

Unified Framework for Real-Time Fluid Simulation in Virtual Rotator Cuff Arthroscopic Skill Trainer (ViRCAST)

Aditya Dendukuri
Computer Science Department
University of California Santa Barbara
Santa Barbara, California
aditya_dendukuri@ucsb.edu

Mustafa Tunc
Google
Mountain View, California
mustafatunc@google.com

Doga Demirel*
Computer Science Department
Florida Polytechnic University
Lakeland, Florida
ddemirel@floridapoly.edu

Sinan Kockara
Computer Science
Rice University
Houston, Texas
skockara@rice.edu

Tansel Halic
Intuitive Surgical
Peachtree Corners, Georgia
tansel.halic@intusurg.com

Abstract— This study introduces a unified real-time fluid simulation framework for virtual surgical simulation. In the context of this work, we aim to simulate thermo- and fluid dynamics in our Virtual Rotator Cuff Arthroscopic Skill Trainer (ViRCAST). Fluid Dynamics directly dictates the flow of the irrigation solutions, and thermodynamics dictates the electrocautery procedure in arthroscopic procedures. An efficient and generalized numerical approach is essential for robust modeling of liquid and heat flow, as the medium of fluid flow changes continuously as the simulation progresses. In addition, these physics models should work in harmony while interacting. We used a numerical computational technique called Smoothed Particle Hydrodynamics (SPH) to model heat transfer and liquid simulation under a single framework. Our formulation intends to virtually simulate the complete fluid motion of the irrigation solution and the mechanism of conduction and convection of heat flow within the shoulder cavity in real time. In addition to the core function of the systems, our study also uses various methods to simulate the interaction between the physical systems and models. In this study, we present the preliminary results of our unified SPH approach to simulate the electrocautery process and liquid flow in the context of arthroscopic rotator cuff surgery in ViRCAST. We investigated the solver's performance by recording the average frames per second (FPS) for the simulation, which is executed for a minute. The average FPS was 202 Hz. We parallelized our framework at the loop level using OpenMP, which resulted in an increase of 46% in FPS.

Keywords— *Computational Fluid Mechanics, Numerical Methods, Smoothed Particle Hydrodynamics, Surgical Simulations*

I. INTRODUCTION

Arthroscopy is a surgical procedure to diagnose and treat injured joints. Arthroscopic Rotator Cuff (ARC) is a minimally invasive procedure that only treats the injured rotator cuff muscles that connect the upper arm to the shoulder blade. Current training methods for ARC are expensive and inefficient. Hence, our team is developing a virtual ARC simulation to counter these challenges. The simulation mainly

focuses on maintaining realism through various factors like physically based rendering (PBR) and physics solvers. This study focuses on the physics aspect of the Virtual Rotator Cuff Arthroscopic Skill Trainer (ViRCAST) [1] to satisfy the realism constraint of the simulation. The study intends to develop a unified model to simulate both fluid and heat flow throughout the simulation in real-time for irrigation and electrocautery procedures, respectively. Due to the complexity of the components involved in the simulation, a numerical approach in designing our solver is necessitated.

Our framework is designed and developed for real-time simulation of the fluid flow in the context of virtual surgical simulation. Therefore, we categorized the existing literature in real-time physics simulation and fluid flow techniques used in the surgery simulations. The Navier Stokes Equations for fluid flow were derived by Claude Navier and George Stokes in 1822 and 1845, respectively [2]. Numerical methods are primarily used to simulate fluid in computers due to their simplistic applications in real-time simulations. Harlow and Welch introduced these methods in the late 1950s and 1960s [3], [4].

The application of the Navier Stokes was only limited to two-dimensional fluids for simulating Planet Jupiter's atmosphere [5]. Foster and Metaxas investigated Eulerian-based fluid simulations in [6], [7] based on the previous work of Kass and Miller [8], which linearized the Navier Stokes Equations and developed a stable solver. Even though computer simulations rely on discrete and approximate results, removing the linearity eliminates most fluid behavior [9]. Foster and Metaxas's works [6], [7] on computational fluids are the most accurate methods in Eulerian fluid simulation. Their primary approach was to apply the finite element method to the Navier-Stokes Equation with the explicit time stepping. Stam in [10] developed a stable density solver using kinetic turbulent wind fields and a stable simulation in [11]. A tree-based method called octree used in water and smoke simulation was introduced in [12]. There were other data structures introduced, like the dynamic tubular grid introduced in [13], the run-length encoded (RLE) sparse level set, which is highly scalable with low memory usage [14], and a hierarchical run-length encoded level set data structure in [15]. Even though there are many methods to optimize scalability and memory usage, the constraints imposed by a mesh still hold, which is a considerable disadvantage in our virtual ARC surgery simulation case.

* Corresponding author

This study was made possible by grants from the National Institutes of Health (NIH) / National Institute of Biomedical Imaging and Bioengineering (NIBIB) 5R01EB005807-11, 3R01EB005807-09A1S1, 1R01EB033674-01, and 5R01EB025241-04. This project was also supported by the Arkansas INBRE program, supported by a grant from the National Institute of General Medical Sciences (NIGMS), P20 GM103429 from the NIH.

Lattice Boltzmann Method (LBE) is a newer approach in computational fluid dynamics based on microscopic models or mesoscopic equations [16]. Lattice Boltzmann's method follows the principle of Ludwig Boltzmann's kinetic theory of gases, which states that gases and fluids can be modeled as molecules colliding with each other. Ladd designed the first framework to numerically simulate fluid using the Lattice Boltzmann method [17]. LBE was also applied to two fluid cases in various studies, for example, work done by Wang and Wang [18].

