Real-Time 3D Fluid Interaction with a Haptic User Interface

Javier Mora[*] Won-Sook Lee[+]

School of Information Technology and Engineering, University of Ottawa

ABSTRACT

Imagine you are playing a videogame in which you impersonate a wizard who needs to create a potion in order to enchant your enemies. Through a desktop haptic probe, shaped as a baton, you are able to stir and feel the magical fluid inside a bowl. As you follow the potion recipe, you feel how the fluid changes its viscosity, density, velocity and other properties. Various hapto-visual user interfaces enable users to interact in three-dimensions with the digital world and receive realistic kinesthetic and tactile cues in a computer-generated environment. So far solid or deformable objects have been experimented for haptic-tactile feedback. In this paper we innovate by devising techniques that enable the haptical rendering of shape-less objects, such as fluids. Focusing on the real-time performance to enhance the user's experience, the system imitates the physical forces generated by the real-time fluid animation, stirring movements and fluid changes. We achieved real-time 3D fluid and overcame the challenges that arise during the integration of both haptics and graphics workspaces, the free-view visualization of 3D fluid volume, and the rendering of haptic forces. These fluid interaction techniques with haptic feedback have wide possible applications including game development and haptic communities.

KEYWORDS: 3D Interaction, Real-time fluid animation, haptics, input devices, visualization.

INDEX TERMS: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism, —Virtual Reality; I.6.m [Computing Methodologies]: Miscellaneous

1 INTRODUCTION

Haptics, which refers to the technology which stimulates the users' sense of touch, has been increasing in popularity because of the powerful enhancements that it brings to the 3D human-computer interaction experience. Haptics allow users to literally touch and feel characteristics about computer-generated objects such as texture, roughness, viscosity, elasticity, and many other properties, and research has mainly been oriented towards the modeling of solid structures. However, little research has targeted the haptic rendering of shape-less objects, such as fluids. Fluid animation is of great popularity in computer graphics and animation. However, it is difficult to achieve a real-time stable simulation due to the heavy computation required to solve the non-linear Navier-Stokes equation.

Our goal is to combine both fields, fluid animation and its haptic rendering, to offer an interactive experience between 3D fluid and the user. Our motivation is to produce a system that brings human-computer interaction to real-time fluid animations, so that users can appreciate and feel the properties of a fluid simulation via a haptic interface.

Several applications could rise from this integration. Videogames, for instance, could be brought to a higher degree of interaction by providing an interface that enables players to feel the stirring of fluids in order to achieve a game task. Nintendo's recent Wii games [23] are an example of the industry's interest for higher interactive applications. Haptics would allow players to feel the physical properties of in-game objects. In addition, medical applications could imitate the blood flow in a patient's cardiovascular system. In combination with audio and video displays, this technology may also be used to train people for tasks requiring hand-eye coordination. It may also be used as assistive technology for the blind or visually impaired.

We aim to provide the human kinesthetic senses to be stimulated through computer-controlled forces which convey to the users a sense of natural feel about 3D fluid while he/she interacts with it. Our paper focuses on two main issues. Firstly, we examine how to stably represent a 3D fluid simulation in real-time and render it on the screen at an acceptable frame rate of approximately 30 frames per second. Secondly, we examine how to haptically render the simulated shape-less fluid to the user. The haptic probe must also interact with the fluid surface and be able to modify the current flow generated by the simulation. In addition, we also discuss haptic gesture recognition as an interactive application for haptic games.

Achieving a high-quality fluid animation in real-time is quite challenging as it demands constant CPU intensive calculations. Fluids are inherently complex – in general the surface changes quickly, and they are influenced by a variety of conditions (boundaries, aerodynamics etc). Fluids have a lot of detail, and there is no real constraint (e.g. droplets, waves). In order to make a simulation algorithm suitable for an interactive real-time application, there are several things to consider due to the limitation of on-site calculation memory size and computation time. In addition, from the total time available for a single frame, physically-based simulation (calculation of the dynamics) only gets a fraction besides other tasks such as graphic and haptic rendering. In off-line simulations adaptive time-stepping can be used in situations where stability problems arise. In contrast, a real-time application runs at a fixed frame rate. The simulation method must be stable for the given time step size no matter what happens. External forces and the movement of boundaries can get almost arbitrarily high. A real-time simulation method needs to be able to cope with all those situations and must remain stable under all circumstances. Of course it is not possible to reduce the computational and spatial complexity so drastically and increase the stability so significantly without some trade off with the quality of the result. Therefore, what we require from a real-time simulation method is visual plausibility and not necessarily scenes that are undistinguishable from the real world.

