Abstract

Visualisations and computations of a real time fluid dynamic rendering have always been a fascinating research topic in the field of computer graphics. However, most existing virtual reality surgery simulators tend to avoid the fluid dynamic model involvement in their system. This is due to the fact that real-time fluid dynamic visualisation is computationally expensive. The calculation may slow down the rendering time and reduce force feedback interactions during simulations. With the availability of high performance graphic hardware, the improvement of visual quality and computational accelerations become easier to achieve. In this paper, we propose fast and efficient fluid dynamic visualisations for a heart surgery simulation. The algorithm utilizes the GPUs capable streaming computations to generate physically-based computational fluid dynamic for bleeding in real time. Our approach is based on the Navier-Stokes Equations that is implemented on two-dimensional data structure which stores height fields, blood quantity, and dissolving blood velocities. We use the cubic interpolated propagation as a fluid solver of blood movement. In this paper, the blood flowing on the surface of a beating heart when the surgical knife cut the heart muscle surface in a certain thickness. The blood will move from the sources and follow the height map of the heart surface. The comparison of the frame rates of surgery simulation with and without fluid inclusion is presented and proves that our approaches are effective enough for evolving fluid dynamic visualisation during surgical simulations.

Keywords: GPU, heart, surgery simulation

1 Introduction

Although the introduction of minimally invasive techniques and new stabilizing devices has undergone resurgence of the beating heart of the coronary artery bypass, the development of a method for training surgeons to perform accurate anastomoses despite cardiac movement and to develop the skills are still needed for keeping a reliable result (Stanbridge, David O’Regan et al. 1999). Consequently, simple methods to train surgeons in performing these operations are urgently needed. While the application of robot-assisted coronary artery surgery has been introduced, more advance technology may be involved to reach optimal results.

In the last few years, the developments of computer graphic technology and hardware have evolved into an effective tool not just in generating more realistic visualization but also in improving the speed of computation. A virtual reality (VR) simulation now becomes possible to be established in real time with force feedback interactions. This fact bring VR to be widely used as a training tool in many applications (Stone 2000). One of the potential applications is in medical industries, particularly for surgery simulations. This techniques can provide a realistic, safe, and controllable environment for novice doctors to practice surgical operations, allowing them to make mistakes without serious consequences (Anonymous 2001; Higgins and Koucky 2002).

Fluid dynamic modelling can enhance the realism in surgical simulation. Without fluid, the surgical simulator is dry and clean. However, most surgery procedure involves fluid. Particularly in heart surgery the fluid interaction during simulation is important. Although fluid dynamic inclusion may have not much effect to force feedback during the simulation, the existing fluid cannot be avoided (Bhasin, Liu et al. 2005) during surgical operations. Besides, it is providing a realistic visual cue, since the existing fluid can obscure surgeon’s view and make operation difficult. The fluid model can be useful in trainings such as sucking, draining, and flooding, or tackling unpredictable blood splashing from the heart.

The fast improvements and its programmability of graphics hardware (Owens, Luebke et al. 2005) allow us to fulfill computationally demanding tasks in many purposes. This has led us to develop a new mechanism in designing surgical simulator. The simulator includes computation of fluid movement based on General-purpose computation on graphics processors (GPGPU). Our approach provides the balancing between computation and real-time visualisations for designing user interactivity during surgical simulation designs.

This paper proposes force and torque models for generating feedback on both the fluid and heart muscle movement into a haptic device during heart beating surgery simulations. The users not only see the fluid, but also can feel and interact with them (blood and water). We introduce fast solutions of Navier-Stokes Equations (NSEs) using a modified cubic interpolated propagation...
scheme. The framework is used to provide real-time computations with realistic 3D visualizations.

2 Related Work
In the last few years, many approaches have been developed to compute fluid dynamic models. Some of them also have done research in the area of fluid dynamic simulation and virtual reality. This section discusses some previous work and will divide discussion into four sections: fluid modelling, haptic renderings, fluid dynamic solver, and GPU programming.

