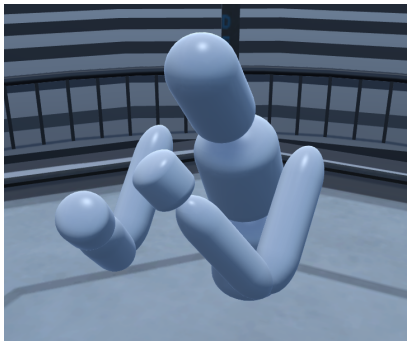


Fig. 2: Upper-body character model in its reference martial arts pose.

B. Novel Middle-Level Control for Martial Arts Games

The overall schema of our control system is shown in Fig. 3. In each frame, the current task of the character is determined and then a low-level controller plans a sequence of actions in order to complete the task. Then, the first action in the sequence is executed and the simulation goes to the next frame. This form of online optimization, i.e., 1) finding a multi-step solution, 2) applying the initial step of the solution, and 3) moving/rolling the horizon forward, is called the rolling horizon control (also known as receding horizon control) [35]. Note that the character is not necessarily given a new task in each frame in which case the task from previous frame is considered as the current task.

Tasks: Characters can have 2 main tasks: 1) *Move* hand to a specific position, and 2) *Punch* opponent either in the head or in the chest. The *Move* task is enabled using mouse drag in which case the desired position is determined based on hand's current position and mouse drag direction. A single click on opponent's head or chest enables the *Punch* task for those parts. Both tasks can be executed using either hands depending on the clicked mouse button (i.e., the left click for the left hand and the right click for the right hand). There is also a dummy *Null* task defined for specifying a character with

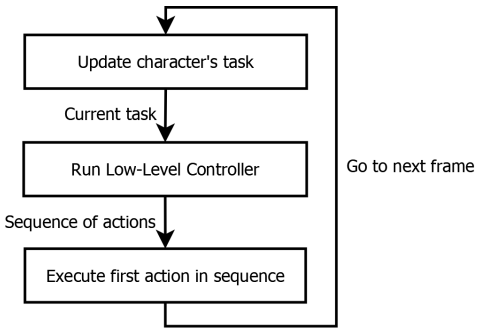


Fig. 3: Core loop in intelligent middle-level control.

no tasks. Character's task is set to *Null* if 1) current task is completed, or 2) time spent on task has passed some threshold $\tau_{Max\ Task}$.

Heads-Up Display (HUD) for tasks: The current task of character is indicated using a simple color coding as follows: 1) when the character is given a punching command, the operating hand and punching target are highlighted using green color, 2) when a successful punch is executed, the punching target will become red, and 3) when a moving command is set for one of the hands, a yellow line connects the current position of the hand to its desired position.

Additional input keys: Each player can use W/S keys to move his/her character on a horizontal line. We added this ability because in martial arts it is very important for the characters to adjust distance to their opponent. Finally, each player can rotate the camera around his/her character using A/D keys.

C. Low-Level Controller

In each frame, we need a low-level controller for achieving the character's current task. To this end, we developed an online simulation-based algorithm for physically-based continuous control. This algorithm uses CMA-ES in a rolling horizon manner along with custom population seeding techniques and outputs the best found action sequence up to a fixed horizon $\tau_{Horizon}$. CMA-ES is a common evolutionary algorithm that assumes a multi-variate Gaussian distribution as the underlying data distribution. In each iteration of CMA-ES, a new population is generated using some assumed distribution. Then, after evaluating the fitness of each individual, CMA-ES updates mean and covariance matrix of Gaussian distribution by selecting a subset of population with highest fitness values [13]. Population seeding means that in each iteration of CMA-ES, some proportion of the population is generated using external distributions. To the best of our knowledge, this is the first time that CMA-ES is used in a rolling horizon manner for online control of 3D physically-based simulated characters.