Figure 1 shows the high level architecture (HLA) for our system. It displays the two main stages of our system and illustrates the data flow for generating hapto-visual 3D flows from the fluid simulation. These stages are categorized as the processing stage, and the rendering stage. The processing stage computes the fluid simulation and the grid deformation, combined

[*] e-mail: jmora091@uottawa.ca
[+] e-mail: wslee@uottawa.ca

IEEE Symposium on 3D User Interfaces 2008
8-9 March, Reno, Nevada, USA
978-1-4244-2047-6/08/$25.00 ©2008 IEEE

to form a 3D fluid, at each time-step. The rendering stage consists of presenting the output to the user via the haptic device, through haptic rendering, and via the monitor, through graphic rendering. In section 2 of this paper we present a literature review on haptics as well as on real-time fluid animations. The integration of graphics and haptics workspaces is discussed in section 3. In section 4 we introduce the processing stage of the system. Force rendering is explained in section 5. Three dimensional fluid visualization and the results are discussed in section 6 and section 7 respectively. As an application in computer games, a haptic gesture recognition module is described in section 8. We conclude in section 9.

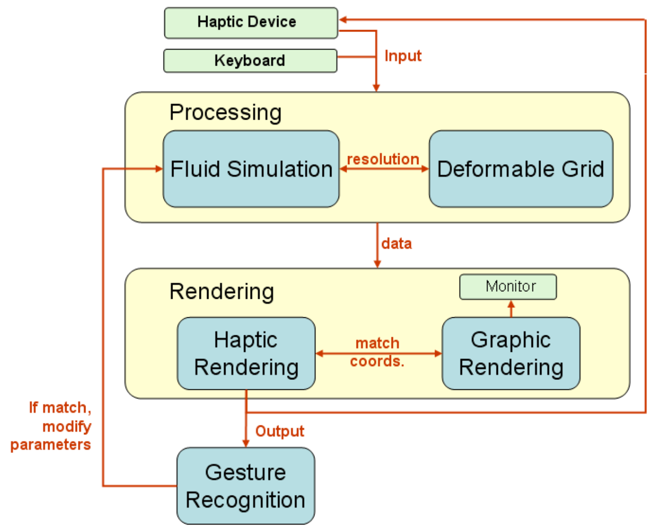

Figure 1. Overall flow of our system

2 Literature Review

The amount of literature regarding haptic technology and rendering has increased substantially in recent years. A more complete background on haptic rendering and haptics in general can be found in other articles [22][4][5][31]. Most typical examples of real-time haptic applications are in games. Experimental haptic games such as HaptiCast [26] and Haptic Battle Pong [8] have been generating brainstorming ideas for assessing haptic effects in game design. In HaptiCast, players assume the role of a wizard with an arsenal of haptically-enabled wands which they may use to interact with the game world. Haptic Battle Pong uses force-feedback to haptically display contact between a ball and a paddle. However, interaction with the game environment is limited since players can feel only the transient forces generated as the paddle strikes the ball. Kauffman et al. [11] present interesting haptic sculpting tools to expedite the deformation of B-spline surfaces with haptic feedback and constraints, but they do not explore any feedback integration with fluid simulations. Graphic frames usually need to be rendered at 30fps to have a visually plausible effect. However, the suggested haptic update rate is of 1 KHz [4], which is an important characteristic of realistic haptic interactions.

There are several competing techniques for liquid simulation with a variety of trade-offs. These methods originated in the computational fluid dynamics community, and have steadily been adopted by graphics practitioners over the past decade. For the simulation of water in real-time, some main simplified methods have become popular in recent years. Procedural water is a procedural method that animates a physical effect directly instead of simulating the cause of it [7][1]. HeightField approximations are appropriate if we would only be interested in the animation of the two dimensional surface of the fluid. Kass and Miller linearize the shallow water equations to simulate liquids [19]. Particle systems are another simplification method that would be a good candidate to represent a splashing fluid or a jet of water [20][12][28]. For computer animators, the main concern is to achieve an efficient and visually plausible effect of a stable real-time fluid interaction, while physical accuracy is of second priority [14].

2.1 Haptic fluids and our approach

Dobashi and his team [32] created a model that approximates real-world forces acting on a fishing rod or kayak paddle by doing part of the math in advance of the simulation: the forces associated with different water velocities and different positions for the paddle or fishing lure were pre-calculated and saved in the database. In addition, their simulation is based on a much larger setup, including two projection screens and large haptic equipment. In contrast to this, our intention is to enable to render real-time fluid calculations on a personal computer or a laptop with a low-end desktop haptic device. To cope with these resource limitations, we take an approach in simulating real-time 3D fluids, by rendering grid-structured deformable 2D layers of fluid simulation. Here the dynamic simulation is represented based on textured fluid where Jos Stam's 2D real-time fluid methods [15][13] are extended to 3D for the fluid parameters. We model density as a set of particles (centers of grid cells) that move through a velocity field, described by the Navier-Stokes equations.