Since our case deals with complicated and dynamic boundary conditions in the case of moving tear and scar tissues, particle-based Lagrangian methods are chosen. Lagrangian approaches are mainly motivated by the idea behind Lagrangian mechanics. Instead of using a static reference like a mesh to represent a system, it describes the properties of the system independent of the mesh. Reeves conducted the first study to have a particle-based approach to simulate fuzzy objects in [19]. There were more particle-based approaches, such as [20], which mainly focused on simulating ocean waves. However, SPH is the best approach for real-time arthroscopic surgery simulation due to its position-based framework.

Smoothed Particle Hydrodynamics (SPH) was initially introduced in [21], [22] for astrophysical models and was gradually applied to other systems. Viscous fluids were first implemented using SPH in [23]. SPH was first applied to free surface flows in [24]. In 1995, SPH was applied to gaseous fluids and fire [25]. The following year, SPH was applied to highly deformable models [26]. In 1999, the advection term was added to the current fluid SPH model [27]. The development of SPH for fluid flow for real-time applications was first achieved in [2], [28], which also incorporated the interface tension force in the fluid. However, the interface tension force is only applicable when there is more than one fluid in the system. In addition, the former did not involve efficient neighbor detection, where they explicitly step through all the particles in a defined boundary box in each simulation step. The need for a more efficient method for neighbor detection was employed in [28] using spatial hashing developed by [29]. A spatial map was used to keep track of particles nearby. However, there is a frequent problem with this method in which close proximity of a specific particle is not being hashed, which makes the simulation performance sensitive to the selected cell size and fluid volume within the cell. Solenthaler et al. [30] presented the prediction-correction scheme to determine the particle pressures, eliminating density fluctuations.

The performance bottleneck of real-time simulation of fluids is mitigated with GPU computation. GPU was used for the first time to accelerate SPH for cloud simulation in [31]. The GPU can increase the number of particles in real-time [32]. Flex by NVIDIA is a popular particle-based solver for real-time fluid simulations. Flex is accelerated by GPU threads. However, the GPU threading is only compatible in systems with NVIDIA GPUs with CUDA support. Eulerian Fluids were also developed on the GPU in [33]. Interactive SPH Simulations of GPU were implemented in [34]. Flex by NVIDIA is a popular particle-based solver for real-time fluid simulations. Flex is accelerated by GPU threads. However, the GPU threading is only compatible in systems with NVIDIA GPUs with CUDA support. Eulerian Fluids were also developed on the GPU in [33]. Interactive SPH Simulations of

GPU were implemented in [34]. However, the incorporation of the multi-physics in the same simulation frame where the models require frequent data exchanges remains unexplored. The scalability of the GPU-based approaches is also limited in performance-dependent scalability (e.g., memory, threads, data structure limits) primarily specific to a GPU card of a particular vendor.

The need for simulating fluid motion in surgical simulations is mostly for simulating blood flow within the vessels and the tissues. The bridge between these two fields was initiated when Oppenheimer et al. [35] developed a comprehensive image-based representation of blood flow in endourologic surgery simulation, enabling simulation developers to evaluate bleeding behavior. Müller et al. [28] provided an SPH-based blood flow model, which provided a realistic implementation of bleeding. However, this model does not incorporate heat flow or irrigation solutions. Other works are based on fluid dynamics in surgical simulations for specific cases like the initial stages of acute cardiac compensation. Since no fluid dynamics is defined for irrigation methods in arthroscopy procedures [36], numerical solutions are the best approach to simulate a unified solver for all physics-based units in the simulation. Therefore, our work is adopting and extending the method used by Müller et al. [28] for a novel solver to cover heat and fluid (Irrigation and blood) flow in one module.

The most common numerical techniques to solve the heat equation are finite element methods (FEM) or finite difference methods (FDM) [37]. More popular methods exist where a Eulerian grid is used for mesh-free simulations. Some examples of these methods are the diffusion approximation method [38], the meshless element free Galerkin (EFG) method [39], [40], and the moving least square method [41]. Since our goal is to form a unified framework for representing the continuum mechanics of fluid and heat flow, the Lagrangian approach is more suitable, especially for the model undergoing topology changes, which is always a case encountered in ARC surgeries.

SPH was applied to thermodynamics for large-scale time stepping in [37]. This method was utilized to study crystal growth since mesh-free methods work much better with continuously developing crystal growth surfaces [37]. Other improvements were made to the heat flow model to increase the accuracy of solutions, like using the equation of state for natural convection [42]. SPH was applied to heat diffusion through porous media in [43].

In this work, we use the SPH-based method to simulate real-time fluid and heat flow. Numerical simulation of these physical phenomena requires a robust and flexible computation framework as the simulation should incorporate constantly changing boundary conditions and complex geometry. Moreover, simulation should maintain seamless and efficient interactions between electrocautery - bone and tissue, irrigation solution - bone and tissue, and heated tissue and irrigation solution interactions. Therefore, we propose a unified framework to handle this intermingled level of interactions in the ViRCAST framework.

