GPU Accelerated Viscous-fluid Deformable Registration for Radiotherapy

Technical report

Karsten Østergaard Noe[1] Kari Tanderup[2] Jacob Christian Lindegaard[2] Cai Grau[2]
Thomas Sangild Sørensen[3]

[1]Department of Computer Science, University of Aarhus
[2]Department of Oncology, Aarhus University Hospital
[3]Centre for Medical Image Computing, Department of Medical Physics and Bioengineering, University College London

Keywords: **3D Image registration, GPGPU, Radiotherapy, IGRT, Dose Planning**

Abstract

In cancer treatment organ and tissue deformation between radiotherapy sessions represent a significant challenge to optimal planning and delivery of radiation doses. Recent developments in image guided radiotherapy has caused a sound request for more advanced approaches for image registration to handle these deformations. Viscous-fluid registration is one such deformable registration method. A drawback with this method has been that it has required computation times that were too long to make the approach clinically applicable. With recent advances in programmability of graphics hardware, complex user defined calculations can now be performed on consumer graphics cards (GPUs). This paper demonstrates that the GPU can be used to drastically reduce the time needed to register two medical 3D images using the viscous-fluid registration method. This facilitates an increased incorporation of image registration in radiotherapy treatment of cancer patients, potentially leading to more efficient treatment with less severe side effects.

1 Introduction

Radiation therapy is an essential part of the treatment of cancer diseases. It can cause severe side-effects, so the radiation dose should always be conformed to the diseased volume as well as possible. However, organ and tissue deformations between radiotherapy sessions represent a significant challenge to this dose conformation. Margins are added to the tumour volume to ensure that the entire tumour receives sufficient dose in each treatment session. Currently, huge efforts are put into minimizing this margin by applying different imaging modalities to guide the planning and delivery of radiation (IGRT: image guided radiotherapy). Recent technological advancements have made it possible to perform CT imaging in the treatment room in connection with each radiotherapy session [1]. Furthermore, increased access to MR, CT and PET scanners has made it possible to perform repetitive scans during radiotherapy treatment. The repetitive acquisitions provide valuable information about organ deformation and movement over time, which can potentially be used for ongoing dynamic dose optimization of the treatment. To draw the full advantage of this, advanced methods of image analysis are required to handle organ deformations.

By the use of a *registration* method each voxel in one image is correlated to a position in another image. All 3D acquisitions from a series of treatments can be registered to the same reference dataset. This results in a geometrically resolved view (GRV [2]) making it possible to perform autosegmentation, to calculate the accumulated dose distribution, and to evaluate tumour growth/regression [3].

This paper presents the implementation and initial evaluation of such a registration method. We reimplemented the viscous-fluid registration method by Christensen et al. [4] on a parallel processor, the GPU. The main drawback of Christensens method has been very long computation times. It is demonstrated that it is possible to implement the viscous-fluid registration method in a way that utilizes the computational power of modern graphics hardware and hereby achieve significantly faster registration times. The prime motivation for using a mainstream graphics card is that it constitutes a very cost effective way of obtaining a high amount of computational power. The feasibility of using the registration method for radiotherapy 3D images is demonstrated through applications on both MRI and CT

images.

The outline of the first part of the paper will be as follows: first the problem of deformable image registration is presented together with a few registration methods which are directly related the method presented in this paper. This is followed by a more detailed introduction to the viscous-fluid method. After this our GPU based implementation is described.

2 Deformable registration

The problem of registering medical images has been the subject of many previous research projects. These have resulted in a variety of different registration algorithms. The task of medical image registration is to find a mapping between two images of the same patient acquired at different times describing the trajectory of each physical point. For use in radiotherapy planning a deformable (also called non-rigid) registration method is needed. This means that it will not be sufficient only to translate and rotate one image with respect to the other.

As described in e.g. [5] image registration can be defined as finding the functions h and g in the following mapping between two 3D images I_1 and I_2:

$$I_2(x, y, z) = g(I_1(h(x, y, z))) \tag{1}$$

where I_1 is called the *source image* and I_2 is called the *reference image*. The images I_1 and I_2 can be thought of as an $\mathbb{R}^3 \rightarrow \mathbb{R}$ mapping from 3D coordinates to image intensities.

The function g is called an *intensity mapping function* that accounts for a difference in image intensities of the same object in I_1 and I_2. In other words it is used to describe so-called *photometric differences*. In this paper, g is assumed to be the identity function.

The function h is used to describe *geometric differences*. It is a *spatial 3D transformation* that describes the mapping between the spatial coordinates (x, y, z) and coordinates (x', y', z') so that $(x', y', z') = h(x, y, z)$. These transformations take different forms depending on the registration method used.

Registration methods can be based on information derived from image intensities or from landmark information (such as contours) placed on the images. Also a hybrid models are possible using a combination intensities and landmarks.

2.1 Related Methods

In this section some classical approaches to the problem of performing a non-rigid registration are presented. The common link between these methods are that they are directly related to our implementation and to the viscous-fluid registration method in general. A general survey on image registration methods is outside the scope of this paper. For this see e.g. [6].

2.1.1 Elastic matching using linear elastostatics

The method of matching using the model of a linear elastic continuum was presented in [7]. This model is based on finding an equilibrium between external forces applied to the elastic continuum and internal forces that arise from elastic properties being modeled. The external forces applied are derived based on local similarity of voxels in the images. The internal elastic forces are based on a model in which small local elastic deformations are assumed, and where a linear relationship between the deformation and restoring forces is also assumed. An external force $\vec{b}$ is applied. This leads to the following differential equation which is used for regularization of the registration:

$$\mu\nabla^2\vec{u}(\vec{x}) + (\lambda + \mu)\vec{\nabla}\left(\vec{\nabla} \cdot \vec{u}(\vec{x})\right) + \vec{b}(\vec{x}) = \vec{0} \tag{2}$$

where the vector $\vec{u}$ is the displacement vector used in the description of the spatial transformation, and λ and μ are material constants. This equation must be solved for every voxel in the image volume.