Jos Stam [14] was the first to demonstrate a Navier-Stokes 2D fluid simulation at interactive rates by using a grid-based numerical method free from timestep restrictions [30]. This 2D-based implementation is also shown in their experiment. It is a major achievement that enables real-time fluid dynamics. We compute forces from this type of simulation and further explain it in section 3. Baxter and Lin [30] present a complete thorough section of related work on fluid-haptic topic. They demonstrate an interesting method for integrating force feedback with interactive fluid simulation that represents paint. They also calculate the force-feedback from the Navier-Stokes equations of fluid motion, and adapt their fluid-haptic feedback method for use in a painting application that enables artists to feel the paint based on flat surface. In our paper, we focus on the force feedback that results from the interaction with a constrained pool of fluid.

Karljohan Lundin et al. [17] present a case study where new modes for haptic interaction are used to enhance the exploration of Computational Fluid Dynamics (CFD) data. They haptically represent the air flow around an aircraft fuselage. However, we are more concerned about addressing a different scenario, in which the haptic interface interacts with a bounded fluid simulation that is fast enough to be used in real-time interaction applications. We explore haptic ambient forces to represent differences in fluid densities. An ambient force is a global strength effect that surrounds the haptic probe, regardless of collision with any surface. In addition, we adapt our method to be integrated with a spring-net deformable surface, enabling users to perceive the ripples of interaction in a 3D perspective.

3 Haptic and Fluid Integration

In order to enable haptic interaction, all objects modeled in the graphic workspace also need to be modeled on the haptic workspace. In order for users to feel what they actually see, the position of these 3D models needs to match and correspond on the scene.

Our main research focus is on realistic and interactive fluid animation integrated with haptic. So as partly explained in section 2.1, our 3D real-time fluid is realized by high resolution texture-based representation on a low resolution surface grid where the fluid is calculated by the Navier-Stokes equations. The deformable grid is constructed and the textured fluid is extended to 3D accordingly.

The undulating fluid surface is modeled as a mass-spring particle system [2] that gets deformed when the user tries to enter the fluid domain with the haptic device. Force feedback is felt at this contact point of interaction. Efficiency is achieved by generating surface ripples that are visually plausible and yet based on a fairly low-res system of 15x15 particles for a 2D surface. Then the 3D surface is designed as grid structure of the 2D surfaces. Once inside the fluid domain, and through the tip of the haptic device, the user is able to inject substances that fill into a 3D simulation grid of 15x15x15 which still works interactively in real-time in our experiment with an Intel Pentium powered processor 3.4GHz CPU with 2GB in RAM. The nodes of this grid also oscillate their position based on the deformation of the fluid surface layer. The end-point position of the haptic device will determine the grid cell from which we read and write density and velocity values for our rendering calculations. This three-dimensional deformable grid simulates the fluid based on the Navier-stokes equation [14], and is further explained in the following section.

Due to the suggested haptic update rate of 1 KHz [4], the number of deformable surface particles was kept low to maintain the stability of the system. In our system, the user interacts with the environment using an Omni Phantom [27] haptic device, as shown on Figure 2. Since there are differences between the graphic workspace and haptic workspace, the integration between graphic and haptic workspaces is required.

We are interested in rendering immediate force feedback while maintaining acceptable visual simulation effects. As haptic interaction requires higher frequency updates, we must limit our system to a particular grid size in order to retain stability. However, the graphic workspace and the haptic workspace have different boundary limitations and coordinate systems. Therefore, the point of intersection between the haptic probe and the fluid surface is different in both workspaces. The fluid surface grid size is defined as an *NXN* square, and its coordinates range from *[0…N-1]* in both *X* and *Y* axis of the deformable surface. If we want to know which part of the fluid surface was touched, we need to convert the haptic coordinates into those of the fluid grid. Therefore, we first convert the haptic coordinates into positive values, and then scale them with the graphic workspace boundaries. Based on these conversions, the graphic workspace and haptic workspace are integrated precisely and it shows performance in real-time.

The 3D cursor represents the position of the probe as well as indicates the user's point of interaction. We extended this integration with a deformable surface. The surface deforms with the touch of a haptic probe and gives back the resulting force feedback to the users. Even though higher resolutions of the surface grid provide smoother looks, our experiment shows that a size of 15X15 particles for the deformable surface is the maximum setup permitted to support a reasonable stable and fast real-time simulation. Once we increase the size, the deformable surface would perform too slowly for real-time purposes.

Our deformable grid based fluid representation is chosen for several reasons:

(i) It provides efficiency as we are able to visually represent a high-resolution fluid rendering based on computations from a lower-resolution deformable grid.

(ii) It is not computationally expensive as it allows us to work in three-dimensions without disrupting the real-time interaction requirements of our haptic user interface.

(iii) It provides us with a systematic way of controlling and matching the input/output domain between the haptic workspace and the simulation grid.