We first tried using the combined tree search and supervised learning approach of Rajamäki and Hämäläinen [11], but found that it had difficulties generating extreme dynamic movements such as punches. We then tried our present approach, as the combination of CMA-ES and a spline parameterization was successfully utilized in the dynamic climbing movement synthesis of Naderi *et al.* [14]. They however use CMA-ES for offline optimization instead of in online rolling horizon manner.

Overall Search Method: In each frame, we run CMA-ES update $n_{CMA\text{-}ES\ Updates}$ times. In each update, a population of size $n_{Population\ Size}$ is generated using mean and covariance of CMA-ES and 2 seeding techniques. Then the fitness value of each population member is computed and the CMA-ES updates its mean and covariance. After repeating this process $n_{CMA\text{-}ES\ Updates}$ times, first action in the best found action sequence is returned as the character's next action.

Spline Parameterization: We parameterize each action sequence using a cubic spline of $n_{Spline\ Points} = 3$ control points. This reduces the problem dimension significantly since we do not need to optimize each individual action in the sequence. Plus, interpolation between control points enforces coordination between body joints which results in smooth movements. We also optimize the time variable of each control point; so each spline is defined using $3 \times (16 + 1) = 51$ parameters.

Population Seeding Techniques: In each CMA-ES update, a fixed number of splines are generated using a multi-variate Gaussian distribution. The standard deviation of this distribution is σ_{Pose} and the mean is determined based on the following 2 seeding techniques: 1) $n_{Last\ Best}$ splines are generated by using the best spline found in the last frame as the mean. Note that at first CMA-ES update of each frame, we need to shift the last best spline by one frame so it becomes valid in the current frame. 2) $n_{Default\ Pose}$ splines are generated by using the default martial arts pose shown in Fig. 2 as the mean.

Fitness Computation: All action splines are evaluated in parallel using forward simulation up until some horizon $\tau_{Horizon}$ by assuming time step of $\Delta t = 1/30$ seconds and computing the reward (negative cost) in each time step. At the end of forward simulation, the fitness value of each action spline is equal to the average fitness value of all visited states during its simulation. The fitness value of a state s is computed by summing over the negation of 3 cost components as follows (the goal is to maximize the fitness):

$$Fitness\,(s) = -\left(Cost_{Pose}\,(s) + Cost_{Move}\,(s) + Cost_{Punch}\,(s)\right) \tag{1}$$

where values of cost components $Cost_X\,(s)$ are computed as follows:

1) $Cost_{Pose}\,(s)$: Cost of pose deviation is computed by finding the angle between current rotation of each bone ($q^b_{current}$) and its desired rotation ($q^b_{desired}$) as shown in the reference martial arts pose in Fig. 2. The cost is computed as:

$$Cost_{Pose}\,(s) = \sum_b \left(\frac{\angle\left(q^b_{current}, q^b_{desired}\right)}{\sigma_{Pose}} \right)^2 \tag{2}$$

where σ_{Pose} is used for indicating how much difference in rotation can be tolerated for each bone.

2) $Cost_{Move}(s)$: Cost of moving hand h is simply defined using the distance between current position $p^h_{current}$ and desired position $p^h_{desired}$ of the hand as follows:

$$Cost_{Move}(s) = \left(\frac{\|p^h_{current} - p^h_{desired}\|_2}{\sigma_{Move}} \right)^2 \quad (3)$$

where σ_{Move} is used for tuning the amount of tolerance for difference between current and desired positions.

