Personal Project Experience and Interests

Zhiwei (Zavier) Zhao

Computer Science and Engineering, UCSD, La Jolla, California, 92037

zhz214@ucsd.edu

Welcome to visit my personal website:

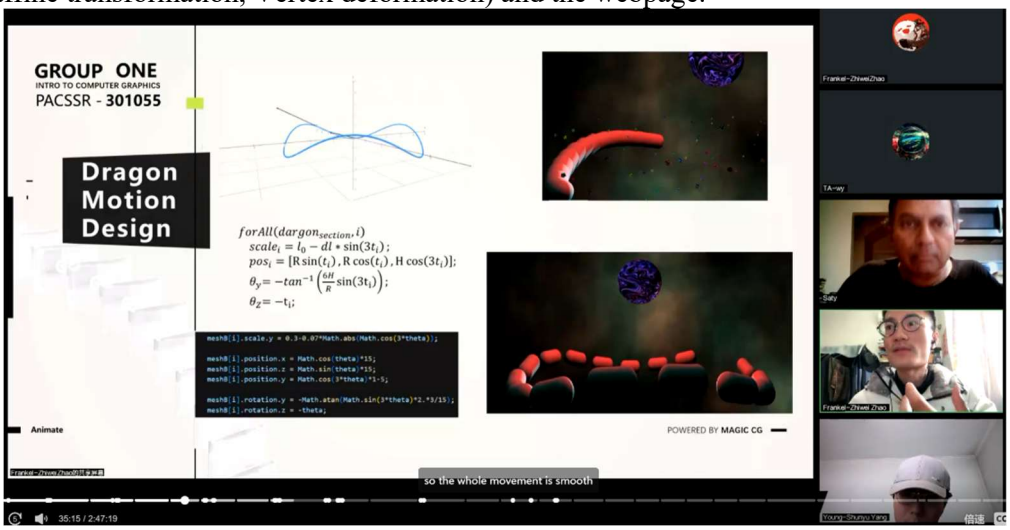
furkathertaha.github.io

and GitHub homepage:

github.com/Furkathertaha

1. Web3D Online Practice

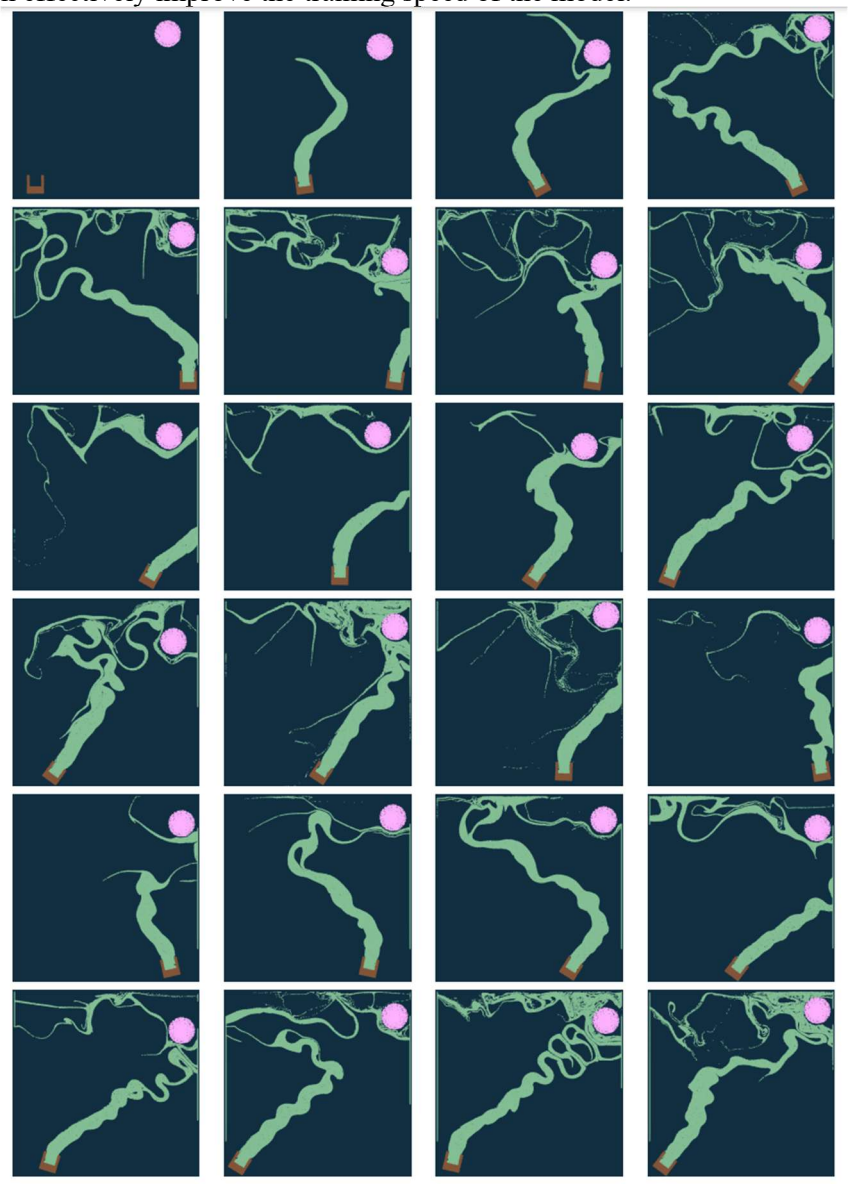
This project was remotely guided by Professor Saty Raghavachary from the Computer Science Department at the University of Southern California. It covered the history of the animation industry's development, the industrial applications of various graphics technologies, and the current development of the integration of the 3D industry with AI. The project mimicked the complete production process of Pixar's Animation Feature, and a team was formed to complete the construction of a 3D webpage. I led an international group (including students from UC Irvine and the University of Glasgow), coordinated the output of STL (no UV) models, GLSL-Shader production, and the completion of three.JS animations (model affine transformation, Vertex deformation) and the webpage.



