Journal of Cancer Treatment and Research
2015; 3(5): 53-65
Published online November 6, 2015 (http://www.sciencepublishinggroup.com/j/jctr)
doi: 10.11648/j.jctr.20150305.11
ISSN: 2376-7782 (Print); ISSN: 2376-7790 (Online)

3D Visualisation of Tumour-Induced Angiogenesis Using the CUDA Programming Model and OpenGL Interoperability

Paul M. Darbyshire

Computational Biophysics Group, Algenet Cancer Research, Nottingham, UK

Email address:
rd@algenet.com

To cite this article:
Paul M. Darbyshire. 3D Visualisation of Tumour-Induced Angiogenesis Using the CUDA Programming Model and OpenGL Interoperability. *Journal of Cancer Treatment and Research*. Vol. 3, No. 5, 2015, pp. 53-65. doi: 10.11648/j.jctr.20150305.11

Abstract: In this paper we solve a complex discrete-continuous model of tumour-induced angiogenesis using an explicit time-stepping FDM and simultaneously simulate the model dynamics in 3D. The interoperability between the CUDA programming model and the graphics hardware through OpenGL allows us to generate dynamic interactive 3D realistic visualisations. We use CUDA for the complex parallel calculations and deploy OpenGL for on-the-fly 3D visualisation of the numerical simulations. Clearly, being able to link the numerical results of complex mathematical models to interactive 3D visualisations that can literally update instantaneously to varying model parameters, should provide an invaluable tool for clinical physicians and research scientists. We also give an overview of current medical imaging techniques for studying microcirculatory and blood flow dynamics at the cellular level and indicate how the results presented here could offer potential for future developments in this area.

Keywords: 3D Cancer Modelling, 3D Visualisation, Medical Imaging, High Performance Computing, Compute Unified Device Architecture (CUDA), Graphical Processing Unit (GPU), Open Graphics Library (OpenGL)

1. Introduction

Over the last decade, *high-performance computing* has evolved dramatically, in particular because of the accessibility to *graphics processing units* (GPUs) and the emergence of GPU-CPU *heterogeneous* architectures, which have led to a huge shift in the available medical applications for supercomputing and parallel programming. Such *in silico* experiments focussed on the dynamics of tumour growth and other related biological phenomenon have become readily accepted by the scientific community both as a means to direct new research and a route to generate new hypotheses and testable predictions [1-5]. Once we have a model that is validated it is possible to more efficiently predict what should be happening in a particular experiment. With such a predictive model it is much easier and faster to perform *in silico* experiments to test hypotheses and predictions than running time consuming and costly laboratory experiments. More recently, the advantages of supercomputing and parallel processing techniques has highlighted the speedup, amongst other benefits, from the numerical solution of complex mathematical models of tumour dynamics [6-10]. In a previous paper, the authors developed a 3D parallel algorithm based on a time-stepping *finite difference method* (FDM) to solve a hybrid continuous-discrete model of tumour-induced angiogenesis [10]. The numerical solution was implemented on the GPU and results indicated an impressive increase in execution time over that of a conventional C++ algorithm. Whilst also highlighting the many optimisation techniques available to further improve the performance of the algorithm. In this paper, the authors develop the algorithm further and utilise the graphical interoperability available on parallel platforms to visualise the angiogenetic process in 3D. It is envisaged that being able to link the numerical results of complex biological models to an interactive 3D visualisation that can literally update instantaneously to varying model parameters, should be an invaluable tool for clinical physicians and research scientists. Moreover, such rapid and interactive virtual experimental representations can also facilitate oncological research and pharmaceuticals in developing and testing new anti-cancer treatment strategies.

In order to progress from the relatively harmless avascular phase to the potentially lethal vascular state, solid tumours must induce the growth of new blood vessels from existing ones, a process known as *angiogenesis*. The morphological events that are involved in angiogenesis have been highlighted through studies both involving *in vivo* and *in vitro* angiogenetic assays [11]. Physiological angiogenesis is a highly organised sequence of cellular events comprising vascular initiation, formation, maturation, remodelling and regression, which are controlled and modulated to meet tissue requirements. In contrast, pathological angiogenesis is less well controlled and although the initiation and formation stages occur, the vessels rarely mature, remodel or regress in disease. Whilst early models of angiogenesis were focused on accurately replicating key observed behaviours during this process, more recent models have been able to test specific hypotheses and suggest useful strategies for anti-angiogenic drug development. It has been known for some time that a series of biologic *on* and *off* switches regulate the process of angiogenesis providing a tumour with the ability to trigger the formation of its own vascular network by the secretion of chemical factors [12]. The main *on* switches are known as angiogenesis-stimulating growth factors and *off* switches as angiogenesis inhibitors. In order to monitor and supply sufficient amounts of essential nutrients to the surrounding tissues, blood vessels also have hypoxia-induced sensors, or receptors that assist in vessel remodelling to adjust the blood flow accordingly. From these vascular networks, blood vessels provide essential nutrients and oxygen throughout the body. Indeed, a key mechanism of anti-angiogenic therapy is to interfere with the process of blood vessel growth and literally starve the tumour of its blood supply. Recently, a new class of cancer treatments that block angiogenesis have recently been approved and available to treat cancers of the colon, kidney, lung, breast, liver, brain, ovaries and thyroid [13-17].

Angiogenesis can be considered a complex biological phenomena and one that at a system level is dynamic, spatially heterogeneous, frequently non-linear, and spans many orders of magnitude, both spatially and temporally. Mathematical and computational models of vascular formation have generated a basic understanding of the processes of capillary assembly and morphogenesis during tumour development and growth [18, 19]. However, by the time a tumour has grown to a size whereby it can be detected by clinical means, there is a strong likelihood that it has already reached the vascular growth phase and developed its own blood circulatory network. For this reason, a thorough understanding of the behavioural processes of angiogenesis is essential. Indeed, the development of realistic mathematical and computational models of such processes is a powerful method of testing hypotheses, confirming biological experiments, and simulating complex dynamics. Moreover, the ability to realistically create *virtual* 3D visualisations of dynamic biological processes provides further support to quantitative analyses. Several other authors have already attempted to model the process of angiogenesis using 3D visualisation techniques [20-22] but to our knowledge have yet to take advantage of the benefits of using high performance computing.

Pathological angiogenesis has been extensively explored through mathematical modelling over the past few decades, specifically in the contexts of tumour-induced angiogenesis and subsequent vascularisation. More recently, hybrid models that integrate both continuous and discrete processes of biological phenomena on various temporal and spatial scales have come to the fore [23]. These models represent cells as individual discrete entities and often use continuous nutrient concentrations to model cellular behaviour due their microenvironment. The model presented in this paper is of a hybrid type in which a system of couple nonlinear partial differential equations (PDEs) describe the continuous chemical and macromolecular dynamics and a discrete cellular automata-like model controls cell migration and interaction of neighbouring cells. Mathematically, FDM are the first port of call for solving complex biological phenomenon described by nonlinear PDEs. However, they require intensive computational resources which generally leads to significant and time-consuming expense. The advantages of explicit time-stepping in FDM over many other types of solutions is that they lend themselves well to exploitation in a completely *data-parallel* context. In such cases, GPUs can be used to greatly accelerate numerical simulations and offer an extremely valuable advanced computational technique for tackling such problems. The *compute unified device architecture* (CUDA) programming model is especially well-suited to address problems that can be expressed as data-parallel computations [24]. Moreover, the interoperability between the CUDA programming model and the graphics hardware through OpenGL allows us to simulate more dynamic and interactive 3D realistic visualisations. We can use CUDA for the complex parallel calculation and deploy OpenGL for 3D on-the-fly visualisations of the numerical results on-screen. In this paper we solve a complex discrete-continuous model of tumour-induced angiogenesis using an explicit time-stepping FDM whilst dynamically visualising the growth of a 3D vascular network driven by tumour-induced angiogenesis.

