Published in final edited form as:

Int J Med Robot. 2010 December ; 6(4): 431–443. doi:10.1002/rcs.353.

GPU-based Efficient Realistic Techniques for Bleeding and Smoke Generation in Surgical Simulators

Tansel Halic, M. Sc.[Ph.D. student],

Department of Mechanical, Aerospace and Nuclear Engineering, at Rensselaer Polytechnic Institute, Troy, USA

Ganesh Sankaranarayanan, Ph.D.[Post Doctoral Research Associate], and

Department of Mechanical, Aerospace and Nuclear Engineering, at Rensselaer Polytechnic Institute, Troy, USA

Suvranu De, Sc.D.[Associate Professor]

Department of Mechanical, Aerospace and Nuclear Engineering, at Rensselaer Polytechnic Institute, Troy, USA

Abstract

Background—In actual surgery, smoke and bleeding due to cautery processes, provide important visual cues to the surgeon which have been proposed as factors in surgical skill assessment. While several virtual reality (VR)-based surgical simulators have incorporated effects of bleeding and smoke generation, they are not realistic due to the requirement of real time performance. To be interactive, visual update must be performed at least 30 Hz and haptic (touch) information must be refreshed at 1 kHz. Simulation of smoke and bleeding is, therefore, either ignored or simulated using highly simplified techniques since other computationally intensive processes compete for the available CPU resources.

Methods—In this work, we develop a novel low-cost method to generate realistic bleeding and smoke in VR-based surgical simulators which outsources the computations to the graphical processing unit (GPU), thus freeing up the CPU for other time-critical tasks. This method is independent of the complexity of the organ models in the virtual environment. User studies were performed using 20 subjects to determine the visual quality of the simulations compared to real surgical videos.

Results—The smoke and bleeding simulation were implemented as part of a Laparoscopic Adjustable Gastric Banding (LAGB) simulator. For the bleeding simulation, the original implementation using the shader did not incur in noticeable overhead. However, for smoke generation, an I/O (Input/Output) bottleneck was observed and two different methods were developed to overcome this limitation. Based on our benchmark results, a buffered approach performed better than a pipelined approach and could support up to 15 video streams in real time. Human subject studies showed that the visual realism of the simulations were as good as in real surgery (median rating of 4 on a 5-point Likert scale).

Conclusions—Based on the performance results and subject study, both bleeding and smoke simulations were concluded to be efficient, highly realistic and well suited in VR-based surgical simulators.

Keywords

Computer Graphics; Bleeding and Smoke Simulation; Physics-based Simulation; Minimally Invasive Surgery; Haptics; Rendering; Virtual Reality; Surgical Simulation

Introduction

In laparoscopic surgery, a cautery tool is used to divide tissue with minimal blood loss. As tissue is burnt, copious smoke is generated. Sometimes, the tissue may also bleed if coagulation is not efficient. Both smoke and bleeding provide vital visual cues to the surgeon. Surgical errors may lead to excessive bleeding and possibly even death. Prior studies (1)(2) indicate that surgical smoke produced during surgery could obscure the surgeon's field of view and is a potential risk that could lead to surgical errors with severe risk to patient safety. Smoke concentration and increased exposure of the tissues to smoke may elevate the risk of chemical poisoning. Using visual cues from smoke, overheating of the tissue, which may lead to excessive scarring, may be prevented.

Both blood and smoke, therefore, provide important visual cues which a surgeon-in-training must be able to recognize to make time-critical decisions. Hence recent surgical literature prescribes the use of such cues as important assessment factors in surgical education. In (3) (4), e.g., control of bleeding was measured as a parameter while assessing the surgical skill of medical trainees and it was found to correlate well with skill level. The volume of the smoke is correlated to the cutting time (2). Too much smoke is considered detrimental to the patient. These and other studies establish the need to include bleeding and smoke generation as features to be integrated into surgical simulators for skill training.

Virtual reality (VR)-based surgical simulators are being increasingly adopted for training surgeons and residents in many medical schools and hospitals across the USA and many other countries in the world (5). These trainers have the potential for training with unlimited practice material in a safe environment with real time objective assessment without the need for a proctor. With ever expanding computational power and advancements in graphical rendering hardware, the realism of the VR-based surgical simulators has improved drastically (6).

The need to incorporate the effects of bleeding and smoke in VR-based surgical simulators has been well-recognized and several previously developed simulators (7–20) include these effects. In a cricothyroidotomy simulator developed by Bhasin et al. (7), bleeding simulation was implemented to add realism when cutting tissues with a scalpel. Although their method had a visually appealing result, it requires the bleeding image to be updated in every CPU cycle, which could be prohibitive for more complex procedural simulations. A similar approach for simulating bleeding in a laparoscopic surgery of the liver was used in (8). The visual realism was not enhanced compared to previous work but they introduced more control such as gravity, surface friction for blood flow. However, the authors noted performance bottlenecks in rendering complex bleeding scenarios. The work in (9) focused on an image-based approach to simulate blood flow in a virtual prostate surgery simulator. Bleeding during the procedure was simulated with pre-recorded movies of the diffusion of a red dye in a fluid, superimposed onto the camera view of the simulator. Though it provided realism to the simulator, as we will discuss later, this technique cannot be generalized to all types of bleeding experienced in a surgery. The authors in (10) developed a fast technique to simulate blood stream and blood flow for an orthopedic surgery simulator. They simulated blood flow from the vessels during the incision using a specialized Physics Processing Unit (PPU) hardware and a GPU for rendering bleeding surfaces. However, these specialized