The assumption of linear deformation and small deformation in general means that the linear elasticity registration method is not suited for registration problems with large geometric displacements. Such large deformations are often encountered in radiotherapy.

2.1.2 Methods using demons

In [8] Thirion introduced the concept of demons. These are "effectors" spatially fixed on the boundary of an object O in the reference image. A local force is applied to those voxels in the source image that have been transformed by the spatial transformation to be near the demon. This force is calculated based on a polarity of source voxels. Each voxel is labeled as either "inside" or "outside" O and the force is directed to push the voxel into O.

A demon scheme is presented in [8] in which a demon is placed in every voxel and a method called optical flow is used to find the force at each point. In this scheme boundaries of O are imaginary boundaries following surfaces with identical intensities. The polarity is determined based on the intensity gradient of the image. The allowed transformations are described using a vector field where each voxel

has an associated deformation vector describing where this voxel is mapped to in the reference image. To regularize the flow a Gaussian filter is used. For the interpolation, trilinear filtering is used. The voxel based demon method is described in detail in [9]. In [10] this method is validated with the use of images of the prostate. In this work they achieved a speedup of the registration by 40 % by adding an *active force* term moving the target image.

2.1.3 Previous work on image registration on the GPU

In [11] Strzodka et al. present an implementation of a registration algorithm that makes use of the architecture of modern graphics cards. In this work Strzodka et al. utilize the same hardware platform similarly to what is presented in section 4. Although they search for an optimal vector field mapping that minimizes the same energy function as in the viscous-fluid method presented in section 3, their method is based on non-physical regularization of the transformation. It uses an explicit Euler scheme for time discretization and a finite-element method for spatial discretization. To speed up the registration and to avoid local minima in the search, a multiscale based solver is used. Besides using a different model, another difference between the two projects is that Strodzka only works with two-dimensional images.

In contrast to Strzodka et al. we have chosen to use a method with a physically based motion model. We believe this makes it easier to understand the behavior of the method.

3 Viscous-fluid registration

A registration method that was designed to handle large geometric displacements between two images is the *viscous-fluid registration* method presented in [12] and [13]. The general idea in this method is to use a motion model that is derived from continuum physics and describes the motion of a viscous fluid for regularizing the registration process. Using this model homeomorphic mappings can be achieved even for image registrations that require large deformations to be described.

The driving force in the viscous-fluid registration is a *body force* vector field that is derived on the basis of image intensities. The force vectors are found by evaluating the gradient of a cost function describing the squared error difference in the two images to be matched.

The method is very time consuming because it requires an iterative solution of a partial differential equation (PDE) and in each iteration another PDE must be solved to find a vector field of velocities. In 1994 Christensen et al. made an implementation of the method on a so-called massively parallel (MasPar) supercomputer with which they obtained registration times in the order of 9-10 hours for a 128x128x100 volume. They used successive over-relaxation to solve for the velocity vector field. In [14] and [15] Bro-Nielsen et al. present a method that speeds up the calculation of the velocity vector field. This is done by deriving a convolution filter based on an eigenfunction basis of a linear operator that (when applied to velocities) yields a relationship between velocities and body forces.

Due to the use of multiresolution techniques it is difficult to directly compare registration times between the works of Christensen and Bro-Nielsen. Bro-Nielsen does not specify registration times for 3D registration, but in 2D his examples perform comparable with that of Christensen's MasPar implementation using only a single workstation. In [16] registration time of Bro-Nielsens method is compared to other methods for solving the viscous-fluid PDE also taking the accuracy of the solution into consideration. Here it is concluded that due to the need for a considerable size of Bro-Nielsen filter to get an accurate solution, the numerical method of successive over relaxation (SOR) is faster.

A fast implementation is achieved in [17] through the use of a full multi-grid solver for solving the viscous-fluid PDE and use of this solution method to also achieve a multiresolution implementation.

In [14] a comparison is made between the demon scheme with a demon in every voxel and the viscous-fluid registration method using Bro-Nielsen's filter for finding the velocities. Bro-Nielsen concludes that since driving forces are very similar and due the use of explicit Euler time integration in both methods, the main difference is the use of a Gaussian filter for regularization instead of his filter for solving for velocities. He proceeds to show that the Gaussian filter is a very low-order approximation to his filter.

In [4] the viscous-fluid registration method is extended to include the use of landmark information. A hybrid model is presented in which regions of interest placed by doctors are converted to binary volumes. These volumes are included when body forces are calculated which makes it easier to guarantee that important structures in the images are matched. This hybrid model is used to match images from patients being treated for cervical cancer. For some patient material the method does not completely succeed in registering the images due to ill-posed deformations of organs. To remedy this, a purely landmark based approach called *fluid-landmark registration* is used as a preprocessing step where matching is done on the basis of strategically placed points [18].

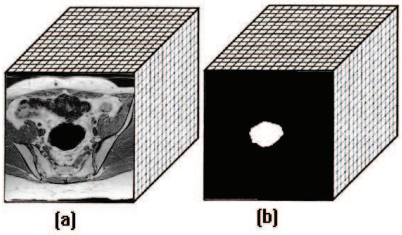

Figure 1: Illustration of the representation of images acquired in brachytherapy treatment. Lattice-representation of the image can be seen on in (a) and (b). (a) Grey-scale intensities with voxels that are inside regions of interest set to 0 (black color). (b) Binary mask representing a uterus ROI. Voxels inside this ROI are set to 1 (white) and voxels outside are set to 0 (black). If more than one region of interest had been used, more of the binary mask-lattices would have been added.

