

Beyond the Ragdoll: Designing a Safe-Falling Controller for Physically Simulated Characters

Lovro Boban^a and Mirko Sužnjević^b

University of Zagreb Faculty of Electrical Engineering and Computing, Zagreb, Croatia

Keywords: Reinforcement Learning, Computer Animation, User Study.

Abstract: In this paper we propose the design for a procedural, physically simulated animation system that produces safe falling animations using reinforcement learning. A character controller is trained to minimize external and internal forces on the humanoid character's body after it is pushed backwards on a flat surface. We design a questionnaire and conduct a user study that compares the reinforcement learning approach with a motion capture approach using subjective ratings on various aspects of the character's movement. Our findings, based on a sample size of (n = 25), indicate that users prefer the motion capture approach on 6 out of 8 aspects, but prefer the reinforcement learning approach on the aspect of reactivity.

1 INTRODUCTION

Character animation is an integral part of many video games, and despite growing visual fidelity over the years, one area of animation is still not very developed - interactive, procedural animation. While many video games do use minor procedural elements in their animations such as inverse kinematics for correct foot placement on uneven surfaces and cloth simulation for secondary motion, rarely do games use procedural animation for whole body motions. This can result in static animations that do not meaningfully adapt to the environment, thereby creating a disconnect between the player character and their environment, lowering immersion.

Procedural, physically simulated animations, however, do show up with some regularity in many video games in the form of "ragdolls". When a character trips and falls or is pushed, their body goes limp and they fall to the ground, often without even trying to protect themselves from injury during the fall. While some games do have so-called "active ragdolls" which curl up to protect themselves or try to retain balance before falling over, this area of animation is still underexplored.

In this paper we present the design of a procedural safe falling animation system based on reinforcement learning (RL) and we present the results of a study


that compares the system with a more traditional animation system based on motion capture. Section 2 looks at other works that have contributed to the topic of safe falling or producing animations using machine learning. Section 3 describes the design of our animation system. Section 4 describes the design and the results of the comparative user study, including a discussion of the results. Finally, section 6 summarizes the results and talks about where future work might be headed.


2 RELATED WORK

2.1 Safe Falling in Robotics

One discipline of engineering that is concerned with the execution of safe falling strategies is robotics. Aside from sensing the environment and navigating through it, safe falling methods for robots are of great interest to roboticists because an uncontrolled fall could result in serious damages to the robot, its environment, or living beings around the robot.

(Rossini et al., 2019) developed a method to compute the optimal falling trajectory for a robot to maximally reduce the velocity at points where it impacts the ground by squatting and stretching as it falls. The fall is modeled as a series of telescopic inverted pendulums - the standing robot is modeled as a four-link inverted pendulum, then as a three-link inverted pen-

^a  <https://orcid.org/0009-0003-4225-7146>

^b  <https://orcid.org/0000-0003-0334-8179>

dulum with a different pivot point when it drops to its knees and so forth. This method enables the robot to drop both forwards and backwards from a standing position. (Ha and Liu, 2015) take a similar approach by modeling the fall of the robot as a series of inverted pendulums. Multiple contact points with the ground are planned such that they maximally distribute the kinetic energy of the fall across each point instead of one contact point taking the brunt of the fall. This approach can manage more dynamic scenarios and can execute a rolling strategy if the original velocity of the robot is so high as to require it.

Whereas the aforementioned approaches seek to minimize damage to the robot itself, (Yun et al., 2009) take a different approach by steering the robot during its fall in such a way as to avoid falling on nearby objects and damaging them. The relation between the robot's center of mass and its base support polygon determines the balance of the robot and the direction of its fall. The robot can then change its fall direction by altering the base support polygon by stepping intelligently so that the leading edge of the polygon is oriented away from the object that it wishes to avoid.

