Real-time Synthesis of Bleeding for Virtual Hysteroscopy

János Zátonyi [a,*], Rupert Paget [a], Gábor Székely [a], Markus Grassi [a], Michael Bajka [b]

[a] *Computer Vision Laboratory, Swiss Federal Institute of Technology Zurich, CH-8092 Zurich, Switzerland*

[b] *Clinic of Gynecology, Dept. OB/GYN, University Hospital of Zurich, Switzerland*

Abstract

In this paper we present a method for simulating bleeding in a virtual reality hysteroscopic simulator for surgical training. The simulated bleeding is required to be visually appealing while at the same time instantaneously responsive to any feedback that the surgeon may be conducing to the virtual environment. In order to meet these real-time requirements, we have based the simulation on graphical fluid solvers. These solvers primarily work best over a 2D domain. For correct visualization in the hysteroscopic simulator, it is, however, necessary to perform the simulation fully in 3D. Therefore in this paper we also present the design modifications for 3D graphical fluid solving and show how to use parallelization to maintain real-time behavior. We also discuss how the incorporation of massless particles into the simulation can improve the visual quality of the results by limiting numerical dissipation effects.

Key words: surgical simulation, virtual reality, flow simulation, hysteroscopy, bleeding

* Corresponding author. Address: Computer Vision Laboratory, Swiss Federal Institute of Technology Zurich, Sternwartstrasse 7. CH-8092 Zurich, Switzerland. Phone: +41 1 632 3965, Fax: +41 1 632 1199.

Email addresses: zatonyi@vision.ee.ethz.ch (János Zátonyi), rpaget@vision.ee.ethz.ch (Rupert Paget), szekely@vision.ee.ethz.ch (Gábor Székely), mgrassi@vision.ee.ethz.ch (Markus Grassi).

1 Introduction

Hysteroscopy is the standard visualization of the inner surface of the uterus performed by using one hull containing both the endoscope and the surgical instrument introduced through the cervix into the uterus. To aid visualization and surgical intervention, a hydrometra is established by applying fluid under pressure to distend the uterine cavity. It is the second most often performed endoscopic procedure after laparoscopy in gynecology [1]. Diagnostic hysteroscopy is used for clarification of pathologic alterations in case of symptoms or findings by imaging modalities like irregular bleeding, endometrial thickening, suspected tumors and problems of infertility or sterility. If necessary, it may be subsequently continued with therapeutic hysteroscopy, e.g. for removal of polyps and myomas, endometrial ablation, resection of adherences or uterine septa, cateterism of the fallopian tube or removal of intrauterine devices [2].

The side effects and iatrogenic injuries depend on the procedure that is performed: while the pure diagnostic hysteroscopy presents almost no major problems, therapeutic hysteroscopy is associated with a small number of well-known serious complications. They primarily result from either the inadequate technique of the surgeon, or the fluid overload of the patient. The only way to acquire sufficient experience is with the repetitive training of specific skills, procedures and complication management. However, patient involvement should be avoided during early phase of the surgeon's learning curve.

Currently the basic visio-spatial and manipulative skills are taught by in vitro methods using inanimate objects such as sheep bladders or bell peppers [3]. These units allow the surgeon to learn how to navigate under monoscopic visual feedback, as well as perform basic manipulative components of an intervention. In this way the surgeon develops competence in completing a particular task, but because the real-life effect is lost, one obtains only limited training in dexterity and surgical problem solving. Primarily, there is a lack of realistic tissue reactivity and one cannot experience the complexities of abnormal anatomy or pathologic situations.

It is believed that a reasonably realistic virtual-reality-based endoscopic simulator training for therapeutic hysteroscopies could contribute to a reduced rate of complications. The ultimate advantage of such a simulator is the potential to provide a realistic and configurable training environment that bridges the gap between basic training and performing the actual intervention on patients, without any restriction on repetitive training. However, the simulator systems proposed to date do not achieve the necessary level of realism that is required for this technology to be widely accepted in the medical community [3,4].

In the simulator, the anatomy and some well defined pathologies must be represented. Realistic real-time simulation of the changes in the operational site due to surgical actions and photo-realistic rendering must be achieved, including the control of the hydrometra by manipulating the liquid inflow and outflow. Finally, the manipulator should allow realistic tactile sensations (realized by force-feedback). In contrast to other virtual reality applications, creating the appropriate virtual environment is straightforward. The simulator with its monitor and manipulator corresponds directly to the setup in the true surgical situation.

In a hysteroscopic simulator, special attention must be paid to the simulation of diffuse intra-uterine bleeding, obscuring the surgeon's view, until the correct actions (adjusting the inflow and outflow of liquid) are performed. It has to be emphasized that the mechanism of fluid management requires specific skills from the surgeon. On one side, proper fluid management is crucial to the patient's safety, on the other side, it is the only way to keep the operation site visible. Consequently, we face the task of synthesizing bleeding, which can only be made realistic if we are also able to model its specific dynamic environment, in which an almost continuous current of the distension liquid is kept alive.

2 Methodology

2.1 Simulation of bleeding

The realistic synthesis of bleeding is an important problem without a defined solution. Temporal texture synthesis [5–10] is one possible approach to solving this problem. However, these techniques are limited in their ability to mimic certain temporal phenomena, and do not have the full capability of synthesizing a spatially stochastic, temporally heterogeneous texture like bleeding. Therefore we have chosen a more promising fluid dynamics approach [11]. This choice is motivated by the fact that in the case of hysteroscopy, the bleeding takes place in the cavum uteri distended by liquid.

The precise mathematical equations describing the behavior of fluid flows are the so-called Navier-Stokes equations [12]. They are non-linear partial differential equations and an analytical solution to them is feasible only for the very simple cases. With the rapid evolution of computers, numerical solutions to these equations have come to the front, establishing the domain of computational fluid dynamics (CFD). This branch of fluid dynamics gives a cost-effective way for accurate, real flow simulations specific for engineering purposes. Alternatively, fluid solvers from the computer graphics domain

provide the possibility to achieve fluid-like effects in real-time, where precise physical accuracy is not as important as just plain visual fidelity.

Foster and Metaxas [13–15] used a coarse grid on which they invoked a discretized version of the incompressible Navier-Stokes equations. The non-linear partial differential equations are solved using an explicit finite difference algorithm. However, an implicit time-step restriction is introduced by the Courant-Friedrichs-Levy (CFL) condition in order to avoid instabilities. Basically a time-step must be chosen such that no particle of the fluid may travel more than the distance of one grid cell per time-step. They were able to achieve a frame rate of about 4 frames/sec by using a coarse grid, and a locally modifiable viscosity. This is, unfortunately, too slow for interactive real-time animation.