II. BACKGROUND

A. Background Formulation

1) Classical Fluid Dynamics

An incompressible fluid can be represented accurately using its velocity, pressure, and density fields [2]. These three vector fields can specify the nature and movement of the fluid at any position. The standard fluid flow model is expressed by two partial differential equations collectively called the Navier-Stokes equations (Equations 1 and 2) [2]. These two equations are derived from the continuity equations of the conservation of mass and energy. The first equation is the continuum mechanics version of the law of conservation of mass.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \quad (1)$$

where ρ denotes the density, and v denotes the velocity. The above equation states that the rate of change of the density of the fluid ($\frac{\partial \rho}{\partial t}$) is represented by the divergence of the density field. Since we are dealing with an incompressible fluid, the mass density does not change in the fluid over a period of time. Therefore, $\frac{\partial \rho}{\partial t}$ term vanishes leaving us with $\nabla \cdot (\rho \vec{v}) = 0$. This expression mandates a divergence-free velocity, which significantly simplifies the numerical calculations. The second equation describes the motion and forces inside the incompressible fluid body.

$$\rho \left(\frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} \right) = -\nabla p + \mu \nabla^2 \vec{v} + \rho g \quad (2)$$

where $\mu \nabla^2 v$ is the Laplacian of the velocity vector field and $\mu > 0$. After we employ the zero gradients of the velocity from (2), we get a simplified version as follows:

$$\rho \left(\frac{\partial \vec{v}}{\partial t} \right) = -\nabla p + \mu \nabla^2 \vec{v} + \rho g \quad (3)$$

This equation can be transformed into a numerical model of incompressible fluid motion. The term ρg represents the gravitational field and is the only external force in the Navier Stokes formulation.

2) Thermal Conduction

The governing heat conduction equation over time is given by the diffusion model (4):

$$\rho c_p \frac{\partial T}{\partial t} = \alpha * (\nabla^2 T)_R \quad (4)$$

where α is the thermal diffusivity, c_p is the specific heat capacity, and R is the coordinate space. Since the volume in the shoulder cavity is irrigated with continuous fluid flow, the heat exchange occurs with convection in the heat equation. Free convection is given by the following equation (5):

$$\frac{dT_{convection}}{dt} = -\nabla \cdot (\vec{v}T) \quad (5)$$

where \vec{v} is the velocity of the fluid passing through the surface. Equation (6) can be combined with (5) to generate a combined model for heat transfer in the simulation, which can be converted to the numerical model.

3) Thermal Convection

We employed Newton's law of cooling to model convective heat transfer (6).

$$\frac{dT_{convection}}{dt} = \beta A (T_{surface_{cell}} - T_{contact_{fluid}}) \quad (6)$$

where β is the convective heat transfer coefficient, and A is the surface area of the contact. Since we are calculating the convective heat transfer cell by cell, we approximate the

contact area to be the sphere's circumference with the radius of the smoothing distance. The equation results in:

$$\frac{dT_{convection}}{dt} = \pi h \beta (T_{surface_{cell}} - T_{contact_{fluid}}) \quad (7)$$

where h is the smoothing distance of the particle.

4) Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) is a mesh-free interpolation method in which the system is divided into a discrete set of particles where the solution of the governing equation is computed. SPH finds applications in various fields such as highly deformable body simulations and computational fluid dynamics. At its core, this approach involves partitioning the fluid (or medium) into distinct elements known as particles. Equation 8 describes the interpolation where h is the length of the smoothing function W , r is the descriptor of the medium entity, and r' is the adjacent entities in the range of h . Many kernel functions are defined in the literature [44], [45]. This method discretizes the fluid body into points and uses smoothing kernels to calculate the dynamics of the fluid at any point in the scene. One of the kernel functions, described by Monaghan's cubic spline [44] as shown in Equation 8, calculates the temperature, (specifically in this case, denoted as A), at position r . This calculation depends on the temperatures of all particles within a radial distance h .

$$A_t(r) = \int A(r') W\{|r - r'|, h\} dr' \quad (8)$$

In Equation 8, a particle's impact on a physical property is determined by its proximity to the particle of interest and its density. Commonly used kernel functions include cubic spline and Gaussian functions. The cubic spline function becomes precisely zero for particles situated at a distance equal to two times the smoothing length, $2h$. This reduces computational costs by excluding particles with minimal contributions to the interpolation. To enhance the representation of the SPH kernel in physics, we introduce another particle, denoted as j , and associate it with a fixed volume ΔV_j in a lump shape. This approach defines the computational domain using a finite number of particles. Regarding the mass and density of the particle, the lump volume can be expressed as the ratio of mass to density, m_j/ρ_j [45]. Then, the governing equation for SPH, which is used to approximate a function $A(r)$, is given by [21]:

$$A(r) = \sum_{j=0}^{num_neighbors} m_j \cdot \frac{A_j}{\rho_j} W(|r - r_j|, h) \quad (9)$$

where W is the kernel function, r is spatial location, A is any quantity at r , m_j is the mass of particle j , A_j is the value of the quantity A for particle j , ρ_j is the density of particle j .

The SPH simplifies calculating the Gradient and Laplacian operators. The operators are applied over the smoothing kernel functions. Therefore, the gradient and Laplacian of the SPH approximations are given below in equations (10) and (11) below, where the summation over j covers all particles. Since m is a scalar, the derivative can easily find the gradient of a quantity ∇ .

$$Gradient: A(r) = \sum_j m_j \cdot \frac{A_j}{\rho_j} \nabla W(r - r_j, h) \quad (10)$$

$$Laplacian: A(r) = \sum_j m_j \cdot \frac{A_j}{\rho_j} \nabla^2 W(r - r_j, h) \quad (11)$$

The smoothing kernels are applied to every force term in the Navier-Stokes formulation (2). Calculating the pressure force by using equation (8) seems reasonable. However, we should also consider that the pressure force must be symmetrical since the neighboring particles have varying pressure values. The SPH formulation symmetrizes the gradient term by sending the density term inside the operators and rewriting the formula [11]. The pressure term in equation (2) becomes:

$$-\nabla_i^2 p = -\rho_i \sum_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) m_j \nabla^2 W(r - r_j, h) \quad (12)$$

The viscosity term also needs to be symmetrized since the neighboring particles may have different velocities. This issue is solved by setting the viscosity function as the velocity difference [3]. After plugging in the velocity difference in (3), we get:

$$\nabla_i v = \sum_j (v_j - v_i) \frac{m_j}{\rho_j} \nabla W(r - r_j, h) \quad (13)$$