While the previously described safe falling methods dealt with low velocity falls from a standing position, (Ha et al., 2012) have developed a real-time controller that allows a physically simulated character to land safely from a variety of heights and initial velocities by rolling upon contact with the ground. The controller first plans a landing strategy and how it will roll once on the ground depending on the initial velocity and rotation. It then continually changes its pose in mid-air to manipulate its moment of inertia in order to arrive in the position calculated at the beginning and executes a safety roll or a dive roll to better disperse the kinetic energy of the fall. Since the calculations are done in real time the controller is robust against external perturbations and is able to land in the desired pose even after unexpected mid-air collisions.

2.2 Machine Learning Approaches to Animation

Machine learning has found many uses in the broad discipline of animation. One example would be pose estimation which is a computer vision task that aims to estimate the position and orientation of a person or object. Pose estimation can help speed up the process of animating from a video reference by obviating the need to manually match the poses between the animated character and the video.

A different approach, that aims to produce the animation itself, is to train the character controllers of physically simulated characters using a group of

machine learning techniques known as reinforcement learning (RL). In contrast to traditional supervised machine learning approaches where the goal is usually classification and where the model is trained on an existing dataset with labeled examples, reinforcement learning models do not use datasets but rather interact with their environment to gain experience and learn a behavior policy that aims to maximize some reward function over time. An example would be an RL model for playing video games where the model is let loose inside an instance of a game and gradually learns to reach an increasingly higher score through trial and error. Some subfields of reinforcement learning are *deep reinforcement learning*, where RL is combined with deep learning (suitable for very large inputs like the entire rendered image of a video game), and *imitation learning*, where instead of a reward function, the learning is done by following example actions demonstrated by an expert with some techniques even having access to the expert at training time meaning that they can query the expert for more examples if necessary.

One such example of an RL agent exploring possible actions in its environment can be found in the work of (Yin et al., 2021) in which they trained a character controller to discover various high jump techniques via deep reinforcement learning. For the run-up, they trained a separate controller to imitate motion capture data of a run-up while also ending up in the desired takeoff state. Since the aim is to explore diverse jumping techniques, they varied the takeoff state using a novel Bayesian diversity search approach. After the takeoff and once in the air, natural looking poses were achieved by a Pose Variational Autoencoder with natural body poses extracted from a motion capture database. To ensure that the task of jumping over the bar wasn't too hard to learn, curriculum learning (Narvekar et al., 2020) was employed wherein the height of the bar, and thereby the challenge, was gradually increased as the policy improved, similarly to how people learn in reality.

(Yu et al., 2019) designed a character controller for a physically simulated figure skater that can perform various tricks. They used video footage of figure skating to extract key poses using pose estimation and applied trajectory optimization to construct a trajectory that visits all poses. Using deep reinforcement learning they generated a robust controller that can perform various figure skating skills. In their subsequent work (Yu et al., 2021), the authors used 3D pose estimation and contact estimation on monocular video footage of highly dynamic parkour motions. For each frame of video, they extracted the 3D pose of the athlete and used a contact estimator to esti-

mate whether the feet and hands were in contact with the environment. Combining pose and contact data, they were able to reconstruct the full motion trajectory from the video of a single panning camera. Using deep reinforcement learning they coached a motion controller inside a physical simulation to replicate the exact movements extracted from the original video.

All of the aforementioned ML approaches do not generate animations in real time, making them unsuitable for interactive scenarios. Luo et al. (Luo et al., 2020) use imitation learning to train a directable, real time motion controller to imitate reference animations of various gaits of a quadruped. A generative adversarial network is then used to train high level directives like speed and direction of movement and, finally, deep reinforcement learning is employed to train the controller to be more resistant to external perturbations. Tessler et al. (Tessler et al., 2023) take a similar approach by using motion capture data to train a humanoid motion controller to mimic it, but instead of training directly on the data they train on its statistical distribution thereby allowing for natural looking variations.