hardware based method did not scale well due to the limited data transfer rates. Moreover, the end of support for specialized PPU hardware limited the applicability of this technique. In a virtual hysteroscopy surgery simulator (11), blood flow in a uterine cavity was simulated in 2D. Even though the bleeding simulation was realistic, the 2D approach was limited to a fluid environment. In a temporal bone dissection surgery simulator (12), both bleeding and irrigation with water was simulated in 2D using GPU based technique. The simulator demonstrated satisfactory results from a user study. The major drawback of this method is that it cannot simulate 3D effects. Simulation of both smoke and bleeding in an endoscopic simulator was implemented using animated images (13). In a minimally invasive surgical simulator (21), smoke arising during coagulation was simulated using a set of prerecorded images. Though a smoke like effect was created, their method cannot add variations in the simulated smoke. Textured particle systems (22) were used to generate smoke effects in (23) and (14). However, the disadvantage of this method is that the smoke looks rather discrete when the camera is zoomed in.

The Computer Graphics community has developed strikingly realistic techniques of animating fluid flows. One of the most notable efforts was by Stam and Fiume (24) to simulate smoke, steam or liquid by solving Navier-stokes equation within a 3D grid. Another approach by Fedkiw et al. (25) used vorticity confinement to capture the small features of the fluid and rendered the smoke with photon mapping technique for higher realism. In the work by Kim et al. (26), simulation of fluids at multiple levels of details was achieved using wavelets. However, none of these methods are suitable for real time environments as they are intended for animations in which each frame is computed over many hours using high performance computers.

VR-based surgical simulators must be *interactive*, i.e., the computations must be performed in *real time*. For haptic (touch)-enabled systems, response forces to the users must be updated at 1 KHz while visual updates must be maintained at 30 Hz for monocular and 60 Hz for stereo rendering. This is a challenging task as a fully functional surgical simulator typically integrates multiple tasks concurrently including realistic rendering of the surgical scene, collision detection, computation of tissue response, accurate tool-tissue interactions, haptic feedback. Incorporating all of these requires significant computational power. Since the various components of the surgical simulation system run at different frequencies, they must also be synchronized for data integrity and consistency.

Real time techniques for the generation of bleeding and smoke effects may be categorized into three major groups: non-physical, semi-physical and physical. The non-physical approaches are based on rendering techniques where the focus is to create visually appealing rather than physically accurate results. These include techniques such as dynamic-textures (15) (13), 3D textures (14), gravity maps and height maps (27), procedural-textures and billboards (24). The semi-physical approaches (16)(17)(28)(29) usually consist of either simplifying the physics by projecting 3D equations onto a 2D grid (16) or simplifying the underlying equations (17). Techniques based on particle system following Newtonian dynamics may also be grouped in this category (28). The physical approaches offer the most accurate simulation of the underlying physics. These include techniques based on finite differences (12) (11) (30) or smoothed particle hydrodynamics (SPH) (31)(18). These methods are, however, computationally very demanding and the cost increases dramatically with increase in grid size or the number of particles. Secondly, the additional burden for collision detection/response during the tool tissue interactions require large amounts of computations and can cause performance bottlenecks. Thirdly, simulations with GPU allows for computations to be performed independent of the CPU. However, the CPU-GPU data transfer limitations may restrict the maximum number of particles or grid size in a

simulation, thereby affecting the realism. Finally, time step and numerical errors may pollute the solution.

Some of the approaches for both smoke and bleeding described previously were also employed in several open source surgical simulation frameworks (32) (33) (34). SPRING framework (32)(15) offers a particle based approach in which each motion of a particle is computed independently. They achieved the rendering of the smoke by using texture billboards and textured lines for blood trails. The SOFA framework (33) provides a real-time fluid library that can be classified as a physics-based approach. They support SPH and Eulerian 2D/3D based fluids. The rendering is achieved through real-time mesh generation of the fluid surface. The GiPSi (34) framework also supports FEM based fluids. Open Tissue (35) and PhysX frameworks (36) are other frameworks with which employ the SPH method.

From our literature review on existing real time techniques of smoke generation and bleeding, it is apparent that non-physical methods are the way to go for present day surgical simulators as the major goal is to create a realistic visual effect without inducing any additional load on the CPU. This is critical for current surgical simulation systems because of the enormous computational load already placed on the CPU from simulating deformations, collisions and response characteristics. Hence, in this paper, we have developed an image based non-physical approach that uses pixel and vertex shader level techniques in the GPU to utilize its parallel many core architecture (37), for both computations and rendering.

Unlike the existing non-physical methods based on GPUs, our approach is significantly different by not only effectively utilizing the GPUs but also overcoming the CPU-GPU data transfer bottleneck. Specifically, we have proposed two buffering schemes to optimize the amount and rate of data transfers between the CPU and GPU. Unlike previous techniques, our method for bleeding doesn't dependent on the size and number of organ geometry or its associated textures and can be applied in any complex simulation scene without significant reduction in performance. Both bleeding and smoke simulation have been successfully incorporated and tested in a simulator for training surgeons in the Laparoscopic Adjustable Gastric Banding (LAGB) weight loss surgical technique. Human user studies have been carried out to establish the realism of the simulations.

