Interim Report

FYP Number: fyp23001

Project Title: Physical-based Character Animation in Fluids

Group Member:
Sinan Wang (3035770599)
Zebin Guo (3035770915)

Supervisor:
Professor Taku Komura

ABSTRACT

In this report, our final year project is presented, offering a detailed exposition than our initial project plan. The project encompasses two key components: physical simulation and character animation. For physical simulation, we employ neural networks, specifically a world model, to accurately simulate fluid dynamics. Leveraging this model, control policies are trained to control characters within these simulated fluids, aiming for realistic animation. An aspirational goal is to extend this control from the virtual realm to actual physical environments. Since our last report, we have successfully developed a complete solver as planned. However, we've faced challenges with storage capacity, prompting a potential revision of our strategies. Our immediate focus is to delve into differentiable fluid simulation and self-supervised learning literature, assisting our decision-making process regarding strategic adjustments on the training process of the world model.

ACKNOWLEDGEMENT

We would like to express my sincere gratitude to my supervisor, Taku Komura, for his invaluable guidance and support to our final year report. His expertise and mentorship have been instrumental in shaping the outcome of this project. Thank you for your unwavering dedication and belief in our abilities.

ABBREVIATION TABLE

Abbreviations	Explanation
MDN-RNN	Mixture Density Network-Recurrent Neural Network
SOTA	State-of-the-art

Table of Contents

List of Figures

List of Tables

1. Introduction

With the advance of the Metaverse, physical simulation and character animation are becoming increasingly popular. Many prior works associated with fluid simulation have diligently sought to emulate the physical rules in virtual environments. Noteworthy among these are endeavors employing fluid solvers, such as utilizing the stream function (Ando, 2015) to mimic fluid behavior based on vorticity. Despite the precision and comprehension of physical laws in these simulations, employing pure mathematical calculations within physical solvers has revealed inherent drawbacks. For example, numerical computation bans the simulated model from future learning and adaptation.

Fortunately, the emergence of the world model paradigm is a promising avenue. This paradigm represents a new direction for physical simulation. It integrates neural networks to comprehend virtual environments, as evidenced by notable achievements from PlaNet (Hafner, 2019) to DayDreamer (Wu, 2023) conducted by Google Research. However, these accomplishments focus on techniques in training control policies within uncomplicated environments like planes and vacuum, but these ideal conditions are unreachable in the real world. Therefore, this project is motivated by the concept of the world model. We shift the focus to realizing a dynamic robot simulation within complicated fluid environments. This environment centered on vorticity, adding verisimilitude to the simulation of a complex fluid environment. Hence, the project is titled "Physical-Based Character Animation in Fluids," aiming to achieve a comprehensive world model for fluids and deliver a control policy of rigid-body agents based on this world model. This project is to prove the effect of differentiable world models and their potential to substitute traditional fluid. Our project will progress in two stages. Firstly, we deploy a differentiable world model using neural networks. The world model inputs previous states and actions and predict the current. Also, this model retains hidden states, facilitating the memorization of states predating the previous state. Secondly, the project formulates a control policy for robots, accepting current and hidden states as input and producing corresponding actions. These actions are then iteratively fed into the world model to perpetuate the generation of future movement sequences. An additional reward function based on states, is manually defined as the optimization objective. Consequently, the deliverables of this project comprise a specialized world model and a control

policy tailored for vorticity-based fluid environments.

The subsequent sections of this report unfold as follows. First, the rest of **section 1** will introduce basic concepts like supervised learning and physical-based animation, then move to the objectives and deliverables. Subsequently, in **section 2,** we present an adapted outline of the world model for fluid environments, accompanied by an operational framework of training control policy. We will also elaborate on some choice justifications in this project. In **section 3**, current results and the data scale problem will be discussed, and two solutions to the data storage problem, namely self-supervised learning for the world model and autoencoder method for data compression, are proposed. Further, in **section 4,** we delineate the milestones achieved and a tentative schedule for future work. **Section 5** is the conclusion of this report.