Reinforcement learning methods have also been applied to the problem of safe falling covered in section 2.1 such as in the work of (Kumar et al., 2017). An actor-critic architecture was used to develop a controller which minimizes the maximal impulse during the robot’s fall with each contacting limb of the robot having an actor-critic pair assigned to it. The discrete problem of contact planning is solved by determining the limb whose corresponding critic has the highest value function at a given point in time and having it be the one that will contact the ground next. The continuous control problem is solved via the optimization of the actors.

3 ANIMATION SYSTEM

Our safe falling animation system utilizes a machine learning approach wherein a reinforcement learning agent learns to control a fully physically simulated humanoid character by controlling the torque generated at each of the character’s joints. This places our approach into the category of procedural animation since there exists no predefined set of static animations. Instead, our system dynamically produces animations in real time by physically simulating the motion of the character.

We choose to use reinforcement learning because, to our knowledge, there doesn’t exist a comprehensive database of realistic falling animations that could be used for training a controller by using imitation

learning. Furthermore, the few examples of motion capture falling animations we could find have exaggerated motions that aren’t naturalistic.

However, to help with convergence we also employ curriculum learning where we first present the agent with an easier version of the problem (in our case that’s a less strict reward function) after which we gradually increase the difficulty once the agent surpasses a certain predefined reward threshold or if enough steps have passed. This helps the agent to quickly converge to an approximate solution in the first step of the curriculum, after which the solution is refined through smaller tweaks in the following steps of the curriculum. This can be seen on Figure 1 where the reward for the approach without curriculum learning returns a lower reward at the end despite both approaches having an equally strict reward function at that point.

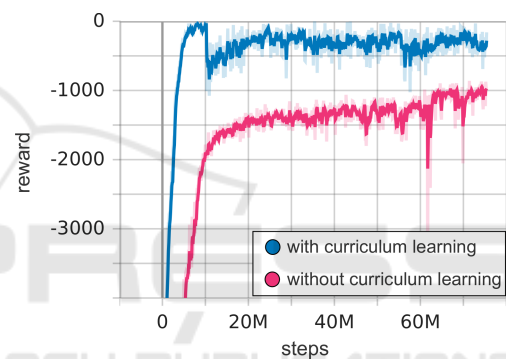


Figure 1: Learning curve with and without curriculum learning.

We use the MuJoCo¹ physics engine for physical simulation and the PPO algorithm with default parameters to train the controller. Using the Gymnasium² API, we design a custom environment for the RL agent to train in. Since Gymnasium already has the *Humanoid* environment with a defined humanoid character, we choose to adapt it for our needs.

3.1 Custom Learning Environment

We adapt the already existing *Humanoid* environment that comes with Gymnasium, which is itself based on the environment introduced by (Tassa et al., 2012). Just like the original environment, our environment consists only of the humanoid character and the floor that it stands on, although we have changed the character model by adding a neck joint which enables the character to tuck their head in.

¹<https://mujoco.org/>

²<https://gymnasium.farama.org/index.html>

At the beginning of each training episode, the character is pushed backwards in order to trigger a fall. To ensure that the controller learns to handle different push forces, the strength of the initial push is randomly chosen from a predefined range. If the character hits its head strongly enough, the episode terminates prematurely and a big negative reward is given. The reward function is structured thusly:

$$\text{reward} = \text{healthy} - \text{head_hit} - \text{ctrl} - \text{constr} \quad (1)$$

with the components having the following meaning:

- *healthy* - healthy reward, a constant positive reward given for each elapsed training step.
- *head_hit* - a fixed negative reward given upon a strong enough head hit. Prematurely terminates the episode.
- *ctrl* - a negative reward proportional to the square of the agent's control signals.
- *constraint_cost* - a negative reward proportional to the constraint forces acting on each part of the character's body. This represents outside forces acting on the body (analogous to impact forces) as well as forces that keep the joints together (roughly analogous to forces going through the tendons and ligaments). It's only counted for those forces that go above a certain threshold.