3) $Cost_{Punch}(s)$: Punching is the most complicated cost component in our work. In a good punch, the hand touches the target with highest possible speed. Then the hand should get back to its relaxed position quickly so the character's guard is not down for a long time. In order to favor these movements, punching cost is computed as follows:

$$\begin{aligned} &Cost_{Punch}(s) = \\ &\mathbf{1}_{punch\ not\ happened?} \cdot \left(\frac{\|v^h_{current} - v^h_{desired}\|_2}{\sigma_{Hand\ Velocity}} \right)^2 - \\ &\mathbf{1}_{punch\ happened\ now?} \cdot PunchPower\left(v^h_{current}\right) + \\ &\mathbf{1}_{punch\ happened\ before?} \cdot \left(\frac{\|p^h_{current} - p^h_{desired}\|_2}{\sigma_{Hand\ Relax\ Position}} \right)^2 \end{aligned} \quad (4)$$

where $\mathbf{1}_A$ is the indicator function and is equal to 1 if the condition A is true, and 0 otherwise. Current and desired velocity of hand are denoted by $v^h_{current}$ and $v^h_{desired}$, respectively. Similarly, $p^h_{current}$ and $p^h_{desired}$ denote current and desired position of the hand, respectively. Similar to previous cost components, $\sigma_{Hand\ Velocity}$ and $\sigma_{Hand\ Relax\ Position}$ determine the amount of tolerance for difference between current and desired values of the hand velocity and position, respectively. The function call $PunchPower\left(v^h_{current}\right)$ maps current velocity of the hand h to a number in the range $[1000, 3000]$.

D. Network Architecture

One of the challenges for developing this prototype was how to design the network architecture. A critical limitation of previous multi-agent studies is that they mostly use competitive self-play RL, which is very slow and unreliable to train. We decided to run optimization in an interleaved manner to double our computing power. Our architecture is shown in Fig. 4. In this architecture, both the server and client do their own simulations in parallel by running the low-level controller on their devices. Then each player sends its next action to its opponent device for final simulation. Due to floating-point computation errors, the simulations on different devices are very likely to deviate. For solving this issue, server sends world state and score values to the client after its simulation is done and the client will then synchronize itself with the server.

Another important concern in this part was how partial information is handled. Each agent stores opponent's last action spline. Then, when evaluating new candidate splines, opponent's spline is simulated up until a fixed horizon $\tau_{Opponent\ Horizon}$ that is smaller than planning horizon, i.e., $\tau_{Opponent\ Horizon} < \tau_{Horizon}$. We did this because it is similar to real life where each character can anticipate movements of other characters by some error.

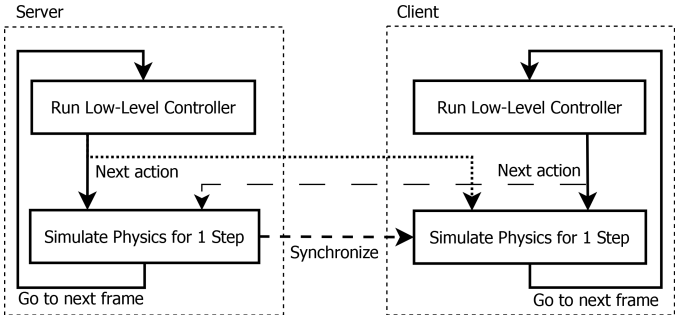


Fig. 4: Network architecture and interactions between server and client.

IV. Evaluations

A. Experimental Setup

We ran a user study involving 12 participants to assess the potentials of intelligent middle-level control in our martial arts game prototype. The participants had varying proficiency in video games and martial arts. Screenshots of our 2-player game are shown in Fig. 1 (each player controls the left character on his/her display).

Scoring: Each successful punch is rewarded with 1 to 10 points depending on its impact. The winner is the first player who gets 100 points.

Slow motion modes: The game was run in slow motion so that players have enough time for planning their movements. We hypothesized that our middle-level control interface could create a strategic *"body chess"* experience instead of a fast-paced action game. However, the camera rotates in real-time speed so the players are able to quickly adjust their point of view to see possible openings for attacks. Since we were not sure what is the right tempo for this game, we tested 3 different slow motion modes in this study. The chosen speeds for slow motion were 0.12x, 0.16x, and 0.2x. In order to find the best slow motion mode, each pair of players played one match with each slow-motion setting, with ordering of the settings counterbalanced between pairs. For the 6 pairs, each possible ordering was tested once.