Surface tension force is calculated by introducing a new scalar quantity called to the SPH fluid. These scalar values are set to 1 for every particle and zero everywhere else in the scene. This allows us to form a border between the surface and particle domain. Surface Tension force is not a part of the Navier Stokes formulation but can be used as a boundary condition since surface tension only acts on the surface of the fluid. A smoothed version of the color function is calculated using SPH, as shown in (14). The volume fraction function is commonly denoted as the color function, represented by c . The normal vector, denoted as n , is calculated as the normalized gradient of this color function: $n = \nabla c / |\nabla c|$.

$$c_i = \frac{m_j}{\rho_j} \nabla W(r - r_j, h) \quad (14)$$

Since the scalar function is explicitly dependent on the position of the constituent particle relative to its neighbors, the function's gradient would give the inward surface normal of the fluid at that particular point. Next, we calculate the Gaussian Curvature of the fluid, κ , where κ is the average curvature obtained by taking the divergence of the normal vector ($\kappa = -\nabla \cdot n$). The term is set to negative to get a positive value for the curvature, as shown below, where n is the surface normal vector and results are normalized with the norm of n :

$$\kappa = -\frac{\nabla n}{|n|} = -\frac{\nabla^2 c}{|n|} \quad (15)$$

After we compute the surface curvature, we can calculate the surface tension force. To calculate the surface tension force, we need to calculate the surface traction (force per unit area) of the specific particle. The formula for the surface traction t is found in [2], which is defined as:

$$t = \sigma \kappa \frac{n}{|n|} \quad (16)$$

where σ is the surface tension coefficient. The surface tension force is calculated by multiplying the surface traction with a normalized scalar field $|n|$ [2].

$$f^{surface\ tension}_i = |n_i| * t_i = -\sigma \nabla^2 c_i \frac{n_i}{|n_i|} \quad (17)$$

The SPH formulation and smoothing kernels can also simulate the heat model. Each SPH heat particle represents the

temperature within its smoothing distance. To smooth the temperature field for a particular particle, we apply the Laplacian for the $(\nabla^2 T)$ term in (4). After applying the formulation and solving for the temperature value explicitly for the next time step, we get the following:

$$T_i^{m+1} = T_i^m + 2\alpha dt \sum_{j=1}^N \frac{m_j T_i^m - T_j^m}{\rho_j x_{ij}} \frac{\partial W(x, h)}{\partial x_{ij}} \quad (18)$$

5) Smoothing Kernels

Different kernel functions are adopted for the specific terms since the given terms in the partial differential equations have unique constraints. The most standard kernel used would be the Poly6, a 6th-degree polynomial kernel introduced in [2]. This kernel is only invoked if $0 \leq r \leq h$. The gradient of the Laplacian of the poly6 kernel is used to calculate the surface normal.

$$W_{poly6}(r, h) = \frac{315}{64\pi h^9} (h^2 - \|r\|^2)^3 \quad (19)$$

with the gradient:

$$\nabla^2 W_{poly6}(r, h) = \frac{315}{64\pi h^9} (h^2 - \|r\|^2)(3h^2 - 7\|r\|^2) \quad (20)$$

The spiky kernel is used for the pressure calculations since the pressure term cannot go below zero since it would speed up particles in some instances. Since we are dealing with the gradient pressure term, we will use the gradient of the spiky kernel function [2] as follows:

$$\nabla W_{spiky}(r, h) = \frac{-45}{\pi h^6} \frac{\vec{r}}{\|r\|} (h - \|r\|)^2 \quad (21)$$

The viscosity kernel is used for calculating the viscosity force.

$$\nabla^2 W_{viscosity}(r, h) = \frac{45}{\pi h^6} (h - \|r\|) \quad (22)$$

These mathematical formulations make making an all-rounded physics solver very convenient due to the robust particle-based feature.

B. Real-Time Simulation in ViRCAS

1) Particle Object

We have a single definition for every physics object in the simulation. This object will store its positional and physical information. We differentiate between fluid, conduction, and convection by using particle identifiers, which can be tweaked. The particle's attributes are listed in Table 1.

TABLE I. Particle Attributes

1. Position	} Basic Attributes
2. Velocity	
3. Acceleration	
4. Density	
5. Mass	
6. Status	
7. Fluid id	} Particle identifiers
8. Conduction id	
9. Convection id	
10. Pressure Force	} Fluid Flow Terms
11. Viscosity Force	
12. Surface Tension Force	
13. Gravitational Force	} Heat Flow Terms
14. Conductive Temperature	
15. Convective Temperature	

2) Solver Algorithm

The solver goes through two main stages: initialization and simulation. The initialization stage involves setting up the solver and processing the scene. This scene also consists of creating the heat and fluid particles and placing them in the solver. The steps of the solver algorithm are given in Figure 1.

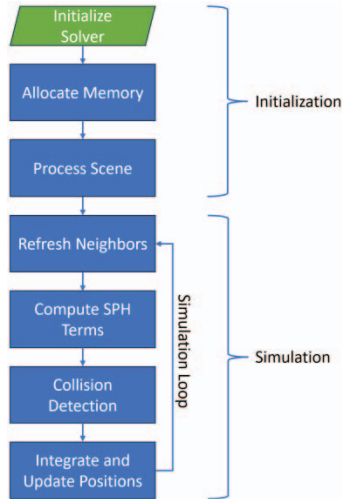


Fig. 1. Algorithm for the Solver.

3) Initialize Solver and Allocate Memory

This stage is invoked when the simulation starts. This phase takes in the maximum number of particles, smoothing distance, and maximum neighbors as input and initializes the solver with these properties embedded. The next step is to allocate memory for all the data structures and initialize universal constants like the gravitational and gas constants. This section also involves constructing the hash map based on the maximum particles and neighbors.