Stam [16] proposed an unconditionally stable fluid solver that allowed for arbitrary time-steps and therefore the possibility for real-time visualization. Instead of the explicit Eulerian integration scheme, both semi-Lagrangian [17] and implicit methods were used to solve the incompressible Navier-Stokes equations. To maintain conservation of mass, Stam [16] used the Helmholtz-Hodge Decomposition to remove the divergent field within the velocity. This divergent field equated to the pressure field in the Navier-Stokes time derivative equation of velocity [18]. Stam's method of projection to calculate the divergent field decoupled the pressure component from the velocity component of the Navier-Stokes equations.

The Stam method of solution has been further refined for specific applications. Fedkiw et al. [18] incorporated vorticity confinement [19] to alleviate the effect of artificial numerical dissipation. Others [20–23] have used passive particles instead of a density field for modelling liquid surfaces and explosions, where the liquid surfaces were further enhanced by a dynamic level set. However, in our application we do not need to model a liquid to air interface. We merely have the interaction of two liquids that can be modelled with the same parameters, as the non-Newtonian properties of the blood can probably be neglected if only a visually pleasing appearance of the solution has to be achieved.

An alternative approach to fluid animation is with Smoothed Particle Hydrodynamics (SPH) [24–26]. Unfortunately, as reported in [23], the standard SPH approach can be computationally rather expensive when using a large number of particles, since one has to keep track of the nearest neighbors and solve fluid equations for velocity and pressure.

In our approach, we have chosen to use an adaptation of Stam's [16] method of solution, as it seems to reasonably resemble the dynamics of our physical model, while giving us the ability to render the fluid animation in real-time. The velocity field describes the motion of the distension liquid, and the ad-

vection of the density in this field describes the motion of the blood. Thereby rendering just requires assigning a constant red color to each cell, with its alpha value defined by the density.

2.2 Basic flow model

The state of a volume of fluid is determined by its density, temperature and velocity. In the following model both the density and the temperature of the fluid are considered constant. The behavior of the velocity field is then expressed by the following incompressible Navier-Stokes equations:

$$\nabla \cdot \mathbf{u} = 0 \quad , \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F} \quad , \tag{2}$$

where $\mathbf{u}$ is the velocity vector, p is the pressure, ρ is the density of the fluid, ν represents the kinematic viscosity coefficient and $\mathbf{F}$ corresponds to the vector of external body forces.

The quantities $\mathbf{u}$ and $\mathbf{F}$ are varying both in space and time. The spatial coordinates can represent either 2D (x, y) or 3D (x, y, z) positions. Equation (1) represents the physical principle that the mass in the fluid is conserved, while (2) is the momentum equation of the fluid flow. For the derivation of the equations we refer the reader to [27] [12].

The velocity and pressure fields in the Navier-Stokes equations are originally coupled. A single equation can be obtained for the velocity if (1) and (2) are combined, and the mathematical theorem known as Helmholtz-Hodge decomposition is used [12]. The Helmholtz-Hodge decomposition theorem plays a fundamental role in the numerical approximation of physical models like the Navier-Stokes equations [28] [12]. It states that any vector field decomposes into the sum of a gradient vector field and a divergence-free one. More formally:

$$\mathbf{w} = \mathbf{u} + \nabla q \quad , \tag{3}$$

where $\mathbf{u}$ is divergent free, i.e. $\mathbf{u}$ satisfies (1), and ∇q is the gradient field of the scalar field q. Let us apply the gradient operator ∇ to both sides of (3):

$$\nabla \cdot \mathbf{w} = \nabla^2 q \quad . \tag{4}$$

This is a so-called Poisson equation which can be solved for the scalar field q with given boundary conditions. Once it is obtained, its gradient field can easily be subtracted from the velocity field $\mathbf{w}$ in order to get the divergent free $\mathbf{u}$ velocity field.

Introducing the projection operator $\mathcal{P}$ from [12], which maps $\mathbf{w}$ onto its divergence-free part $\mathbf{u}$, we have

$$\mathbf{u} = \mathcal{P}\mathbf{w} = \mathbf{w} - \nabla q \quad . \tag{5}$$

Applying this operator $\mathcal{P}$ to both sides of the incompressible Navier-Stokes equations (2), we get

$$\frac{\partial \mathbf{u}}{\partial t} = \mathcal{P}\left(-(\mathbf{u}\cdot\nabla)\mathbf{u} + \nu\nabla^{\mathbf{2}}\mathbf{u} + \mathbf{F}\right) \quad , \tag{6}$$

using the identities $\mathcal{P}\mathbf{u} = \mathbf{u}$ and $\mathcal{P}\nabla q = 0$.

This form (6) of the Navier-Stokes equations de-couples the pressure and expresses the momentum equation in terms of $\mathbf{u}$ alone. Specifically, the gradient part of the right-hand side in (6) represents the pressure gradient.

The velocity field is going to be used to move around a substance within the fluid, whereby the evolution of the substance is governed by the following transport equation:

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u}\cdot\nabla)\rho + \kappa\nabla^{\mathbf{2}}\rho + \mathbf{S} \quad , \tag{7}$$

where ρ is the scalar field of the density of the substance, κ is the diffusion constant and S is the scalar field of the amount of substance injected. With this model both the motion of the fluid and the related propagation of substances (like blood) can be handled.

3 Numerical Implementation

There are various ways of implementing the above model [18,29–31]. The differences are mainly in regard to possible improvements in either the visual appearance or computational efficiency. In our implementation we optimize the methodology for real-time visualization of bleeding in the uterine cavity. The general method is to iteratively solve all the terms in (6) and (7) sequentially. The corresponding loop is illustrated schematically in Fig. 1.

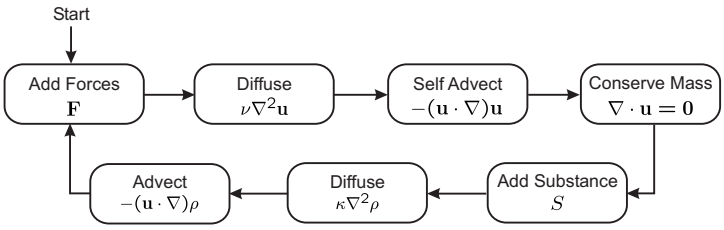


Fig. 1. Flowchart of the solution. Top line contains the evolution of the velocity field, Eqns. (1) and (6). Bottom line contains the evolution of the substance field, Eqn. (7).