4 Real-time fluid simulation

Our fluid animation method is based on the classic Navier-Stokes Equation. The Navier-Stokes Equation can be presented in both velocity and density fields [14]:

$$Velocity: \frac{\partial u}{\partial t} = -(u \cdot \nabla)u + v\nabla^2 u + f \tag{1}$$

$$Density: \frac{\partial \rho}{\partial t} = -(u \cdot \nabla)\rho + k\nabla^2 \rho + S \tag{2}$$

These equations describe the behavior of fluid at one point in a fluid volume. Here, u is the vector-valued velocity at that point, t is time, v is the kinematic viscosity of the fluid, f represents the external force, ρ represents its density, S represents the external substance that is being added to the fluid, p is pressure, k represents a constant at which density tends to diffuse, "$\cdot$" denotes a dot product between vectors, and "∇" denotes the vector of spatial partial derivatives.

The two similar representations of Navier-Stokes equations indicate that we could solve both velocity and density fields in a similar fashion. The incompressibility property of fluids determines that there is an additional constraint, known as the continuity equation. This serves to ensure the conservation of mass. It is a constant-density fluid which could be presented as $\nabla \cdot u = 0$. Due to the application purpose of this paper, we do not describe too many details about the mathematic principles behind the Navier-Stokes equations. In Jos Stam's previous work, the main process for computing density consists of three major steps: adding force, diffusing fluids, and moving fluids. At initialization, the fluids are discretized into computational grids. The velocity field is defined at the center of each cell. For each step, the fluid solver would calculate the parameters in the equation in order to make the simulation real-time. His method applies these three main steps for both the fluid's density and velocity properties due to their similarity. The back-tracing method [14] is the main reason for making the interactive fluid stable and efficient, but it is also the main reason for causing not high-level accuracy and unrealistic visual effects. However the visual effects are still quite acceptable.

We integrate Jos Stam's 2D real-time fluid simulation method into a deformable surface with depth effect and force feedback in 3D space. When users touch the fluid surface, through the haptic interface, they can perceive the resulting surface deformations. When they stir the fluid, they can see changes in the fluid's density and velocity and simultaneously feel the resulting force. The force felt depends on the velocity, direction, density and viscosity properties. These values are calculated and displayed each time the graphic workspace is updated.

4.1 Multiple Fluid Simulation

A substance represents a matter with given properties that enters the base fluid simulation. The injection of a new substance into the simulation is accompanied by short, transient haptic impulses which represent force recoil. Our system also allows for the combination of multiple substances on top of the base fluid. Therefore, different calculation grids are maintained for each substance. Each substance has its own characteristics and colors.

The resulting rendered force is a weighted combination of each substance's grid involved in the mix. Figure 3 shows an initial red substance which is later mixed with a denser green substance. The result is a yellowish blend which combines the contributed haptic properties of both substances.

5 Force rendering

In contrast with conventional haptic systems, our reaction force and torque feedback originate from two sources; (i) deformable surface – that accounts for elastic forces, and (ii) fluid simulation – provides values for viscosity, density, velocity, and inertia. At a basic level, a haptic device generates force-feedback based on the position of the probe's end-effector and the Haptic Interface Point (HIP). These two positions are initially the same, but as the player manipulates the haptic device, the HIP might traverse a collision surface. A force is then rendered at the haptic device which is directly proportional to the vector (times the stiffness scalar) between the device's end-effector and the position of the HIP. In Figure 2, the HIP's position has penetrated a static obstacle (e.g. the baton has touched a wall of the bowl). Since the end-effector cannot move to the HIP's position, a spring force is displayed at the haptic device and the users can feel a collision response.

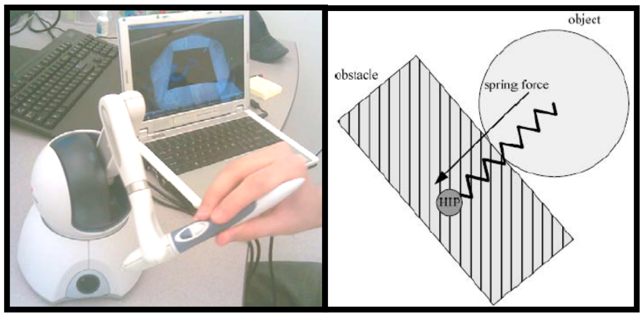

Figure 2. Left image shows the Sensable Phantom Omni Haptic Device. Right thumbnail shows an illustration of the concept behind haptic force rendering.

Elastic spring forces are controlled by stiffness properties that are particular to rigid surfaces, like the walls of a bowl. However, when the probe enters an area of fluid, the force felt is that of a viscous force rather than a spring force. The fluid does not actually repel the probe, but just slows down its stirring movement, according to the density grid computed at the time. The higher the density contained at a grid cell, the harder it is to stir through it. The moving fluid also affects the position of the probe. If the fluid's velocity field is running to the left and the user tries to stir to the right, for instance, a higher force feedback will be felt until the velocity field has adapted itself to the new input forces. Following the law of inertia, the probe will remain in movement unless acted upon by an outside force.