In our implementation of the viscous-fluid registration method both voxel intensities and regions of interest (ROIs) that are manually contoured by oncologists can be used to calculate the driving potential. To describe this, an image set $T_i(\vec{x})$ is defined as in [4] for every image study i as follows:

$$T_i(\vec{x}) = \begin{bmatrix} T_{i0}(\vec{x}) \\ T_{i1}(\vec{x}) \\ T_{i2}(\vec{x}) \\ \vdots \\ T_{iN}(\vec{x}) \end{bmatrix} = \begin{bmatrix} \text{Gray scale (zero for } \vec{x} \text{ in masks)} \\ \text{Binary mask for organ 1} \\ \text{Binary mask for organ 2} \\ \vdots \\ \text{Binary mask for organ N} \end{bmatrix}$$

where $\vec{x} \in \Omega$

See figure 1 for an illustration. This way of representing images means that every image $T_i(\vec{x})$ contains $N+1$ parts. The first part is the gray scale intensities in the range from 0 to 1. The last N parts are binary masks. Each of these are 1 for $\vec{x}$ that are inside the corresponding contoured ROI and 0 for other $\vec{x}$. The vector $\vec{x}$ is a position in the scanned volume Ω.

3.1 Frame of reference

The description of a transformation between image study i and j is denoted $\vec{h}_{ij}$. It is based on an Eulerian frame of reference, where values are associated with fixed spatial positions in the reference volume through which the "particles" of the source image move. In other words, the particles are tracked with respect to the final coordinates. Informally this means that for each voxel point in the lattice that forms the resulting image of the registration, a vector is stored pointing back to the point in the source image, that has been deformed to this final coordinate. When using an Eulerian frame of reference, we are sure that every voxel in the resulting image is accounted for.

This frame of reference is in contrast with a Langrangian frame of reference in which the positions of the particles are tracked with respect to the source coordinate system. Here we are not sure that the positions of the particles being tracked correspond to pixel coordinates in the reference coordinate system.

The vectors pointing from the resulting image (described in the reference system) back to the source image constitute a displacement field $\vec{u}$ for each registration. The relationship between $\vec{u}$ and $\vec{h}_{ij}$ is as follows:

$$\vec{h}_{ij}(\vec{x}, t) = \vec{x} + \vec{u}(\vec{x}, t) \tag{3}$$

When working with the Eulerian frame of reference, we need to use the *material derivative* $\frac{d}{dt}$, which is defined as $\frac{d}{dt} = \frac{\partial}{\partial t} + \sum_{i=1}^{3} v_i \frac{\partial}{\partial x_i}$. The material derivative evaluates to the time rate of change of a quantity as seen by a piece of material that is momentarily at position $\vec{x}$ at time t and which is traveling at velocity $\vec{v}$. The $\frac{\partial}{\partial t}$ term is the local derivative that accounts for changes in the desired quantity that happens as a function of time at $\vec{x}$. The term $\sum_{i=1}^{3} v_i \frac{\partial}{\partial x_i}$ is called the convective derivative. This term takes into account that the desired quantity has a different value at different positions.

The material derivative can be used to get an expression for the velocity field by taking the material derivative of $\vec{u}$:

$$\vec{v}(\vec{x}, t) = \frac{d\vec{u}(\vec{x}, t)}{dt} = \frac{\partial \vec{u}(\vec{x}, t)}{\partial t} + \sum_{i=1}^{3} v_i(\vec{x}, t) \frac{\partial \vec{u}(\vec{x}, t)}{\partial x_i} \tag{4}$$

As explained above $\frac{\partial \vec{u}(\vec{x},t)}{\partial t}$ corresponds to the normal (local) velocities being the displacements differentiated w.r.t. time. The $\sum_{i=1}^{3} v_i(\vec{x}, t) \frac{\partial \vec{u}(\vec{x},t)}{\partial x_i}$ term reflects that the relationship between $\vec{u}$ and $\vec{v}$ is not linear in the Eulerian description. Without it curved trajectories over time would not be possible.

Equation 4 can be used to find an explicit Euler equation for time integration. The Euler formula has the following form: $u(\vec{x}, t + \delta) = u(\vec{x}, t) + \delta \frac{\partial \vec{u}(\vec{x},t)}{\partial t}$. Here δ is a small time step. After finding $\frac{\partial \vec{u}(\vec{x},t)}{\partial t}$ from equation 4, this discretization gives:

$$\vec{u}(\vec{x}, t+\delta) = \vec{u}(\vec{x}, t) + \delta\vec{v}(\vec{x}, t) - \delta \sum_{i=1}^{3} v_i(\vec{x}, t) \frac{\partial \vec{u}(\vec{x}, t)}{\partial x_i} \quad (5)$$

where $\vec{v}$ is a velocity vector.

To sum up, what we have so far is a description of how to iteratively find the displacement field using a velocity field in every step. In the next section how to find this velocity field will be explained.

3.2 Motion model

Equation 5 provides us with a way of iteratively updating the displacement field using small time steps δ. Between each update, a field of velocity vectors that describe the motion of particles must be calculated. These velocity vectors are calculated from a motion model derived from continuum mechanics describing the motion of a viscous fluid. The motivation for this model is the need to be able to describe very complex local deformations while still allowing large deformations. The existence of very fine structures in the displacement field means that a lot of parameters are needed to describe the transformation (a three-dimensional vector per voxel). The viscous-fluid model is designed to perform a regularization to ensure continuity and smoothness of the transformation. It is described by the following PDE of velocities [12]:

$$\mu\nabla^2\vec{v}(\vec{x}, t) + (\lambda+\mu)\vec{\nabla}\left(\vec{\nabla}\cdot\vec{v}(\vec{x}, t)\right) + \vec{b}(\vec{x}, \vec{h}_{iR}(\vec{x}, t))) = \vec{0} \quad (6)$$

where ∇^2 is the Laplacian operator, $\vec{\nabla}$ is the gradient operator, and $\vec{\nabla}\cdot\vec{v}$ is the divergence of $\vec{v}$. The material constants λ and μ describe the viscosity of the fluid. $\vec{v}(\vec{x}, t)$ is the velocity of the template being deformed. The vector $\vec{b}(\vec{x}, \vec{h}_{iR}(\vec{x}, t)))$ is the *body force* that is used as driving potential in the registration process. How to find the body forces will be explained in the next section.