Details of the Animation webpage: https://furkathertaha.github.io/assets/threeJS/sf/long1.html
3D webpage source code: https://github.com/Furkath/threeJS-GLSL_STL-3D-webpage-design

2. Deep Reinforcement Learning in CG

According to the theme of the Machine Learning Reproduce Challenge (MLRC) 2023, we reproduced, expanded, and optimized the deep learning application in graphics in recent years' papers published in ACM TOG. As the group leader, together with my classmates, I chose the work of controlling the position and posture of a small ball using deep reinforcement learning (Soft Actor-Critic Algorithm) in fluid-rigid body simulation as the topic and plan to submit it to the MLRC2024 conference. Currently, the optimization of 2D animation has been completed: Pytorch is used to build a small-scale AlexNet as the AutoEncoder of fluid spatial information, and the definition of Reward is improved to prevent convergence difficulties caused by over-sparsity. Taichi was fully utilized to write the interactive animation of MLS-MPM fluid and rigid body, optimize the physical process of their collision, and utilization of GPU parallel particle memory. The latent space information after simulation and structural optimization can effectively improve the training speed of the model.

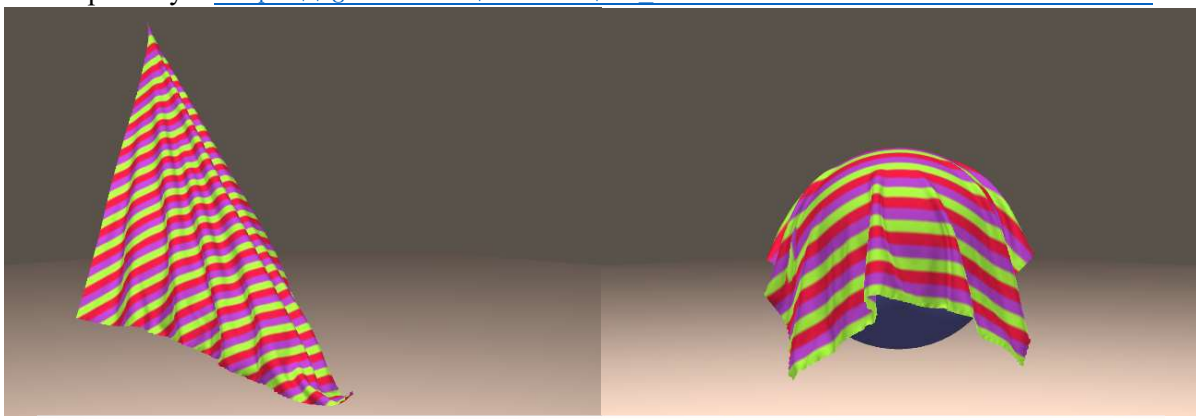


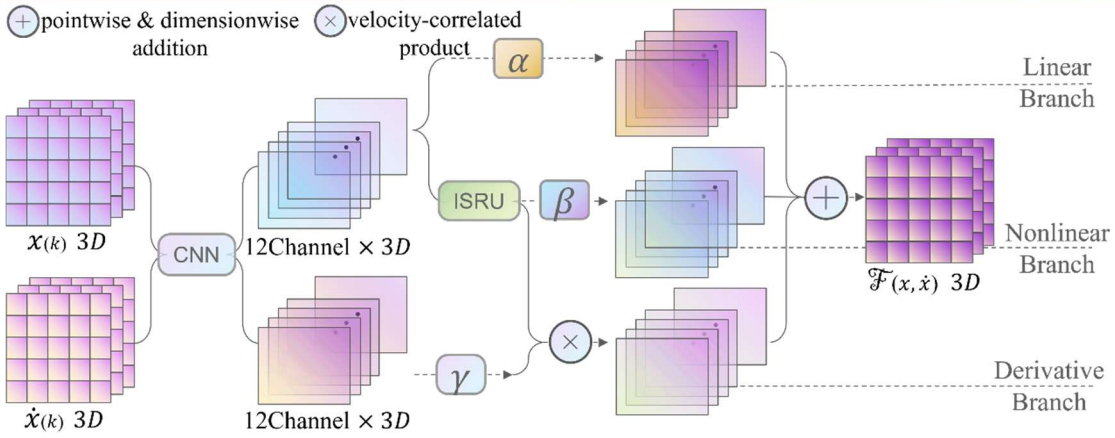
Code repository：https://github.com/Furkath/DRL_controlled_fluid-rigid_simulation
Research Report：
https://github.com/Furkath/DRL_controlled_fluid-rigid_simulation/blob/master/Document.pdf

3. Fabric Simulation by ML Model with Embedded Physics