The force feedback calculation is based on the results generated by the equations of an incompressible Navier-Stokes fluid simulation [14], which enables the generation of forces and torques for use with haptic devices. The bowl can also be touched through the haptic interface, giving the player a sense of boundary limitations for the interaction. The deformable surface uses the classical fourth-order Runge–Kutta (RK4) method to solve the ordinary differential equations (ODE) formed by the applied forces and the constrained spring-network of particles.

The fluid surface is deformed as the haptic probe pushes through it. This deformation is gradually transmitted and damped to the lower layers based on their depth and fluid density. After a certain pop-through force threshold, the probe is able to penetrate the surface and interact with the inner 3D fluid. As a consequence, the sense of viscosity can be rendered in any direction of interaction as the probe moves on the three-dimensional scene. In order to simplify and increase the stability of the haptic-graphic simulation, the fluid is contained in a constrained grid environment. The fluid may not tear apart nor spill over the container. Some drawbacks of this proposed deformable surface fluid representation include limitations on the force values that can be rendered, tradeoffs on real-life physics, and restrictions on the grid size of the simulation.

6 Volumetric rendering methods

The OpenHaptics Toolkit [25] provides high-level haptic rendering and is designed to be familiar to OpenGL API programmers. It allows significant reuse of existing OpenGL code and greatly simplifies synchronization of the haptics and graphics threads. However, OpenGL does not support a direct interface for rendering translucent (partially opaque) primitives. Transparency effects may then be created with the blend feature and carefully ordering the primitive data. When using depth buffering in an application, the order in which primitives are rendered is important. Fully opaque primitives need to be rendered first, followed by partially opaque primitives in back-to-front order. If objects are not rendered in this order, the primitives, which would otherwise be visible through a partially opaque primitive, might lose the depth test entirely. In order to visualize the fluid simulation, different techniques were considered for the rendering process, taking into account the limitations of OpenGL rendering capabilities. Some of these techniques included:

(i) Filling the 3D grid by rendering a considerable amount of points in a regular periodic pattern. The color and position of each point would be determined by a linear combination of their proximity to each of the closest grid cell density values. The result is an open-space visualization, similar to floating dust. However, this reduces performance because of the computations needed for each point.

(ii) Making use of OpenGL's fog features. Each of the 3D grid cubes can be treated as a constrained fog space. The fog density values match the simulation density values; however, since fog doesn't allow for multiple colors, our multi-substance simulation cannot be represented using this technique.

(iii) Rendering smaller alpha transparent cubes that are sorted based on their distance to the camera. The density value of a grid vertex is used as a cue for its color. Each small cube face interpolates the color of the four vertices that form it. However, the walls of each inner cube are also visible, which makes the rendering not as smooth as desirable.

The preferred technique consisted of slicing the 3D grid into planes that are perpendicular to the line of camera sight, as shown on Figure 4. We apply gradual alpha transparency to the slices and sort them based on their distance to the camera. Alpha channels are dynamically adjusted based on density values during the simulation. We continuously construct a 3D texture based on the fluid simulation data, and proceed to texture the slices with this information. The texture color is based on the density of the fluid cells. The higher the density, the brighter the color is. A 3D texture is a series of (width * height * depth * bytes per texel) bytes where width, height, and depths are powers of 2. The result, as shown on Figure 5, allows the user to perceive the simulation in any three-dimensional angle.