In practice, each quantity is defined on a spatially discretized domain. In the first stage we implemented the model in 2D, hence our domain is a squared grid aligned with a Cartesian coordinate system. Instead of the uniform grid configuration considered in [16,30] we used the so-called staggered grid arrangement as in [15,18]. This means – as shown in Fig. 2 – that the density of the substance (ρ) and the external forces ($\mathbf{F}$) are defined at the center of each cell, while the velocity field is orthogonally separated into its scalar components (u,v) and defined on the faces of the grid cells. In the following, these velocity components are referenced locally, i.e. the neighboring velocities for cell (i, j) are indexed as $(i+\frac{1}{2}, j)$, $(i-\frac{1}{2}, j)$ and $(i, j+\frac{1}{2})$, $(i, j-\frac{1}{2})$. In accordance with the results in [18], our experiments also showed the visual benefits of less numerical dissipation by applying this staggered arrangement.

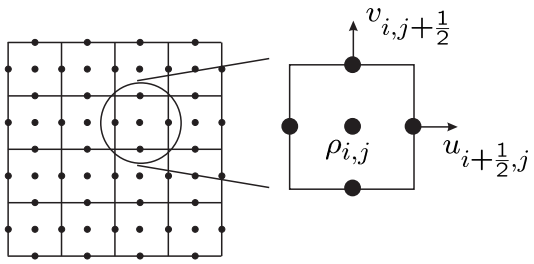


Fig. 2. Spatially discretized 2D staggered grid configuration

The simulation is advanced by updating the grid from a previous status over a predefined time-step Δt. First, the velocity components of the fluid are updated in four major steps. Step one is the addition of the external force field to the velocity field. The corresponding forces can describe e.g. pre-configured actions or more specifically, in our case, the influence caused by manipulating the liquid inflow and outflow during the surgical intervention. The forces are multiplied by the time-step, then averaged to the grid faces on which the velocities are defined and the corresponding scalar components are added to the velocities:

$$u_{i+\frac{1}{2},j} \; += \frac{(F^x_{i,j} + F^x_{i+1,j})}{2} \cdot \Delta t \quad , \quad v_{i,j+\frac{1}{2}} \; += \frac{(F^y_{i,j} + F^y_{i,j+1})}{2} \cdot \Delta t \quad , \qquad (8)$$

where F^x, F^y represent the x and y components of the external force.

The second step is the diffusion of the velocity field. It is solved in the same way as for the diffusion of the substance in step two of the density solver. However, instead of a diffusion rate κ, a viscosity ν is used, and each component of the velocity field is solved separately. For easier conceivability, the details of this approach are explored later as part of the density solver.

The third step is the self-advection. It can be interpreted as the velocity field moving itself. It is a non-linear problem, but Stam [16] showed how a stable solution, viable for computer graphics, can be obtained using the semi-Lagrangian integration scheme [17]. The motivation behind the semi-Lagrangian scheme is to alloy the regular arrangement of Eulerian schemes and the better stability of Lagrangian ones. This is done by moving the data relative to a spatially fixed grid and considering a different set of particles at each time-step. The actual set is chosen such that the particles arrive at the points of the regular Cartesian grid at the end of a time-step. In a less rigorous way, it basically means that each velocity component is traced back in time through the velocity field to find where it came from. This can be done using either a linear particle tracer or a second order Runge-Kutta method [32]. From our experiments the more elaborate Runge-Kutta approach did not provide sufficient improvements to justify the increase in computation time. At the earlier points, interpolation is needed to obtain the earlier velocities. This may be achieved via simple linear, or the so-called monotonic cubic interpolation method as proposed in [18]. Although the monotonic cubic interpolation scheme reduces the simulation speed considerably, the realism of the appearance of the fluid improved adequately (especially when used during the density solution step). Finally the calculated velocities are transported back to the origin of the back-tracing operation.

The fourth step is mass conservation according to (1). It is a physical constraint that every fluid has to conserve mass, meaning that the flow into a cell should be equal to the flow coming out of the cell. After the preceding steps (adding external forces, viscous diffusion, self-advection), this condition is not fulfilled. Mass conservation is important also from the computer graphics point of view, as it forces the flow to swirl and have appealing vortices enhancing its realism.

The application of the Helmholtz-Hodge decomposition (4) results in the following equation in the discretized domain:

$$\frac{(u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}) + (v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}})}{h} = \frac{p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{i,j}}{h^2}, \quad (9)$$

which incorporates the task of solving a sparse linear equation system for the unknown p values on the right-hand side.

There are diverse methods for computing a solution [16], but we found that the simple iterative Gauss-Seidel relaxation solver [30] was the fastest while still giving visually satisfactory results. After the p values are obtained the velocities can be updated according to (3):

$$u_{i+\frac{1}{2},j} \; -= \; \frac{p_{i+1,j} - p_{i,j}}{h} \quad , \quad v_{i,j+\frac{1}{2}} \; -= \; \frac{p_{i,j+1} - p_{i,j}}{h} \quad . \tag{10}$$

To minimize numerical dissipation and maintain the realism of the flow, we also added vorticity confinement [18] which enhanced these vortices through an additional force component.

When solving for the velocity field, proper boundary conditions have to be set. Primarily we are concerned with boundary conditions corresponding to a stationary wall, with inflow and outflow occurring along portions of the boundary. Foster and Metaxas [13] discuss how to achieve these conditions on a staggered grid.

Let us now move ahead to the second major part of the model. The density equation (7) governs the evolution of the substance injected into the fluid. The method of solution is similar to the one introduced previously. We start with an initial density field and solve the terms in (7) sequentially. The first step is the addition of the density values of an external source (S), which are multiplied by the time-step and added to the density field. In our case, external source is e.g. the spurting blood from vessel on the wall of the uterus.

$$\rho_{i,j} \; += S_{i,j} \cdot \Delta t \quad . \tag{11}$$

The second step is to solve for the diffusion of the density. Due to the discretized domain, the second spatial derivative is approximated by finite differences. Taking the second difference in both direction (2D model) we get:

$$\frac{\partial \rho}{\partial t} = \kappa \nabla^2 \rho \approx \kappa \frac{\rho_{i-1,j} + \rho_{i+1,j} + \rho_{i,j-1} + \rho_{i,j+1} - 4\rho_{i,j}}{h^2} \quad . \tag{12}$$

The straightforward implementation of a diffusion task like (12) can cause stability problems during the simulation if the diffusion rate or the time-step is set too large or the grid spacing is too small. To overcome this problem Stam [16] proposed an implicit, thus stable solution method for the diffusion term: find the density field which would diffuse backward in time to give the densities we started with. Formally, we instead solve the following equation for the cell i, j:

$$\rho^t_{i,j} = \rho^{t+\Delta t}_{i,j} - \frac{\kappa \Delta t}{h^2} \left(\rho^{t+\Delta t}_{i-1,j} + \rho^{t+\Delta t}_{i+1,j} + \rho^{t+\Delta t}_{i,j-1} + \rho^{t+\Delta t}_{i,j+1} - 4\rho^{t+\Delta t}_{i,j} \right) \quad . \tag{13}$$

In this way we again face the problem of solving a system of linear equations, since the terms $\rho^{t+\Delta t}$ on the right hand side are unknown. Due to the reasons mentioned above, the Gauss-Seidel relaxation [30] was used as well, but here 4-5 iterations are enough for a realistic impression. Coming back to the previously mentioned viscous diffusion of the velocity, exactly the same idea can be applied there but for each velocity component the diffusion rate κ is replaced by the viscosity ν.