Materials and Methods

In this section, we present techniques of simulating smoke and blood flow in surgical simulators. These techniques have been integrated into a virtual laparoscopic adjustable gastric banding (LAGB) simulator (38–41). Finally, we discuss our protocols for human subject studies to gauge the visual realism of the simulation scenes.

Smoke generation

In order to simulate smoke, videos from actual laparoscopic surgery were analyzed. Based on this, two different types of smoke were identified: *environmental smoke* which often covers the laparoscopic camera view during cauterization and *source smoke* which originates at the tip of the cautery tool. The *source smoke* slowly diffuses over the simulation scene, accumulating and generating the *environmental smoke* over time due to the limited ventilation capacity inside the operative space.

The technique that we have developed for smoke generation in virtual environments is based on the principle of capturing sample videos of smoke and overlaying them on the OpenGL frame buffer by removing the background (42) (see Figure 1). In our work, the removal of

the background was performed at the GPU (at the pixel shader), where the video frames are blended together to produce the final effect as shown in Figure 1. In addition, we perturb the transparency values (alpha values) using a Perlin noise texture to create variations in the smoke. The key difference in rendering the *source* and *environmental smoke* is that for the *source smoke* the tool tip coordinates have to be converted to screen coordinates for matching the start position of the smoke, whereas for the environmental smoke, it can be directly overlaid onto screen coordinates since the laparoscopic camera angle is usually fixed for the most part of the surgery. Figure 2 shows the schematic of rendering of both types of smoke simulations. In both smoke types, we extract the background by first separating the brightness (luma) and color (chroma) channels of the smoke video frames and then applying a filter whose threshold values are chosen based on the quality or noise of the smoke videos. Since the filtering and the blending operations are performed at GPU (pixel shader), it is computationally very efficient utilizing the multiprocessor banks of the GPU (43). The performance of this method is discussed in the *Results* section.

Retrieving video frames from storage in real-time is the major bottleneck for our approach for smoke generation. Considering the 60 frames per second (FPS) requirement of graphics, retrieving the video frames becomes time-critical. Any delay in this process, which includes reading video data from storage, decoding the compressed data to proper format, resizing, etc., not only affects the realism of the generated smoke but also the performance of the rendering of other objects in the scene. This is because in addition to time spent for video pre-processing tasks, in every rendering frame the video frames should also be transmitted from CPU to the GPU. In order to overcome this limitation, a separate module has to be designed for retrieving video frames. We have developed two different schemes for this purpose: pipeline and buffering. These two schemes are designed to perform I/O asynchronously with the rendering task.

In the *pipeline scheme*, the video frames are updated only once during each rendering cycle. At the end of each rendering cycle in the pipeline scheme, availability of video frames in memory is checked for retrieval. If the frame is ready then the texture in video memory on GPU is updated with the decoded video (where raw RGB data is stored) on the CPU side. When the data transfer from CPU to GPU is completed, the signal for retrieving next video frame to be used in the following rendering cycle is also given to the module so that it can continue retrieving the frames in parallel. It can be clearly seen in Figure 3 that the graphics rendering stays idle whenever there is a delay in retrieving the video frames.

In the *buffering scheme*, we utilize the idle regions of the pipeline scheme by employing internal buffers for the video frames. These buffers help storing the video data before they are actually used. Typical cycles of both retrieving videos frames and rendering are illustrated in Figure 3. Although buffering helps to have more than one video frame for the rendering task and removes any bottlenecks that arise from the lag in retrieving video frames, it does not reduce the time spent in CPU-GPU data transfer. In order to optimize the CPU-GPU data transfer, two video frames are updated simultaneously in the same rendering cycle. One update is for the current frame and the other for the next one. In the next rendering cycle, the CPU-GPU data transmission is eliminated since the texture is already in the GPU memory. As a result, the buffering scheme, leads to significant reduction of the CPU-GPU data transfer load. In general, the buffering scheme (unlike the pipeline scheme) always reads from the memory locations called buffers that keep the video data. The buffering scheme waits on the data if all video data are consumed in the buffers. The pseudo code for the buffering algorithm can be found in Table 1. In our current implementation of this buffering scheme, consecutive video frames are skipped for performance enhancement. However, more complex buffering schemes based on hardware capabilities may be implemented using our framework. The flow diagram of both pipeline and buffering

schemes are presented in Figures 4 and 5. Experimental results comparing both the schemes are presented in the *Results* section of the paper.

Simulation of bleeding

Bleeding occurs during surgery when either the tissue is divided using a blunt dissector or a cautery tool. Bleeding starts at the contact location of the tool tip with the tissue and spreads around it rapidly. Bleeding is stopped by coagulating the blood using an electrocautery tool. The simulation of bleeding in this work is based on the assumption that the bleeding is started at the contact point of the tool tip at the tissue, and coagulated within a short period of time. In order to simulate this behavior, an animation variable for each of the vertices of the mesh in the scene is defined in the mesh structure. The animation variables are the floating point values defined for each vertex that relates to the overall spread of blood on the tissue surface. The rate of change of these variables represents the bleeding speed over a surface. These animation variables are then passed through the two main GPU processing phases (vertex shader and pixel shader) before the final image is rendered. The animation variables are used in the GPU (pixel shader) to shade the surface of the polygons.