4) Process Scene

This stage is responsible for setting up the particles for conductive heat transfer. Since we aim for a unified framework, our solver should be able to differentiate between the heat and fluid particles. Since we are only concerned with heat transfer in the rotator cuff area in the ViRCAST framework, we turn the models into discrete points for our particle-based approach using ray casting. After creating the particle model, we update our hash map with our new heat particles. Therefore, this stage is responsible for producing heat particles. It also aids in the convection calculation process since we can invoke the convection term whenever a fluid particle gets in the same hash cell as a heat particle. This is also the final process of the initialization phase as we link the muscle and tissue in the models to our heat model.

5) Refresh Neighbors

This is the first stage in the simulation phase and will be invoked at every step as the simulation progresses. Before calculating the physical quantities, we need to update the neighbors of every particle since a fluid body is highly dynamic with no positional constraints. We incorporated a spatial hash map in the algorithm since it has a neighbor retrieve complexity of $O(1)$ [2]. The idea is to dynamically store the particles' indexes in a spatial hash map so that particles close to each other will be hashed into the same cell. Therefore, all the particles in the same cell would be added to the neighbor list of the specific particle. We used the

proposed hash function given in [29]. The main task of this process is to rebuild the hash map according to the new positions of all the particles. This process also updates the particles' convection ID, which will be used as a reference to invoke convection in the next stage. The algorithm for neighbor detection is illustrated in Figure 2.

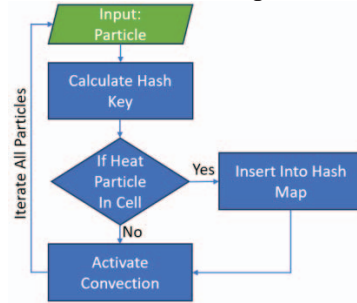


Fig. 2. Algorithm for Neighbor Detection.

6) Compute SPH Terms

This stage is the heart of the solver since it calculates all the physical properties of the fluid and the heat. This stage is set to handle fluid conduction and convection terms in one sub-process, as given in Figure 3.

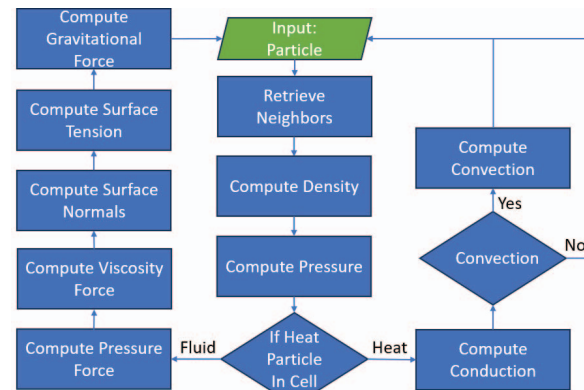


Fig. 3. Algorithm for SPH Calculations.

7) Collision Detection and Resolution

Collision detection and resolution are performed using the spatial hash map since it stores the mesh particles. Therefore, if a fluid particle gets placed in the same hash cell as a mesh particle, we check if the fluid particle is in a defined radius of a mesh cell or not. If the fluid particle is in the mesh cell, it will be deflected according to the surface normal of the mesh point. The algorithm is given in Figure 4.

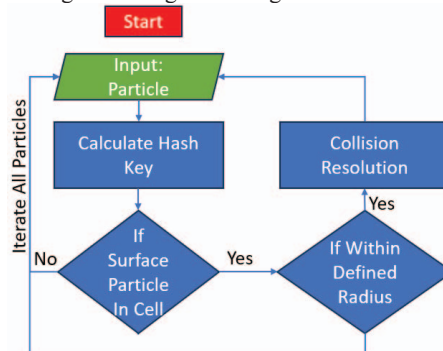


Fig. 4. Algorithm for Collision Detection.

8) Integrate and Update Positions

An implicit approach named Leapfrog Integration was used to update the particles' positions and velocity. The formulations are given below, and the delta time value is set to 0.02s:

$$v_{t+1} = v_t + \frac{(a_t + a_{t+1})}{2} \Delta t \quad (23)$$

$$r_{t+1} = r_t + v_{t+1} \Delta t \quad (24)$$

where t is time, v is velocity, a is acceleration, and r is position.

9) Interaction of Physics with Other Systems in ViRCAS

The fluid and heat simulation interacts with the rest of the systems in the simulation, like scar tissue and bone. Therefore, it is crucial to integrate these systems into the simulations as a combined unit instead of a background process layering the main simulation. The fluid should influence the movement of the scar tissue around the shoulder cavity, and the heat should affect the various tissues and muscles around that region. Therefore, the modules simulating heat and fluid will output their respective field maps to the main simulator. In that way, the rest of the systems in the main simulator can access the information required from the heat map at every step and integrate those values into the simulation.

The field simulates the irrigation procedures. The heat map, however, will influence the soft tissues' state. The fluid solver will generate the pressure field map and dynamic models, such as scar tissue, which will move to simulate electrocautery procedures. We implemented these interactions using the same spatial hash functions used in the neighbor search. The fluid will transpose the pressure field into a spatial hash map and output it to the simulator. Therefore, it will access the force field from the hash map whenever the scar tissue is rendered and move according to the field. Similarly, the tissue models will access the hash map from the heat values using the vertex positions from the mesh and will change the state accordingly. The entire system is illustrated in Figure 5, and a close-up image of the entire system is given in Figure 6. In Figure 5, the arrows represent the fluid direction.