As mentioned earlier, the curriculum changes the reward function to be more forgiving at earlier steps by increasing the threshold above which constraint forces will be counted in *constraint_cost*. Additionally, we found that disabling *head_hit_cost* in the first step of the curriculum helps to produce a more natural looking results where the character doesn't contort itself into unnatural shapes in order to avoid hitting its head.

3.2 Resulting Character Controller

As can be seen in the accompanying video³, the trained character controller exhibits different behaviors depending on the strength of the initial push. At weaker and medium push strength, the character pushes off of the ground in order to create a forward rotation which counteracts the initial backwards rotation of the push. The character lands in a crouch with its hands touching the floor in front of it to increase stability. At higher push strength, the character performs a roll onto its back (see Figure 2) because the push is so strong that the character cannot generate enough forward rotation to counter it. However, a

backwards roll also helps to prolong the fall, spreading out the impact over a longer period of time thereby making it safer. The backwards roll can be seen in real life in situations when one is falling backwards at significant speeds, e.g. when landing in parkour. Additionally, another behavior emerges during the backwards roll, namely the character puts one of its arms above and behind the head to halt the roll and prevent it from going over the head.

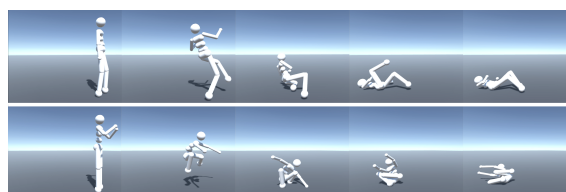


Figure 2: Example motions from the two animation systems compared in the user study. Top row - motion capture, bottom row - our RL system.

4 USER STUDY

In order to test how our animation system compares to a more established approach, we have conducted a user study where we compare animations produced by our approach to animations recorded via motion capture. To enable us to effectively compare animations produced by our system with other types of animation, we made a custom program using the Unity game engine.

In order to direct focus onto the animations themselves instead of the 3D humanoid model, we use the same, abstract humanoid model that comes with MuJoCo for both animation systems. This also helps to avoid presentation differences between the motion capture's 3D model and our system's 3D model from impacting user scores. Another reason for choosing to use MuJoCo's model over a more realistic humanoid model is the fact that the animation data from motion capture can be trivially mapped onto MuJoCo's simpler humanoid model whereas there is no simple and unique mapping from joint positions in MuJoCo onto a more complex humanoid model. Since MuJoCo's model has fewer degrees of freedom than the animation skeleton of the motion capture animations (it lacks hands and feet) the motion capture skeleton would be left underactuated.

4.1 Program for Comparing Animations

We designed a program for comparing animations which showcases one animation system at a time with

³<https://bit.ly/4fVCwWH>

the ability to switch between animation systems at will. These two animation systems are our RL animation system and a simple motion capture animation system. The RL and motion capture systems are represented by two scenarios, Scenario X and Scenario Y, respectively. The scenario names are not descriptive so as not to give away the nature of the animation system currently in use, since this could inadvertently influence the participant’s ratings.

The main view of the program can be seen on Figure 3 where the character is placed near the middle of the screen with UI elements on the right side which enable the user to push the character backwards. The slider controls the strength at which the character will be pushed and clicking the “Push” button will initiate a falling motion using the current animation system. After an animation finishes, the character appears back at its original standing spot, ready to be pushed again.

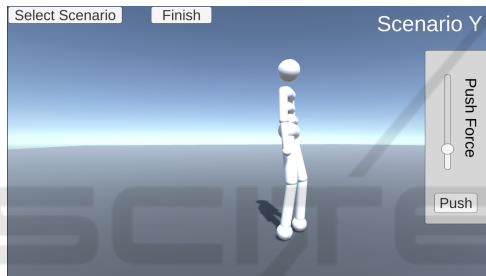


Figure 3: Main view of the animation comparison program.