1.1. Background

1.1.1. Physical Simulation

Physical simulation involves using computers to model the real physical world, a concept that originated from applied mathematics and physics. In these simulations, real-world objects are often abstracted into simpler virtual mathematical entities, such as circles and rectangles. These simplified objects are assigned properties like mass and size, and their interactions adhere to established physical laws. For instance, a basketball's fall in reality can be simulated as a sphere's descent in a virtual environment, with the simulation calculating the ball’s velocity.

Over time, our capability to simulate the physical world has significantly expanded. We've progressed from simulating just solids to now also simulating fluids and their interactions with solids. This project specifically focuses on fluid simulation, including the challenges of solid-fluid coupling.

Originating from Computational Fluid Dynamics, fluid simulation is an extensively researched field, drawing interest from mathematicians, physicists, and computer scientists. The cornerstone of most fluid simulation problems is the Navier–Stokes (NS) equations, a set of partial differential equations that describe fluid flow. Our task is to solve these equations

computationally. A critical aspect of this process is addressing the solid boundary conditions, ensuring accurate representation of solid-fluid interactions.

1.1.2. Character Animation

Character animation is a specialized area within the field of computer graphics, focusing on bringing virtual characters to life through movement. It is a blend of art and technology, where artists and animators use software tools to create the illusion of thought, emotion, and personality in digitally rendered characters. Initially rooted in traditional animation techniques, character animation in computer graphics has evolved significantly with advancements in technology.

In computer graphics, character animation is often driven by a combination of methods, including keyframe animation, motion capture, and procedural techniques. Keyframe animation, a traditional approach, involves manually setting specific frames where the character assumes distinct poses. Motion capture, on the other hand, involves recording the movement of a real actor and translating it onto a digital character, offering a more realistic motion. Procedural techniques, such as simulations for cloth or hair dynamics, add further realism to characters, especially in their interactions with the environment.

Moreover, character animation in computer graphics is not just about movement but also about creating characters that can interact believably with virtual environments and other characters. This requires a deep understanding of anatomy, physics, and storytelling, making it a multidisciplinary endeavor. The ultimate goal is to create characters that audiences can connect with, contributing to the overall narrative and experience of the digital medium, be it in video games, films, or virtual reality.

1.2. Motivation

A significant portion of character animation research predominantly focuses on the interaction with solids, with relatively few forays into the dynamics involving fluids. Yet, in our everyday lives, fluids—from the air we breathe to the water we interact with—play an indispensable role. This discrepancy highlights a crucial area of potential exploration and innovation in character

animation. The integration of machine learning techniques, as exemplified by the implementation of reinforcement learning in projects like DeepMimic (Peng, 2018), demonstrates promising advancements in this field. Furthermore, the progressive developments in fluid simulation within computer graphics, coupled with the emergence of world models which try to simulate the world by neural networks, present a compelling foundation for our project. These advancements not only signify the maturing of the field but also provide a starting point.

1.3. Objectives

Our project is guided by four sequential objectives. Firstly, our aim is to develop a fluid solver that is not only fast and robust but also proficient in handling fluid-rigid coupling effectively. Secondly, Leveraging the fluid solver, a differentiable world model is trained to simulate fluid environment and the fluid-rigid coupling. Thirdly, building upon the world model, control policies are trained to control characters within fluid environments, with the goal of achieving physically realistic animation. Finally, we aspire to extend this virtual control to tangible, real-world applications, if feasible.

1.4. Deliverables

Our deliverables are set to include a comprehensive program that, upon receiving inputs such as a virtual character, a virtual environment, and a command (e.g., 'move left'), will enable the character to move left on the screen in a manner that is physically realistic. Additionally, if feasible, we aim to produce a technical paper documenting our methodologies and findings.

1.5. Related Work

In this section, we mainly focus on the literature review for the physical simulation aspect. Regarding the character animation part, our investigation is in its preliminary stages, as we have yet to delve extensively into the relevant literature.