Goals of User Study: The participants were asked to complete a questionnaire about the most important strengths and weaknesses of the interface. Since there are no games similar to this interface, it was not feasible to conduct a comparative quantitative evaluation. Thus, we designed the questionnaire using open-ended and qualitative questions on the following themes, with the goal of informing future work by both us and other designers and researchers:

1) What kind of combat techniques does the game allow the players to do?

2) Does the game allow novel gameplay behavior to emerge?
3) How much precision does the low-level controller have in executing players' commands?
4) Which slow motion mode is most suitable for achieving fun gameplay involving high-quality martial arts movements?
5) How can this martial arts interface be improved?

B. Results

Now we report the results obtained from 12 participants in our user study. Empty answers and answers that were irrelevant to the asked questions, are not included.

Slow Motion Mode: Slow motion modes of 0.12x, 0.16x, and 0.2x were chosen by 1, 4, and 7 people, respectively. The reported reasons are as follows:

1) The version with 0.12x speed does not produce fighting experience as the punches do not seem to be effective.
2) In the slowest mode (0.12x), it is difficult to anticipate the movements with good precision.
3) In two faster versions (0.16x and 0.2x), it is easier to control the character and react to opponent's moves.
4) The version with 0.2x speed is chosen by most of the participants because it produces the feeling of action more than other versions.

Our initial belief was that slower versions are easier to work with as they allow more time for planning and anticipation of movements. However, the results suggest that fast-paced gameplay may be more important for martial arts games.

Strategies: The main strategies reported by participants are as follows (some participants used more than one strategy):

1) Staying close and attacking aggressively (5 participants).
2) Using one hand for blocking and the other one for punching (4 participants).
3) Waiting for opponent to attack and then going for the punch (2 participants).
4) Looking for an open side and punching from that side (1 participant).
5) Getting hands through the defense of opponent (1 participant).

The variety of reported strategies and the gameplay videos suggest that middle-level control provides a good testbed for supporting different styles of gameplay.

Movement Precision: 8 participants reported that character executes the commands with high precision. Other 4 participants stated that controlling character in slow motion mode is difficult. This suggests that the control algorithm, despite its flaws, is doing a good job in synthesizing dynamic movements, but the control interface was not optimal for all participants.

Best Part: The participants were asked to name the best part of their experience while working with the interface. The answers are as follows:

1) Changing the color of body parts when punching (3 participants).
2) "*I like the idea of controlling both hands very much*".
3) "*The way that the hand and body part lit up when punching*".
4) "*The distance between characters can be adjusted in a good way*".
5) "*Seeing the game from the top*".
6) "*Fun to play against a friend*".
7) "*When it comes to punching, the character was quite creative*".
8) "*Different camera angles, realistic approach*".
9) "*Seeing nice landed punches*".
10) "*Nice to win*".

Worst Part: The participants have reported the followings as the worst parts of the interface:

1) Moving arms using mouse drag (3 participants).
2) "*Nothing was strikingly bad; but I had some trouble recollecting the right/left-click-equals-right/left-hand rule. At times, I found myself just clicking whatever clicked*".
3) "*It felt like the camera was so close to the body that you almost would like it to be first-person, and especially the camera angle above the head felt like it was from so much above it was not fun to use*".
4) "*Estimating of the time that it takes to hit*".
5) "*When trying to sweep the hands of the opponent away, it wasn't that responsive or intuitive to use*".
6) "*Both parties just end up punching each other, the game is over very quickly, and is not that fun*".
7) "*Unexpected rise of points in opponent's points when in close fight*".

Suggestions for Improvement: The participants also made the following suggestions for improving the game:

1) Adding game-like visual and audio effects (7 participants).
2) "*I think it would be cool if you could somehow with mouse make your punches' curves. Maybe dragging the mouse to show the desired curve movement for the hits*".
3) "*Moving the entire body could be possible*".
4) "*Moving hands by clicking and not dragging would make it easier to adjust hand positions*".
5) "*Allowing to crouch which makes it easier to block punches*".
6) "*Allowing to hit arms to incapacitate the other player from blocking punches using them*".