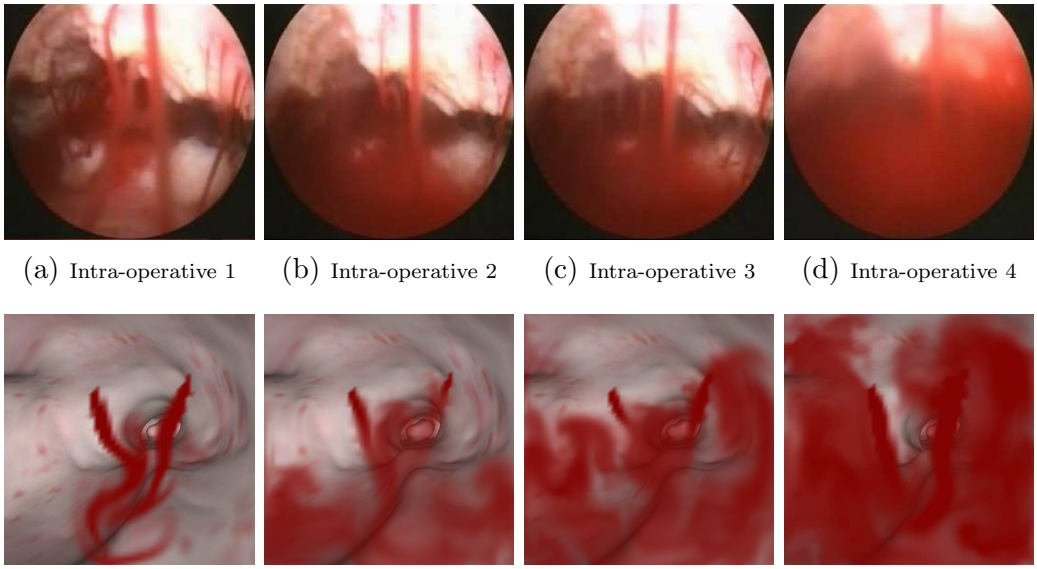
There is one more term in (7) which has to be solved. It states that the density of the substance should follow and be advected by the – previously updated – velocity field. In order to preserve the stability of the solver, here again the semi-Lagrangian integration scheme is used [16]. We trace back the midpoints of each cell through the velocity field. Then the new density values are interpolated at these points and their values are transferred to the grid centers where we started at. For our purposes we used the monotonic cubic interpolation scheme [18] due to its advantageous properties in providing a more realistic appearance of the substance.

3.1 2D implementation of the model

The 2D version of bleeding was implemented for the hysteroscopy simulator. Figure 3 gives a visual comparison between real bleeding recorded during hysteroscopic surgery, when fluid inflow and outflow were stopped, and our simulated version. The 2D plane displaying the bleeding was placed between the scene (uterine cavity) and the camera (hysteroscope). In this case the boundary conditions on all the four edges of the 2D plane were set as a wall, which results in a closed domain. This corresponds to a clinical setup when the inflow and outflow valves are closed. The visualization was done in OpenGL and the best rendering performance was achieved if the blood flow was considered as a dynamic texture, blended with the background scene. The synthesis gave high visual fidelity, but what is more important, real-time frame rate was achieved with up to 60 frames/sec on a 64×64 grid or 15 frames/sec on a 128×128 grid on a Sun Ultra Sparc III processor.

3.2 3D extension of the model

The approach to merge a 3D surgical scene with a semi-transparent 2D texture plane, describing the visual appearance of the fluid environment, provides acceptable results if the endoscopic camera is kept statically in a corresponding predefined position. During interventions, however, the endoscope position is constantly adjusted, which completely ruins the visual realism of this simplified solution. Moreover we need to accommodate for the realistic in- and



(a) Intra-operative 1 (b) Intra-operative 2 (c) Intra-operative 3 (d) Intra-operative 4

(e) Simulated 1 (f) Simulated 2 (g) Simulated 3 (h) Simulated 4

Fig. 3. Top row: original images recorded during hysteroscopic surgery in the case of bleeding when fluid inflow and outflow were stopped. Bottom row: frames from the real-time 2D synthesis. See Movie 1 and 2.

outflow of the distension liquid as well, which cannot be fulfilled with a 2D simulation either. In order to resolve these issues, the fluid simulation has to be performed in a 3D environment.

The model itself (introduced in Section 2.2) generalizes well to this case and even the extension of the implementation to 3D can be considered as rather straightforward. First of all, the staggered grid changes to cubic cells as illustrated in Fig. 4 and therefore each location in the grid has six neighbors. This

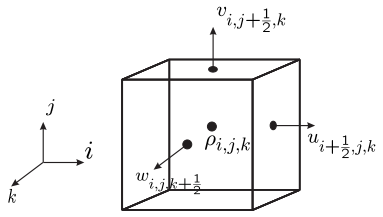


Fig. 4. Spatially discretized 3D staggered grid cell.

leads to additional terms in the numerical approximations of the differential operators used by the solver. The external forces also have three components (F^x, F^y, F^z) resulting in the additional z coordinate term to be added to the corresponding velocities:

$$w_{i,j,k+\frac{1}{2}} \; += \frac{(F^z_{i,j,k} + F^z_{i,j,k+1})}{2} \cdot \Delta t \quad . \tag{14}$$

The central equation in the mass-conservation step (9) has to be expanded

with the terms responsible for the additional spatial dimension:

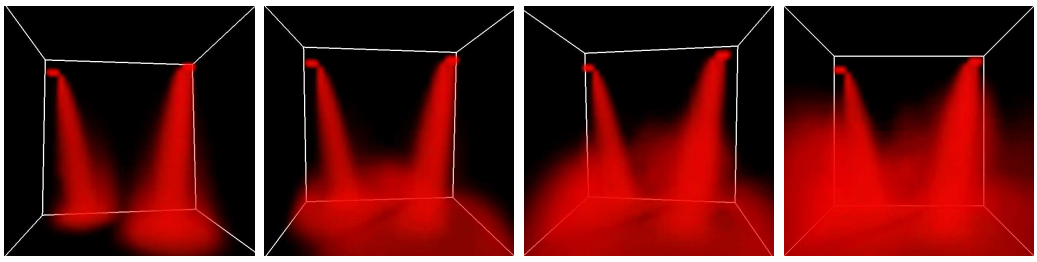
$$\frac{(u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}) + (v_{i,j+\frac{1}{2},k} - v_{i,j-\frac{1}{2},k}) + (w_{i,j,k+\frac{1}{2}} - w_{i,j,k-\frac{1}{2}})}{h} = \frac{p_{i+1,j,k} + p_{i-1,j,k} + p_{i,j+1,k} + p_{i,j-1,k} + p_{i,j,k+1} + p_{i,j,k-1} - 6p_{i,j,k}}{h^2} \quad . \tag{15}$$

Analogously, the subsequent update step shown in (10), has to incorporate the third velocity component:

$$w_{i,j,k+\frac{1}{2}} \; -= \; \frac{p_{i,j,k+1} - p_{i,j,k}}{h} \quad . \tag{16}$$

Similar changes apply to (13). Obviously, the calculation domain for the numerical solver grows by the third power of the resolution leading to very large problems. The real-time requirement forces us to choose a moderate resolution for the grid, while on the other hand a higher resolution would lead to a more satisfactory visual appearance of the blood flow.

In Fig. 5 the results of the 3D bleeding simulation are shown with changing camera position. In the results shown, the grid size was set to 32x32x32 and only linear interpolation was used. The mass field representing the blood was rendered as a dynamic 3D texture with OpenGL. The three-dimensional texture approach is a straightforward way to visualize volumetric data. The images clearly show that the result has an overly smoky appearance which is clearly visually different from the expected behavior of blood dispersed in liquid. This clearly undesirable effect is caused by using the semi-Lagrangian method which results in significant numerical dissipation.



(a) Frame1 (b) Frame2 (c) Frame3 (d) Frame4

Fig. 5. 3D simulation of bleeding with pure semi-Lagrangian method. Notice the change of the camera position during the simulation. See Movie 3.

3.3 Marker particles

This deficiency suggests the reconsideration of the work of Foster and Metaxas [13] and Foster and Fedkiw [20], where they simulated the liquid flow using

passive particles. Their approach makes it possible to keep the favorable unconditionally stable semi-Lagrangian method for solving the velocity field, while efficiently correcting the inherent numerical dissipation with marker particles. This also allows the velocity field to be calculated on a fairly coarse grid, while at the same time the marker particles can delineate the fluid position on a completely independent finer grid.

In [13] and [20] the key motivation is to precisely track the (free) fluid surface at the fluid-air interface. That is why in these works additional, computationally more demanding methods are involved in tracking the fluid position: free surface particles and height field in [13] and level sets in [20]. In our particular case, namely bleeding in the distension liquid, we need to consider a fluid-fluid interface instead of a fluid-air one. This distinction makes it possible to use massless particles without tracking a precise interface, which would be computationally too expensive for real-time simulation. According to our results, the slight false diffusion (noise) occurring at the blood-liquid interface is visually acceptable.

The marker particles are introduced at the bleeding points, and convected with the local velocity. The position of a particle is updated according to $pos^{t+\Delta t} = pos^t + \mathbf{v} \cdot \Delta t$, where the velocity $\mathbf{v}$ is determined by linear interpolation over the nearest cell velocities.

Although the semi-Lagrangian approach applied in the velocity advection step allows for larger time steps, this is only at the cost of increased dissipation. In order to preserve the liquid-like visual effect, the time-step used in the explicit time-marching update of the particles needs to be limited according to the CFL condition [13]. This means that a given marker particle cannot move more than one-cell distance within a time-step, i.e $\Delta t_{particle} < \Delta x/|\mathbf{v}|$. In accordance with the remarks in [13], we have chosen the velocity field time-step four times bigger than the one given by the CFL criterion for $\Delta t_{particle}$. Hence the marker particles are advected by means of four consecutive steps before they are visualized. In regard to the velocity field, we use the same model as introduced previously, except that the viscous term is solved using central differencing as described by Foster and Metaxas [13]. This gives more accurate results as opposed to the ones given by the implicit diffusion technique.

3.4 Parallelization

The implementation of the model outlined above in three-dimensions, though visually already appealing, did not fully satisfy the real-time condition for our application (see Section 4. for the results). One reasonable way to speed up the simulation is to consider parallelization of the implementation.

There exists a wide body of work on the parallelization of CFD solvers. They study strategies starting from grid generation and domain decomposition to as far as linear algebra algorithms [33]. Due to the fact that the model in our bleeding simulation is a rather specialized and simplified version of CFD (being motivated by computer graphics applications rather than physically accurate simulations) it cannot simply be inserted into an existing package or library. Nevertheless, the general ideas used in the parallelization strategies are worth considering.

Regarding the velocity solver, the local nature of the involved operators, strongly motivates the tessellation of the spatial domain into partitions, which can be processed in a parallel way. In our case the straightforward partitioning is to cut the domain into horizontal layers as shown in Fig. 6. Each partition contains a predefined number of layers from the data array. To reach a reasonable load balance among the available processors, the data is equally distributed among them.

The multi-threaded environment we used was developed in-house based on the Posix library [34]. This environment enables the threads to have convenient access to a shared memory area. Concurrent reading from this memory area does not require special action, thus prevents us from time consuming synchronization among the reading threads. More importantly it also relieves us from the challenge of precise domain decomposition which would be necessary for the Lagrangian technique we use. What we have to guarantee is that the threads do not write to the same memory area at the same time. This is easily ensured since the general configuration during the velocity field calculation is such that the threads read from data arrays belonging to time-step t, and write to a memory area corresponding to time-step $t + \Delta t$ and moreover only to the partitions which exclusively belong to them (see the upper part of Fig. 6). When a consistent state of a shared memory array is required (e.g. before calculation of the viscous terms, between the consecutive iterative steps in the mass conservation procedure, or before the particles are advected) the threads have to be synchronized.