2. Biological Description

Solid tumours generally undergo a period of avascular growth, after which they become dormant for a sustained period without access to a sufficient supply of essential nutrients, such as oxygen and glucose. Beyond a certain size (~2 mm) diffusion alone is insufficient for the provision of such nutrients; the surface area to volume ratio is too low and as such the developing tumour begins to starve. In response to this state of hypoxia, cancer cells send out signals to cells of nearby blood vessels by secreting a number of chemicals, known collectively as *tumour angiogenic factors* (TAF) [25-27]. Tumour angiogenesis stimulators include chemicals that belong to *fibroblast growth factor* (FGF) and *vascular endothelial growth factor* (VEGF) families. One important function of FGF is the promotion of endothelial cell proliferation and the physical organisation of endothelial cells into tube-like structures. Also, some anti-angiogenic drugs block VEGF from attaching to the receptors on the endothelial cells that line the blood vessels in order to stop them from growing. In fact, no metabolically active tissue in the body is more than a few hundred micrometres from a blood capillary.

Once secreted, TAF diffuse into the surrounding tissue and set up an initial steady-state concentration gradient between the tumour and any pre-existing vasculature. Endothelial cells situated in nearby parent vessels degrade their own basal lamina and begin migrating into the *extra cellular matrix* (ECM) [28, 29]. The ECM is a complex mixture of macro-molecules, containing collagens, fibronectin etc., which functions as a scaffold for endothelial cells to grow on. The degradation of the basal lamina leads to damage, and potential rupture, of the parent vessel basement membrane. Such damage allows fibronectin from the blood to leak from the parent vessel and diffuse into the surrounding tissue [30-32]. Small capillary sprouts form from several endothelial cell clusters and begin to extend towards the tumour, directed by the motion of the leading endothelial cell at the sprout tip, until the finger-like capillaries reach a certain length. At this point, they tend towards each other, and form loops before fusing together in a process known as *anastomoses* [25, 26]. Following anastomoses, the primary loops start to bud and sprout repeating the process and further extending the newly formed capillary bed. Figure 1 shows diagrammatically the general shape of the capillary sprouts and their finger-like structure.

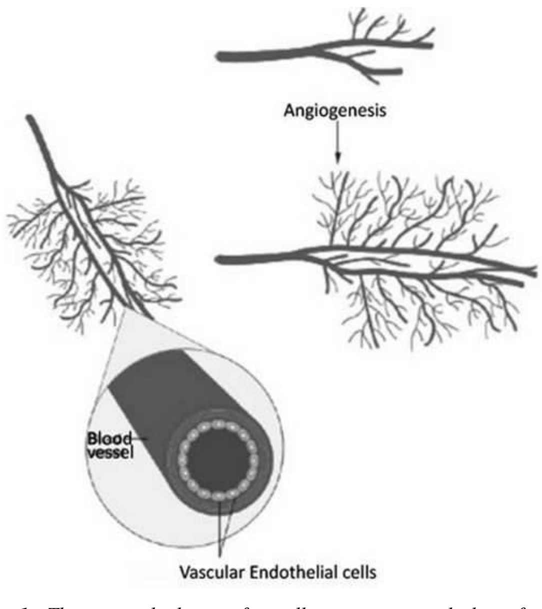


Figure 1. *The general shape of capillary sprouts and their finger-like structure.*

Further sprout extension occurs when some of the endothelial cells on the sprout-wall begin to proliferate. Cell division is largely confined to a region just behind the cluster of endothelial cells that constitute the sprout-tip. This process of sprout-tip migration and proliferation of sprout wall cells forms solid strands of endothelial cells within the ECM. As the sprouts approach the tumour, branching rapidly increases and produces a *brush border* effect, until the tumour is finally penetrated [29]. Once a supply of essential nutrients reaches the tumour, through this newly formed blood circulatory system, it enters the phase of vascularisation as shown in Figures 2 and 3. To support continued growth, the vascular system constantly restructures itself implying that angiogenesis is an on-going process, continuing indefinitely until the tumour is removed or destroyed. Indeed, angiogenesis also enables the tumour to spread to other parts of the body through the blood stream significantly increasing the probability of mortality from cancer due to *metastasis*.

Figure 2. *A tumour being surrounded by a vascular network of blood vessels.*

Figure 3. *A tumour reaching the vascular phase as a result of angiogenesis.*

Indeed, angiogenesis is an essential component of the metastatic pathway. The new blood vessels that are formed allow the cancer cells to leave the original site of the cancer and spread to distant organs through the blood. Moreover, the higher the density of new blood vessels within a tumour, the higher the risk of metastasis.

3. A Continuous-Discrete Model of Tumour-Induced Angiogenesis

3.1. The Continuous Model

For a more rigorous mathematical proof, readers are directed to [9, 33] and references therein. Here we simply summarise the main mathematical development so as to focus on the main issues of the paper. If we denote the endothelial cell density by n, the TAF and fibronectin concentration by c and f, respectively the complete system of scaled coupled nonlinear PDEs describing tumour-induced angiogenesis can be written as:

$$\frac{\partial n}{\partial t} = D\nabla^2 n - \nabla \cdot (\chi(c) n \nabla c) - \rho \nabla \cdot (n \nabla f) \tag{1}$$

$$\frac{\partial f}{\partial t} = \beta n - \gamma n f \tag{2}$$

$$\frac{\partial c}{\partial t} = -\eta n c \tag{3}$$

The chemotactic migration is characterised by the function $\chi(c)$, given by:

$$\chi(c) = \frac{\chi}{1+\alpha c} \tag{4}$$

$\chi(c)$ reflects the fact that chemotactic sensitivity generally decreases with increased TAF concentration. A description of each of the parameters, and their respective values, are shown in Table 1.

***Table 1.** Parameter descriptions and values used in the coupled nonlinear PDE model (1) – (4) [33].*

Parameter	Description
α	Decay factor
β	Fibronectin production coefficient
γ	Fibronectin degradation
D	Random motility diffusion coefficient
η	Rates of TAF uptake
ρ	Hapotactic coefficient
χ	Chemotactic coefficient

Our system is assumed to hold on a 3D spatial domain Ω (i.e. a volume of tissue) with appropriate initial conditions; $c(x, y, z, 0)$, $f(x, y, z, 0)$ and $n(x, y, z, 0)$.The tumour cells are assumed to be confined within a domain $\Omega \in [0,1]^3$in which *no-flux* (Neumann) boundary conditions $\partial\Omega$, are imposed on the boundaries of Ω. After the TAF has reached the parent vessel, the endothelial cells within the vessel develop into several cell clusters which eventually form sprouts [33]. For simplicity, we assume that initially five clusters develop along the x-axis at $y \approx 1$, with a circular tumour located at $y = 0$ and the parent vessel of the endothelial cells at $y = 1$ as shown in Figure 4.