My research topic is AI-empowered PDE system solving algorithms. My group has long been engaged in research on the application of machine learning in the simulation of thermal fluids and viscoelastic objects. Inspired by the work of applied mathematicians in Physics-informed-neural-networks, PDE-Nets, and Physics-recurrent-convolutional-neural-networks in recent years, all general PDE systems (including physics-based animation simulations) can adopt non-data-driven physical residuals as the optimization target for the learning process. Related work in the field of computer graphics can also be seen in TOG's Neural Cloth, etc. Nevertheless, typical physical problems have very obvious characteristics in spatial interaction and time progression, which can optimize network structure to avoid the need for general models to increase depth to fit data, thereby causing difficulties in training, convergence, and reasoning. Therefore, I try to use a network structure that includes CNN (analyzing spatial interaction) and embedded physical features (simulating the nonlinear relationship of various forces) to learn the stepping of spring-mass systems in explicit integration. The code is written in Pytorch and Taichi, and preliminary results have been demonstrated, with NN and PBS effects being similar. NN parameters usually have higher degrees of freedom than real physical parameters, but there is no overfitting in this Priori Test. This article was published in a CS conference, and further optimization of CNN's parameter quantity and structural features will be conducted later. Attempts will be made to add learning of cloth self-collision and implicit integration to this DL framework.

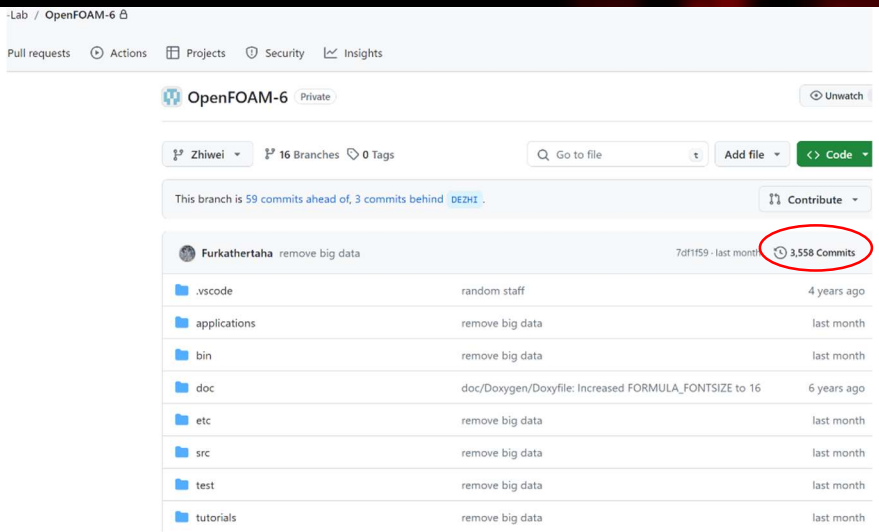
Code repository：https://github.com/Furkath/DL_Framework-for-PBS-Cloth-Simulation