Despite the regularization performed, the viscous-fluid model allows large deformations by penalizing large velocities instead of large displacements. Where the linear elastic deformation model assumes that the restoring force is proportional to linear deformation, the viscous-fluid model instead assumes that the restoring force is proportional to the *velocity* of the motion (compare eq. 6 with eq. 2), and large displacements are relaxed over time.

3.3 Driving potential

The driving potential in the fluid registration model (eq. 6) is a field of body forces $\vec{b}(\vec{x}, \vec{h}_{iR}(\vec{x}, t)))$. The body forces are designed to minimize the following SSD (sum of squared differences) cost function [4]:

$$C(\vec{h}_{iR}) = \sum_{k=0}^{N} \alpha_k \iiint_{\Omega} |T_{ik}(\vec{h}_{iR}(\vec{x})) - T_{Rk}(\vec{x})|^2 dV \quad (7)$$

where R denotes the reference image. This cost function constitutes the similarity metric for the viscous-fluid registration method. It penalizes many and large differences in voxel intensities and binary mask values. The α_k values are weights describing the importance of the entries of T.

The body forces are found as the Gâteaux-Differential of the cost function yielding [4]:

$$\vec{b}(\vec{x}, \vec{h}_{iR}(\vec{x}, t)) = -\sum_{k=0}^{N} \alpha_k (T_{ik}(\vec{h}_{iR}(\vec{x}, t)) - T_{Rk}(\vec{x})) \cdot \vec{\nabla} T_{ik}|_{\vec{h}_{iR}(\vec{x}, t)} \quad (8)$$

The expression $\vec{\nabla} T_{ik}|_{\vec{h}_{iR}(\vec{x}, t)}$ means the evaluation of the gradient *in the Lagrangian reference frame* of T_{ik} at position $\vec{h}_{iR}(\vec{x}, t)$. From eq. 8 a direction of the force is found in which the deformation is locally pushed towards a lower value of the cost function.

4 GPU based implementation

A trend of increased programmability in recent years has lead to graphics hardware that is fully programmable. Although the main function of this programmability is enabling custom lighting effects in computer games and other graphical applications, the major graphics hardware producers have also seen a marked in the use of their hardware for general purpose computations. As a consequence of this the hardware vendors Nvidia and ATI[1] have each released a more dedicated framework for doing scientific computing on graphics hardware. These frameworks are called CUDA[2] and CTM[3] respectively. They remove some of the previous limitations of doing scientific computations on the GPU - more on this in section 4.3.

The motivation for using the graphics processing unit (GPU) of modern graphics cards for general computations is that this platform provides a high ratio between computational power and cost. Modern graphics cards have a

[1] Now AMD

[2] See `http://developer.nvidia.com/object/cuda.html`.

[3] See `http://ati.de/companyinfo/researcher/documents/ATI_CTM_Guide.pdf`.

highly parallelized architecture that enables multiple calculations to be processed simultaneously leading to very fast processing rates. The key to utilizing this parallelizability is the formulation of your numerical problem in a SIMD (single instruction multiple data) fashion. This means that the same calculations must be performed on a large amount of data elements. The SIMD architecture means that the GPU is well suited for accelerating lattice calculations.

In this paper we show that the viscous-fluid registration method described in section 3 can be implemented in a way that exploits the computational capabilities of a modern GPU. The implementation is not based on either CTM or CUDA but utilizes hardware using the standard graphics pipeline. We present the basic principles on programming the GPU by using a streaming approach which will prepare the reader for understanding any of the three frameworks for general purpose computations on the GPU (GPGPU) mentioned above.

The following sections describe the discretizations used to evaluate the above equations from a lattice based description of the vector fields involved and our mapping of these lattice computations to graphics hardware.

4.1 Discretization and algorithm

To solve the equations in the viscous-fluid registration method numerically on a spatial lattice, suitable discretizations need to be made. In this work finite difference approximations are utilized.

By rewriting the expression in (6) and applying second order finite difference approximations, a system of linear equations on the form $A\vec{v} = -\vec{b}$ can be derived that describes an approximation to (6) on a 3D lattice. Here the vectors $\vec{v}$ and $\vec{b}$ are lexicographical orderings of the components of all $\vec{v}$- and $\vec{b}$-vectors in the lattice. To find an approximate solution to this, we use the numerical method of Jacobi iteration. For each lattice point (corresponding to a voxel in the scanned images) the corresponding velocity vector is found iteratively in a number of passes where the following update rule is used:

$$v_i^{(n)} = \frac{-b_i - \sum_{j \neq i} a_{ij} v_j^{(n-1)}}{a_{ii}} \tag{9}$$

In each pass a more accurate approximation to the correct solution is found by applying the update rule to velocity vectors from the result of the previous pass. At each lattice point velocity vectors at all 26 neighbors in the lattice need to be accessed for this update rule to be evaluated. Points on the boundary of the lattice are handled by *sliding boundary* conditions allowing vectors to point along the boundary edge but not into or out of it.

Second order finite difference approximations are also used for evaluating (5) and (8) on the lattice. Here trilinear filtering is used for looking up values in between lattice points.

The method of Jacobi iteration is chosen because the update of each lattice point in an iteration is independent on the updated value of all other lattice points. This makes it suitable for an implementation on the graphics hardware we have been working with.

When updating displacement vectors from the velocity field, it is not sufficient to just use a discretized version of (5) since singularities probably will arise after a number of iterations. To prevent this from happening, it is necessary to perform *regridding* [13] when the Jacobian gets near zero at any point. During regridding the deformation currently described by the displacement field is applied to the source image. The result of this is made the new source image used in the subsequent registration iterations.

An overview of the used viscous-fluid registration method can be seen in algorithm 1. Prior to the viscous-fluid registration an automated rigid registration is performed to account for a global displacent of images w.r.t. each other. This is currently done using the Insight Segmentation and Registration Toolkit[4].

Inspired by [13] a *perturbation field* is calculated in algorithm 1 for each time step. This field is used for checking the Jacobian without updating the displacement field, and it is also used to find a suitable time step δ.