Solving Navier-Stokes (NS) equations for fluid simulation in computer graphics gained popularity starting with the works of Foster and Metaxas (1997) and the introduction of 'Stable

Fluids' by Stam (1999). These methods predominantly employed pressure projection to ensure the velocity vector field's solenoidality, meaning the field remains divergence-free. However, it was observed that these methods often struggled to maintain the persistence of fluid vortices (which would dissipate quickly) and sometimes failed to fully enforce solenoidality. In response, Elcott et al. (2007) suggested using the vorticity formulation of the NS equations, with vorticity as the primary variable for fluid simulation. Subsequently, numerous variants of vorticity-based formulations emerged, such as those by Angelidis and Neyret (2005), based on Lagrangian vortex filaments, Gamito et al. (1995), focusing on vortex particles, and Pfaff et al. (2012), centered on vortex sheets. Ando (2015) also made significant contributions by utilizing the stream function to simulate multi-phase flows, incorporating solid-fluid coupling solutions.

The methods previously mentioned are both accurate and physically correct. However, their limitations lie in speed and, crucially, the lack of differentiability, which is a key requirement for training agents in the character animation segment of our project. Addressing this challenge, Ha (2018) introduced the concept of the 'world model,' an innovative approach that leverages neural networks to simulate the real world, which is both rapid and differentiable. As artificial intelligence and machine learning continue to grow, similar groundbreaking works have emerged, including projects like DayDreamer (Wu, 2023).

2. Methodology

In this section, out approach is elaborated in greater detail, which is divided into four distinct steps. The initial step involves constructing the fluid solver. Following this, we deploy robots to interact with the solver, gathering data over time to create a comprehensive dataset. The third step is utilizing this dataset to train our world model. Because of the storage issue of the supervised learning approach, as discussed in **Section 4.2**, another self-supervised learning approach utilizing the differentiable feature of Navier-Strokes Equation will be discussed and listed as an additional method to train the world model. Ultimately, the trained world model is employed to train our control policy for controlling the characters. The first three steps belong to the physical simulation part, while the last step is the character animation part. These four steps are combined together and presented in Figure 1.

2.1. Fluid Solver Construction

To understand how the fluid solver works, it's crucial to first understand what a fluid solver is. Essentially, a fluid solver predicts future states of a system based on its current state and actions taken. In this context, 'states' refer to all the physical properties of the fluids and any solids within them. For example, this includes the velocity vector field, which indicates the speed and direction of the fluid motion at every point in the simulation domain. 'Actions' describe the interventions in the system, like a robot preparing to jump by exerting forces on its leg. Since the robot body is connected by joints, these forces are represented as torques on joints like described in Figure 2. To perform actual calculations, these abstract concepts are translated into mathematical objects. Consider our simulation domain as a 1-meter cube. To handle this computationally, we divide it into a 256 x 256 x 256 grid. Formally, the velocity vector field becomes a tensor of shape (256, 256, 256, 3), representing a vector of 256 vectors, each sized (256, 256, 3). The actions are also modeled as vectors.

The challenge then is predicting the state tensor and action vector for a future time step, given their current state and action. The time step is usually a small time amount, say 0.001 seconds. The state tensor at time t is denoted by s_t, and the action tensor at time t is denoted by a_t. To predict future states, the NS equations are solved numerically by a vorticity-based method. Vorticity, essentially the curl of the velocity, is the primary variable in our approach (where curl is a vector calculus operator indicating the rotation of the vector field).

In the solution process, we first 'advect' the vorticity field using the velocity field. Advection, in simple terms, is like tracking the movement of smoke in the air; it's about how the vorticity field changes and moves over time. After advection, a Poisson equation is solved to obtain the next time step's velocity field from the vorticity field. Solving this equation is akin to filling in a puzzle where we know how the vorticity swirls and twists, and we need to figure out the resulting fluid motion. These two steps, advection and solving the Poisson's equation, are complex and involve many technical details that we won't delve into here.