2.1 Fluid modelling for computer graphic
Recently, the modelling of fluid dynamic has received much attention in the computer graphics society. Although, there are many good works related to this field, this paper only introduce some recent closely related works.

A fluid modelling in computer graphic with interactive animation was pioneered by (Stam and Fiume 1993). This study simulated a fluid turbulence based on advection-diffusion model of particle-based gaseous phenomena. They used clustering algorithm to get efficient fluid animations and renderings. This work had been extended by (Foster and Metaxas 1996) with introduction of a more complex fluid behaviour. However, those studies suffer from blowing when a bigger time step is applied. A semi-Lagrangian method with unconditionally stable fluid simulation, then, was introduced by (Stam 1999) to overcome that problem. Some other researches to augment fluid modelling such as: particle-based fluid animation (Kruger, Kipfer et al. 2005; Muller, Solenthaler et al. 2005) and turbulent water over natural terrain had been continuously developed for various applications. All of these studied mainly discuss on the ways of finding efficient NSEs solution of motion for liquid with realistic looking behaviours for practical animations. Other methods as proposed in (Matthias and Faller 2009) for tracking fluids over surface, in (Adrien, Andrew et al. 2006) for real time interaction between fluid and objects, and in (Benjamin and Erik 2009) for real-time fluid simulation, look superior than the proposed method; however, these methods only interact with mouse or digitally generated object. Their robustness, real-time interaction, and its computation efficiency, have not been tested for multisensory hardware interaction such as haptic.

In the last ten years the introduction of graphic processor unit (GPU) has also embossed fluid modelling in several ways. GPU not just provide better visualizations, but also support efficient computations with its parallelism and programmability. The application of GPU for flow simulation was introduced by (Csebfalvi and Szirmary-Kalos 2003). This study not only focuses on visual fidelity as done in (Stam and Fiume 1993; Foster and Metaxas 1996; Stam 1999; Muller, Solenthaler et al. 2005; Losasso, Irving et al. 2006), but also attempts to accelerate the flow simulation in real time speed, while maintaining physical accuracy. The parallel algorithm to approach shallow water equation on GPU as extension of (Kass and Miller 1990) work was implemented by (Noe and Trier 2004). Moreover, the GPU applications for interactive fluid motion in terrain rendering with erosions (Anh, Sourin et al. 2007; Mei, Decaudin et al. 2007) also become a current research interest in computer graphics and animation. While these studies show potential outcome in increasing the visual quality and the user interactivity, none of them involved force feedback interactions in their fluid visualizations.

2.2 Haptic rendering for fluid interaction
The exploration of force feedback design and computations for haptic instruments has been done for several years; however, there are only a few that concern with force feedback design for fluid media. Haptic rendering method was introduced by (Avila and Sibierijski 1996). This study, then, inspired many applications such as: surgical cutting or trimming (Thomas V and Cohen 1999), sculpting (Kim and Park 2004), and volume visualization (Lundin, Gudmundsson et al. 2005). Whereas a force feedback integration with interactive fluid model has been introduced in (Baxter and Lin 2004). This method enables force and torque generations in virtual painting applications. There is no other study exploring this field except in (Lundin, Silen et al. 2005) for presenting fluid dynamic data and in (Bhasin, Liu et al. 2005) for simulating droplet fluid. Both of them more concentrate on the fluid model visualization rather than its interaction to haptic device.

2.3 The development of fluid solver
There many floating numerical fluid dynamic solvers have been developed; however, the discussion in this section only focuses on the numerical solvers that rely on the cubic interpolated propagation (CIP) scheme. This method was firstly introduced in (Yabe, Ishikawa et al. 1991) as a universal solver for hyperbolic equation by cubic interpolation. It is then used for solving fluid dynamic equation as a conservative semi-Lagrangian solver for a solid, liquid and gas. Some studies prove that this method is efficient enough to solve hydrodynamic equation in all states (Yabe, Ogata et al. 2002).