The animation system that uses motion capture is constructed using an animation state machine consisting of 4 states: *idle*, *light*, *medium* and *hard*. Depending on the the strength of the push, the system transitions from the *idle* state into one of the other three. This way, the motion capture animation system can still react in proportion to the strength of the push, but not as granularly as the RL animation system. To load and play motion capture animations, the program uses Unity’s inbuilt functionality. The position and rotation of the animation bones are read from the animation clip and the 3D model is manipulated to exactly match the current pose from the animation clip.

On the other hand, to play animations from MuJoCo, the program has to communicate with a running instance of MuJoCo’s physics simulation. On each new frame in Unity, the program requests and subsequently receives pose data of the humanoid (in the form of the position and rotation of each bone) from the running physics simulation. This pose data is then used to make the 3D model assume the exact pose that the simulated humanoid currently has in the physics simulation. Upon clicking the “Push” button, the program sends the strength of the push (read from

slider) to the MuJoCo instance which resets the environment with the received push strength as the parameter and starts simulating. When the episode finishes, the simulation pauses until another command to reset the environment is received.

4.2 Participants

We involved 25 participants in the study (14 female and 11 male). We chose the number of participants in accordance with ITU-T recommendations (ITU-T, 2022). Participants are aged between 22-67 with the average age being 30.16 and the median age being 28.

The participants report their level of familiarity with 3D action video-games on a 5-point Likert scale with “Never played” on the lowest end of the scale and “Played extensively” on the highest end of the scale. The genres of sports games and combat oriented games were listed as examples of 3D action video-games. Participants also report how important animation realism and animation variety are to them in the context of video games, also on a 5-point Likert scale with “Not important at all” on the lowest end of the scale and “Very important” on the highest end of the scale. The distribution of the participants’ experience with 3D action video-games is shown in Table 1 and the reported importance of animation realism and animation variety is shown in Table 2.

Table 1: Participants’ experience with 3D action video-games.

	familiarity level				
	1	2	3	4	5
# of responses	5	3	4	6	7

Table 2: Participants’ reported importance of animation realism and animation variety.

	importance				
	1	2	3	4	5
realism	0	1	8	9	7
variety	0	5	6	11	3

Participants were also asked whether they had participated in activities where falling is commonplace (e.g. martial arts, rollerblading, gymnastics, parkour etc.) because of the assumption that those who do have experience with falling might answer questions about safe falling differently to those who do not.

4.3 Questionnaire

Our questionnaire consists of eight questions whose aim is to assess the users’ personal opinion on 8 different aspects of the animations produced by our

RL animation system, namely *animacy*, *anthropomorphism*, *gracefulness*, *surprise*, *likeliness*, *injury*, *realism* and *reactivity*.

Questions about *animacy* and *anthropomorphism* are meant to gauge whether the character passes as human, questions about *gracefulness* and *injury* are meant to gauge the perceived level of the character's skill, questions about *surprise*, *likeliness* and *realism* are meant to gauge whether the character is convincing, and the question about *reactivity* is meant to gauge whether our RL approach is significantly more interactive than motion capture.

Inspired by the Godspeed Questionnaire for Human Robot Interaction (Bartneck et al., 2009), we opted to use 5-point semantic differential scales meaning that participants were asked to indicate their position on a scale between two bipolar words, the anchors. Five questions were posed as semantic differential scales: *animacy* (*Mechanical/Organic*), *anthropomorphism* (*Not human-like/Humanlike*), *gracefulness* (*Clumsy/Graceful*), *surprise* (*Surprising/Expected*) and *reactivity* (*Unchanging/Reactive*). The question about animacy was taken directly from the Godspeed Questionnaire, whilst the one about anthropomorphism was adapted from the questionnaire. We used 5-point Likert scales for the other three aspects by posing the following questions - "Is a person likely to fall similarly to how the character fell?" for *likeliness*, "Does it look like the character injured themselves?" for *injury*, and "Does it look like the character's movements follow the laws of physics?" for *realism*.