Algorithm 1: Overview of the viscous-fluid registration method.

```
Let source and reference images be specified
Let all other lattices reset to zero vectors
t ← 0
repeat
    Calculate body forces
    Solve for instantaneous velocities
    Calculate the perturbation field
    From this perturbation field calculate a step size δ
    if regridding is needed then
        Perform regridding
    else
        for s ← 1 to s_max do
            Update displacements using step size δ / s_max
    t ← t + 1
until t >= t_max or δ > δ_max ;
```

[4] See `http://www.itk.org/`

4.2 A brief introduction to 3D graphics rendering

Our implementation is based on graphics hardware through the use of ordinary 3D graphics drivers. Therefore a very short introduction to conventional 3D graphics rendering will be given. In a conventional 3D graphics application, the graphics card is usually used to render 3D geometry consisting of triangles. With the introduction of *vertex programs* and *fragment programs*, programmers of 3D computer graphics have gained control over the computations on the GPU. Vertex programs are pieces of code to be executed for each vertex (triangle corner), and fragment programs are pieces of code to be executed once for each "candidate pixel" that is contained in the triangles. The process of generating these pixels (fragments) by filling out triangles is called rasterization. Due to the architecture of the graphics cards vertex programs and fragment programs are each processed in parallel.

Vertex- and fragment programs can read data from *textures* which are images stored in memory on the graphics card. Conventionally these are used for mapping visual images onto the triangles for display, but they can also be used for storing other kinds of data (e.g. vectors) off screen.

4.3 Doing general computations on the GPU

In our viscous-fluid implementation the computational capacity of modern graphics hardware is utilized for general computations by adopting a so-called stream programming model, in which fragment programs are used as computational kernels operating on streams of fragments stored in textures.

When one would loop over all data points in a CPU based implementation of a numerical calculation, the same is achieved in our GPU based version by drawing a polygon or a line that covers the pixels corresponding to the data values that we wish process. In this way rasterization serves to invocate computation and the fragments are processed in parallel.

Feedback from previous iterations is achieved by using special buffers (p-buffers or frame buffer objects) for storing the resulting pixels from one rendering pass in a texture which can be used as input in a subsequent pass. To keep track of positions in textures to be read in fragment programs, it is required to designate *texture coordinates* to vertices, which are then interpolated throughout triangles. In this way the fragment program at each pixel knows the relationship between the position in the lattice of the data element beeing processed and the corresponding texture position.

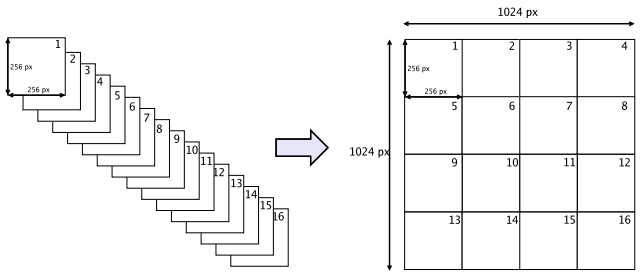

Figure 2: Mapping a 3D lattice onto a 2D texture. This example shows the placement of the slices of a 256x256x16 array on a 1024x1024 2D texture.

The individual steps of algorithm 1 are implemented as fragment programs. Each program is responsible for evaluating the results of one of the steps at the designated lattice point corresponding to a single pixel. The parallized execution of the fragment programs on the GPU allows for a substantial acceleration of the calculations.

A limitation to the above way of using the GPU for calculations is that it is not possible to output resulting values from a calculations to other grid points than the one associated with the current pixel position. We say that it is possible to *gather* but not *scatter* data. In this work we have circumvented this by choosing Jacobi Iteration for solving the linear system derived by discretizing the viscous-fluid PDE. In the CUDA and CTM frameworks scatter operations have been made possible, giving new possibilities when using the newest generation of graphics hardware for general computations. In the CUDA framework we furthermore have access to an amount of very fast memory shared between a number of kernels.

4.4 Mapping data to textures

In order to implement the fluid registration algorithm on the GPU we need to perform calculations on three-dimensional arrays. There are multiple ways of representing 3D arrays on the graphics card. We choose to represent them as "flat 3D textures"[19], which are essentially two-dimensional textures on which the slices of 3D arrays are placed next to each other. As an example, the placement of the slices of a 256x256x16 array on a 1024x1024 2D texture is shown in figure 2. Similar layouts are used for images of other dimensions.

Contrary to the alternative of using a dedicated 3D texture this method requires address translation to be made.

The advantage with the flat representation is that it is possible to update the entire lattice in a single render pass. Also due to internal memory layout in current graphics cards, this representation is more efficient than using a 3D texture.

For storing vector values, 32 bit float RGB textures are used. These are textures with a Red, Green, and Blue color component at each pixel. Each component is an IEEE floating point value with 32 bit precision.

For storing the voxel intensities and regions of interest, 8 bit RGBA textures are used. These textures contain only a single byte for each color component and are used for storing both the voxel intensities of the source and reference images and the binary volumes of the regions of interest. In the red color component, the intensity of the corresponding voxel of the source image is stored. Equivalently the intensity of the reference volume is stored in the blue component. The green and alpha components are used to store the boolean values from the regions of interest of the source and reference images respectively. As the regions of interest are defined not to overlap, a unique byte value is designated to each region of interest, and this byte is stored in the texture. The regions of the source and target images that correspond to the same organ are given the same value.

When doing calculations in fragment programs (specifically when looking up neighbor values) a conversion from 3D lattice positions to 2D texture coordinates is necessary. Two approaches to this problem are used.

Values at arbitrary positions need to be looked up when calculating body forces. These positions are found through a look-up in the texture containing displacement vectors. To handle this the fragment program is given information about the number of slices to the left and right on the current line of slices in the texture. This information is used to make a local displacement without knowledge of the global position in the texture.

In the fragment programs implementing the remaining steps of the viscous-fluid method, it is known which slices are needed in the computations. Therefore the corners of these slices are simply specified when drawing the quadrilateral that is used to invoke computation of a slice. In this way coordinates are interpolated in hardware and the fragment program can know the texture coordinates of all neighboring values.