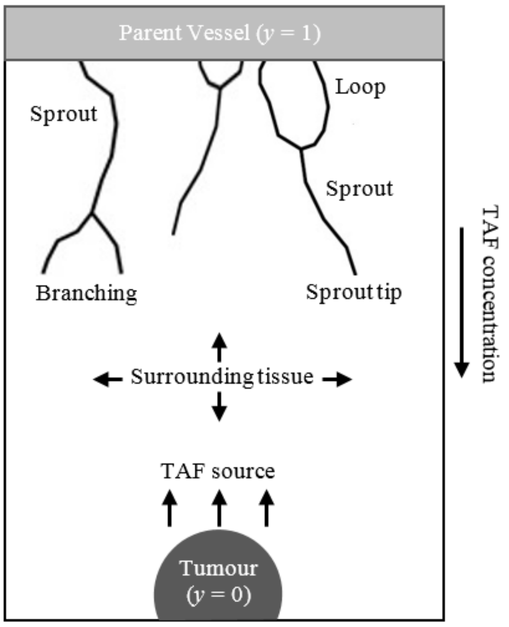
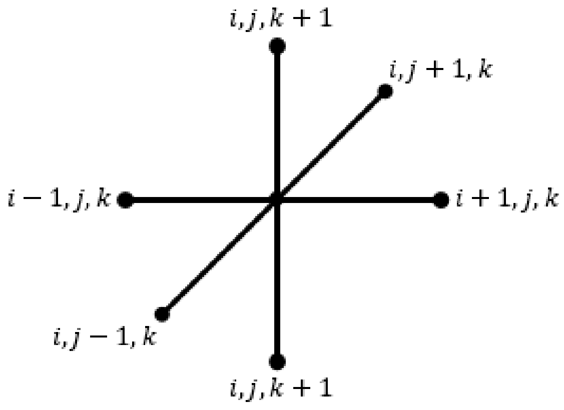


***Figure 4.** A schematic representation of the positions of the parent vessel and circular tumour as well examples of branching at a sprout tip and looping of two capillary sprouts.*

3.2. The Discrete Model

The technique of tracing the path of an individual endothelial cell at a sprout tip was first proposed by Anderson *et al.* [34]. The method involves using standard FDM to discretise the continuous model described in (1)-(4) over a 3D uniform grid. Then, the resulting coefficients of the finite difference seven-point stencil are used to generate the probabilities of movement of an individual endothelial cell in response to its local microenvironment. 3D stencil computations are those in which each node in a 3D grid is updated with a weighted average of the six neighbouring node values. Two schematic diagrams of a 3D finite difference seven-point stencil are shown in Figure 5.



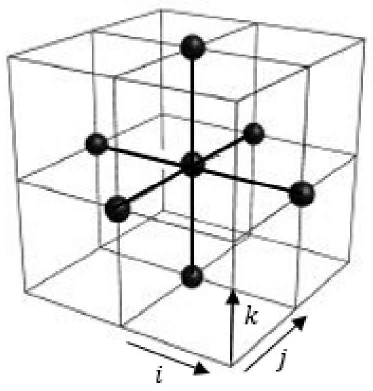


***Figure 5** Schematic diagrams of the finite difference 7-point 3D stencil.*

We first discretise the continuous model by approximating the 3D domain $\Omega \in [0,1]^3$ on a uniform grid of node length, width and depth h, and time t by increments of size k. By applying a *forward* finite difference scheme, the *fully-explicit* discretised version of the continuous model can be obtained. The discretisation for n, f and c are shown below:

$$n_{i,j,k}^{q+1} = n_{i,j,k}^{q}P_0 + n_{i+1,j,k}^{q}P_1 + n_{i-1,j,k}^{q}P_2 + n_{i,j+1,k}^{q}P_3 + n_{i,j-1,k}^{q}P_4 + n_{i,j,k+1}^{q}P_5 + n_{i,j,k-1}^{q}P_6 \quad (5)$$

$$f_{i,j,k}^{q+1} = f_{i,j,k}^{q}\left[1 - k\gamma n_{i,j,k}^{q}\right] + k\beta n_{i,j,k}^{q} \quad (6)$$

$$c_{i,j,k}^{q+1} = c_{i,j,k}^{q}\left[1 - k\eta n_{i,j,k}^{q}\right] \quad (7)$$

Where *i, j k,* and *q* are positive parameters which specify the location on the grid and the time step, i.e., $x = i\Delta x, y = j\Delta y, z = k\Delta z,$ and $t = k\Delta t$. P_0–P_6 are functions of both fibronectin and TAF concentrations at nearby neighbouring points of an individual endothelial cell. The complete set of parameter values used for the numerical simulation and the exact forms of P_0–P_6 can be found in [9, 20, 33]. The coefficients P_0–P_6 can be thought of as being proportional to the probabilities of endothelial movement. That is, the coefficient P_0, is proportional to the probability of no movement, and the coefficients P_1, P_2, P_3, P_4, P_5, and P_6, are proportional to the probabilities of moving left, right, up and down, out of and into the plane, respectively. Each numerical simulation is based on an increased size of array width i.e. a finer grained uniform 3D grid. We use a constant iteration size of 1,000 time steps to allow for an adequate convergence of the numerical solution. At each time step, the numerical simulation involves solving the discrete model to generate the seven coefficients P_0–P_6. Based on the values of these coefficients, a set of seven probability ranges are determined based on the following criteria:

$$R_0 = 0 \text{ to } P_0 \quad (8)$$

$$R_m = \sum_{n=0}^{m-1} P_n \text{ to } \sum_{n=0}^{m} P_n \quad (9)$$

Where m = 1…6. A uniform random number is then generated on the interval [0, 1], and, depending on the range into which this value falls, the current individual endothelial cell will remain stationary (R_0), move left (R_1), right (R_2), move up (R_3), down (R_4), out of (R_5), or into the plane (R_6). Each endothelial cell is therefore restricted to move to one of its six orthogonal neighbouring grid nodes or remain stationary at each time step. We further assume that the motion of an individual endothelial cell located at the tip of a capillary sprout governs the motion of the whole sprout. This is not considered unreasonable since the remaining endothelial cells lining the sprout-wall are contiguous [35]. We further assume that each sprout tip has a probability, P_b of generating a new sprout (branching) and that this probability is dependent on the local TAF concentration. It is also reasonable to assume that the newly formed sprouts do not branch until there is a sufficient number of endothelial cells near their tip. We will assume that the density of endothelial cells required for branching is inversely proportional to the concentration of TAF, since new sprouts become much shorter as the tumour is approached [35]. Based on these assumption we can write down the following three cellular rules:

Rule 1: New sprouts reach maturation after a length of time (ψ = 0.5) [33] before branching,

Rule 2: Sufficient local space exists for a new sprout to form, and

Rule 3: Endothelial cell density, $n > n_b$, where $n_b \propto \frac{1}{c_{i,j}}$.

We also assume that if a sprout tip encounters another sprout, then anastomosis can occur and a loop is formed. As a result of a tip-to-tip anastomosis, only one of the original sprouts continues to grow (purely random) and the other fuses to form the loop [25]. Figure 4 shows a schematic of the branching at a sprout tip and looping of two capillary sprouts. In addition, endothelial cell doubling time was estimated at 18 hours [36] and this is factored into our discrete model such that cell division occurs behind a sprout tip every 18 hours. We assume that this has the effect of increasing the length of a sprout approximately one cell length every 18 hours. Due to the inherent randomness of the discreet model, proliferation will occur asynchronously, as observed experimentally [25].

4. Implementation

4.1. Hardware

The CUDA C++ program was developed in Microsoft® Visual Studio 2012 using CUDA version 7.0 and tested on an Nvidia GeForce® GTX™ 780 GPU based on the Kepler GK110 architecture with Compute Capability 3.5. The Compute Capability describes the features of the hardware and reflects the set of instructions supported by the device as well as other specifications, such as the maximum number of threads per block and the number of registers per multiprocessor. Moreover, hardware design, number of cores, cache size, and supported arithmetic instructions are different for different versions of Compute Capability. Higher compute capability versions are supersets of lower (i.e., earlier) versions, so they are backward compatible. The graphics engine was developed using interoperability between the CUDA programming model and OpenGL 3.x instructions. The operating system was Windows 8.1.

4.2. The CUDA Programming Model

The CUDA programming model provides an *application program interface* (API) that exposes the underlying GPU architecture; a collection of *single instruction, multiple data* (SIMD) processors capable of executing thousands of *threads* in parallel. A version of SIMD used by GPUs is the *single instruction, multiple threads* (SIMT) architecture in which multiple threads execute an instruction sequence. In CUDA C, an instruction sequence is written into a specific function known as a *kernel* that can be executed on a device N times in parallel by N different CUDA threads, *asynchronously*. Unlike a C function call, all CUDA kernel launches are asynchronous so that control returns to the CPU immediately after the CUDA kernel is invoked [24, 37].