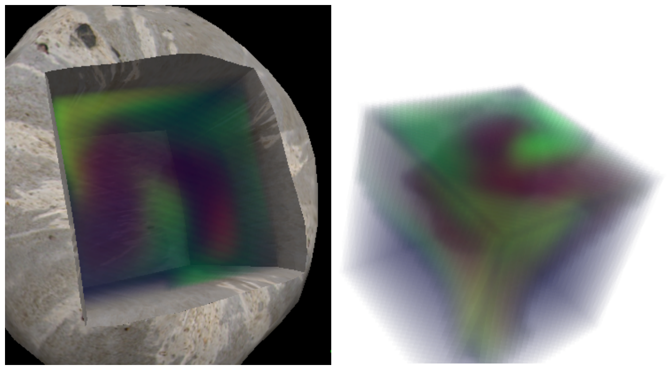

Figure 3. 3D interaction when user mixes multiple fluid substances

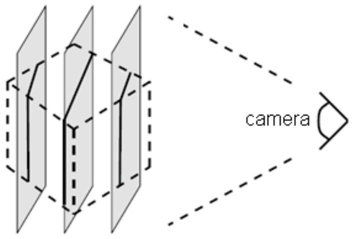

Figure 4. Volumetric rendering using 3D Texture slices

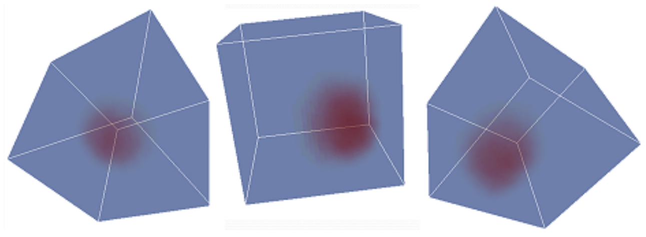

Figure 5. Fluid domain perceived from different angles

7 Results

Screenshots are shown in Figure 6. They present the fluid being stirred by the haptic device. We can see that the intensity of the color represents the density value of a cell. In addition, the fluid follows the velocity field that is being generated by the probe movements. Figure 6 also shows the velocity field that guides the movement of forces across the surface. After removing the probe out of the bowl, the fluid keeps moving itself and gradually reduces its waviness. In the same manner, if the user lets loose of the haptic device while still inside the fluid, the probe will continue to follow the flow's path. Performance is maintained at speed since the simulation is not volumetric but rather based on discreet extensions of two-dimensional layers.

In order to better appreciate the quality of haptic rendering, the user is able to mix and toggle between different force rendering modes. Each of these modes may be enabled or disabled according to the user's preferences. These force modes focus on particular aspects of the force feedback. The deformable-surface mode enables the user to feel the ripples on the fluid surface. The viscosity mode challenges the user to move around dense fluid. The flow mode guides the haptic probe, hence the user's hand, through the velocity field so that the user perceives the formed currents. The flow-resistant mode enables the user to modify the velocity field by applying forces that resist the current flow. As a result, the system serves as an experimental framework to analyze haptic experiments for fluid simulations. Forces can also be visually appreciated by looking at the color of the baton during the simulation. These colors are dynamically modified according to the rendered forces. Users are able to associate the physical force feeling with the visual cue: a green shaded baton for light forces, a yellow color for moderate forces, and red for strong forces.

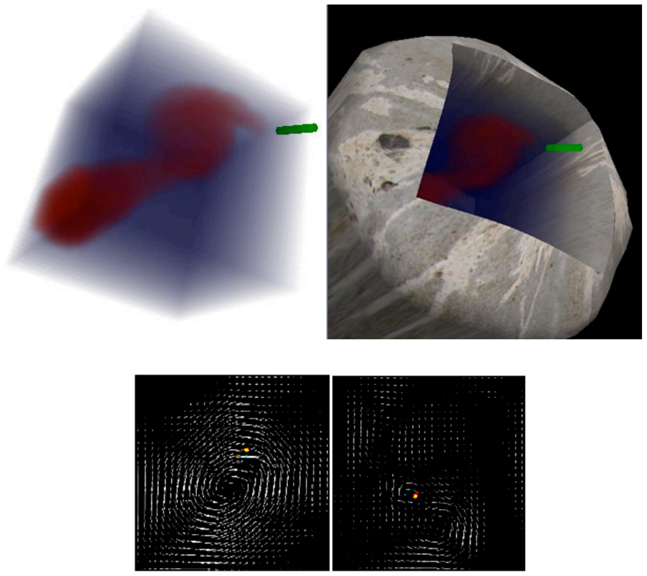

Figure 6. These screenshots show the fluid being stirred by the haptic probe. The different colors represent the different levels of fluid density. 2D version of Velocity field (bottom) shows the resulting current flow of the interaction.

8 Enhanced user interaction application – haptic gesture recognition

This section describes how this haptic fluid interactive system can be enhanced by integrating it with gesture recognition. A possible application can be a game situation such as the player impersonates the role of a witch. Following a specific recipe, a magic potion needs to be created. It would require the right ingredients, mixed at the right moment, with the proper stirring movements and force. Once the player succeeds, the system is able to trigger customized modifications to the fluid properties. The fluid might change color, viscosity and elasticity parameters among other characteristics. The decision might be made through a haptic motion recognition module as one of possible ways that will allow game developers to take full advantage of the high degree-of-freedom input capabilities of modern haptic devices. Haptic devices provide more valuable parameters (force, torque, velocity, etc.) than conventional graphics users interfaces. It does not only allow us to recognize 3D coordinates, but also to use force-feedback data as extractable features. These parameters are used to raise the recognition rate of user motions. For instance, a harsh circular movement will be recognized differently than a gentle circular movement. Even though these are both circular movements, different forces were applied.

Haptic biometric behavioral applications [21] show the importance of force and torque for the purpose of recognition. We present how to recognize a few simple figures, also known as gestures, which would trigger the right potion spell, for instance, three consecutive circular motions or the shape of a star. This would reduce the complexity of the task, and therefore it would be more feasible for the recognition to be performed in real-time, parallel to the haptics and graphics fluid simulations. This module follows Dopertchouk's concepts of motion recognition [24] and is organized in three major steps: Creation and storage of the master gesture templates, normalization of the strokes, and recognition of the shapes. The gesture templates are recorded from predefined sample haptic inputs. We read the proxy position of the haptic device and store the 4D coordinates as a sequence of points in the workspace, mainly *x, y, z* and force data. When players stir the potion mix, some of the gesture shapes may be different in size, speed, or position. Even thought the shape results might look similar to the naked eye, these shapes would look like completely unrelated sets of coordinates to the computer. Therefore, we need to normalize the captured strokes.