4.4 Study Procedure

Testing begins with participants signing a written consent form that describes the kinds of data that is collected. Further, participants are asked to provide demographic information (gender and age) and to describe their level of experience with 3D action video-games, and how important animation realism and animation variety are to them. Additionally, participants are asked whether they have participated in an activity where falling is commonplace

After providing demographic data, the participants begin using the animation comparison program which randomly shows them one of two scenarios. Participants are instructed to push the character as many times as they want while varying the push strength with the aim of developing an opinion on the character's movement. After the participants feel that they have watched the character fall a sufficient number of times, they fill out the questionnaire described in 4.3. Once they've filled out the questionnaire, the

participants return to the program and switch to the other scenario. They repeat the process of forming an opinion on the character's movement after which they again fill out the same questionnaire as before, this time rating the animation system from the second scenario. Finally, after having filled out questionnaires for both scenarios, they choose which scenario they prefer overall - *Scenario X*, *Scenario Y* or *No preference*.

4.5 Study Results

The participants' ratings for each question in the two questionnaires can be seen in Figure 4 in the form of a diverging bar chart. The proportions of responses to each of the questions are shown in their corresponding stacked bar chart and all of the stacked bar charts' "neutral" sections (representing the proportion of "3" ratings on a 5-point scale) are aligned in the middle to allow for a better visual overview. The more a bar chart is to the right, the more participants answered towards the right part of the scale, and vice versa.

What can be seen is that for every question except for *injury* and *reactivity*, the motion capture system received higher scores. Comparing whether there is a difference between systems by running a Wilcoxon Signed-Rank Test on each pair of questions, it is shown that there does exist a statistically significant difference (*Organic* $p = 0.002$, *Humanlike* $p < 0.001$, *Graceful* $p = 0.042$, *Expected* $p = 0.002$, *Likely* $p < 0.001$, *Realism* $p = 0.002$, *Reactivity* $p = 0.038$) for every question except (*Injury* $p = 0.94$), even after adjusting p-values via the Benjamini-Hochberg procedure because of testing multiple hypotheses. This is consistent with the answers of overall preference where, out of 25 participants, 22 prefer the motion capture system, 2 have no preference, and 1 prefers the RL system.

Running ANOVA (Analysis of Variance) tests to check whether falling experience influences injury ratings and whether 3D action video-game experience influences reactivity shows no statistically significant correlations (*Injury* $p = 0.670$, *Reactivity* $p = 0.958$)

In conclusion, the results indicate that users do perceive the RL system to be more reactive, which is in agreement with the stated aim of developing an RL based animation system. On the other hand, the system doesn't produce animations that look safer than motion capture and, additionally, all other aspects of the animation suffer to varying degrees.

4.5.1 Participants' Comments

Looking at participants' comments about the RL system, one of the more common comments is that the