4.3. CUDA and OpenGL Interoperability

OpenGL is one of the most common programming interfaces used for 2D and 3D visualisation of scientific results and data. OpenGL is platform independent and the most widely supported, and best documented 2D and 3D graphics API available. To get started with OpenGL and CUDA operability we need to initialise the OpenGL driver by calling the standard *GL utility toolkit* (GLUT) setup functions. A typical OpenGL initialisation procedure is shown in Code Listing 1. Note that it is necessary to create a valid OpenGL rendering context and call glewInit() to initialise the extension entry points. If glewInit() returns GLEW_OK, the initialisation succeeded and it is then possible to use available extensions as well as core OpenGL functionality.

Code Listing 1 *OpenGL initialisation*

```
const unsigned int width = 512;
const unsigned int height = 512;

bool initGL(int argc, char **argv)
{

        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE |GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize(width, height);
        glutInitWindowPosition(100, 100);
        glutCreateWindow("3D Tumour-Induced Angiogenesis ");
        glClearColor(0, 0, 0, 1);

        GLenum err = glewInit();
                if (GLEW_OK != err){
                        fprintf(stderr, "GLEW error");
        return 1;
        }

        glViewport(0, 0, width, height);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(60.0, (GLfloat)width / (GLfloat)height, 0.1, 10.0);

        return 0;
}
```

Code Listing 2 *CUDA initialisation*

```
void initCUDA(int argc, char** argv){

        cudaDeviceProp  prop;
        int dev;

        memset(&prop, 0, sizeof(cudaDeviceProp));
        prop.major = 1;
        prop.minor = 0;
        cudaChooseDevice(&dev, &prop);
        cudaGLSetGLDevice(dev);
}
```

After the OpenGL initialisation, we can proceed to select a CUDA device on which to run our application. On many systems, this is not a complicated process, since they will often contain only a single CUDA-enabled GPU. However, an increasing number of systems contain more than one CUDA-enabled GPU, so we implement a method to choose one as shown in Code Listing 2.

Essentially, this code tells the runtime to select any GPU that has a *compute capability* of version 1.0 or better. It accomplishes this by first creating and clearing a cudaDeviceProp structure and then by setting its major version to 1 and minor version to 0. It passes this information to cudaChooseDevice(), which instructs the runtime to select a GPU in the system that satisfies the constraints specified by the cudaDeviceProp structure.

We need to know the CUDA device ID so that we can tell the CUDA runtime that we intend to use the device for CUDA and OpenGL interoperability. We achieve this with a call to cudaGLSetGLDevice(), passing the device ID dev we obtained from cudaChooseDevice()[24, 37].

The OpenGL and CUDA APIs share data through a commonly-accessible memory in the *framebuffer* through which OpenGL stores data in abstract buffers known as *buffer objects*. The actual CUDA and OpenGL interoperability occurs when a CUDA kernel maps a buffer into a CUDA memory space. A resource must be registered to CUDA before it can be mapped using the functions in OpenGL. These functions return a pointer to a CUDA graphics resource of the form cudaGraphicsResource. Registering a resource is potentially high-overhead and therefore typically called only once per resource. A CUDA graphics resource is unregistered using cudaGraphicsUnregisterResource(). Once a resource is registered to CUDA, it can be mapped and unmapped as many times as necessary using cudaGraphicsMapResources() and cudaGraphicsUnmapResources(). cudaGraphicsResourceSetMapFlags() can be called to specify usage hints (write-only, read-only) that the CUDA driver can use to optimise resource management. A mapped resource can be read from or written to by kernels using the device memory address returned by cudaGraphicsResourceGetMappedPointer() for buffers and cudaGraphicsSubResourceGetMappedArray() for CUDA arrays. There are two main OpenGL memory objects that CUDA manipulates; namely:

1. *Pixel buffer objects* (PBO) – a region of memory used by OpenGL to store *pixels*.
2. *Vertex buffer objects* (VBO) – a region of memory used by OpenGL for 3D *vertices*.

To pass data between OpenGL and CUDA, we must first create a buffer that can be used with both OpenGL and CUDA APIs. We declare two global variables that will store handles to the data we intend to share between OpenGL and CUDA. We need two separate variables because OpenGL and CUDA will both have different *names* for the *same* buffer [24, 37]. Code Listing 3 shows how we typically generate, bind, register and subsequently delete a *vertex buffer object* (VBO) between OpenGL and CUDA.

Code Listing 3 *Generate, bind, register and delete VBO*

```
#define grid_width 512
#define grid_height 512

GLuint vbo;
cudaGraphicsResource *resource;

void createVBO(GLuint* vbo){

        glGenBuffers(1, vbo);
        glBindBuffer(GL_ARRAY_BUFFER, *vbo);
        unsigned int size = grid_width * grid_height * 4 * sizeof(float);
        glBufferData(GL_ARRAY_BUFFER, size, 0, GL_DYNAMIC_DRAW);

        cudaGraphicsGLRegisterBuffer(&resource, *vbo, cudaGraphicsMapFlagsNone);
}

void deleteVBO(GLuint* vbo){

        glBindBuffer(1, *vbo);
        glDeleteBuffers(1, vbo);
        *vbo = NULL;

        cudaGraphicsUnregisterResource(resource);
}
```

glGenBuffers() generate the relevant buffer object names, in this case one buffer object named vbo. No buffer objects are associated with the returned buffer object names until they are first bound by calling glBindBuffer(). glBufferData() creates a new data store for the buffer object currently bound to GL_ARRAY_BUFFER. The new data store is created with the specified size in bytes and usage i.e., GL_DYNAMIC_DRAW. Effectively, the call to glBufferData() requests the OpenGL driver to allocate a buffer large enough to hold the required amount of data. In subsequent OpenGL calls, we can now refer to this buffer with the handle vbo, while in CUDA runtime calls, we refer to this buffer with the pointer resource. In its initial state, the new data store is not mapped, it has a NULL mapped pointer, and its mapped access is GL_READ_WRITE. cudaGraphicsGLRegisterBuffer() registers the buffer object specified by the buffer for access by CUDA. cudaGraphicsRegisterFlagsNone specifies no hints about how the resource will be used except that it will be read from and written to by CUDA. Since we would like to read from and write to this buffer from our CUDA C kernels, we will need more than just a handle to the object but an actual address in device memory that can be passed to our kernel. We achieve this by instructing the CUDA runtime to map the shared resource and then by requesting a pointer dptr to the mapped resource. A typical implementation of mapping and unmapping resources is shown in Code Listing 4.

Code Listing 4 *Mapping and unmapping resources*

```
        cudaGraphicsMapResources(1, &resource, NULL);
        float4 *dptr;
        size_t  size;
        cudaGraphicsResourceGetMappedPointer((void**)&dptr, &size, resource);

        dim3 block(8, 8, 1);
        dim3 grid(grid_width/block.x, grid_height/block.y, 1);
        update_kernel<<<grid, block>>>(dptr);

        cudaGraphicsUnmapResources(1, &resource, NULL);
```

cudaGraphicsResourceGetMappedPointer() returns in dptr a pointer through which the mapped graphics resource resource may be accessed. We can now use dptr as we would use any device pointer, except that the data can also be used by OpenGL as, for example a pixel source. As we can see in Code Listing 4, an *execution configuration* defines both the number of threads that will run the kernel plus their arrangement in a 1D, 2D, or 3D computational grid [24, 37]. In its simplest form, the kernel is defined using the following CUDA C syntax:

__global__ *kernel*<<<grid, block>>>();

Threads are grouped into *blocks* and blocks are grouped into *grid*s as shown schematically in Figure 6. There is a limit to the number of threads per block, for the Kepler GK110 architecture a thread block may contain up to 1,024 threads. On the GPU, each multiprocessor is responsible for handling one or more blocks in a grid which is further divided into a number of streaming processors each handling one or more threads in a block.