More advance CIP study (Song, Shin et al. 2005) attempts to improve the stable fluid (Stam 1999) from numerical dissipation. This scheme is not just applicable for dissipative media (like fog and smoke), but also non-dissipative liquid (water). While, the CIP produce low dispersion compare to other method (Yabe, Takizawa et al. 2007), it need no dimensional splitting (Kim, Song et al. 2008). These provide user to simulate fluid in all state simultaneously. Lastly, although the CIP has been accepted as an alternative method for fluid dynamic solver that fits the need of computer animations, to the best of author knowledge, there is no other study to date that uses this method in surgery simulation including force feedback interaction device.

In this paper, we introduce our approach in combining CIP fluid solver and parallelism in GPU. The rest of the paper is organized as follows: Force model computation is described in Section III; GPU implementation for visual rendering and fluid solver computation is described in Section IV; Section V describes the experimental setups and the results; and lastly, Section VI describes a conclusion of the study and planned work for the future.
3 Force models computation

A force model is defined as a force generating feedback to haptic devices or other kinaesthetic feedback tools in a virtual environment. This force has a means of mapping from the haptic device position to a force vector (Anonymous 2005). There are techniques for integrating force models into virtual environment. This paper concentrates on integrating force as the existing fluid in the simulation.

3.1 Fluid solver formulation

Theoretically, fluid dynamic equations are described by Navier-Stokes equations (NSEs). In compressible Newtonian fluids, the motion equations are derived from a combination of the transport of a momentum in the fluid media;

\[
\frac{\partial u}{\partial t} = -(u \cdot \nabla)u - \frac{1}{\rho} \nabla p + \frac{\mu}{\rho} \nabla^2 u + F,
\]

(1)

and mass conservation function;

\[
\nabla \cdot u = 0
\]

(2)

where \( u, p, \rho \) are a velocity field of fluid, a pressure, and the density of the fluid respectively. Whereas \( \mu \) is a viscosity coefficient, \( F \) represents external force (the gravity, forces delivered through haptic), and \( \nabla \) is the differential operator, respectively. The fluid movement can be predicted by solving the Equations (1) and (2). The flow simulation can be run in staggered grids that can be predicted by solving the Equations (1) and (2).

In addition, If \( \varphi \) is the function of the volume of fluid fraction, the fraction of fluid volume in each cell is transported by using advection equation

\[
\frac{\partial \varphi}{\partial t} = -(u \cdot \nabla) \varphi
\]

(3)

The solver of this equation can be generated by using a combination of the CIP method and the advection form (\( -(u \cdot \nabla)u \)). Then, the interface between liquid and solid is traced by distributing Equations (4) into advection phase (Eq.6) and non-advection (Eq.5). If \( g = -(u \cdot \nabla)f \), then we can formulate that:

\[
\frac{\partial \varphi}{\partial t} + u \cdot \nabla \varphi = g
\]

(4)

\[
\frac{\partial \varphi}{\partial t} = g
\]

(5)

\[
\frac{\partial \varphi}{\partial t} + u \cdot \nabla \varphi = 0
\]

(6)

The solution of those equations in the spatial domain \((x, y, z)\) can be performed by solving the non-adveective part (Eq.5) using a finite difference and then advect the result as shown in Eq.6. Numerical solution for this, on the other hand, can be calculated in a grid of cells of the space domain.

Furthermore, the advection phase can be computed by shifting a cubic interpolated profile into space according to the total derivative equations. Eq.6 is also called as level set equations, where the surface of liquid can be obtained by tracking their positions for which \( \varphi = 0 \).