The flow diagram of the bleeding algorithm is shown in Figure 6. The whole process is divided into three major steps based on the hardware regions where the computations are taking place. The major CPU side tasks include pre-computation, detection of vein density and increments to the animation variables based on the bleeding speed. Once this is completed the data is then streamed to the fragment shader. The final step is performed by the fragment shader where the bleeding color and the color gradient are computed.

In Figure 7a these animation values at the vertex shader are shown as the blood spreads over the tissue surface. The interpolated values are then compared to the noise value obtained from the Perlin texture plus a predefined threshold. When the interpolated value exceeded this noise plus threshold value, bleeding is rendered in the scene. The Perlin noise texture allows us to generate complex regions of blood pooling with smooth edges. Since the animation values are interpolated over the polygons, using only a predefined threshold instead of the smoothing noise would create a sharp polygonal appearance, which is undesirable as seen in Figure 7b.

When the bleeding is initiated, the vertex animation value is incremented proportionally to the appropriate bleeding speed which was determined by observing actual surgery video clips. In order to spread the bleeding on the surface of the tissue, we simply use the connectivity of the surface mesh and vertex neighbors. The extent of the bleeding on the surface is controlled by an upper threshold for the vertex animation value.

Bleeding and Smoke in a Virtual Laparoscopic Adjustable Gastric Banding (LAGB) Simulator

The algorithms presented in the previous section were implemented in a procedural simulator for the laparoscopic adjustable gastric banding (LAGB) procedure. LAGB is a minimally invasive weight loss surgical procedure that is performed on morbidly obese patients. In this procedure, an adjustable band is placed around the stomach to prevent excessive food intake by providing early satiation (44) leading to subsequent weight loss. The details of the surgery and phases may be found in our previous study (39). Screenshots of the simulator with bleeding and smoke generation are presented in Figure 8.

In this LAGB procedure, fats and ligaments are scored using a hook cautery, frequently generating smoke of both *source* (at the hook tip) and *environmental* types in the scene. So, both types of smoke were simulated. Bleeding during the cautery process is initiated only

when tissue is actually cut. During the cauterization process, localized heat is applied to the tissue. In this work, we assume that the amount of heat and its effects on the nearby tissue is proportional to the duration of the application of the cautery. A more physics-based technique may, of course, be developed in which the bio-heat equation (45) may be solved to compute temperature profiles. As the cauterization continues, the region affected by the heat is expanded radially (46). If a pre-determined temperature threshold is exceeded, the tissue within the region is divided and bleeding is initiated based on an algorithm described below and presented in Figure 9.

Bleeding in the LAGB simulator is initiated based on the density of the blood vessels at the cautery tool tip. The texture used in the LAGB simulation is rather detailed with an intricate network of blood vessels. It is reasonable to assume that tissue regions with a higher density of blood vessels will tend to bleed more. Since very small blood vessels may not produce visible bleeding, a pre-set density threshold was used to determine when to initiate bleeding. In order to compute the density of blood vessels in the vicinity of the tool tip, we identify the *region of influence* or ROI of the tool tip. The blood vessels are modeled as 2D structures on the ligaments and organ textures. A band pass filter is then applied to the converted image to filter out the blood vessels from the original image. The obtained image is however pixelated due to nature of the threshold filtering. A Gaussian filter is then applied to smooth out any discrete bumps. The resultant final image has the same size as the original texture with only smoothed blood vessels, which is then used for computing the blood vessel density. The density is then simply sampled from this filtered image during the simulation of bleeding.

Human User Studies

The realism of the bleeding and smoke simulations was evaluated using an Institutional Review Board approved human subject study conducted at the Advanced Computational Research Laboratory in Rensselaer Polytechnic Institute, Troy, NY. The purpose of the study was to compare videos of surgical scenes showing smoke and blood and compare that with the computationally generated scenes. Since this study did not require medical knowledge, which would actually bias the results, the subject population was chosen from the engineering student community at Rensselaer. Several bleeding and smoke occurrences during real surgery were captured and edited into a final video of five minutes of duration. Similar bleeding and smoke scenarios were then simulated and captured in a video. Each subject in this study was shown first the clips from the real surgery video followed by the clips from the simulated bleeding and smoke simulations. The subjects were allowed to rewind and watch the clips as many times as needed. They were then asked to fill in their subjective feedback rated using a five point Likert (47) scale, ranging from strongly disagree to strongly agree with neither agree nor disagree as the neutral response.

A total of 20 subjects participated in the study. There were 18 males and two females. The data from the experiments were collected and descriptive statistics were computed. Table 3 shows the feedback questionnaire and selected descriptive statistics such as median, first and third quartile, minimum and maximum values. Figure 11a shows the box plot of feedback for questions related to the bleeding simulation. It can be seen that all questions obtained a median score of 3.0 or more. The highest rated response was to Question 3 which is critical as it refers to the realism of the appearance and color of bleeding in the simulated environment. The smoke simulation obtained very positive ratings with a median score of 4.0 or higher for all of the questions as seen in Figure 11b. Question 2, related to the realism of the smoke from the cautery tip in the virtual environment, and received the highest scores.