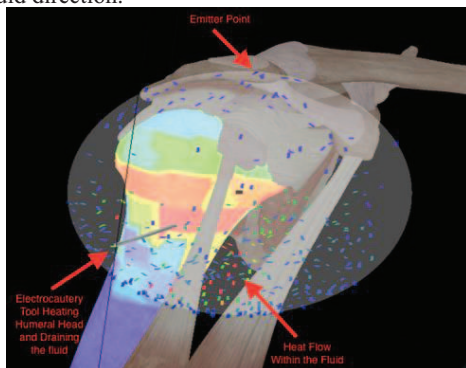


Fig. 5. Illustration of the entire surgical system.

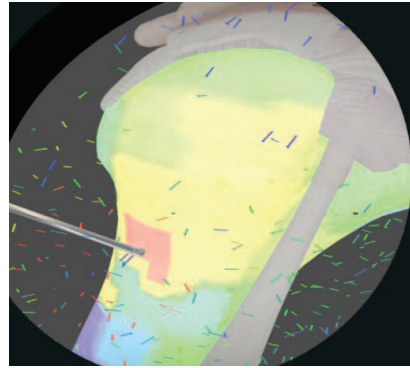


Fig. 6. A closer view of the surgical system for better depiction.

10) Fluid Suction

The fluid suction was simulated by adding a point with a negative pressure difference in the solver particle pool. Therefore, whenever the solver runs, the pressure difference will be added to all the resulting forces and simulate the fluid being sucked into the point. Fluid particles are deactivated once they enter the vicinity of the suction point. This makes the implementation very convenient as the suction point can be moved freely, and the physics will act independently of any external interference, exploiting the robust nature of position-based dynamics. Another great advantage lies in the fact that the number of particles is controlled in the scene with suction, hence adding up to the performance of the simulation. Figure 7 demonstrates this added effect. The circular boundary was removed to illustrate this effect in more detail. The below figure demonstrates the suction point attached to the electrocautery tool (cylindrical device in the middle bottom part of Figure 7).

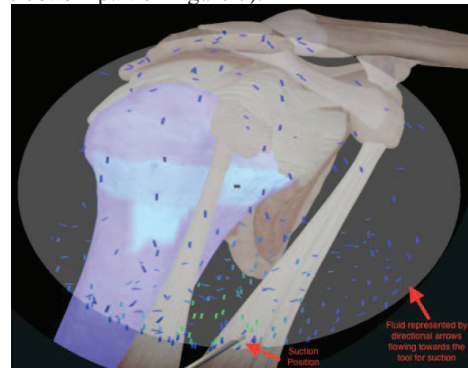


Fig. 7. Demonstration of dynamic suction point.

III. RESULTS

We have investigated the solver's performance by recording our simulation's average frames per second (FPS). The simulation was performed for one minute, and the results were as follows (data represents the crucial functions of simulation), as seen in Figure 8a. In Figure 8a, all functions were timed within one time period (i.e., $1/202 \sim 0.00495s$), and the simulation maintained an average frame rate of 202 Hz over 60 seconds.

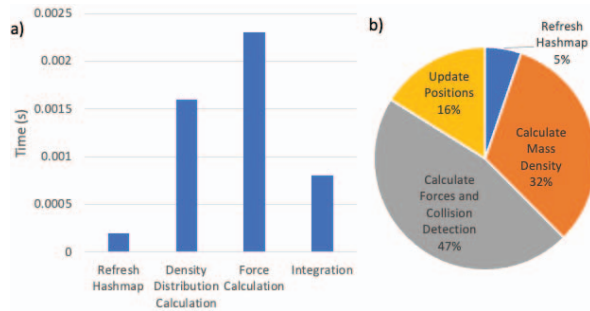


Fig. 8. a) Elapsed time for the serial algorithm at a discrete time in a one-time period and b) duration in percentage by the components for every time period.

We recorded the performance of every component in the solver to locate specific locations of the bottlenecks. We tackled the bottlenecks, such as position update, hash map refresh, mass density calculation, force calculation, and collision detection, by multithreading the algorithm for all the components. The bottlenecks are visualized in Figure 8b, showing the percentage of time utilized by every component for every period. As we can see, the particle query (collision detection and force calculation) is the most expensive part of this algorithm, which was 47% of the time elapsed for computation, while refreshing the hash map only 5% of the time elapsed time. Based on this data, we focused on optimizing the performance accordingly. We also investigated the impact of the smoothing distance on the solver, which has a linear $O(n)$ impact as expected, as seen in Figure 9.

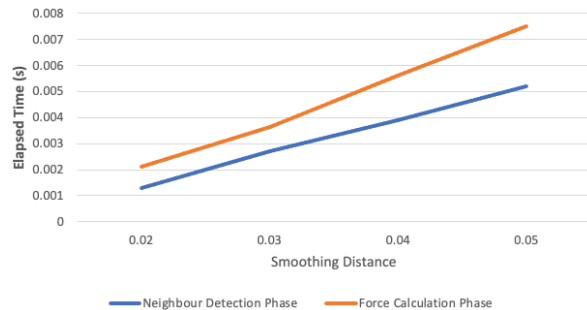


Fig. 9. Smoothing distance impact on the solver.

1) Parallelization of Framework: Multithreading Fluid Solver

We parallelized our framework using OpenMP. The parallelization is performed at the loop level. Every simulation component is parallelized, and the results are given in Figure 10. The solver's performance, on average, increased by 46% after the parallelization. The average FPS increased to 294 Hz. The comparison between serial and parallel performance is reflected in the chart below in Figure 11.

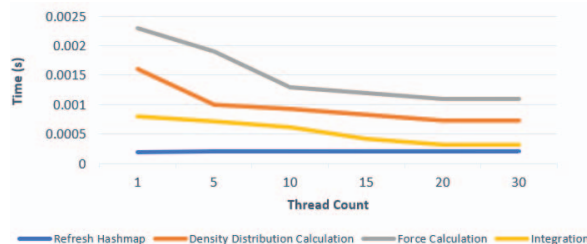


Fig. 10. Elapsed time for the parallel algorithms.