5 Results

In this section the results of accelerating the viscous fluid registration method are presented. Timing experiments were performed in which the time required for registering images from radiation therapy on the GPU was compared to registration time on an identical CPU based reference implementation. Here "identical" means that this CPU version also uses Jacobi Iteration for solving the viscous-fluid PDE. All optimization flags were turned on in the compiler. However no hand optimization was done on the CPU based code. This results in a comparision which is somewhat unfair because we have not taken advantage of the more general memory model in our CPU implementation.

The timing experiments revealed that the GPU version is approximately 25 times faster than this CPU version on an Intel P4 3.2 GHz machine with an Nvidia geforce 7900GTX graphics card. It must be noted that initial testing of the method on an Nvidia geforce 8800 GTX graphics card in some case even yielded results that were approximately twice as fast as the times reported in this paper. The actual time needed to register two images is very dependent on the magnitude of deformation that needs to be described. With the GPU implementation typically between 100 and 400 time steps are used and for a resolution of 256x256x32 this requires between 5 and 20 minutes to be processed when using 100 Jacobi iterations per time step.

To demonstrate the feasibility of using the registration method for radiotherapy 3D images, the method is applied to both MRI and CT images. These experiments are described in the following sections.

5.1 CT registration experiments

To investigate the ability of our implementation of the viscous fluid registration method to handle the CT image modality, registration experiments were performed on pairwise combinations of three 3D CT images acquired in connection with the treatment of a head and neck cancer patient. Slices of two images can be seen in figure 3 (upper left and lower left parts).

Before the deformable registration was run an automated rigid registration was performed on the basis of bone structures. A total of 170 viscous-fluid time steps were used to register the two images of dimension 256x256x64 seen in figure 3. The result of the registration is illustrated in the upper and lower right parts of the figure. The registration method handles these CT images quite well. As can be seen in figure 3 the algorithm is succesful in describing the difference in tumor size (arrow). In general the few head and neck CT registration experiments performed gave encouraging results.

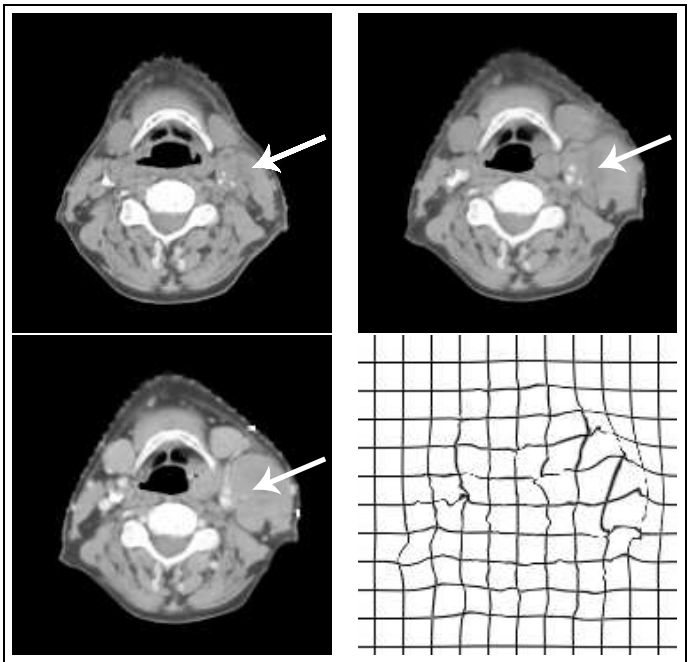

Figure 3: Images depicting the results of registering two CT images acquired in connection with the treatment of a head and neck cancer patient. All images depict the same slice. The upper left part depicts the undeformed source image followed by the deformed source image (upper right). The lower left part shows the reference image followed by an image displaying the result of applying the transformation to a rectilinear grid.

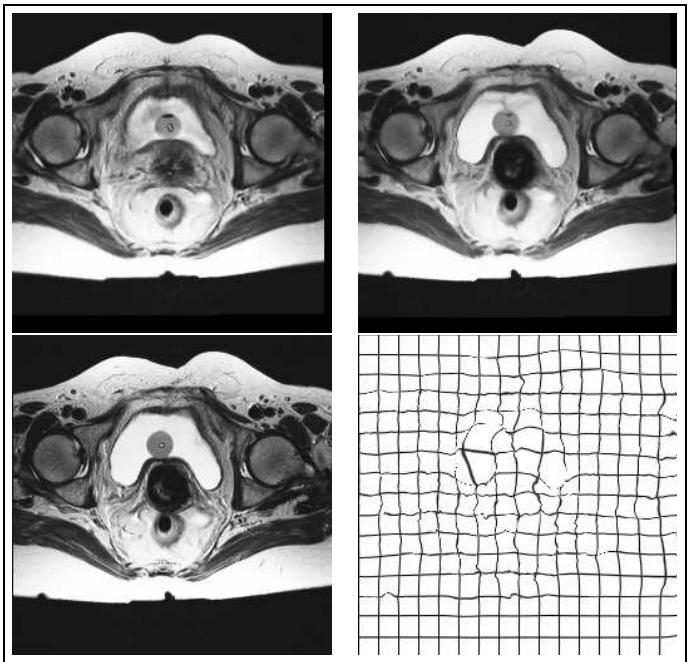

Figure 4: Images depicting the results of registering two MR images acquired in connection with the treatment of a cervical cancer patient. For a description of the different parts of the figure, see the caption of figure 3.

5.2 MRI Registration experiments

We have also tested our implementation on magnetic resonance images. This was done by registering a number of 3D MRI images from cervical cancer patients.

In most cases the viscous-fluid registration was capable of handling the cervical cancer images reasonable well. However some images could not be registered by this method without the use of regions of interest (more on this below). An example of a successful registration can be seen in figure 4. An automatic rigid registration was performed based on image intensities before the deformable registration was started. 130 time steps were used for the shown registration. It can be seen that the difference in bladder positioning is nicely described by the method even without the use of regions of interest.