Then the exact solution of Eq.6 can be formulated as

\[
\varphi(x, t) = \varphi(x - ut, 0)
\]

(7)

If the velocity is assumed to have a constant value within a short time, we get

\[
\varphi(x, t + \Delta t) \approx \varphi(x - u\Delta t, t)
\]

(8)

Because of the CIP method uses not only the function values at the grid points, the spatial derivatives (Eq.5) at those positions then are used for constructing the profile inside the grid cell to trace \( \varphi \) and \( g \). The values \( \varphi \) and \( g \) at the two grid points is interpolated by a cubic polynomial. Thus, we can obtain the profile at the next time step \( n+1 \) by transporting \( u\Delta t \) and we have;

\[
\varphi_i^{n+1} = F(x_i - u\Delta t) = a_i \xi^3 + b_i \xi^2 + c_i \xi + \varphi_i^n
\]

(9)

\[
g_i^{n+1} = dF(x_i - u\Delta t) = 3a_i \xi^2 + 2b_i \xi + c_i
\]

where

\[
a_i = \frac{g_i + g_{i+1} + 2(g_{i+1} - g_i)}{D^2} + \frac{2(g_i - g_{i+1})}{D}
\]

\[
a_i = \frac{3(\varphi_i - \varphi_{i+1})}{D^2} + \frac{2g_i + g_{i+1}}{D}
\]

and \( \xi = -u\Delta t \), when \( u>0 \) we have \( D = -\Delta x \), \( i' = i-1 \) otherwise when \( u<0 \), \( D = \Delta x \), \( i' = i+1 \). However the use of cubic profile (Eq.9) can lead to instabilities. The modification as proposed in (Song, Shin et al. 2005) solve this problem and is claimed always stable. This scheme implement higher dimensional CIPS based on the monotonic CIP solver.

![Figure 1](image)

The physical value estimation in advection phase on the departure point of the arbitrary contour.

The numerical solver in CIP scheme employs a spatial profile that rely on different interpolations to propagate the solution along the characteristics (Yabe, Mizoe et al. 2004). The physical value on \( \varphi \), is calculated locally within the computed cell (the original equation). Then, an estimated value is directly advected toward the grid point calculations at the next time step value (Fig.1). This step provides results in more accurate modelling of the real situation, even in fairly coarse grid. While other methods require multi-points to construct the profile in the one dimensional case, the CIP can be constructed only from a single cell. This advantage is useful for treating boundaries. Beside the CIP method provides a stable result, less diffusive, and has third order accuracy (Usunmi, Kunugi et al. 1997; Song, Shin et al. 2005); the numerical solver of fluid motions provide the solutions
simultaneously to all interfaces in all states without dimensional splitting (Kim, Song et al. 2008).

### 3.2 Interaction forces

The external forces or forces driven by user interaction, $F_e$, in (Eq.1) can consist of gravity ($F_g$), surface tension ($f_s$), a user interactive force (haptic) ($F_u$), or a combination of them. These forces can be represented as below:

\[ F_e = F_g + f_s + F_u \]

When the surface tension is treated as a body force and no explicit information on the geometry and position of the liquid surface is required, the surface tension can be expressed as a continuum surface force as follow (Brackbill, Kothe et al. 1992):

\[ f_s = -\sigma \delta_k (\phi) \frac{\partial \phi}{\partial n} \nabla(\phi) \]

where $\sigma$ and $k(\phi)$ are surface constant coefficient and local curvature value respectively, and $\delta_k (\phi)$ delta function that is formulated as:

\[ \delta_k (\phi) = \begin{cases} \frac{1}{2\pi} \log(e) & |\phi| \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \]

Interactive force from user through a haptic device can be formulated as the equation of motion for a rigid body (Eq.10).

\[ m \ddot{x} = g - \int p \, n \, ds \text{ (10)} \]

where $m$ is the mass and $x$ the centre of gravity of the rigid body. (Eq.1) requires the solver for the fluid pressure or commonly called as pressure projection. This projection is solved using the Poisson’s equation with assumption that the liquid has no viscosity and has an external force that is determined by the force released from/to haptic interactions. After the fraction of the volume of fluid in each cell is obtained in the simulation, a fluid surface is created and it is smoothened by subdivision, if necessary. Assuming that $\mathbf{u}'$ is the velocity result obtained by processing (Eq.1), the pressure can be expressed as the Poisson equation

\[ \nabla \cdot \left( \frac{\nabla p}{\rho} \right) = \nabla \cdot \mathbf{u}' \]