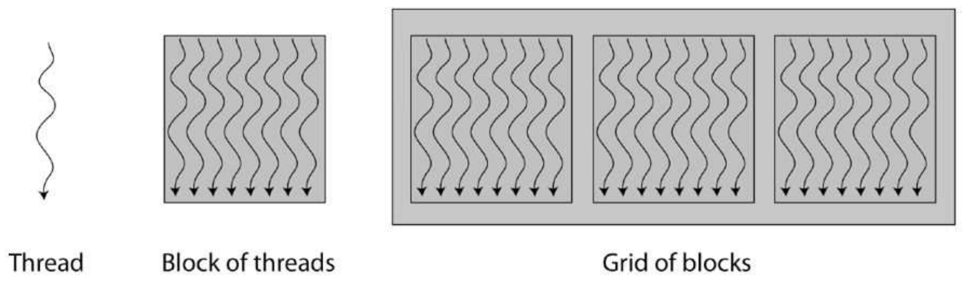


Figure 6. *A schematic representation of threads, blocks and grids.*

A block is 1D, 2D, or 3D with the maximum size of the x, y, and z dimensions being 1,024, 1,024, and 64, respectively, such that $x \times y \times z \leq 1{,}024$ i.e., the maximum number of threads per block. Blocks are subsequently organised into a 1D, 2D or 3D grid with the maximum size of the x, y, and z dimensions being 2^{31}-1, 65,535, and 65,535, respectively. An example schematic of a block and grid set up is shown in Figure 7. There are also a maximum of 65,536 registers available per block.

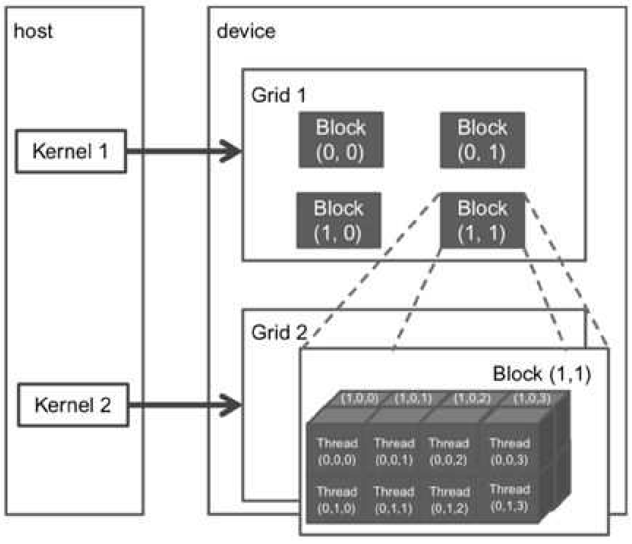


Figure 7. *An example CUDA thread grid and block.*

Threads are organised in a two-level hierarchy. At the top, a grid is organised into a 2D array of blocks. The number of blocks in each dimension is specified by the first parameter given in the kernel launch grid. At the bottom level, all blocks of a grid are organised into a 3D array of threads. The number of threads in each dimension of a block is specified by the second parameter given in the kernel launch block. Each parameter is a dim3 CUDA C data type, which is essentially a struct with three fields'. x,. y, and. z all initialised to 1. Since grids are only a 2D array of block dimensions, the third field is often ignored; but still initialised to one.

As shown in Code Listing 4, we must unmap our shared resource using cudaGraphicsUnmapResources(). This call is important to make prior to performing any rendering since it provides *synchronisation* between the CUDA and graphics portions of the application. Specifically, it implies that all CUDA operations performed prior to the call to cudaGraphicsUnmapResources() will complete before ensuing graphics calls begin. Algorithm 1 shows the CUDA update kernel for the time-stepping finite difference solution to our hybrid continuous-discrete model.

Algorithm 1 *CUDA C update kernel*

```
Get current global indices $(i, j, k)$
Get array offset
Determine coefficients: $P_0, P_1, P_2, P_3, P_4, P_5, P_6$
Determine probability ranges: $R_0, R_1, R_2, R_3, R_4, R_5, R_6$
Generate uniform random number [0, 1]
Determine cell movement: none, left, right, up, down, in, out
Translate to vertex coordinates $(x, y, z)$
Update
        if ($i < Nx - 1$ && $j < Ny - 1$ && $j < Nz - 1$ && $i > 0$ && $j > 0$ && $k > 0$) then
                Update nodes: $n^{q+1}_{i,j,k}, f^{q+1}_{i,j,k}, c^{q+1}_{i,j,k}$
        end if
```

Finally, we render each result to the screen from the repeated calls to the update kernel as shown in Code Listing 5.

Code Listing 5 *Display*

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

glBindBuffer(GL_ARRAY_BUFFER, vbo); // Bind the buffer
glVertexPointer(4, GL_FLOAT, 0, 0);
glEnableClientState(GL_VERTEX_ARRAY);
 glColor3f(1, 0, 0);

for(int i=0 ; i<grid_width*grid_height; i+= grid_width)
        glDrawArrays(GL_LINE_STRIP, i, grid_width);