An additional possibility for parallelization of the solver beyond the domain decomposition is to let the calculations of the three velocity components be performed simultaneously. In our model, such a case occurs at the advection step for the u, v and w component of the velocity, and similarly for the calculation of the three components of the viscous term. Although occasionally these methods read from the same arrays, they write their results to separate memory blocks. The general method of parallelization of the velocity solver is depicted in Fig. 6.

The processing step of the marker particles offers the possibility for parallelization as well. One feasible way is to split up the list containing the particles

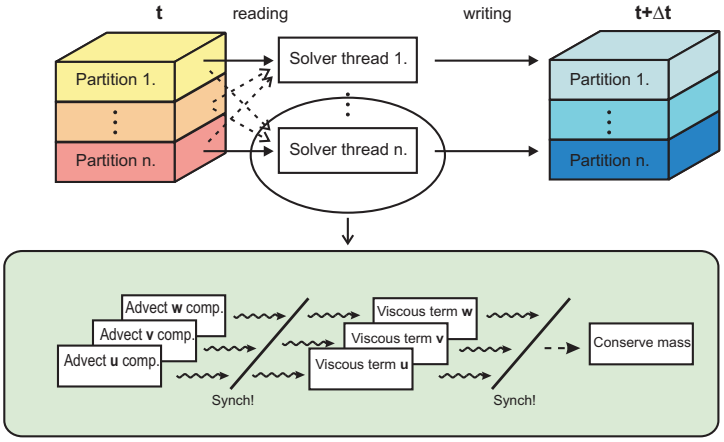
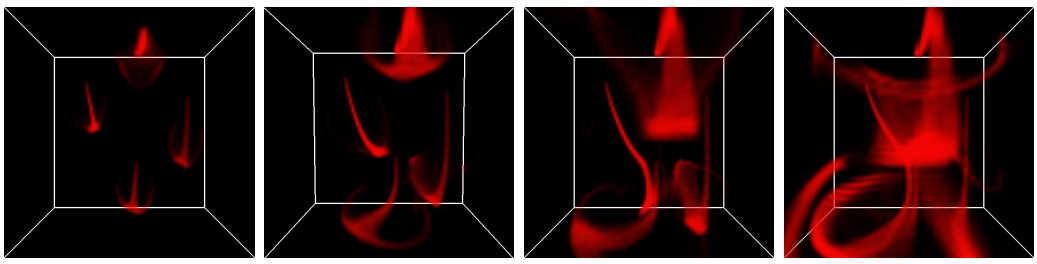


Fig. 6. The setup of parallelization.

among the threads. Since the particles are completely independent from each other, these fragments of the list can be processed asynchronously. Furthermore, similarly to the velocity solver, the update of the x, y and z components of the particle positions can be also performed asynchronously.

4 Results

In Fig. 7 the results of the 3D bleeding simulation are shown, when the camera changes its position. In this case, the solver grid size was $22 \times 22 \times 22$, and the maximum number of marker particles was 200 000. The calculation time for the velocity field took 18ms, while the update of the 200 000 particles was approximately 250ms. These values correspond to a single threaded computation.

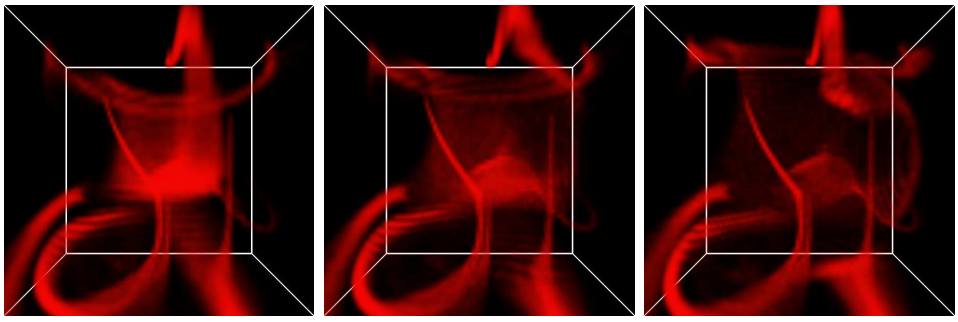


(a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Frame 4

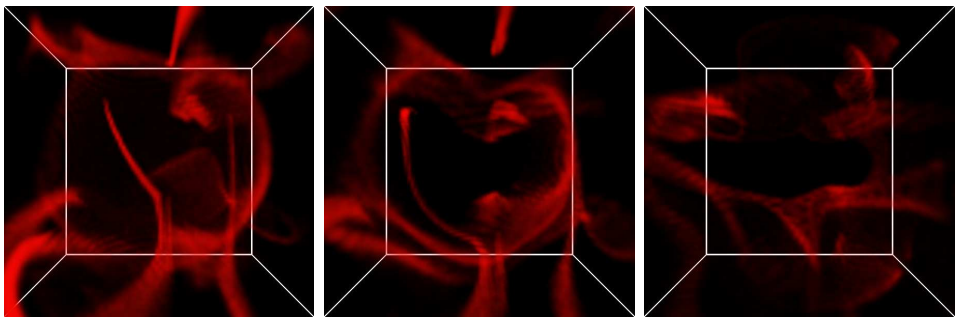
Fig. 7. Simulated bleeding for virtual hysteroscopy in 3D with the marker particle approach. See Movie 4.

The synthesis is also responding to changes in the virtual environment. One essential aspect in hysteroscopy simulation is the accounting for the effect of inflow and outflow fluid management. The surgeon can clear up the view by opening the in- and outflow valves on the tool. In Fig. 8. we show what happens to the scene when both flows were switched on in frame 2. As in the realistic

case, the clear liquid is injected at the tip of the hull close to the camera, while the suction is induced through the holes on the hull behind the camera. We account for these with properly set boundary conditions. In the case of inflow, constant velocity is applied to the grid cells belonging to a circular patch around the center of the front wall. The direction of this velocity is towards the rear wall. For outflow, we allow the velocities of the grid cells within a ring around the inflow section of the front wall to exit without constraint. If a marker particle has left the domain it is simply deleted.



(a) Frame 1 (b) Frame 2 (c) Frame 3



(d) Frame 4 (e) Frame 5 (f) Frame 6

Fig. 8. Interacting with the environment. Inflow and outflow were switched on in Frame 2. See Movie 4.

We used the 3D texturing feature of OpenGL in order to visualize the flock of marker particles. For every frame a three-dimensional histogram is established from the actual standings of the particles and this histogram is treated as volume data. The bin size of the histogram determines the voxel size of the 3D texture and hence the resolution of the blood flow. The weight of a given bin in the histogram represents the alpha value for that voxel. Obviously, opacity increases with the number of particles that are present in a voxel. The 3D texture size used for the simulations was 64^3.