The cervical cancer MR images pose a more difficult task for the registration method than the above CT image since the assumption of an identity intensity mapping is often violated due to differences in intestine content. An example of this can be seen on the left hand side images of figure 5. For registering these 256x256x32 images 110 time steps were used.

Looking at the results on the right hand side of figure 5 it can be seen that the uterus has been wrongly expanded to fill out a surrounding piece of intestine with a similar intensity (arrow). In this example the use of a region of interest describing the position of the uterus will be advantageous because the algorithm can use this prior information to distinguish between organs.

The clincal registration experiments above can be hard to evaluate because no gold standard exists describing the "correct" transformation. With the purpose of introducing a correct registration result, a number of experiments have been carried out in which images acquired by MRI in connection with radiotherapy treatment have been deformed using mathematical functions. In the example demonstrated here, rotational displacements around the center are applied. How large an angle the rotation is has a sine dependence to the distance from the center.

Images depicting the result of registering an image from a brachytherapy treatment to a version transformed with the

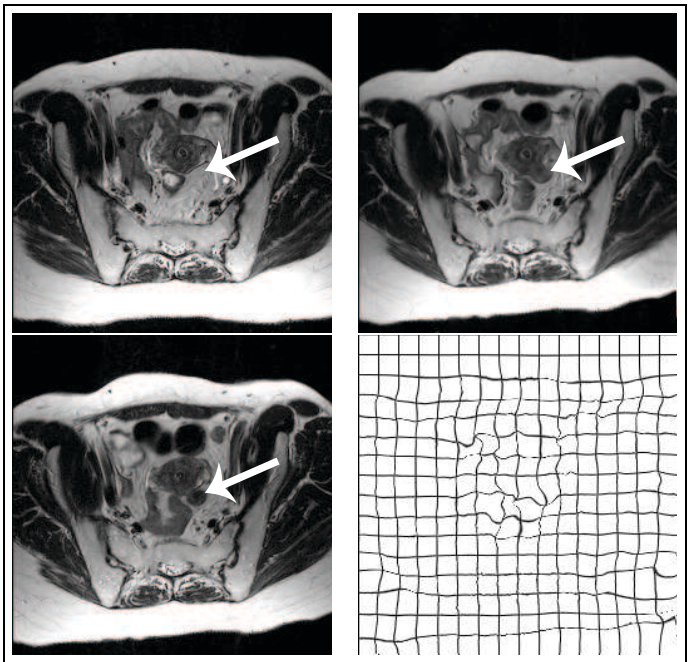

Figure 5: Images depicting the results of registering two MR images acquired in connection with the treatment of a cervical cancer patient. This is a difficult registration task due to differences in intestine content. For a description of the different parts of the figure, see the caption of figure 3.

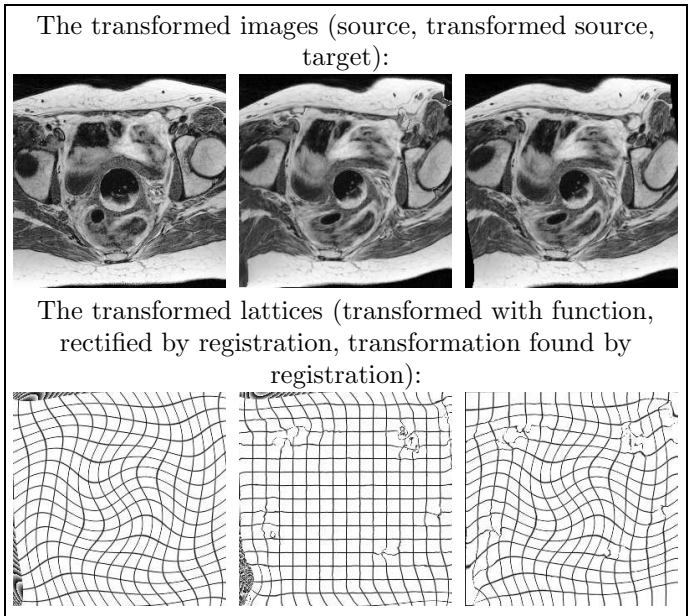

Figure 6: Images depicting the result of registering an image from a brachytherapy treatment to a mathematically transformed version of that image. On the top row the source image, transformed source image, and reference image is found. Here slice 9 of 16 is shown. On the bottom row: to the left the mathematical function is used to transform a grid. On the right side the transformation found by registration is shown. In the middle the found displacements are subtracted from the mathematical transformation.

function in the wave experiment can be seen in figure 6. Besides the source, transformed source, and reference images, an image is also shown where the mathematical function is used to transform a grid (on the bottom left). On the bottom right the transformation found by registration is shown. In the middle the found displacements are subtracted from the mathematical transformation. This can be used to see if the registration algorithm finds the same transformation as was defined mathematically. If this is the case, a rectilinear grid should emerge. It can be seen that the registration method handles the wave experiment rather well with exception to the boundary areas where our boundary conditions prevent the method from finding a match.

Work on clinical evaluation of the registration results is in progress at the Dept. of Oncology, Aarhus University Hospital [20].

6 Discussion

In this section a number of issues are presented that one needs to be aware of when using the viscous-fluid method. Also a discussion of the use of the GPU for accelerating the method is made.

6.1 Shortcomings with viscous-fluid registration

For the presented method to work well, intensities need to be consistent meaning that the same part of an organ in two images need to be the same (or very similar) intensity. This is due to the assumption of the intensity mapping function being the identity function. For CT scans this assumption is generally met, but in connection with MRI, using the same protocol and placement of coils is required.

Due to the choice of similarity measure between images, the viscous-fluid method with its current implementation of driving forces cannot be used for multimodality registration (image fusion). In the face of images of different modality, the only way of using our implementation is to base registration on binary masks alone though.

As a result of the way driving forces are calculated in the viscous-fluid registration, it does a better job at describing expansion rather than rotation. In practice this can give problems when an organ (e.g. the uterus scanned in connection with cervical cancer treatment) is bent in one image and straight in another. In this case it can happen that some parts of the source image are incorrectly compressed while other parts are incorrectly expanded to fill out the positioning of the organ in the reference image leading to an inadequate registration. To remedy this problem we are currently working on including deformable models for organ simulation in the registration method.