glDisableClientState(GL_VERTEX_ARRAY);
glutSwapBuffers();
```

glVertexPointer() specifies the location and data format of an array of vertex coordinates to use when rendering. To enable and disable the vertex array, we call glEnableClientState() and glDisableClientState() with the argument GL_VERTEX_ARRAY. If enabled, the vertex array is used when glDrawArrays() is called which can specify multiple geometric primitives. Instead of calling an OpenGL procedure to pass each individual vertex, texture coordinate, edge flag, or colour, it is possible to prespecify separate arrays of vertices and colours and use them to construct a sequence of primitives with a single call to glDrawArrays(). When glDrawArrays() is called, it uses a count of sequential elements from each enabled array to construct a sequence of geometric primitives. glutSwapBuffers() performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers() promotes the contents of the back buffer of the layer in use of the current window to become the contents of the front buffer. The update typically takes place during the vertical retrace of the monitor, rather than immediately after glutSwapBuffers() is called.

5. Results and Discussion

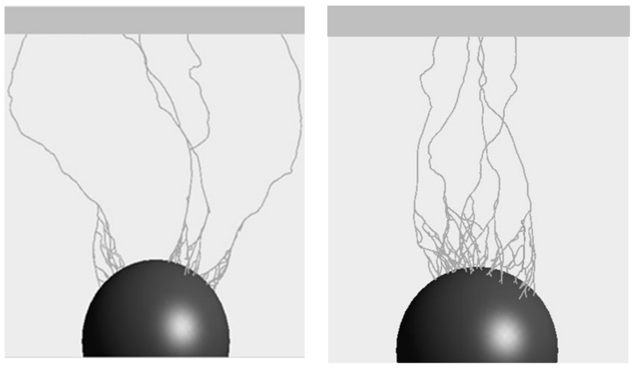


Figure 8. *3D visualisation of tumour-induced angiogenesis initiated from two different initial randomly generated sprout tips based on our hybrid discrete-continuous model.*

Figure 8 shows two 3D visualisations of tumour-induced angiogenesis initiated from two different initial randomly generated sprout tips based on the hybrid discreet-continuous model discussed above. Notice the occurrences of anastomoses has the developing sprout tips merge and loop into one another. The brush border effect is also evident has the capillary sprouts get closer to the tumour. Our results show that the hybrid discrete-continuous representation of tumour-induced angiogenesis is indeed a valid model of the process. The 3D visualisations are not only rapid but can be animated and dynamically altered during each run of the application.

Textures, a feature from the graphics world, are images that are stretched, rotated and pasted onto polygons to form 3D graphics. Textures enable fast random access to arrays and use a cache to provide bandwidth aggregation. Moreover, Kepler GPUs and CUDA 5.0+ introduce a new feature called *texture objects* that greatly improves their potential. Texture objects use the new cudaTextureObject_t class API, whereby textures become first-class C++ objects and can be passed as arguments just as if they were pointers. There is no need to know at compile time which textures will be used at runtime, this enables much more dynamic execution and flexible programming. Figure 9 shows two examples of applying different levels of *filtering* to each texture based on the new texture object API. The images show that we can uncover more features by controlling the granularity and contrast of the textures. Indeed, implementing more complex mathematical models that take into, for example blood flow dynamics could make it possible to use such techniques for accelerated 3D image processing, visualisation and dynamic interaction.

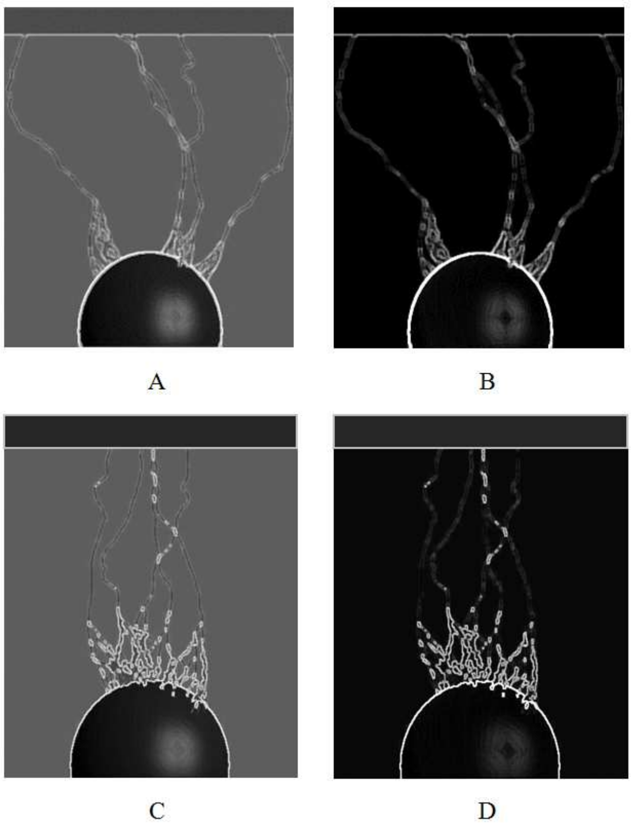


Figure 9. *The effects of applying a high (A and C) and low (B and D) level of filtering using texture objects.*

6. Medical Imaging Techniques

Examination of any photomicrograph relating to vascularisation immediately demonstrates why the modelling of fluid flow through a vascular network is such a challenging task. Fluid mechanical issues notwithstanding, the underlying network topology is itself rather complex, consisting of tortuous interconnected blood vessels embedded within a host tissue. Significant gaps remain in our understanding of the mechanisms that determine the spatial organisation of angiogenic growth and the topology of the resulting vascular network. Advances medical imagining technology for studying microcirculatory and blood flow dynamics at the cellular level will hopefully help close this gap.

6.1. Laser Speckle Imaging

When an object is illuminated by laser light, the backscattered light will form a random interference pattern consisting of dark and bright areas known as a *speckle pattern*. If the illuminated object is static, the speckle pattern is stationary. When there is movement in the object, such as red blood cells in a tissue, the speckle pattern will change over time. Such changes will be usually be recorded with a type of *charge-coupled device* (CCD) camera. Depending on the degree of movement in the imaged area, the level of blurring will differ; the more movement there is in an image, the more blurred it will appear. The level of blurring is quantified by the *speckle contrast* which has been found to correlate with blood flow. Figure 10 shows several images produced using *laser speckle imaging* (LSI); a standard lamp illumination (A) where the level of blood flow in each vessel is unknown, a raw speckle image (B) when laser excited shows a grainy blurred image from light collected by moving blood cells. By applying a *convolution filter* it is possible to obtain a high definition *speckle contrast image* (C). Finally, by making several assumptions on the velocity of blood flow, a *speckle flow index map* (D) can be produced.

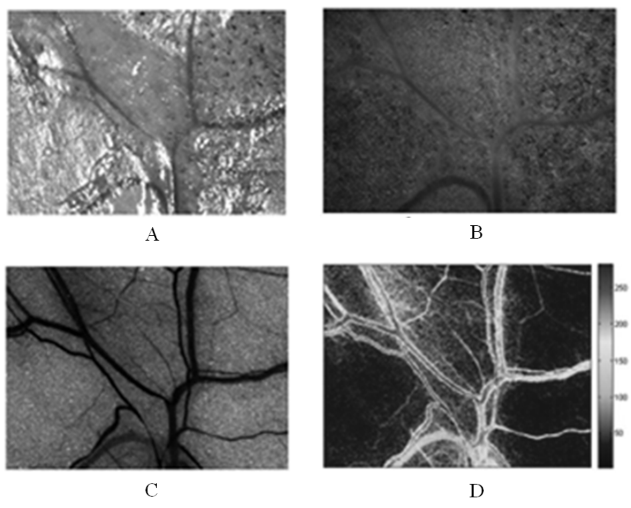


Figure 10. *Laser speckle imaging. A. Reflectance image B. Raw speckle image C. Speckle contrast image D. Speckle flow index map.*

LSI is routinely used to measure blood flow as well as being prominent in clinical research to study the microvascular response of a patient to therapeutic treatments and strategies, in both pre-clinical and clinical trials.

6.2. Laser Doppler Imaging

In contrast to LSI, *laser Doppler imaging* (LDI), the temporal intensity fluctuations of each speckle (or a collection of speckles) is monitored at high sampling frequencies (~MHz). In this case, an increase in fluctuation frequency is associated with faster blood flow. The main functions of the microcirculation are the transport of blood cells and chemicals, such as nutrients and oxygen to tissues, aid in blood pressure regulation and to act as a thermo-regulator. However, the microcirculation can show extreme dynamics. Under normal conditions, the blood perfusion can differ several 1000% between a cold and warm fingertip, for example. It also exhibits large spatial variations and may vary up to 100% in forearm skin if the measurement site is moved by as much as 1 mm. Blood perfusion measurements using laser Doppler techniques can capture these extreme dynamics and large spatial variations as shown in Figure 11.

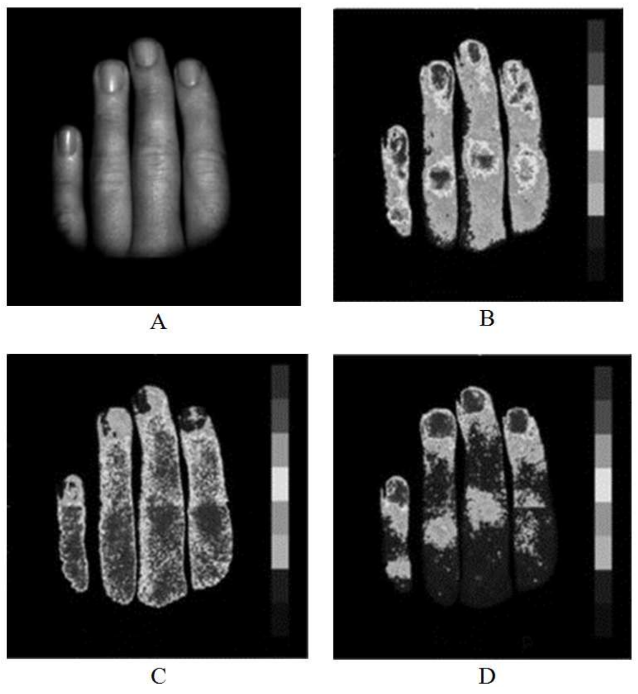


Figure 11. *Laser Doppler imaging. A. Intensity image B. Perfusion map C. Concentration map D. Speed map.*

6.2. Intravital Microscopy

Complete surgical resection of the primary tumour is still one of the most efficient cures for cancer. Unfortunately, cancer can progress to a stage at which tumour cells leave the primary tumour and spread to distant tissues and organs to form secondary tumours, a process known as *metastasis*. Complications caused by metastases are the major cause of cancer-related death, and this process is far from being fully understood. Although *histological techniques* have provided important information on metastasis, they give only a static image and therefore lack a detailed interpretation of this highly complex and dynamic process. New advances in *intravital microscopy* (IVM), such as *two-photon microscopy*, *imaging chambers*, and *fluorescent resonance energy transfer* techniques, have recently been used to visualise the behaviour of single metastasising cells at subcellular resolution over several days, yielding new and unexpected insights [38].