In Table 1 and 2 the results of the parallel implementation of the 3D solver and marker update are listed. In order to show the performance of the parallelization the speedup (S) and efficiency (ϵ) are also indicated as measures. The speedup is the ratio of the execution time on a single processor (τ_1) to the time on n processors (τ_n). The time τ_n incorporates the execution time of

Table 1
Average calculation time in ms (τ_n), speedup (S) and efficiency (ϵ) in percentage for the 3D numerical solver

	Grid size: 22 x 22 x 22			Grid size: 40 x 40 x 40		
# threads	τ_n	S	ϵ	τ_n	S	ϵ
1	15.4	–	–	89	–	–
2	10.5	1.47	73.3	50.4	1.77	88.3
3	9.1	1.69	56.4	38	2.34	78.1

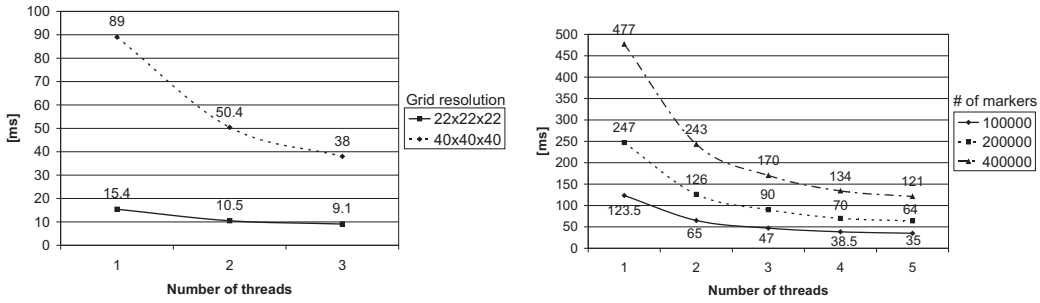
Table 2
Average calculation time in ms (τ_n), speedup (S) and efficiency (ϵ) in percentage for the marker update

	# of markers: 100 000			# of markers: 200 000			# of markers: 400 000		
# threads	τ_n	S	ϵ	τ_n	S	ϵ	τ_n	S	ϵ
1	123.5	–	–	247	–	–	477	–	–
2	65	1.9	95	126	1.96	98	243	1.96	98.1
3	47	2.63	87.6	90	2.7	91.5	170	2.81	93.5
4	38.5	3.21	80.2	70	3.5	88.2	134	3.56	89
5	35	3.53	70.6	64	3.9	77.2	121	3.94	78.8

the parallel and serial parts of the calculations including the parallelization overhead. This overhead is due to running the algorithm on parallel processors and can result from the hardware, operating system or the algorithm itself. The efficiency is the ratio of the real speedup to the ideal one, when the execution time is not affected by parallelization overhead. In Fig. 9 the average calculation times are plotted. The measurements were taken on a Sun Fire 6800 HPC server, with 12 Ultra Sparc III processors. The values listed for the marker update correspond to the four-step advection and include the time needed for establishing the three-dimensional histogram. If an alternative rendering method is used, where such a construction of a volumetric data is unnecessary, about 10% of the computation time can be spared.

As one can observe from the data, the calculation time decreases with the number of threads, however the speedup does not scale in a linear fashion in conjunction with the number of threads. In any case the efficiency increases with the problem size. Our investigations suggest that the suboptimal scaling stems from the memory-intensive nature of the algorithm. As such, generic cache coherency contention and false sharing can occur very easily and results in poor scaling. These phenomena can arise in all cache-based, shared memory multiprocessor systems. The correction for cache coherency contention can only be done by thoroughly considering the architecture of the underlying hardware and applying appropriate tuning techniques using appropriate data array representation and algorithmic modifications.

Unfortunately, dynamically changing and swapping the content of the 3D tex-



(a) Solver (b) Marker update

Fig. 9. Average calculation times in ms.

ture buffer in OpenGL substantially degrades its performance irrespective of the resolution of the texture. This strains the real-time rendering capability of our simulated blood flow and explicitly outweighs the performance increase gained by the parallelized calculations. The visualization system we used was a Sun Fire V880 with a XVR-4000 graphics accelerator, where the dynamic update of the texture limited the frame rate to 6-7 fps, causing a bottleneck in the performance of the simulation.

5 Conclusion

We have designed a state-of-the-art graphic flow simulation offering realistic visualization of blood flow for virtual surgery. The work was focused on the special requirements of a bleeding simulation for virtual hysteroscopy, where the blood is present in the uterine cavity distended by liquid. We modelled this phenomenon as an interaction between two inert fluids, for which no chemical processes were involved. Furthermore, no fluid to air interface was considered, only basic fluid to object interaction (i.e. at the walls).

The quantitative validation of the development techniques is by far not straightforward. It would be very difficult, if not impossible to collect experimental data about 3D flow conditions and blood dispersion in a reasonably realistic clinical environment. Comparisons with results of commercial numerical solvers, which allow us to perform the calculations under less restrictive assumptions while abandoning real-time requirements would be possible, but it is still uncertain how closely even such simulations can approximate real life conditions. Most importantly it is not clear, how to convert the certainly existing numerical deviations in the flow field and blood concentration to a reliable measure allowing us to evaluate the visual quality of the simulation.

We therefore propose and plan to build up a validation environment directly addressing the clinical target of the simulation by measuring the effects of using

the system on the actual working performance of the surgeon. In other words, we are interested in a performance measure which shows how the learning-curve of a surgeon is influenced by using the simulator. We are currently setting up targeted clinical studies, which will allow to quantitatively assess selected surgical skills and to observe their dependence on repetitive training using the simulator unit under development.

For the time being we can evaluate the success of our research effort only in a subjective manner based on three principal factors. One of these is the ability to achieve real-time simulation along with the possibility of interaction with the virtual environment. Secondly, the appearance of the synthesized bleeding should be visually convincing and appear to be reasonably realistic for the surgeon being trained. Finally, the simulated bleeding has to pose the same challenges in the virtual environment as the clinician faces during real surgery.

The results presented in this paper show that the method, originating from the Navier-Stokes fluid equations, can be solved in real-time with satisfactory frame rate in 2D, while currently available computational resources allow only moderate frame rate in the desired 3D case. Parallelization offers an appealing way to speed up these calculations. Graphical fluid simulation allows the bleeding to be interactive and responsive to any changes made to the virtual environment in real-time. The visual fidelity originally had an overly smoky appearance due to the applied semi-Lagrangian technique, but this was significantly improved with marker particles leading to a more liquid-like quality. Rendering the blood flow with dynamic 3D texture offers convincing visual realism but currently at the expense of rendering performance.