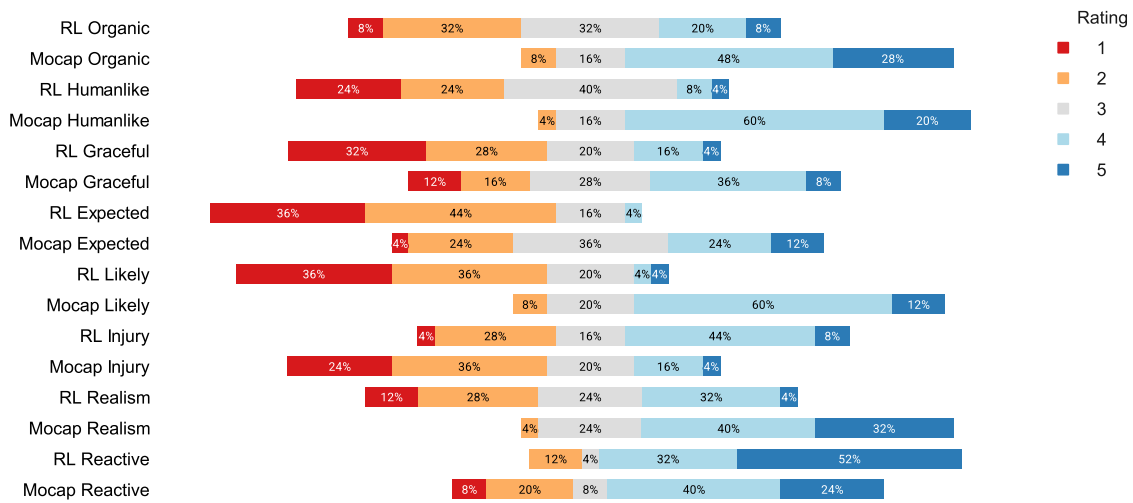


Figure 4: Participants’ ratings for each question on a diverging bar chart.

character is twitchy and too curled up after landing from a push at low strength, with one participant commenting that it reminds them of insect behavior. Participants also expressed confusion about where on the body the character is pushed which might have made it more difficult for them to appropriately rate *likeliness* and *reactivity*. For the motion capture system, multiple participants have expressed that the fall looks overly dramatic or like the character is faking the push. For both systems, participants have commented that high velocity falls look more natural.

5 DISCUSSION AND FUTURE WORK

Regarding high velocity falls looking more natural in both systems, for RL this could be the case because the character rolls onto their back instead of just landing in a crouch. For motion capture this could be the case because at high velocities the stunt actor has less room to act or dramatize since they have to take more care not to hurt themselves.

Regarding lower *animacy* and *anthropomorphism* scores for the RL system, we plan to improve this by further tweaking the reward function to reduce any twitchy motion. We also plan to make the humanoid model more realistic by making individual parts of the body more resilient or less resilient against large forces in accordance with human anatomy, thereby guiding the controller towards more realistic behavior. *Likeliness* and *anthropomorphism* could also be improved by involving human feedback in the training loop like is done in Reinforcement Learning from Human Feedback or through a method similar to (Ha

and Liu, 2014) where commands like “straighten the legs more” could be issued to the optimizer.

The majority overall preference for the motion capture system may be explained by the choice of a flat environment without any obstacles scattered throughout which favors motion capture since it is likely the exact environment in which the original motion was captured by the performer. On the other hand, a procedural approach, such as the one we developed, shows its strength in more complex, unpredictable environments with uneven ground or obstacles in the way of the fall. In such complex environments, motion capture animations might clip through or get stuck on surrounding geometry without the appropriate physical reaction that procedural approaches can provide. In our future work we plan to train the controller in environments that could involve uneven terrain, slopes, obstacles, drops, etc. These more complex environments might, however, make convergence more difficult.

An alternate procedural approach that focuses on realistic motion would be to use motion capture as a baseline by employing imitation learning, with reinforcement learning allowing for the introduction of reactivity, similarly to (Luo et al., 2020), although this might limit the controller to only those scenarios available in motion capture which is almost always on flat ground.

Ultimately, despite the RL system’s on average lower scores on various aspects compared to the motion capture system, our RL animation system can nevertheless be considered as a viable alternative to motion capture in cases where reactivity and animation variety are the primary concerns. This is often the case in games which have motion controls, notably VR games, that allow the player to interact with

virtual characters at a much more granular level than in traditional games. Nevertheless, further efforts in improvement of overall acceptability of RL generated animations are needed.

6 CONCLUSION

In this paper we describe the design of a safe falling animation system based on reinforcement learning and present the results of a user study comparing said animation system with an animation system based on motion capture.

Our animation system consists of a reinforcement learning agent that controls an articulated, fully physically simulated 3D humanoid character. The character is pushed backwards while standing on flat ground and the controller tries to minimize the impact of the character with the ground and the forces inside the character's joints. The PPO algorithm is used for training the controller and curriculum learning is employed in order to help with convergence.