Tumour cells have to acquire certain traits that allow them to escape from the primary tumour site and home in on and colonise a secondary site. These gained properties, such as survival, invasiveness and motility, allow tumour cells to move into the surrounding tissue, where they enter blood or lymph vessels. Once in circulation, tumour cells are transported to a secondary site, where they can grow to form *metastatic foci* or become *dormant*. To investigate these dynamic processes underlying metastasis, most studies rely on techniques that are only able to provide a static view, visual inspection of tumour size and end-stage measurements (e.g., the number of metastatic foci). In addition, these techniques analyse large numbers of cells, which obscures the signalling properties and activities of individual cells. This results in loss of crucial information concerning the adaptive properties of the offending tumour cells that spread and form metastases. Recent advances in IVM techniques have made it possible to visualise the metastatic process at a subcellular resolution in real time *in vivo*. Moreover, a number of new IVM techniques have become available with different properties in relation to imaging depth, resolution, timescale and applications [39-41].

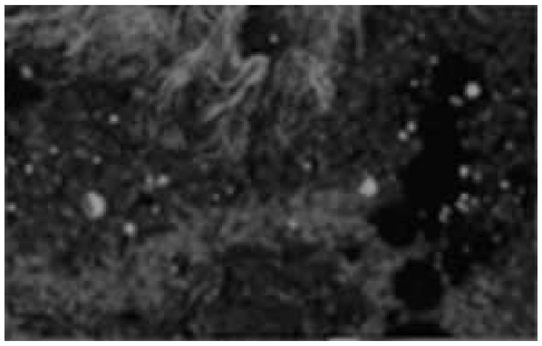


Figure 12. *An example image taken using intravital microscopy. Tumour cells (white spots) are present in a vessel that collects blood from a C26 colorectal tumour (scale 10m).*

As shown in Figure 12, to metastasise, tumour cells (white spots) have to escape from the primary tumour and colonise a distant site. During this process, cells develop traits, such as *invasiveness*, *motility*, *attachment*, *chemo sensing*, that allow them to detach from the primary tumour, invade the interstitial matrix, overcome the barrier of the basement membrane, enter the blood vessel, be transported to a distant site, exit the blood vessel and finally grow to form metastatic foci. So far, most of the advances have resulted in the ability to image the earlier steps of metastasis, including *migration*, *invasion* and *intravasation*. Only a small number of studies have attempted to image cells in organs that are prone to metastasis, such as the lungs, bone marrow, lymph nodes and spleen. Unfortunately, these experiments rely on surgical dissection to expose the imaging site, which hampers long-term imaging and therefore the visualisation of colonisation and dormancy. Indeed, the next generation of imaging chambers are aiming to visualise the spleen, liver and lymph nodes. Although IVM has been successful in providing new insights into the early stages of metastasis, most studies are only observational. In future developments, it will be important to move the field from observational IVM to experiments that can also characterise the underlying molecular processes. For example, new cancer models should be imaged that have been engineered to manipulate the behaviour of cancer cells by inducible expression of oncogenes or signalling proteins [42].

6.4. Computerised Tomographic Angiography

Angiography is an imaging technique used to visualise the inside, or lumen, of blood vessels and organs of the body. Computerised tomographic (CT) angiography is a method of combining the technology of a conventional CT scan with that of traditional angiography to create detailed images of the blood vessels in the body. In a CT scan, x rays and computers create images that show cross-sections, or slices, of the body. Angiography involves the injection of contrast dye into a large blood vessel, usually in your leg, to help visualize the blood vessels and the blood flow within them (see Figure 13).

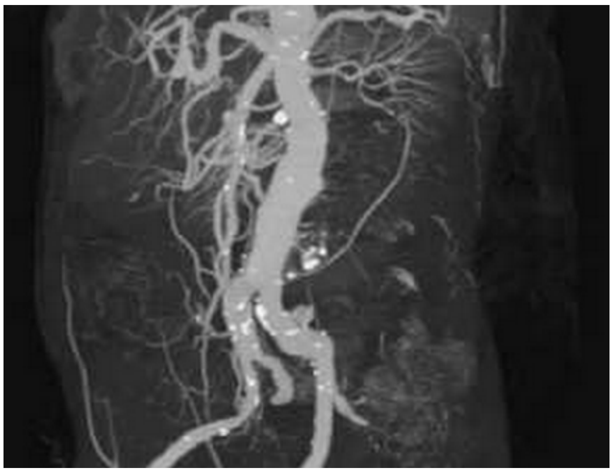


Figure 13. *An example image of the abdomen taken using CT angiography.*

An entire anatomic region can be scanned while the intravenous contrast is in the arteries and before it passes into the veins. Along with improved scanner speed, the ability to create thin-section images, around 1 mm or less, has become practical. This combination has allowed for the creation of multiplanar images to depict anatomy and pathologic conditions without the previous limitation of only being able to view the images in the axial plane. Nowadays, conventional diagnostic angiography is being replaced by CT angiography to evaluate the aorta, major vessels of the abdomen and pelvis, and the arteries of the thighs and legs. The resultant image *volumes* can be viewed in any plane and in several ways, including maximum intensity projection, shaded surface display, and volume rendering.

From our perspective, the above medical imaging techniques provide us with food for thought in which to develop our research when considering graphics interoperability for 3D visual representations using high performance computing. Indeed, future work will involve a study of the blood flow within the vasculature surrounding the tumours whilst also investigating enhanced methods to visualise the supply of targets to a tumour through the blood vessel *superhighway*. By developing more complex mathematical models with built in hemodynamics, such as blood pressure, viscosity, flow rates and other mechanical stress factors, we can further understand angiogenesis has a mechanism for targeting cancer directly. Moreover, being able to interact with such models in real time will allows us to experiment with parameters and dynamically change the topology and investigate other strategies for targeted therapy. For example, we could investigate the effects of capillary pruning and clipping along with self-fusion and devise an optimum pathway to the tumour to increase speed of delivery of anti-cancer treatments.

7. Conclusions

With more controlled texture mapping and programmable texture objects it should be possible to investigate the trajectories and interactions of capillary vessels as they move into the tumour mass. Moreover, developing complex mathematical models of fluid dynamics and many-body interactions will allow us to investigate the possibility for targeted treatment strategies on a fully interactive virtual platform. Indeed, by developing complex dynamic models of microvascular networks it should be possible to study in more detail the blood *superhighway* within the tumour using advanced computational techniques and 3D visual effects along the same lines as those presented in this paper. The authors are currently developing new and innovative algorithms that will realise the goal of having a full interactive 3D virtual laboratory to aid oncologists, researchers and others in the fight against cancer.

References

[1] Molnár, Jr. F., Izsák, F., Mészáros, R., and Lagzi, I. Simulation of reaction–diffusion processes in three dimensions using CUDA. *Chemometrics and Intelligent Laboratory Systems.* 108, 76–85. 2011.