Results

Both bleeding and smoke simulations were implemented using shaders on GPUs. The code was written with GLSL shader and tested on an Intel quad core 2.83 GHz PC with 1.75 GB physical RAM and with two GPU GeForce 9800 Gx2, each with two cores with 512MB RAM each. On the GPU, the bleeding code was integrated into the existing shader, which allows bump mapping and wet surface effects on the organs. The smoke was implemented as an additional shader that is switched on/off during the cauterization.

We measured the performance of the two smoke generation algorithms presented in *smoke generation* section: pipeline and buffering scheme by timing the various core tasks in these two schemes. Table 2 shows the timing results.

The core tasks involve initialization of the video file which is performed at the beginning of the simulation, is called the *initialization of video stream* phase. In each rendering frame cycle, the video stream is retrieved from the local storage and decoded into the proper format to use as a texture. This phase is termed *average video frame retrieving*. After the frame is retrieved, the raw data is copied into the memory as indicated under the *buffer copy time* in Table 2. The copied data is then sent to GPU and successively processed by the shader. These activities are referred to as *average video streaming to GPU* and *shader processing time*, respectively. Another indicator of the performance of both the schemes is the video rendering in frames per second (FPS) which is also presented in our measurements.

As seen from the data, the total video processing time for the buffering scheme is nearly half of that of the pipeline scheme. This is due to the combined data transmission of the video frames and pre-update scheme of the frames. As expected, both the shader processing time and the initialization of video stream time are similar. The computation (computations in the shader) or I/O (video loading from storage, texture accessing in the shader) are also akin in both schemes.

In the buffering scheme, as the decompressed video is stored in an allocated buffer, the time required to copy the video frame to the buffer is doubled. As we require the buffering scheme to allocate two internal buffers, a two-fold slowdown is not unusual. *Average video frame retrieving* shows similar timings. This timing is sampled within each scheme's internal retrieval of the video frames. Therefore, they show approximately the same timings. *Average video streaming to GPU* in buffering scheme is also half of that of pipeline version.

As seen from Figure 10a, the average CPU-GPU timing of the pipeline scheme is substantially greater than the buffered scheme after five video streams. This is expected due to aggregate texture transmission and cache hits of the CPU-GPU transfers. The buffered scheme outperforms the pipeline in I/O time as seen in Figure 10b. Since the CPU-GPU transfer is drastically improved in the buffered scheme, the video FPS is higher as seen in Figure 10c. As in average CPU-GPU transfer, the video FPS also exhibits noticeable slowdowns beyond five video streams. This is due to the total video retrieval time spent during the I/O which cannot be completed within ≈ 16 milliseconds (60 Hz). Therefore, the rendering frame has to wait for I/O completion. For buffer updates the time to copy the decompressed video data into the CPU buffer is presented in Figure 10d. No noticeable differences in buffer update are observed between the two schemes. In the buffered scheme, all available buffers are filled immediately after starting the program and, as a result, the availability of buffers is critical for the performance of this scheme. In our testing of up to 15 videos, no significant memory problems were observed. In terms of overall performance, the I/O time is the dominant factor.

Discussion

The ratings from subject feedback show that there is an overall agreement in realism of the smoke and bleeding simulations. From Figure 11a it can be seen that the subjects tend to disagree most on the amount of bleeding in the simulation compared to actual surgery video (Question 7). Post questionnaire discussion indicated that the amount of simulated bleeding was far greater compared to the video clips. Though not in the same degree, the subjects also indicated that the amount of smoke simulated in the scene was greater than the smoke scene in the video clips from actual surgeries, which explains the relatively poor response to Question 9. We plan to modify our algorithm to limit the amount of blood in the scene as well as alter our alpha channels to dissipate some of the smoke in the scene.

Concluding Remarks and Future Directions

In this work, an effective GPU-based technique for both bleeding and smoke simulation was presented. Both were successfully implemented in the LAGB simulator as a case study. Computationally expensive physical simulations of bleeding and smoke were avoided by using image-based techniques. Both the bleeding and the smoke techniques utilized GPU, thereby eliminating overloading of the CPU. For smoke generation, two different schemes for video retrieving were developed to overcome the I/O bottleneck. Based on the benchmarks, the simulations had no negative impact on the rendering performance. From the human subject experiments for evaluating the realism of smoke and bleeding simulations, we conclude that both simulations are considered realistic, while there is some disagreement as to the quantity of blood and smoke generated in the simulator, which is straightforward to overcome.

In the future, we plan to incorporate more attributes to both smoke and bleeding simulations to adapt them to different surgical scenarios. We plan to adjust the intensity of the smoke based on the density of the fat layer under the cauterized region. In addition, during the cauterization of the fat layers, tiny fat particles dislodge from the fat tissue and fly around in the environment. We will incorporate this in our simulation by manipulating the fragments of the smoke at the fragment shader level. We also plan to optimize the texture loading to increase the performance of the rendering by using off-screen rendering techniques that will reduce bottleneck problems between CPU and GPU data transfer (48). Moreover, based on the feedback during the evaluation study and the quantitative results obtained from the questionnaire, we will adjust the level of bleeding to match the levels recorded during actual surgeries.

Acknowledgments

The authors gratefully acknowledge the support of this NIH/NIBIB through grant # R01EB005807.

Grant

Support of this work through the NIH/NIBIB grant # R01EB005807.