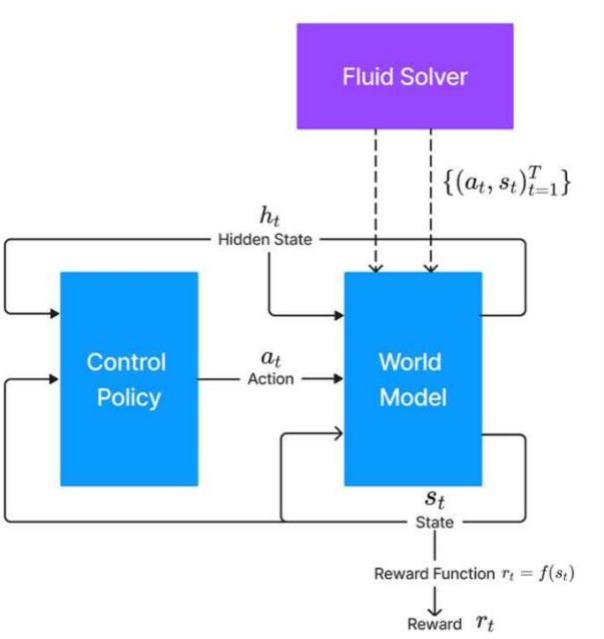

Figure 1. The framework of this project. The fluid solver provides action-state pairs for the world model. The world model then uses the data to learn a general transition model $p(s_t|s_{t-1}, a_{t-1})$. Then with a learned world model, a control policy can be trained using the traits of neural network backpropagation from the user-defined energy function.

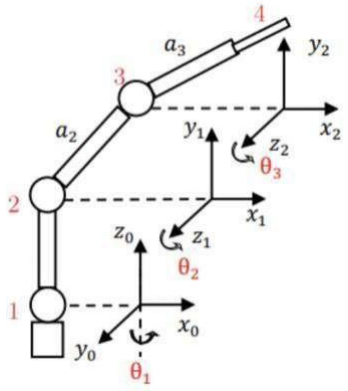

Figure 2. Example of joint representation of robots. The position and orientation represent each joint. For each joint, the orientation is represented by the respective angles in 3 domains of freedoms. (Zhou, 2023)

2.2. Dataset Creation

After the fluid solver is built, we let some robots make random actions in the solver, and collect the state-action pair, denoted by $\{(a_t, s_t)$, t ranges from 0 to the end of the simulation$\}$.

2.3. World Model Training by Supervised Learning

A completed fluid solver should be capable of handling various agents with different joints, processing exerted torques and calculating joint conditions in the next time slot. For simplicity, the

fluid solver can produce multiple $\{(a_t, s_t)_{t=1}^{T}\}$ sequences, where T is the terminated time step. Therefore, the construction of the fluid solver can provide sufficient data, as shown in Figure 1. With sufficient data, the world model and control policy can be trained separately. This project selects the agent model proposed by World Models (David Ha, 2018) as the prototype of our models. We reuse the concept of the MDN-RNN world model and the linear control policy model proposed in that paper. However, this report removes the factor of observations and VAE models, which introduce latent vectors. We also change the optimization method for training policy, ensuring the differentiability, which is the original intention of introducing the world model, is well maintained. This model is chosen because it is a SOTA work while maintaining great simplicity and impendence of different models, leading to much easier implementation and lower requirements for computing technologies.

Recall that the world model is trained as an RNN model, predicting the current state using the previous state and action. In this scenario, this world model initializes a hidden state and accepts the previous state and action, as shown in Figure 3. Then, using an RNN model, a distribution of future states can be predicted.

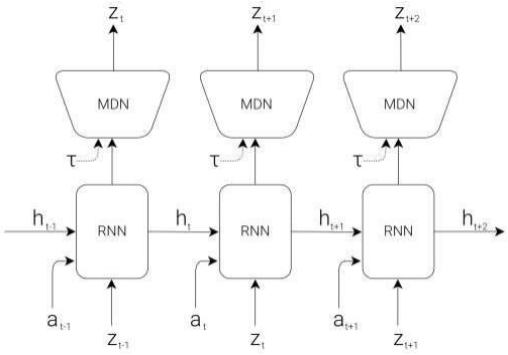

Figure 3. Illustration of MDN-RNN model (from World Models). The hidden state in each time step is transmitted to the next after augmenting current information with some impact factors. (David Ha, 2018)

In our world model, we use the MDN layer with noise to learn the pattern of the predicted distribution of the output of the RNN model, which is the next state in our case. A random noise is introduced to enhance prediction uncertainty by arbitrarily modifying several parameters. With the help of noise, the MDN-RNN model can reach more uncertain scenarios. This uncertainty can bolster the model's understanding of the surrounding environment. The intuition is trivial. Imagine

child A learns to walk on a straight road and child B learns to walk on a tortuous path. Typically, B will learn better with sufficient time, as he gains more experience dealing with uncertain circumstances. However, learning generalizations from uncertainty requires more time and resources, and the choice of degree of randomness is also significant, as we need to circumvent our model to learn pure randomness to "cheat" the training examiner.