We conduct a user study comparing the RL approach with a motion capture approach whilst keeping the surface presentation the same by using the same humanoid 3D model in both animation systems. The participants are instructed to make the character fall by pushing it at varying strengths and to develop an opinion on the resulting movement. They are subsequently given a questionnaire in which they rate the character's movement on 8 different aspects on 5-point semantic differential scales. This testing procedure is carried out for both animation systems individually.

Results show a statistically significant difference in ratings between the two systems for all but one aspect (*injury*), with the motion capture system being rated more favorably in 6 out of 8 aspects, and the RL system being rated more favorably in *reactivity*. The impact of background variables (experience with falling, previous gaming experience) on ratings of *injury* and *reactivity*, respectively, is shown to not be statistically significant.

ACKNOWLEDGEMENTS

This research was fully supported by the Croatian National Recovery and Resilience Plan (NPOO) under the project Research and Development of Multiple Innovative Products, Services and Business Models Aimed at Strengthening Sustainable Tourism and the Green and Digital Transition of Tourism (ROBO-CAMP), with grant number NPOO.C1.6.R1-I2.01.

REFERENCES

- Bartneck, C., Kulić, D., Croft, E., and Zoghbi, S. (2009). Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *Intl. journal of social robotics*, 1:71–81.
- Ha, S. and Liu, C. K. (2014). Iterative training of dynamic skills inspired by human coaching techniques. *ACM Transactions on Graphics (TOG)*, 34(1):1–11.
- Ha, S. and Liu, C. K. (2015). Multiple contact planning for minimizing damage of humanoid falls. In *2015 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS)*, pages 2761–2767. IEEE.
- Ha, S., Ye, Y., and Liu, C. K. (2012). Falling and landing motion control for character animation. *ACM Transactions on Graphics (TOG)*, 31(6):1–9.
- ITU-T (2022). P. 910: Subjective video quality assessment methods for multimedia applications. pages 15–16.
- Kumar, V. C., Ha, S., and Liu, C. K. (2017). Learning a unified control policy for safe falling. In *2017 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS)*, pages 3940–3947. IEEE.
- Luo, Y.-S., Soeseno, J. H., Chen, T. P.-C., and Chen, W.-C. (2020). Carl: Controllable agent with reinforcement learning for quadruped locomotion. *ACM Transactions on Graphics (TOG)*, 39(4):38–1.
- Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *The Journal of Machine Learning Research*, 21(1):7382–7431.
- Rossini, L., Henze, B., Braghin, F., and Roa, M. A. (2019). Optimal trajectory for active safe falls in humanoid robots. In *2019 IEEE-RAS 19th Intl. Conference on Humanoid Robots (Humanoids)*, pages 305–312.
- Tassa, Y., Erez, T., and Todorov, E. (2012). Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE.
- Tessler, C., Kasten, Y., Guo, Y., Mannor, S., Chechik, G., and Peng, X. B. (2023). Calm: Conditional adversarial latent models for directable virtual characters. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–9.
- Yin, Z., Yang, Z., Van De Panne, M., and Yin, K. (2021). Discovering diverse athletic jumping strategies. *ACM Transactions on Graphics (TOG)*, 40(4):1–17.
- Yu, R., Park, H., and Lee, J. (2019). Figure skating simulation from video. In *Computer graphics forum*, volume 38, pages 225–234. Wiley Online Library.
- Yu, R., Park, H., and Lee, J. (2021). Human dynamics from monocular video with dynamic camera movements. *ACM Transactions on Graphics (TOG)*, 40(6):1–14.
- Yun, S.-k., Goswami, A., and Sakagami, Y. (2009). Safe fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping. In *2009 IEEE intl. conference on robotics and automation*, pages 781–787. IEEE.