References

1. Barrett WL, Garber SM. Surgical smoke: a review of the literature. *Surgical endoscopy*. 2003; 17(6):979–987. [PubMed: 12640543]
2. Mattes D, Silajdzic E, Mayer M, Horn M, Scheidbach D, Wackernagel W, Langmann G, Wedrich A. Surgical smoke management for minimally invasive (micro) endoscopy: an experimental study. *Surgical Endoscopy*. 2010:1–10.
3. Eubanks, S.; Swanström, LL.; Soper, NJ. *Mastery of endoscopic and laparoscopic surgery*. Lippincott Williams & Wilkins; 2000.

4. Moorthy K, Munz Y, Forrest D, Pandey V, Undre S, Vincent C, Darzi A. Surgical Crisis Management Skills Training and Assessment. *Ann Surg.* 2006 Jul; 244(1):139–147. [PubMed: 16794399]
5. Scerbo M. Medical virtual reality simulation: Enhancing safety through practicing medicine without patients. *Biomedical Instrumentation and Technology.* 2004; 38(3):225–228. [PubMed: 15174366]
6. Huynh N, Akbari M, Loewenstein JI. Tactile Feedback in Cataract and Retinal Surgery: A Survey-Based Study. *Journal of Academic Ophthalmology.* 2008; 1(2):79–85.
7. Bhasin Y, Liu A, Bowyer M. Simulating surgical incisions without polygon subdivision. *Studies in Health Technology and Informatics.* 2005; 111:43–49. [PubMed: 15718696]
8. Neyret F, Heiss R, Sénégas F. Realistic rendering of an organ surface in real-time for laparoscopic surgery simulation. *The Visual Computer.* 2002; 18(3):135–149.
9. Sweet R, Porter J, Oppenheimer P, Hendrickson D, Gupta A, Weghorst S. Third Prize: Simulation of Bleeding in Endoscopic Procedures Using Virtual Reality. *J Endourol.* 2002; 16(7):451–455. [PubMed: 12396436]
10. Qin, J.; Pang, WM.; Chui, YP.; Xie, YM.; Wong, TT.; Poon, WS.; Leung, KS.; Heng, PA. Hardware-accelerated Bleeding Simulation for Virtual Surgery. *MICCAI 2007 Workshop Proceedings*; 2007. p. 133
11. Zátönyi J, Paget R, Székely G, Grassi M, Bajka M. Real-time synthesis of bleeding for virtual hysteroscopy. *Medical Image Analysis.* 2005; 9(3):255–266. [PubMed: 15854845]
12. Kerwin T, Shen HW, Stredney D. Enhancing Realism of Wet Surfaces in Temporal Bone Surgical Simulation. *IEEE transactions on visualization and computer graphics.* 2009; 15(5):747–758. [PubMed: 19590102]
13. Baur C, Guzzoni D, Georg O. VIRGY: A Virtual Reality and Force Feedback-Based Endoscopic Surgery Simulator. *Studies in health technology and informatics.* 1998:110–116. [PubMed: 10180525]
14. K\ühnapfel U, Cakmak HK, Maa\ss H. Endoscopic surgery training using virtual reality and deformable tissue simulation. *Computers & Graphics.* 2000; 24(5):671–682.
15. Daenzer, S.; Montgomery, K.; Dillmann, R.; Unterhinninghofen, R. Real-Time Smoke and Bleeding Simulation in Virtual Surgery. *Medicine meets virtual reality 15: in vivo, in vitro, in silico: designing the next in medicine.* 2007. p. 94
16. Basdogan, C.; Ho, CH.; Srinivasan, MA. *Medicine Meets Virtual Reality: the Convergence of Physical & Informational Technologies; Options for a New Era in Healthcare.* Vol. 20. San Francisco, CA: 1999. Simulation of tissue cutting and bleeding for laparoscopic surgery using auxiliary surfaces; p. 38-44.
17. Mukai, N.; NAKAGAWA, M.; Kosugi, M. Real-time Blood Vessel Deformation with Bleeding Based on Particle Method. *Medicine meets virtual reality 16: parallel, combinatorial, convergent: NextMed by design.* 2008. p. 313
18. Müller M, Schirm S, Teschner M. Interactive blood simulation for virtual surgery based on smoothed particle hydrodynamics. *Technology and Health Care.* 2004; 12(1):25–31. [PubMed: 15096684]
19. Liu W, Sewell C, Blevins N, Salisbury K, Bodin K, Hjelte N. Representing Fluid with Smoothed Particle Hydrodynamics in a Cranial Base Simulator. *Studies in health technology and informatics.* 2008; 132:257. [PubMed: 18391299]
20. Székely G, Brechb\ühler C, Dual J, Enzler R, Hug J, Hutter R, Ironmonger N, Kauer M, Meier V, Niederer P, et al. Virtual reality-based simulation of endoscopic surgery. *Presence: Teleoperators & Virtual Environments.* 2000; 9(3):310–333.
21. Meseure P, Davanne J, Hilde L, Lenoir J, France L, Triquet F, Chaillou C. A physically-based virtual environment dedicated to surgical simulation. *Surgery Simulation and Soft Tissue Modeling.* 2003:1002–1002.
22. McReynolds, T.; Blythe, D.; Grantham, B.; Nelson, S. *SIGGRAPH'98 Course Notes.* 1998. Advanced graphics programming techniques using OpenGL; p. 90-99.
23. Agarwal, R.; Bhasin, Y.; Raghupathi, L.; Devarajan, V. Special visual effects for surgical simulation: Cauterization, irrigation and suction. *Medicine meets virtual reality 11: NextMed: health horizon.* 2003. p. 1