2.4. World Model Training by Self-Supervised Learning

Supervised learning is adopted widely in simulating virtual environment. However, the storage problem of dataset persists. From **Section 2.2**, we are required to store enough action-state pairs of robots and fluid particles to expect a fair training effect, which is usually out of personal reach and considered hard to be utilized in training process, with more details discussed in **Section 4.2**. Therefore, we propose an alternative self-supervised method to train the world model, stemming from the idea from Li et al. (2023). As in simulation process in the fluid solver, the fluid state is updated iteratively, with the process of fluid advection, pressure projection and coupling of fluid and rigid particles. As the stepping process can be viewed as the transmit of a neural network, we can carefully define an energy function which serves as the objective function to minimize on and calculate the gradient of variables in every simulation process. Compared to generate thousands of sequences of samples to generate a dataset for supervised learning, we only require the production of several samples of frames as “ground truth”. After the initial states of fluid and rigid particles are set, they are simulated to the target frames and generate the output using pre-defined energy function, which usually constitutes the displacement and velocity difference of simulated particles and target particles. The difference is then backpropagated to optimize the trajectory and initial state of particles.

In this approach, the problem can be formulated as an optimization problem involving design variables, namely the initial states of both agents and fluids. Initially, we randomly select the initial states of rigids and fluids, and subject them to simulation using the vorticity method for multiple time steps. Following the predefined simulation duration, the obtained simulation result is then compared to the ground truth using a specific energy function. Subsequently, the partial derivatives with respect to the initial states, denoted as $\frac{\mathrm{d}s^{n+1}}{\mathrm{d}s^{0}}$, are calculated using the chain rule. To optimize the initial states, the backpropagation technique can be employed. The differentiation procedures of

PDEs should be carefully designed and require experiments to check if potential problems exist, like gradient explosion, as mentioned in DiffFR (Li et al., 2023).

2.5. Control Policy Training

The control policy is defined as a function which maps the current state to the current action, given the virtual character, environment, and the objective that we want the character to achieve. Assuming we've acquired a well-trained world model, our control policy can be described by a straightforward linear equation: $a_t = W[z_t h_t] + b$. The state z and the hidden state h for the time step t are combined to characterize past states. Having the current state, we are then able to compute the current reward. Given that our control policy model is a single-layer linear model, and the world model employs an RNN neural network, the gradient can be backpropagated via the chain rule. Consequently, traditional machine-learning techniques can be used to optimize the control policy parameters (Das, 2017).

Further, using the self-supervised learning approach in section 2.4 enables a similar training procedures as the world model by fixing the initial position and orientation of the agent as part of the known “state”, and we train other states using back propo

3. Current Results and Difficulties

3.1. Current Stage and Reproduction Result Analysis

Our fluid solver is currently operational, but its proficiency in handling solid-fluid coupling requires further enhancement. Concurrently, we are exploring alternative methods to train the world model, with preliminary results presented in Table 1. Due to a significant storage challenge, we have not utilized our own dataset. To illustrate, consider a simulation domain of (256, 256, 256) and the requirement to train the world model with 10,000 samples. This would necessitate approximately 1875 Gigabytes of storage, which is far beyond our capacity. As a result, we are contemplating a shift from supervised to unsupervised learning methods, which would alleviate the need for such an extensive dataset.

Original Score	Our best score			
	Training step = 1e3	Training step = 1e4	Training step = 1e5	Training step = 1e6
1092±556	146.2	455.7	641.2	611.3

Table 1. Comparison Between the Paper Result and Our Reproduction Result. Even when we train the code for 100 times of the original work, the best performance of our result only reaches about 60% of the original score.