[2] Kirtzic, J. S., Allen, D. and Daescu, O. Applying the Parallel GPU Model to Radiation Therapy Treatment. *International Conference on Parallel and Distributed Processing Techniques and Applications*. 2013.

[3] Nvidia Corporation. How GPU-Driven Drug Discovery is Finding New Targets to Cure Cancer. 2015.

[4] Nvidia Corporation. Compute the Cure: How GPU-Driven Cancer Therapies Overtook One Man’s Astronaut Dreams. 2015.

[5] Nvidia Corporation. Zapping Cost of Cancer Treatment Using Laser-Driven Ion Accelerators and GPU Computing. 2015.

[6] Worecki, M., and Wcislo, R. GPU Enhanced Simulation of Angiogenesis *Computer Science*. 13 (1). 2012.

[7] Borys, D., Psiuk-Maksymowicz, K., and Swierniak, A. Parallel Implementations of Numerical Simulation of the Vascular Solid Tumour Growth Model under the Action of Therapeutic Agents. *Biotechno: Sixth International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies*. 2014.

[8] Darbyshire,P. M. Coupled Nonlinear Partial Differential Equations Describing Avascular Tumour Growth Are Solved Numerically Using Parallel Programming to Assess Computational Speedup. *Computational Biology and Bioinformatics*. Vol. 3, No. 5, 65-73. 2015.

[9] Darbyshire,P. M. The Numerical Solution of a Hybrid Continuous-Discrete Model of Tumour-Induced Angiogenesis is Implemented in Parallel and Performance Improvements Analysed. *European Journal of Biophysics*. Vol. 7, No. 4, 167-182. 2015.

[10] Darbyshire,P. M. Performance Optimisations for a Numerical Solution to a 3D Model of Tumour-Induced Angiogenesis on a Parallel Programming Platform. *Cell Biology*. Vol. 3, No. 3, 38-49. 2015.

[11] Staton, C. A., Reed. M. W. R. and Brown, N. J. A critical analysis of current in vitro and in vivo angiogenesis assays. *International Journal of Experimental Pathology*, 90, 195–221. 2009.

[12] Hanahan, D and Folkman, J. Patterns and emerging mechanisms of the angiogenic switch during tumorigenesis. *Cell*, 86, 353–364. 1996.

[13] Albini, A., Tosetti, A. F., Li, W. V., Noonan, D. M. and Li, W. W. Cancer prevention by targeting angiogenesis *Nature Reviews Clinical Oncology* 9, 498-509. 2012.

[14] Ferrara, N. and Kerbel, R. S. Angiogenesis as a therapeutic target. *Nature*, 438 967–974. 2005.

[15] Carmeliet, P. Angiogenesis in life, disease and medicine. *Nature*, 438: 932–936. 2005.

[16] Bouard S. de, Herlin, P. and Christensen, J. G. Antiangio-genic and anti-invasive effects of sunitinib on experimental human glioblastoma. *Neuro-Oncology*, Vol. 9, No. 4, 412– 423. 2007.

[17] Norden, A. D, Drappatz, J. and Wen P. Y. Novel antiangiogenic therapies for malignant gliomas. *The Lancet Neurology*, Vol. 7, No. 12, 1152–1160. 2008.

[18] Peirce, S. M. Computational and mathematical modeling of angiogenesis. *Microcirculation*, 15(8), 739–751. 2008.

[19] M. Scianna, M., Bell. C. and Preziosi L. A review of mathematical models for the formation of vascular networks. *Oxford Centre for Collaborative Applied Mathematics*. 2012.

[20] Stephanou, A., McDougall, S. R., Anderson, A.R.A. and Chaplain, M. A. J. Mathematical Modelling of Flow in 2D and 3D Vascular Networks: Applications to Anti-Angiogenic and Chemotherapeutic Drug Strategies. *Mathematical and Computer Modelling*, 41, 1137-1156. 2005.

[21] Shirinifard, A. J., Scott Gens, J., Zaitlen, B. L., Popławski, N. J., Swat, M., and Glazier, J. A. 3D Multi-Cell Simulation of Tumor Growth and Angiogenesis. *PLoS One*, 4(10). 2009.

[22] Perfahl, H., Byrne, H. M., Chen, T., Estrella, V., Alarcon, T., Lapin, A., Gatenby, R. J., Gillies, M.C., P.K., Maini, Reuss, M., and Owen, M. R. 3D Multiscale Modelling of Angiogenesis and Vascular Tumour Growth. Chapter in *Micro and Nano Flow Systems for Bioanalysis*. Vol. 2, 29-48. 2012.

[23] Rejniak A. K. and Anderson A.R.A. Hybrid models of tumor growth. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 3(1), 115–125. 2011.

[24] Nvidia Corporation. *CUDA C programming guide*. Version 6.0. 2014.

[25] Paweletz, N. and M. Knierim M. Tumor-related angiogenesis. *Critical Reviews in Oncology and Hematology*, 9, 197–242. 1989.

[26] Paku, S. and N. Paweletz. First steps of tumor-related angiogenesis. *Laboratory Investigation*, 65, 334–346. 1991.

[27] Schor S. L., Schor A. M., Brazill G. W. The effects of fibronectin on the migration of human foreskin fibroblasts and Syrian hamster melanoma cells into three-dimensional gels of lattice collagen fibres. *Journal of Cell Science*, 48, 301–314, 1981.

[28] Bowersox J. C. and Sorgente N. Chemotaxis of aortic endothelial cells in response to fibronectin. *Cancer Research* 42, 2547–2551. 1982.

[29] Quigley J. P., Lacovara J., and Cramer E. B. The directed migration of B-16 melanoma-cells in response to a haptotactic chemotactic gradient of fibronectin. *Journal of Cell Biology* 97, A450–451. 1983.

[30] Stokes C. L., Lauffenburger D. A., and Williams S. K. Migration of individual microvessel endothelial cells: stochastic model and parameter measurement. *Journal of Cell Science,* 99: 419–430. 1991.

[31] Stokes C. L., Rupnick M. A., Williams S. K., and Lauffenburger D. A. Chemotaxis of human microvessel endothelial cells in response to acidic fibroblast growth factor. *Laboratory Investigation,* 63, 657–668, 1991.

[32] Stokes C. L., and Lauffenburger D. A. Analysis of the roles of microvessel endothelial cell random motility and chemotaxis in angiogenesis. *Journal of Theoretical Biology*, 152, 377–403. 1991.

[33] Anderson, A.R.A. and Chaplain, M. Continuous and discrete mathematical models of tumour-induced angiogenesis, *Bulletin of Mathematical Biology*, 60, 857-900. 1998.

[34] Anderson, A., Sleeman, B. D. S., Young I. M., and Griffiths, B. S. Nematode movement along a chemical gradient in a structurally heterogeneous environment: II. Theory. *Fundamental and Applied Nematology*, 20, 165–172. 1997.

[35] Muthukkaruppan, V. R., Kubai, L., Auerbach, R. Tumorinduced neovascularization in the mouse eye. *Journal of the National Cancer Institute*, 69, 699–705. 1982.

[36] Williams, S. K. Isolation and culture of microvessel and large-vessel endothelial cells; their use in transport and clinical studies. *Microvascular Perfusion and Transport in Health and Disease*, 204–245. 1987.

[37] Cheng, J., Grossman, M and McKercher, Ty. Professional CUDA C Programming. Wrox. 2014.

[38] Beerling, E., Ritsma, L., Vrisekoop N., Derksen P. W. B. and Rheenen J. van. Intravital microscopy: new insights into metastasis of Tumors. *Journal of Cell Science*, 124, 299-310. 2011.

[39] Vakoc, B. J., Lanning, R. M., Tyrrell, J. A., Padera, T. P., Bartlett, L. A., Stylianopoulos, T., Munn, L. L., Tearney, G. J., Fukumura, D., Jain, R. K. et al. Three dimensional microscopy of the tumor microenvironment in vivo using optical frequency domain imaging. *Nature Medicine* 15, 1219-1223. 2009.

[40] Abdul-Karim, M.-A., Al-Kofahi, K., Brown, E. B., Jain, R. K. and Roysam, B. Automated tracing and change analysis of angiogenic vasculature from in vivo multiphoton confocal image time series. *Journal of Microvascular. Research.* 66, 113-125. 2003.

[41] Hoffman, R. Imaging cancer dynamics in vivo at the tumor and cellular level with fluorescent proteins. *Clinical and Experimental Metastasis* 26, 345-355. 2009.

[42] Le Dévédec, S., Lalai, R., Pont, C., de Bont, H. and van de Water, B. Two photon intravital multicolor imaging combined with inducible gene expression to distinguish metastatic behavior of breast cancer cells in vivo. *Molecular Imaging Biology*, 13(1):67-77. 2011.