24. Stam, J.; Fiume, E. Turbulent wind fields for gaseous phenomena. Proceedings of the 20th annual conference on Computer graphics and interactive techniques; 1993. p. 376
25. Fedkiw, R.; Stam, J.; Jensen, HW. Visual simulation of smoke. Proceedings of the 28th annual conference on Computer graphics and interactive techniques; 2001. p. 15-22.
26. Kim T, Thürey N, James D, Gross M. Wavelet turbulence for fluid simulation. ACM Trans Graph. 2008; 27(3):50.
27. HLSL_BloodShader.pdf [Internet].[cited 2010 Mar 9] Available from:
http://http.download.nvidia.com/developer/SDK/Individual_Samples/DEMOS/Direct3D9/src/HLSL_BloodShader/docs/HLSL_BloodShader.pdf
28. Holtkämper, T. Real-time gaseous phenomena: a phenomenological approach to interactive smoke and steam. Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa; 2003. p. 30
29. Çakmak, HK.; Kühnnapfel, U. Animation and simulation techniques for vr-training systems in endoscopic surgery. Computer animation and simulation 2000: proceedings of the Eurographics Workshop in Interlaken; Switzerland. August 21–22, 2000; 2000. p. 173
30. Yang, M.; Lu, J.; Safonova, A.; Kuchenbecker, KJ. GPU Methods for Real-Time Haptic Interaction with 3D Fluids. 2008.
31. NVIDIA PhysX Physics Simulation for Developers [Internet].[cited 2010 Mar 9] Available from:
<http://developer.nvidia.com/object/physx.html>
32. Montgomery K, Bruyns C, Brown J, Sorkin S, Mazzella F, Thonier G, Tellier A, Lerman B, Menon A. Spring: A general framework for collaborative, real-time surgical simulation. Medicine meets virtual reality 02/10: digital upgrades, applying Moore's law to health. 2002:296.
33. Allard J, Cotin S, Faure F, Bensoussan PJ, Poyer F, Duriez C, Delingette H, Grisoni L. Sofa-an open source framework for medical simulation. Studies in health technology and informatics. 2006; 125:13. [PubMed: 17377224]
34. Cavusoglu, MC.; Göktekin, TG.; Tendick, F.; Sastry, S. Medicine meets virtual reality 12: building a better you: the next tools for medical education, diagnosis, and care. 2004. GiPSi: An open source/open architecture software development framework for surgical simulation; p. 46
35. Erleben, K.; Spöring, J.; Dohlmann, H. OpenTissue-An Open Source Toolkit for Physics-Based Animation. MICCAI Open-Source Workshop; 2005.
36. Harris, M. Siggraph Asia. 2009. Cuda fluid simulation in nvidia physx.
37. Owens, JD.; Luebke, D.; Govindaraju, N.; Harris, M.; Kruger, J.; Lefohn, AE.; Purcell, TJ. A survey of general-purpose computation on graphics hardware. 2007. p. 80-113.
38. Halic, T.; De, S. Lightweight Bleeding and Smoke Effect for Surgical Simulators. IEEE Virtual Reality Conference; Waltham, MA. 2010.
39. Maciel A, Halic T, Lu Z, Nedel LP, De S. Using the PhysX engine for physics-based virtual surgery with force feedback. The International Journal of Medical Robotics and Computer Assisted Surgery. 2009; 5(3)
40. De, S.; Ahn, W.; Lee, DY.; Jones, DB. Novel Virtual Lap-Band® Simulator Could Promote Patient Safety. Medicine Meets Virtual Reality 16: Parallel, Combinatorial, Convergent: NextMed by Design. 2008. p. 98
41. Sankaranarayanan G, Adair DJ, Halic T, Gromski AM, Lu Z, Ahn W, Jones BD, De S. Validation of a Novel Laparoscopic Adjustable Gastric Band Simulator. 2010 will appear in Surgical Endoscopy.
42. Beato N, Zhang Y, Colbert M, Yamazawa K, Hughes CE. Interactive chromakeying for mixed reality. Computer Animation and Virtual Worlds. 2009; 20(2–3):405–415.
43. Lindholm, E.; Nickolls, J.; Oberman, S.; Montrym, J. IEEE Micro. 2008. NVIDIA Tesla: A unified graphics and computing architecture; p. 39-55.
44. Ren CJ, Fielding GA. Laparoscopic adjustable gastric banding: surgical technique. Journal of Laparoendoscopic & Advanced Surgical Techniques. 2003; 13(4):257–263. [PubMed: 14561254]
45. Pennes HH. Analysis of tissue and arterial blood temperatures in the resting human forearm. Journal of Applied Physiology. 1948 Aug.1(2)

46. Lu, Z.; Sankaranarayanan, G.; Deo, D.; Chen, D.; De, S. Towards Physics-based Interactive Simulation of Electrocautery Procedures using PhysX; Boston. IEEE Haptics symposium; 2010.
47. Likert, R. A technique for the measurement of attitudes. 1932.
48. Bindless Graphics Tutorial [Internet]. [date unknown];[cited 2010 Mar 9] Available from: http://developer.nvidia.com/object/bindless_graphics.html

Figure 1.
Steps for smoke rendering and background filtering on the GPU

Figure 2. Overlaying of blended smoke textures on the rendered pixels at the fragment shader with perturbation of the alpha channel.