Further, the capability of employing the idea from the paper DiffFR (Li et al., 2023) is checked. After successfully reproduce the original paper, I found that the C++ library SPlisHSPlasH is used to simulate the behaviors. In the paper, the differentiation feature of DFSPH method is implemented in this library by the author, and it is further compiled as a python library to be invoked in PyTorch. The torch code written for training is found operational and easy to reuse for our project. Nonetheless, significant efforts will be required in the future to establish a suitable differentiation scheme for our solver.

3.2. Large Dataset Storage Issues in Training the World Model

Even if the overall stream task for training the world model by supervised learning seems impeccable, one major problem exists. The problem is the dataset size in supervised learning of the world model. We are supposed to store the state-action pair of fluid particles and solid agents in the dataset. However, when we move to the actual experiment, the size problem occurs. Imagine the resolution of the environment is 256×256, each pixel's data should be stored in each simulation frame. In the standard setting, we have 60 frames per second to achieve smoothness of simulation. Moreover, according to other simulation works using supervised learning, a sufficient data sample size is necessary and essential for a satisfied machine learning effect. Usually, the sample size should be at least 1000 frames each for multiple sequences. Also, for every pixel in every frame, one state matrix comprising the velocity, position, orientation, angular speed, and external forces is stored with a total dimension of more than 100. As every float number requires 4 bytes in physical memory, the data storage size is more than 100GB, making it too bulky to handle during the training process, and utilizing this large-scale dataset to achieve optimal training results typically requires a substantial model size and extended training duration. The following section will discuss two feasible solutions to address this problem. One solution is to remove the reliance on data, altering it to a carefully designed energy function. Another solution is to compress the data before training while preserving most of the information from the original

data.

3.3. Two Feasible Solutions to the Storage Problem

The first solution is to use self-supervised learning described in **Section 2.4**. As the objective of our project is to mimic the behavior of fluids and agents, we can use solvers to generate the result that serves as the ground truth. Then by the idea of Li et al. (2023), an energy function denoting the distance of the current result and the ground truth can guide the optimization direction of initial parameters. However, this proposal requires the energy function to be defined to maintain the divergence-free property while representing all particles' optimization objectives. Therefore, more investigations are required to define the function before calculating the gradient of states based on the stream function.

The second solution is to compress the data using the auto-encoder method. The idea from Deep Fluids (Kim et al., 2019) shows that using an auto-encoder can compress the state of fluids like smoke to a latent vector space. The representation of the auto-encoder is shown in Figure 4 below. The state of smoke can be well compressed using a loss function that combines supervised learning and an unsupervised learning approach. As our project simulates water, another kind of fluid, this idea is apt for our work. However, compressing data with large sizes requires more time and sophisticated considerations. Additional explorations should be expected to leverage the auto-encoder in our project.

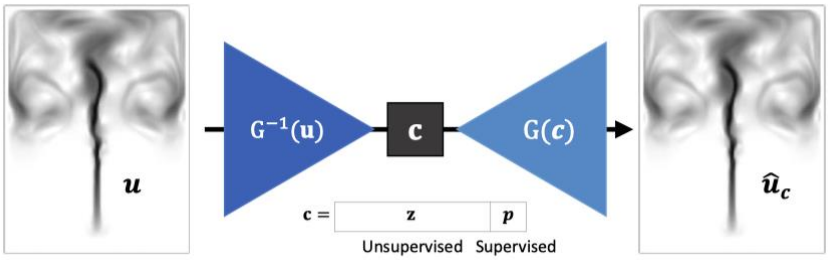

Figure 2. Representation of auto-encoder. The decoder is used to examine the effect of information preservation by re-constructing the original smoke from the compressed data. (Kim et al., 2019)

4. Remaining Working Plan

Given the challenges encountered, our immediate priority is to decide whether a strategic shift is necessary, with two feasible solutions mentioned in the section above. To make this decision, we need to thoroughly review literature pertaining to differentiable fluid simulations, world model and self-supervised learning. Once we've reached a decision, we will proceed with the training of the world model and subsequently focus on refining the control policy based on this model. Our tentative timeline for these activities is outlined in Table 2.