The method discussed in the paper can be directly applied to simulating bleeding in connection with any clinical procedure where the distension of the operation area is accomplished by means of fluid, such as in urology during transurethral resection of the prostate (TURP simulator). The proposed procedure can also be used under more general surgical conditions for predicting the appearance of bleeding after appropriate adjustments allowing the proper handling of the liquid-air interface as described in [13] [20].

Acknowledgment

This research has been supported by the NCCR CO-ME of the Swiss National Science Foundation.

References

[1] R. Sierra, G. Székely, M. Bajka, Generation of pathologies for surgical training simulators, in: Procs. of the 5th International Conference on Medical Image Computing and Computer-Assisted Intervention, Vol. 2, Springer Verlag, 2002, pp. 202–210.

[2] O. R. Köchli (Ed.), Hysteroscopy : state of the art, Vol. 20 of Contributions to gynecology and obstetrics, Basel : Karger, 2000.

[3] K. Montgomery, C. Bruyns, S. Wildermuth, L. Heinrichs, C. Hasser, S. Ozenne, D. Bailey, Surgical simulator for operative hysteroscopy, IEEE Visualization 2001 (2001) 14–17.

[4] W. K. Müller-Wittig, U. Bockholt, J. L. L. Arcos, P. Oppelt, J. Stähler, G. Voss, Lahystotrain - VR-based intelligent training environment for laparoscopy and hysteroscopy, in: Virtual Reality International Conference (VRIC), Vol. 3, 2001, pp. 225–233.

[5] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, M. Werman, Texture mixing and texture movie synthesis using statistical learning, IEEE Transactions on Visualization and Computer Graphics 7 (2) (2001) 120–135.

[6] G. Doretto, S. Soatto, Editable dynamic textures, in: ACM SIGGRAPH 2002 Sketches and Applications, San Antonio, Texas, 2002.

[7] P. Oppenheimer, A. Gupta, S. Weghorst, R. Sweet, J. Porter, The representation of blood flow in endourologic surgical simulations, in: Proceedings of Medicine Meets Virtual Reality, 2001, pp. 365–371.

[8] K. Perlin, F. Neyret, Flow noise, Siggraph Technical Sketches and Applications (2001) 187.

[9] A. Schödl, I. Essa, Controlled animation of video sprites, in: First ACM Symposium on Computer Animation in Conjunction with ACM SIGGRAPH 2002, San Antonio, TX, USA, 2002.

[10] M. Szummer, Temporal texture modeling, Master's thesis, MIT Media Lab Perceptual Computing (1995).

[11] L. Raghupathi, Simulation of bleeding and other visual effects for virtual laparoscopic surgery, Master's thesis, University of Texas at Arlington (December 2002).

[12] A. Chorin, J. E. Marsden (Eds.), A Mathematical Introduction to Fluid Mechanics, 2nd Edition, Springer-Verlag, New York, 1990.

[13] N. Foster, D. Metaxas, Realistic animation of liquids, Graphical Models and Image Processing 58 (5) (1996) 471–483.

[14] N. Foster, D. Metaxas, Controlling fluid animation, in: Proceedings CGI '97, 1997, pp. 178–188.

[15] N. Foster, D. Metaxas, Modeling the motion of a hot, turbulent gas, in: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1997, pp. 181–188.

[16] J. Stam, Stable fluids, in: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1999, pp. 121–128.

[17] A. Staniforth, J. Cote, Semi-Lagrangian integration schemes for athmospheric models: A review., in: Monthly Weather Review, Vol. 119, 1991, pp. 2206–2223.

[18] R. Fedkiw, J. Stam, H. W. Jensen, Visual simulation of smoke, in: Procs. of ACM SIGGRAPH 2001, ACM Press, 2001, pp. 15–22.

[19] J. Steinhoff, D. Underhill, Modification of the euler equations for "vorticity confinement": Application to the computation of interacting vortex rings., Physics of Fluids 6 (8) (1994) 2738–2744.

[20] N. Foster, R. Fedkiw, Practical animations of liquids, in: E. Fiume (Ed.), SIGGRAPH Computer Graphics Proceedings, 2001, pp. 23–30.

[21] D. Enright, S. Marschner, R. Fedkiw, Animation and rendering of complex water surfaces, in: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM Press, 2002, pp. 736–744.

[22] B. E. Feldman, J. F. O'Brien, O. Arikan, Animating suspended particle explosions, ACM Trans. Graph. 22 (3) (2003) 708–715.

[23] N. Rasmussen, D. Q. Nguyen, W. Geiger, R. Fedkiw, Smoke simulation for large scale phenomena, ACM Trans. Graph. 22 (3) (2003) 703–707.

[24] R. Gingold, J. Monaghan, Smoothed particle hydrodynamics: theory and application to non-spherical stars, Mon. Not. R. astr. Soc. 181 (1977) 375–389.

[25] M. Desbrun, M.-P. Gascuel, Smoothed particles : A new paradigm for animating highly deformable bodies, in: Computer Animation and Simulation, 1996, pp. 61–76.

[26] M. Müller, D. Charypar, M. Gross, Particle-based fluid simulation for interactive applications, in: Proceedings of the 2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation, Eurographics Association, 2003, pp. 154–159.

[27] J. D. Anderson, Jr., Computational Fluid Dynamics: The Basics with Applications, McGraw-Hill, Inc., New York, 1995.

[28] F. M. Denaro, On the application of the Helmholtz-Hodge decomposition in projection methods for incompressible flows with general boundary conditions, International Journal for Numerical Methods in Fluids 43 (2003) 43–69.

[29] J. Stam, A simple fluid solver based on the FFT, Journal of Graphics Tools 6 (2) (2001) 43–52.

[30] J. Stam, Real-time fluid dynamics for games, in: Proceedings of the Game Developer Conference, 2003.

[31] F. Losasso, F. Gibou, R. Fedkiw, Simulating water and smoke with an octree data structure, ACM Trans. Graph. 23 (3) (2004) 457–462.

[32] W. H. Press, B. P. Flannery, S. Teukolsky, W. T. Vetterling (Eds.), Numerical Recipes in C. The art of scientific computing, Cambridge University Press, Cambridge, 1988.

[33] P. Wilders, A. Ecer, J. Periaux, N. Satofuka (Eds.), Parallel CFD 2001: Parallel Computational Fluid Dynamics - Practice and Theory, Elsevier Publishing Co., Egmond aan Zee, The Netherlands, 2002.

[34] IEEE Std 1003.1, The Open Group Technical Standard Base Specification, Issue 6 (2004).