First, we need to scale the gesture to a predetermined size (e.g. 1 unit). We do this by finding the length and dimensions of the gesture, and dividing its coordinates by the scaling factor. Second, we need to put the individual points in the gesture at a uniform distance. We do this through a dynamic time warping algorithm [21]. Since we are interested in the geometric shape, it would be irrelevant to know how fast or slow the gesture was drawn. Finally, we need to center the gesture at the origin of the coordinate system through a translation matrix.

We compare two different approaches for the haptic recognition of the gestures: Dot Product and Neural Network. Simple shape recognition was performed through the implementation of a neural network-based recognition engine, using an approach similar to others [16][18], whom also provide good introductions to neural networks. A similar feature extraction procedure was used. However, haptic proxy positions were converted to directional vectors (e.g. Right: *1,0,0; 1,0,0;* ...). A back-propagation algorithm was used to train the neural network with a few basic shapes, run as many epochs and find the minimum sum-of-squares error (SSE) [6] constraint. However, this method proved to be cumbersome to perform in respect to the additional marginal benefit that we would get in the recognition phase. It would be more time-consuming to integrate a new predefined gesture into the system, as the network would need to be retrained. Therefore a simple Dot Product method is chosen for our recognition system. Since both our gesture templates and captured strokes have the same number of points after normalization, we model our gestures as normalized vectors. These are *4XN* dimensional vectors, where *N* is the number of points in the gesture. Using this technique, if you compare two normalized vectors that are exactly the same, the result will be one. The result will be a value slightly less than one for vectors that point in more-or-less the same direction, and the result will be a low number for vectors that point in different directions.

This worked well for simple shape matching and it did not slow down any of the haptic fluid nor the deformable surface computations. From a set of three basic motions (e.g., circle, V shape, and S shape), this module was able to reach recognition rates above 95% for both recognition approaches. In our game scenario, the dot product approach seemed effective enough to recognize potion shapes. Neural networks also provided acceptable gesture recognition rates, but the time allocated for network retraining is cumbersome and tedious for gaming purposes.

9 CONCLUSION

We have shown a novel 3D human-computer system based on haptic-fluid interaction. It is the fluid simulation with the deformable surface together with the force feedback of the haptic device that requires very high-speed interaction rate. The system is stable and efficient. In addition, the realistic looking fluid rendering and haptic feedback have been successfully achieved. Our main contribution is to extend the Human-Computer interface into real-time 3D fluid interaction with force feedback. In summary, the proposed solution consists of the following techniques:

(i) Use of a textured deformable grid to represent the state of the simulation and model the fluid motion based on the Navier-Stokes equations [15]. We achieved successfully the 3D extension of Jos Stam's [14] 2D real-time fluid animation.

(ii) Rendering of 3D fluid in any camera viewport. We cut slices parallel to the camera viewport, and apply a 3D texture based on the fluid's density values. Use alpha blending to achieve a volumetric rendering effect.

(iii) Matching of haptic and graphic coordinates through a mapping function to achieve haptic input/output and maximize workspace area.

(iv) Rendering of a set of haptic effects based on the velocity field and density values of the fluid simulation in order to achieve the sense of a moving shape-less object.

These fluid interaction techniques with haptic feedback have wide possible applications including game development and haptic communities. Haptic gesture recognition, as an application for haptic games, was also experimented. OpenGL was used to implement the graphic framework of the system. We made use of lighting, blending, and shading effects to appreciate the animated fluid ripples. A bowl model was created on Autodesk 3D Studio Max [3] and imported into the scene. OpenHaptics API was used to model the haptic interactivity of the tool. The system performed on an Intel Pentium powered processor with 3.4GHz CPU and 2GB in RAM. A Sensable Phantom Omni [27] device was used as our haptic device. It would be interesting to keep exploring the haptic gesture recognition phase of the project to produce more various effects on the fluid with game scenario. The orientation and workspace of the Phantom series of haptic devices allow the users to make natural, human gestures using a stylus. Even the idea of waving a haptic stylus through the air in order to cast spells is appealing in that it makes the player feel as if they really are wizards. This is a feature-in-progress, but current results look promising.

REFERENCES

[1] A. Fournier and W. T. Reeves. A simple model of ocean waves. In Proc. SIGGRAPH, pages 75–84, 1986.