Figure 3.
Two different schemes employed for video retrieving I/O

Figure 4.
Pipeline video retrieving scheme

Figure 5.
Buffering video retrieving scheme

Figure 6.
The bleeding process flow on CPU and GPU

Figure 7.

(a) Animation value changes during the bleeding (b) Undesirable visual appearance without noise function

Figure 8.
(a), (b) and (c) Snapshots of bleeding and smoke from LAGB surgical simulator

Figure 9. Veins are extracted to initialize bleeding and compute the bleeding amount.

Figure 10.

Performance comparisons of two different schemes (a) CPU-GPU transfer (b) Video processing time (c) Video frames per second (d) Buffer copy

Figure 11.

(a) Box plot of results for bleeding questionnaire (b) Box plot of results from smoke questionnaire

Table 1

Pseudo code for rendering task of buffering scheme

```

Function CPU2GPU
  If (Next Frame textures are sent already)
    return
  Else
    Check the buffers of Video Retrieving Module
    Wait for the buffers
    Transfer the buffers to GPU
  End If
End Function

Function FrameCoalescing
  If (NextFrametextures !=true)
    Call CPU2GPU
    NextFrametextures=true
  Else
    NextFrametextures=false
  End if
End Function

Function initBuffer
  NumberOfBuffers=MAX_TEXTURES/TotalVideos
  Allocate Buffers (NumberOfBuffers)
End Function

Function RenderSmoke
  EnableShader
  Call CPU2GPU
  Render ScreenQuad
  DisableShader
  FrameCoalescing
End Function

```

Table 2

Benchmarks of two different schemes for video retrieving

Pipeline Video Retrieving Scheme						
# of sample videos	Initialization of video stream (ms)	Average video frame retrieving (ms)	Buffer copy time (ms)	Average video streaming time to GPU (ms)	Shader processing time (ms)	Total video processing time (ms)
1	100.258	2.17665	0.235747	1.54582	0.384965	1.64677
2	151.904	4.17174	0.5333594	3.1579	0.447961	3.25741
3	195.64	6.41025	0.870259	4.68073	0.564852	4.78104
5	293.509	12.5976	2.19474	8.27805	1.03806	8.40228
8	430.997	22.4896	4.28879	16.5888	2.17362	16.7192
10	548.292	28.281	5.67987	22.1224	4.3373	22.2582
15	776.62	43.3006	8.65272	32.4184	10.183	32.5438
BufferingVideo Retrieving Scheme						
1	102.164	2.17056	0.341065	0.775565	0.377916	0.873131
2	151.258	4.44614	0.671104	1.555	0.441045	1.65732
3	200.999	6.5289	0.99612	2.46336	0.556842	2.56828
5	299.193	11.0834	1.70013	3.84415	1.02658	3.94769
8	445.926	19.49	3.40009	6.87388	2.15974	7.00681
10	548.132	27.1295	5.28326	10.8744	4.33791	11.0062
15	783.42	41.0246	8.04662	16.1795	10.1966	16.3174
Screen Resolution	1280×1024 pixels					
Video Codec	DivX version 6.9.1					
Video Resolution	720×480 pixels					
						Video rendering (FPS)
						59.4923
						59.468
						59.4719
						59.4575
						47.4496
						37.7391
						26.9911
						59.4874
						59.487
						59.4847
						59.4551
						59.3467
						52.8614
						36.5566

Table 3

Feedback questionnaire and descriptive statistics

Question	Maximum	1 st Quartile	Median	3 rd Quartile	Minimum	Mode
1. The bleeding in the tissue is very similar to the surgery video	5.0	2.75	4.0	4.25	2.0	4.0
2. The smoke originated from cautery tip looks similar to the surgery video	5.0	4.75	5.0	5.0	3.0	5.0
3. The appearance and color of bleeding is as realistic as the surgery video	5.0	3.0	4.0	5.0	2.0	4.0
4. The smoke diffusion in the simulation resembles actual smoke diffusion in surgery	5.0	4.0	5.0	5.0	2.0	5.0
5. The spread of the bleeding in the simulation looks similar to the spread in the surgery video	5.0	2.0	3.5	4.0	2.0	4.0
6. The intensity, and color of the smoke in the simulation looks like the smoke in actual surgery	5.0	4.0	5.0	5.0	4.0	5.0
7. The amount of the bleeding in the simulator was similar to the amount in the surgery video	4.0	2.0	3.0	4.0	1.0	4.0
8. The smoke appearance in the simulation gives the impression of a smoke like in actual surgery	5.0	4.0	4.0	5.0	2.0	5.0
9. The amount of smoke in the simulator was similar to the amount in the surgery video	5.0	3.75	4.0	5.0	2.0	4.0
10. The bleeding in the simulator gave an impression of actual bleeding observed in the surgery video	5.0	2.0	3.5	4.0	2.0	4.0
11. The overall realism of smoke and bleeding is as good as the surgery video	5.0	3.0	4.0	4.0	2.0	4.0