V. Conclusions

We have proposed the concept of intelligent middle-level game control, demonstrated and evaluated through a novel game prototype followed by a user study. This type of game control allows the player to produce novel gameplay through commands that are executed using a low-level controller without using any pre-recorded animations. In our 2-player martial arts game, each player controls a physically-based simulated character by giving commands such as *"punch in the chest using the left hand"*. Then a low-level controller executes the commands using a real-time control algorithm. Our online continuous control algorithm uses rolling horizon CMA-ES along with custom population seeding techniques.

We evaluated this prototype by running a user study involving 12 participants. The results show that the interface allows players to come up with various strategies for fighting. The participants have also reported that low-level controller is able to execute the commands with high precision. The users had some difficulties in mastering some of the mechanics such as camera movement and the *Move* task. However, we believe that the interface has potential for further research and novel game experiences.

VI. Limitations and Future Work

Full-Body Humanoid Characters: Our control algorithm is not currently robust enough to handle full-body humanoid characters in multi-agent environments. We have tested the algorithm successfully in locomotion tasks and single-agent settings. However, heavy perturbations make maintaining balance a very complicated problem. Since using legs is very important in martial arts, enabling this framework to work with full-body humanoid characters is a crucial direction for future work. Full-body characters should also provide enough realism to make the system useful for cognitive, strategic practicing of real martial arts, which we intend to investigate in future user studies.

Interface Design: Some reported that they prefer to see the movements in normal speed after they have given a command. However, most of the participants have stated that this character control system is fun and interesting. We are investigating possible ways for improving this interface.

Machine Learning: Currently our control algorithm does not apply any kind of machine learning for stabilizing movements. We have already got promising results by adding neural networks on top of our control algorithm in single-agent settings. Our tests show that adding machine learning can be a good approach for reducing sampling budget. Therefore, we believe that using machine learning is one of the low-hanging fruits of this work. On the other hand, being able to function without learning helps as it enables fast iteration when designing and testing the interface design.

Other Game Genres: In this work we evaluated intelligent middle-level control only in the context of martial arts games. A reasonable extension to this work is to apply this idea to other games in the sports genre where the quality of movements is important.

Acknowledgments

This work has been supported by the Academy of Finland grants 299358 and 305737.

Appendix

Table I shows the details of all important hyperparameters used in this study.

TABLE I: Algorithm parameters

Parameter	Value
Δt	$\frac{1}{30}$ s
$\tau_{Max\ Task}$	0.5 s
$\tau_{Horizon}$	0.6 s
$\tau_{Opponent\ Horizon}$	0.15 s
$n_{CMA\text{-}ES\ Updates}$	4
$n_{Population\ Size}$	16
$n_{Last\ Best}$	3
$n_{Default\ Pose}$	3
$n_{Spline\ Points}$	3
σ_{Pose}	20°
σ_{Move}	2 cm
$\sigma_{Hand\ Velocity}$	2.5 m/s
$\sigma_{Hand\ Relax\ Position}$	1 cm

References

[1] P. Hämäläinen, X. Ma, J. Takatalo, and J. Togelius, "Predictive physics simulation in game mechanics," in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '17. New York, NY, USA: ACM, 2017, pp. 497–505. [Online]. Available: http://doi.acm.org/10.1145/3116595.3116617

[2] "Toribash," http://toribash.com/, accessed: 2018-02-18.

[3] "QWOP," http://www.foddy.net/, accessed: 2018-02-18.

[4] P. J. Silvia, "Interest-the curious emotion," *Current Directions in Psychological Science*, vol. 17, no. 1, pp. 57–60, 2008.

[5] "Octodad," http://octodadgame.com/, accessed: 2018-03-09.