[2] A. Nealen, M. Muller, R. Keiser, E. Boxerman, Carlson M.. Physically based deformable models in computer graphics. *In Eurographics: State of the Art Report.* (2005)

[3] Autodesk, http://autodesk.com

[4] C. Basdogan. Haptic Rendering Tutorial, http://network.ku.edu.tr/~cbasdogan/Tutorials/haptic_tutorial.html (2007)

[5] C.H. Ho, C. Basdogan, M. Srinivasan. Efficient point-based rendering techniques for haptic display of virtual objects. Teleoperators and Virtual Environments, pp.477-491, (1999)

[6] C. M. Bishop. Neural Networks for Pattern Recognition, Oxford Press, USA, (1995)

[7] D. Hinsinger, F. Neyret, and M.P. Cani. Interactive animation of ocean waves. In Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim., pages 161–166, 2002.

[8] D. Morris, J. Neel, K. Salisbury. Haptic Battle Pong: High-Degree-of-Freedom Haptics in a Multiplayer Gaming Environment. In Experimental Gameplay Workshop, GDC (2004)

[9] D. Nilsson, H. Aamisepp. Haptic hardware support in a 3D game engine. Master thesis, Department of Computer Science, Lund University, May (2003)

[10] F. Conti, F. Barbagli, D. Morris, C. Sewell. CHAI: An Open-Source Library for the Rapid Development of Haptic Scenes. Paper presented at *IEEE World Haptics*, Italy, (2005)

[11] F. Dachille, H. Qin, A. Kaufman. Novel haptics-based interface and sculpting system for physics-based geometric design. Computer-Aided Design, Vol. 33, pp.403-420 (2001)

[12] J. O'Brien and J. Hodgins, Dynamic simulation of splashing fluids, In Computer Animation 95, pages 198–205. (1995)

[13] J. Stam. Interacting with smoke and fire in real time. In: *Communications of the ACM*, Volume 43, Issue 7, pp.76-83. (2000)

[14] J. Stam. Real-Time Fluid Dynamics for Games. In *Proceedings of the Game Developer Conference*, March (2003).

[15] J. Stam. Stable Fluids. In: *SIGGRAPH 99* Conference Proceedings, Annual Conference Series, pp.121-128. (1999)

[16] K. Boukreev. Mouse gestures recognition, http://codeproject.com/cpp/gestureapp.asp (2007)

[17] K. Lundin, M. Sillen, M. Cooper, A. Ynnerman. Haptic visualization of computational fluid dynamics data using reactive forces. In. *Proceedings of the International Society for Optical Engineering*,(2005)

[18] K. Murakami, H. Taguchi. Gesture Recognition using Recurrent Neural Networks. In Proceedings: Conference on Human Factors in Computing Systems, pp.237 - 242, (1991)

[19] M. Kass, G. Miller. Rapid, Stable Fluid Dynamics for Computer Graphics. *ACM Computer Graphics (SIGGRAPH '90)*, 24(4):49–57, August 1990.

[20] M. Müller, D. Charypar, M Gross. Particle-Based Fluid Simulation for Interactive Applications, In Proceedings of SCA 03, pages 154-159. (2003)

[21] M. Orozco, A. El Saddik. Haptic: The New Biometrics-embedded Media to Recognizing and Quantifying Human Patterns. *In proceedings of 13th Annual ACM International Conference on Multimedia* (ACMMM 2005), Singapore, November 06-12, (2005)

[22] M. Srinivasan and C. Basdogan. Haptics in Virtual Environments: Taxonomy, Research Status, and Challenges. In *Computer and Graphics*, 21(4), pp.393-404, (1997).

[23] Nintendo Wii, http://wii.com

[24] O. Dopertchouk. Recognition of Handwritten Gestures, http://gamedev.net/reference/articles, (2007)

[25] OpenHaptics Toolkit. Available at: sensable.com/products-openhaptics-toolkit.htm

[26] S. Andrews, J. Mora, J. Lang, W.S. Lee. HaptiCast: A Physically-Based 3D Game with Haptic Feedback. In Proceedings of FuturePlay (2006)

[27] Sensable Technologies, http://sensable.com

[28] S. Premoze et.al, Particle based simulation of fluids, Eurographics 03, pages 401-410. (2003)

[29] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of SIGGRAPH '99*, 187-194 (1999).

[30] W. Baxter, M.C. Lin. Haptic Interaction with Fluid Media. Proceedings of Graphics Interface. Vol. 62, pp.81-88, Canada (2004)

[31] W. Mark, M. Randolph, J.V. Finch, R.M. Verth, Taylor II. Adding Force Feedback to Graphics Systems: Issues and Solutions. *SIGGRAPH'96*, pp. 447-452, August (1996)

[32] Y. Dobashi, M. Sato, S. Hasegawa, T. Yamamoto, M. Kato, T. Nishita. A Fluid Resistance Map Method for Real-time Haptic Interaction with Fluids. Proceedings of ACM VRST'06. pp. 91-99 (2006)