Date	To-Do
Before 12.31	Make decisions on strategy switch
Before 2.1	Collect the dataset or design energy function as needed
Before 3.1	Complete the training of World Model (Final version)
Before 4.1	Complete the training of robot control policy
Before Final Presentation	Do investigations for complex movements and environments

Table 2. Tentative Schedule of the Project

5. Conclusion

In conclusion, this paper has presented a comprehensive exploration of integrating fluid dynamics with character animation in a computer graphics context, employing advanced machine learning techniques. Our journey began with the development of a fluid solver, optimized for handling solid-fluid interactions. We then shifted our focus towards training a world model, adapting to challenges such as storage limitations by considering a transition from supervised to self-supervised learning.

Looking ahead, our immediate focus is on a thorough review of literature in differentiable fluid simulations and self-supervised learning, which will guide our decision on whether a strategic shift is necessary. This step is crucial for the continued development and training of both the world model and the control policy. The tentative schedule for these phases, as detailed in Table 2, outlines our planned progression. Our ultimate goal remains to achieve a harmonious

blend of physical realism and computational efficiency in character animation within fluid environments. This endeavor not only advances the field of computer graphics but also opens new avenues for realistic simulations in various applications, from entertainment to scientific visualization.

This paper, therefore, contributes not just to the body of knowledge in fluid simulation and character animation, but also exemplifies the innovative application of machine learning in solving complex, real-world inspired computational challenges.

References

Ando, R., Thuerey, N., & Wojtan, C. (2015). A stream function solver for liquid simulations. *ACM Transactions on Graphics*, *34*(4), 1–9. https://doi.org/10.1145/2766935

Angelidis, A., & Neyret, F. (2005). Simulation of smoke based on vortex filament primitives. *Proceedings of the 2005 ACM SIGGRAPH*. https://doi.org/10.1145/1073368.1073380

Das, K., & Behera, R. N. (2017). A survey on Machine Learning: Concept,Algorithms and Applications. *International Journal of Innovative Research in Computer and Communication Engineering*, *5*(2). https://www.rroij.com/open-access/a-survey-on-machine-learning-conceptalgorithms-and-applications-.pdf

Elcott, S., Tong, Y., Kanso, E., Schröder, P., & Desbrun, M. (2007). Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics*, *26*(1), 4. https://doi.org/10.1145/1189762.1189766

Gamito, P., Lopes, P. F., & Gomes, M. R. (1995). Two-dimensional simulation of gaseous phenomena using vortex particles. In *Eurographics (Wien)* (pp. 3–15). https://doi.org/10.1007/978-3-7091-9435-5_1

Ha, D., & Schmidhuber, J. (2018). World models. *arXiv (Cornell University)*. https://doi.org/10.5281/zenodo.1207631

Li, Z., Xu, Q., Ye, X., Ren, B., & Liu, L. (2023). DIFFFR: Differentiable SPH-Based Fluid-Rigid coupling for rigid body control. *ACM Transactions on Graphics*, *42*(6), 1–17. https://doi.org/10.1145/3618318

Peng, X. B., Abbeel, P., Levine, S., & Van De Panne, M. (2018). DeepMimic. *ACM Transactions on Graphics*, *37*(4), 1–14. https://doi.org/10.1145/3197517.3201311

Pfaff, T., Thuerey, N., & Groß, M. (2012). Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics*, *31*(4), 1–8. https://doi.org/10.1145/2185520.2185608

Stam, J. (1999). Stable fluids. *Proceedings of SIGGRAPH '99*. https://doi.org/10.1145/311535.311548

Wu, P., Escontrela, A., Hafner, D., Goldberg, K., & Abbeel, P. (2022). DayDreamer: World Models for Physical Robot Learning. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2206.14176

Zhou, Z., Zhang, Y., & Li, Y. (2023). Model Predictive Control Design of a 3-DOF Robot Arm Based on Recognition of Spatial Coordinates. *2023 9th International Conference on Mechatronics and Robotics Engineering (ICMRE)*. https://doi.org/10.1109/icmre56789.2023.10106581