(Eq.11) can be discretised as:

\[ \sum_{\rho \in \{(i,j)\}} \left( \rho^{+1}_{i,j} + \rho^{-1}_{i,j} \right) p_{i,j} - \rho^{+1}_{i,j} p_{i+1,j} - \rho^{-1}_{i,j} p_{i-1,j} = -\frac{1}{\Delta t} \sum_{\rho \in \{(i,j)\}} \left( \mathbf{u}'_{i,j} - \mathbf{u}_{i,j} \right) \]

This equation reveals that velocity and density values can be taken from cell faces, whereas the pressure values are taken from the centres of neighbouring cells, as shown in Fig.2.

### 3.3 Repelling forces

A force that is received from or released to the finger through haptic device is called as repelling forces. This force can have the following expression:

\[ d\phi = -np(\phi) ds \text{ (12)} \]

where $\phi$ is the position object that may be not located in the grid cell, hence pressure calculation need to be interpolated based on the CIP scheme. The determination of pressure is critical to get precise repelling force and we need an accurate profile of the pressure inside a grid cell. The pressure may leads to the rigid body torque \((\Gamma)\). The formulation of the torque can be given as:

\[ d\Gamma = -np(\phi) \times (\phi - r) ds \]

These values are updated in each simulation step by running computation kernels over the grid. The computation is implemented as a vertex shader that executes on every cell in the grid and the results are written to an output texture. Since GPUs are designed to render into 2D buffers, the execution must be done for each slice of a 3D volume by iterating slice indexes over the entire grid.

The momentum equation is calculated numerically by splitting the equations into advection and non-advection equation groups. The advection equations consist of velocity, density, and level set advections, whereas non advection equations have pressure projection and diffusions.

### 4 Experimental Setup

Data collected in this paper was generated from objects that were digitally created from a human anatomical test sample. The test results demonstrated the efficiency and effectiveness of our proposed framework for a 3D fluid dynamic simulation in real time interactions with force feedbacks. All results were implemented on a personal computer (PIV Dual Core 3.2 GHz, 2GB RAM, 80GB HD) in combination with the graphics chip of a NVIDIA GeForce 8800 GTX 758MB run on 400 MHz RAMDAC Clock Speed. To support the immersion and kinaesthetic felling, the PHANTOM Haptic premium 1.5 (from SensAble) in Reachin display were used.

The visual and haptic rendering were developed based on Reachin API 4.2 and blended in Visual Studio.net, Python 2.4, and VRML. Mostly the computations in this study were implemented with 32-bit floating points on programmable graphics hardware that was assumed suitable to real-world problems.

Flow charts of the simulation system in the experiment can be described as in Fig 3. The system is divided into two main threads of subsystem: slow and fast subsystem. Slow subsystem manages visual rendering with frequency of about 20-60 Hz, whereas fast subsystem controls haptic rendering in the frequency of about 1000Hz.
5 Results and Discussion

This study examines fluid-haptic interactions in several different scenarios of heart surgery simulations. The medical data in this paper were digitally created from a human anatomical test sample (Anonymous 2003).

The average performance observed during the experiment can be seen in the Table 1. The data is observed based the frame rate on of V\text{fps} (Visual frame rate per second) and H\text{fps} (Haptic frame rate per second). The comparison and contrast of the surgery simulation with and without GPU computations are presented. CPU computation means that all the calculation of the solver were undertaken in CPU and the GPU was only for visualization. CPU-GPU computation means that the computations were done on GPU with CPU as a host controller to manage computation and visualization on the GPU.