[6] "Rag Doll Kung Fu," http://www.ragdollkungfu.com/, accessed: 2018-03-09.

[7] T. Geijtenbeek, N. Pronost, A. Egges, and M. H. Overmars, "Interactive character animation using simulated physics." in *Eurographics (STARs)*, 2011, pp. 127–149.

[8] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4906–4913.

[9] P. Hämäläinen, S. Eriksson, E. Tanskanen, V. Kyrki, and J. Lehtinen, "Online motion synthesis using sequential monte carlo," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 51, 2014.

[10] P. Hämäläinen, J. Rajamäki, and C. K. Liu, "Online control of simulated humanoids using particle belief propagation," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 81, 2015.

[11] J. Rajamäki and P. Hämäläinen, "Augmenting sampling based controllers with machine learning," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2017, p. 11.

[12] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[13] N. Hansen, "The cma evolution strategy: a comparing review," in *Towards a new evolutionary computation*. Springer, 2006, pp. 75–102.

[14] K. Naderi, J. Rajamäki, and P. Hämäläinen, "Discovering and synthesizing humanoid climbing movements," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 43:1–43:11, Jul. 2017. [Online]. Available: http://doi.acm.org/10.1145/3072959.3073707

[15] S. Samothrakis, S. A. Roberts, D. Perez, and S. M. Lucas, "Rolling horizon methods for games with continuous states and actions," in *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*. IEEE, 2014, pp. 1–8.

[16] J. Liu, D. Pérez-Liébana, and S. M. Lucas, "Rolling horizon coevolutionary planning for two-player video games," in *Computer Science and Electronic Engineering (CEEC), 2016 8th*. IEEE, 2016, pp. 174–179.

[17] R. D. Gaina, S. M. Lucas, and D. Pérez-Liébana, "Population seeding techniques for rolling horizon evolution in general video game playing," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 1956–1963.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[20] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[21] X. B. Peng, G. Berseth, and M. Van de Panne, "Terrain-adaptive locomotion skills using deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 81, 2016.

[22] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] S. Levine, J. M. Wang, A. Haraux, Z. Popović, and V. Koltun, "Continuous character control with low-dimensional embeddings," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 28, 2012.

[25] S. Clavet, "Motion matching and the road to next-gen animation," in *Proceedings of Game Developers Conference (GDC)*, 2016.

[26] D. Holden, J. Saito, and T. Komura, "A deep learning framework for character motion synthesis and editing," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 138, 2016.

[27] L. Liu, M. V. D. Panne, and K. Yin, "Guided learning of control graphs for physics-based characters," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 3, p. 29, 2016.

[28] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 41, 2017.

[29] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 42, 2017.

[30] S. Krome, J. Holopainen, and S. Greuter, "Autogym: an exertion game for autonomous driving," in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '17. New York, NY, USA: ACM, 2017, pp. 33–42. [Online]. Available: http://doi.acm.org/10.1145/3116595.3116626

[31] T. Merritt, C. L. Nielsen, F. L. Jakobsen, and J. E. Grønbæk, "Glowphones: designing for proxemics play with low-resolution displays in location-based games," in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '17. New York, NY, USA: ACM, 2017, pp. 69–81. [Online]. Available: http://doi.acm.org/10.1145/3116595.3116598

[32] R. Kajastila, L. Holsti, and P. Hämäläinen, "The augmented climbing wall: high-exertion proximity interaction on a wall-sized interactive surface," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 758–769. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858450

[33] M. Sheinin and C. Gutwin, "Quantifying individual differences, skill development, and fatigue effects in small-scale exertion interfaces," in *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*. ACM, 2015, pp. 57–66.

[34] "Open dynamics engine," http://www.ode.org/, accessed: 2018-02-18.

[35] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, *Simulation-based algorithms for Markov decision processes*. Springer Science & Business Media, 2007.