4. Multi-scale Simulation

Choosing the simulation of flames as a representative of multi-scale PDE systems, typical gas-phase fluid simulations generally use mesh division combined with the finite volume method. Therefore, 3D CFD large eddy simulation can serve as a foundation for developing a series of acceleration algorithms that sacrifice the accuracy of small-scale features. These include topology, low-dimensional manifolds, mechanism simplification, Hash dynamic tables, digital twins, and AI data-driven methods. Numerous studies on Fourier Neural Operators (FNO) have also been conducted, with details available in the subsequent project team achievements. I am also responsible for the construction and maintenance of the project team's server, familiar with UNIX command shell principles, as well as system configuration files and various compilation tools. High-precision physical simulations can serve as a baseline for multiple tasks, and the TVD format can be referenced at:
https://github.com/Furkath/TVD_Fluid_Simulator



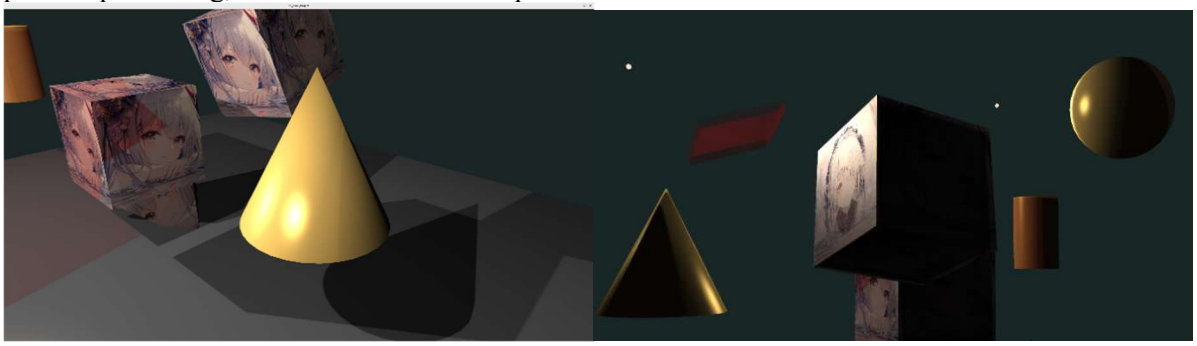
5. Self-docking of Unmanned Surface Vessels

The extension of the capstone design, involved RGBD camera sampling, 3D point cloud reconstruction, Minimum-Snap algorithm for shortest path planning, WIFI module ESP8266 point-to-point communication, L298N motor control, and Arduino microcontroller development. I was responsible for local wireless communication and microcontroller control development, assisting in the denoising algorithm for boat pose recognition, and building a Linux central C++ program based on timestamp stream processing for various tasks.

Code repository：https://github.com/Furkath/autoDock

6. OpenGL Illumination Model Rendering Practice

In this practice, I used the most basic C++ OpenGL API to complete the rendering of desired scenes. It read in Obj files and was coded with the BRDF lighting model in GLSL. I implemented dual-layer mapping, multiple light sources, Shadow Map shadows, translucency, supersampling, and texture optimization. Thoughts: The most basic OpenGL has typical C language characteristics, and the buffer process is quite cumbersome; GLSL ray tracing can try to pass the model as a texture to the GPU for parallel processing, which remains to be implemented.



Code repository：https://github.com/Furkath/GL-Rendering

7. McGill University Entrepreneurship Program

Participated in McGill's exchange program in business, conducted a simulated operation of a Canadian Chinese food delivery service in groups, as well as researched the business model of hockey sticks in the Canadian market in the last century.

8. Personal Interests

I have worked with some digital creation tools, including Blender, SD-WebUI, SD-ComfyUI, MMD, SolidWorks, Unity, MD, ShaderFrog, RenderMan (Aqsis), etc. Among them, I have experience in reviewing the source code of Blender (to copy some mature open-source software graphics algorithms), using geometry nodes, and calling Python API scripts for procedural modeling. I have experience deploying SD-WebUI on ROCm's AMD cards, adding new types of samplers to the source code, and rolling back and resolving version conflicts to repeat the model training process. Particularly with ComfyUI, I have practiced the workflow of video face-swapping.