<table>
<thead>
<tr>
<th>Type of Interaction</th>
<th>CPU Computation (average)</th>
<th>CPU-GPU Computation (average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cuttings with no bleeding</td>
<td>22 (V\text{fps}) 1001 (H\text{fps})</td>
<td>49 (V\text{fps}) 1001 (H\text{fps})</td>
</tr>
<tr>
<td>Blood flowing test without Cutting (Injector)</td>
<td>24 (V\text{fps}) 980 (H\text{fps})</td>
<td>54 (V\text{fps}) 1000 (H\text{fps})</td>
</tr>
<tr>
<td>Cutting and bleeding</td>
<td>10 (V\text{fps}) 960 (H\text{fps})</td>
<td>43 (V\text{fps}) 1001 (H\text{fps})</td>
</tr>
<tr>
<td>Cutting -bleeding in slow motion</td>
<td>8 (V\text{fps}) 940 (H\text{fps})</td>
<td>29 (V\text{fps}) 993 (H\text{fps})</td>
</tr>
<tr>
<td>Cutting -bleeding in fast motion</td>
<td>4 (V\text{fps}) 920 (H\text{fps})</td>
<td>25 (V\text{fps}) 992 (H\text{fps})</td>
</tr>
<tr>
<td>Cutting -bleeding in fast motion (short)</td>
<td>4 (V\text{fps}) 920 (H\text{fps})</td>
<td>23 (V\text{fps}) 955 (H\text{fps})</td>
</tr>
<tr>
<td>Cutting -bleeding in fast motion (Long)</td>
<td>2 (V\text{fps}) 900 (H\text{fps})</td>
<td>18 (V\text{fps}) 953 (H\text{fps})</td>
</tr>
</tbody>
</table>

Table 1: Frame rate CPU versus GPU computations

In Table 1, a cutting represents a combination of the pressure force and the sharpness constant of the blade. Cutting without bleeding represent the voxel removal along the cutting path. When it is combined with bleeding, the computation will be distributed into two tread groups, one group for managing voxel removal and the other for driving the fluid solvers. Slow or fast movement, on the other hand, influences the speed of voxel removal computations and the releasing fluid from the cutting path. When cutting force is applied, the fluid will be released from the cut position and will continue to flow through the heart surface toward the boundary or towards the lowest parts on the surface and toward the centre of the gravity force. An example of snapshot simulation is shown in Figure 4a and 4b. This example describes the cutting simulation followed by blood flowing over the surface in the cutting path during heart beating surgery. It seems that as the faster scalpel move the frame rate getting slower.

Using GPU can increase the performance of the simulation. From Table 1, the performance increase about 2.2 to over 12 times when GPU computation is implemented on the simulation. The more complex the computation, the better the performance with GPU. It may reveal that when the computations need more threads, using GPU provides more advantage.

6 Conclusion

This paper presents a fluid dynamic modelling for heart surgical simulator. Our work focus on the development of the real-time fluid dynamic with haptic interaction
based on the CPU-GPU balancing to get more realism fluid effects in heart beating surgery simulator. By taking advantage of the GPU parallelism coupling with CIP fluid solver that solve multiphase fluid simultaneously, the fluid dynamic simulation can be created in real time without involving supercomputer.

Our results demonstrated the great potential for designing plausible surgical simulator with complex fluid effects in real-time, particularly for virtual reality simulator with kinaesthetically interaction. It is demonstrated that the basic need of fluid effects in surgical simulation can be created such as: cutting and bleeding, fluid injection, and fluid flowing over complex obstacles. They can be effectively simulated on realistic visualization on reasonable frame rate.

Future attempts in improving virtual surgical simulation include multi-fluid interactions, surgical tools simulations (such as: sucker and injector), and a complex solid-fluid interactions (such as mixing blood and other surgical fluids).

Figure 5. Snapshot of cutting without bleeding (Voxel cutting over the surface of the heart).

Figure 6. Combination cutting-opening-bleeding during heart beating surgery simulations: (a) short movement), (b) long movement.

7 References


Lundin, K., M. Sillen, et al. (2005). Haptic Visualization of Computational Fluid Dynamics Data Using Reactive Forces. the Conference on Visualization and Data Analysis, part of IS&T/SPIE Symposium on Electronic Imaging San Jose, CA USA.