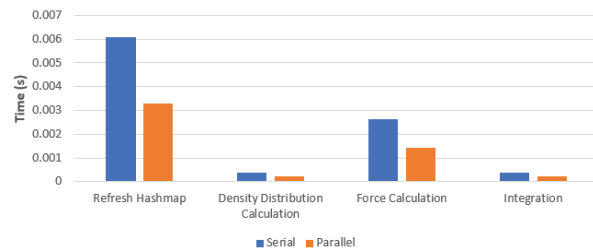


Fig. 11. Comparison of Multithreaded and Serial Performance for each parallelized algorithm.

IV. CONCLUSION AND FUTURE WORK

This work demonstrated our unified Lagrangian physics framework, which handles real-time fluid and heat flow simulations in Virtual Rotator Cuff Arthroscopic Skill Trainer (ViRCAS). The real-time fluid and heat flow interactions enhance the realism aspect of the training simulation. This framework can manage efficient interactions between a) heated tissue – irrigation solution, b) electrocautery – bone and tissue, and c) irrigation solution – bone and tissue. Also, the framework is designed to be robust, making it usable for various other simulations without any significant changes. Also, in this study, we share the initial outcomes of our comprehensive SPH methodology aimed at simulating the electrocautery procedure and fluid dynamics during arthroscopic rotator cuff surgery within the ViRCAS environment. Our study delves into the operational effectiveness of the solver, gauging it through the calculation of average FPS during a one-minute simulation run. The resulting mean FPS registered at 202 Hz. We integrated parallel processing into our framework using OpenMP at the loop level to enhance efficiency, resulting in a noteworthy 46% upsurge in FPS.

As future work, we intend to work on optimization methods to eliminate all the remaining bottlenecks for more stable and faster performance. We plan to extend parallelization further, converting our solver into a parallel GPU version using NVIDIA's CUDA toolkit since the results shown in [32]–[34], [46] were significantly positive.

ACKNOWLEDGMENT

This study was made possible by grants from the National Institutes of Health (NIH) / National Institute of Biomedical Imaging and Bioengineering (NIBIB)5R01EB005807-11, 3R01EB005807-09A1S1, 1R01EB033674-01, and 5R01EB025241-04. This project was also supported by the Arkansas INBRE program, supported by a grant from the National Institute of General Medical Sciences (NIGMS), P20 GM103429 from the NIH.

REFERENCES

- [1] W. Kwabla *et al.*, "Evaluation of WebRTC in the Cloud for Surgical Simulations: A Case Study on Virtual Rotator Cuff Arthroscopic Skill Trainer (ViRCAS)," in *International Conference on Human-Computer Interaction*, Springer, 2023, pp. 127–143.
- [2] M. Müller, D. Charypar, and M. H. Gross, "Particle-based fluid simulation for interactive applications," in *Symposium on Computer animation*, 2003.
- [3] F. H. Harlow and J. E. Welch, "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface," *Phys. Fluids*, vol. 8, no. 12, pp. 2182–2189, 1965.

- [4] R. Courant, E. Isaacson, and M. Rees, "On the solution of nonlinear hyperbolic differential equations by finite differences," *Commun. Pure Appl. Math.*, vol. 5, no. 3, pp. 243–255, 1952.
- [5] L. Yaeger, C. Upson, and R. Myers, "Combining physical and visual simulation—creation of the planet jupiter for the film '2010,'" *Acm Siggraph Comput. Graph.*, vol. 20, no. 4, pp. 85–93, 1986.
- [6] N. Foster and D. Metaxas, "Modeling the motion of a hot, turbulent gas," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997, pp. 181–188.
- [7] N. Foster and D. Metaxas, "Realistic animation of liquids," *Graph. Models Image Process.*, vol. 58, no. 5, pp. 471–483, 1996.
- [8] M. Kass and G. Miller, "Rapid, stable fluid dynamics for computer graphics," in *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, 1990, pp. 49–57.
- [9] J. Stam and D. Brinsmead, "Method of Producing Fluid-Like Animations Using a Rapid and Stable Solver for the Navier-Stokes Equations," Jul. 24, 2001.
- [10] J. Stam, *A general animation framework for gaseous phenomena*. European Research Consortium for Informatics and Mathematics, 1997.
- [11] J. Stam, "A simple fluid solver based on the FFT," *J. Graph. Tools*, vol. 6, no. 2, pp. 43–52, 2001.
- [12] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke with an octree data structure," in *Acm siggraph 2004 papers*, 2004, pp. 457–462.
- [13] M. B. Nielsen and K. Museth, "Dynamic Tubular Grid: An efficient data structure and algorithms for high resolution level sets," *J. Sci. Comput.*, vol. 26, no. 3, pp. 261–299, 2006.
- [14] B. Houston, M. Wiebe, and C. Batty, "RLE sparse level sets," in *ACM SIGGRAPH 2004 Sketches*, 2004, p. 137.
- [15] B. Houston, M. B. Nielsen, C. Batty, O. Nilsson, and K. Museth, "Hierarchical RLE level set: A compact and versatile deformable surface representation," *ACM Trans. Graph. TOG*, vol. 25, no. 1, pp. 151–175, 2006.
- [16] S. Chen and G. D. Doolen, "Lattice Boltzmann method for fluid flows," *Annu. Rev. Fluid Mech.*, vol. 30, no. 1, pp. 329–364, 1998.
- [17] A. J. Ladd, "Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation," *J. Fluid Mech.*, vol. 271, pp. 285–309, 1994.
- [18] T. Wang and J. Wang, "Two-fluid model based on the lattice Boltzmann equation," *Phys. Rev. E*, vol. 71, no. 4, p. 045301, 2005.
- [19] W. T. Reeves, "Particle systems—a technique for modeling a class of fuzzy objects," *ACM Trans. Graph. TOG*, vol. 2, no. 2, pp. 91–108, 1983.
- [20] N. Chiba, S. Sanakanishi, K. Yokoyama, K. Muraoka, and N. Saito, "Visual simulation of water currents using a particle-based behavioural model," *J. Vis. Comput. Animat.*, vol. 6, no. 3, pp. 155–171, 1995.
- [21] R. A. Gingold and J. J. Monaghan, "Kernel estimates as a basis for general particle methods in hydrodynamics," *J. Comput. Phys.*, vol. 46, no. 3, pp. 429–453, 1982.
- [22] L. B. Lucy, "A numerical approach to the testing of the fission hypothesis," *Astron. J.*, vol. 82, pp. 1013–1024, 1977.
- [23] M. Becker and M. Teschner, "Weakly compressible SPH for free surface flows," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2007, pp. 209–217.
- [24] J. J. Monaghan, "Simulating free surface flows with SPH," *J. Comput. Phys.*, vol. 110, no. 2, pp. 399–406, 1994.
- [25] J. Stam and E. Fiume, "Depicting fire and other gaseous phenomena using diffusion processes," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995, pp. 129–136.
- [26] M. Desbrun and M.-P. Gascuel, "Smoothed particles: A new paradigm for animating highly deformable bodies," in *Computer Animation and Simulation '96*, Springer, 1996, pp. 61–76.
- [27] J. Stam, "Stable fluids," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999, pp. 121–128.
- [28] M. Müller, B. Solenthaler, R. Keiser, and M. Gross, "Particle-based fluid-fluid interaction," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2005, pp. 237–244.
- [29] M. Teschner, B. Heidelberger, M. Müller, D. Pomerantes, and M. H. Gross, "Optimized spatial hashing for collision detection of deformable objects.," in *Vmv*, 2003, pp. 47–54.
- [30] B. Solenthaler and R. Pajarola, "Predictive-corrective incompressible SPH," in *ACM SIGGRAPH 2009 papers*, 2009, pp. 1–6.
- [31] M. J. Harris, *Real-time cloud simulation and rendering*. The University of North Carolina at Chapel Hill, 2003.
- [32] T. Harada, S. Koshizuka, and Y. Kawaguchi, "Smoothed particle hydrodynamics on GPUs," in *Computer Graphics International, SBC Petropolis*, 2007, pp. 63–70.
- [33] M. J. Harris, "Fast fluid dynamics simulation on the GPU.," *SIGGRAPH Courses*, vol. 220, no. 10.1145, pp. 1198555–1198790, 2005.
- [34] P. Goswami, P. Schlegel, B. Solenthaler, and R. Pajarola, "Interactive SPH simulation and rendering on the GPU," 2010.
- [35] P. Oppenheimer, A. Gupta, and S. Weghorst, "Endourologic surgical simulations," in *Proceedings of MMVR Conference*, 2001, pp. 365–71.
- [36] J. M. Gomiz, F. L. Mombiola, and J. V. Martin, "Irrigation systems in shoulder arthroscopy," *Rev. Esp. Cir. Ortopédica Traumatol. Engl. Ed.*, vol. 52, no. 4, pp. 250–259, 2008.
- [37] R. Rook, M. Yildiz, and S. Dost, "Modeling transient heat transfer using SPH and implicit time integration," *Numer. Heat Transf. Part B Fundam.*, vol. 51, no. 1, pp. 1–23, 2007.
- [38] T. Sophy, H. Sadat, and C. Prax, "A meshless formulation for three-dimensional laminar natural convection," *Numer. Heat Transf. Part B Fundam.*, vol. 41, no. 5, pp. 433–445, 2002.
- [39] Y. L. Wu, G. R. Liu, and Y. Gu, "Application of meshless local Petrov-Galerkin (MLPG) approach to simulation of incompressible flow," *Numer. Heat Transf. Part B Fundam.*, vol. 48, no. 5, pp. 459–475, 2005.
- [40] I. V. Singh, K. Sandeep, and R. Prakash, "Heat transfer analysis of two-dimensional fins using meshless element free Galerkin method," *Numer. Heat Transf. Part Appl.*, vol. 44, no. 1, pp. 73–84, 2003.
- [41] L. H. Liu, J. Y. Tan, and B. X. Li, "Meshless approach for coupled radiative and conductive heat transfer in one-dimensional graded index medium," *J. Quant. Spectrosc. Radiat. Transf.*, vol. 101, no. 2, pp. 237–248, 2006.
- [42] Y. Zhu and P. J. Fox, "Smoothed particle hydrodynamics model for diffusion through porous media," *Transp. Porous Media*, vol. 43, no. 3, pp. 441–471, 2001.
- [43] I. V. Singh, "Meshless EFG method in three-dimensional heat transfer problems: a numerical comparison, cost and error analysis," *Numer. Heat Transf. Part A*, vol. 46, no. 2, pp. 199–220, 2004.
- [44] J. J. Monaghan, "Smoothed particle hydrodynamics," *Rep. Prog. Phys.*, vol. 68, no. 8, p. 1703, 2005.
- [45] M. Kelager, "Lagrangian fluid dynamics using smoothed particle hydrodynamics," *Univ. Cph. Dep. Comput. Sci.*, vol. 2, 2006.
- [46] D. Demirel, J. Smith, S. Kockara, and T. Halic, "GPU Based Position Based Dynamics for Surgical Simulators," in *International Conference on Human-Computer Interaction*, Springer, 2023, pp. 81–88.