We are not using a multiresolution scheme in our GPU based implementation although this has a number of advantages. Such a scheme is expected to increase the registration speed as well as enabling a systematic way of handling deformations at different scales and making it less likely that the registration will be trapped in a local minimum at an early stage. It is also expected to decrease the sensitivity to rigid displacement.

6.2 Viscous-fluid registration on the GPU

The significant speedup of registration obtained by porting the algorithm to being GPU accelerated can be explained by the superior number of floating point operations per second (Flops) of the GPU compared to the CPU. The Nvidia Geforce 7900GTX GPUs used in this work are capable of producing approximately 250 GFlops[5]. This must be compared to the theoretical maximum throughput of 7.4 GFlops for a 3.8 GHz Intel Xeon CPU. In general the development in computer graphics hardware works in our favor. The capabilities in terms of computing and memory access is steeply increasing. This development is expected to continue in the coming years. Also the possibility of using multiple GPUs in a single PC has the potential of decreasing computation times even further.

However the number of Flops achievable on the two architectures is not sufficient information for determining the achievable speedup when porting to the GPU. What must also be taken into account is the ratio between the number of machine instructions and the number of memory accesses - this is sometimes called the *arithmetic intensity*[19]. In the viscous-fluid implementation - especially in the Jacobi iteration steps - the computational intensity is not high. This means that the limiting factor in the calculation is here not the number of Flops but rather the number of gigabytes per second of memory access.

[5] `http://graphics.stanford.edu/projects/gpubench/`

The viscous-fluid registration method discretized using finite difference approximations is very well suited for being GPU accelerated. One of the reasons for this is that in all steps except the calculation of body forces, all texture coordinates for looking up values can be determined by interpolation between vertex coordinates that are known beforehand. This enables the use of hardware interpolation and cache friendly memory access instead of random access [21].

6.3 Clinical perspectives

Deformable registration has the perspective of being widely integrated into many different steps of the radiotherapy process. The tasks of planning, delivery and evaluation of radiotherapy can all be improved by taking organ deformation into account. The feasibility of introducing deformable registration into these different steps depends on the computation time. Planning and evaluation of radiotherapy are tasks that run over hours or days, and the registration time is less critical although a registration time of hours would limit the usabilitity considerably. Our method which features a registration time of less than 15 minutes would make it possible to maintain a smooth workflow. For deformable registration in connection to delivery of radiotherapy the timing is critical, since the patient will be in the treatment room during the process of deformable registration. Considering the comfort of the patient, the accuracy of the treatment, and the patient flow, the registration should be kept below a minute.

To achieve deformable registration in this time frame requires further optimization of the implementation of the registration method, which is helped along by the rapid development in computer graphics hardware and the possibility of using multiple graphics cards simultaneously.

7 Conclusion

In this paper it has been demonstrated that modern graphics hardware can be used to significantly decrease the time required for registration of medical 3D image using the viscous-fluid registration method. This was done by formulating the required computations in programs runnable on a GPU. With the achieved speed, it will be clinically feasible to include the image registration in the dose delivery and dose planning process for a series of radiotherapy treatments.

Acknowledgments

The authors would like to thank Ole Østerby, and Jesper Mosegaard for their valuable advice during this project.

References

[1] C. Thilmann, S. Nill, T. Tucking, B. Hesse, L. Dietrich, B. Rhein, P. Haering, U. Oelfke, J. Debus, and P. Huber, "Correction of patient positioning errors based on in-line cone beam cts: Clinical implementation and first experiences," *International Journal of Radiation Oncology*Biology*Physics*, vol. 63, pp. 550–551, 2006.

[2] K. K. Brock, L. A. Dawson, M. B. Sharpe, D. J. Moseley, and D. A. Jaffray, "Feasibility of a novel deformable image registration technique to facilitate classification, targeting, and monitoring of tumor and normal tissue." *Int J Radiat Oncol Biol Phys*, vol. 64, no. 4, pp. 1245–1254, Mar 2006. [Online]. Available: http://dx.doi.org/10.1016/j.ijrobp.2005.10.027

[3] W. Lu, G. H. Olivera, Q. Chen, K. J. Ruchala, J. Haimerl, S. L. Meeks, K. M. Langen, and P. A. Kupelian, "Deformable registration of the planning image (kvct) and the daily images (mvct) for adaptive radiation therapy," *Phys. Med. Biol*, vol. 51, pp. 4357–4374, 2006.

[4] G. E. Christensen, B. Carlson, K. S. C. Chao, P. Yin, P. W. Grigsby, K. Nguyen, J. F. Dempsey, F. A. Lerma, K. T. Bae, M. W. Vannier, and J. F. Williamson, "Image-based dose planning of intracavitary brachytherapy: registration of serial imaging studies using deformable anatomic templates," *Int. J. Radiation Oncology Biol. Phys*, vol. Vol. 51:1, pp. pp. 227–243, 2001.

[5] L. G. Brown, "A survey of image registration techniques," *ACM Computing Surveys*, vol. 24, December 1992.

[6] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, pp. 977–1000, 2003.

[7] R. Bajcsy and S. Kovacic, "Multiresolution elastic mathcing," *Computer Vision, Graphics, and Image Processing*, vol. 46, pp. 1–21, 1988.

[8] J.-P. Thirion, "Non-rigid matching using demons," in *Computer Vision and Pattern Recognition, CVPR'96*, San Francisco, California, USA, June 1996.

[9] J. P. Thirion, "Fast non-rigid matching of 3d medical images," pp. 47–54, 1995.

[10] H. Wang, L. Dong, J. O'Daniel, R. Mohan, A. S. Garden, K. K. Ang, D. A. Kuban, M. Bonnen, J. Y. Chang, and R. Cheung, "Validation of an accelerated 'demons' algorithm for deformable image registration in radiation therapy," *Phys. Med. Biol.*, vol. vol. 50, pp. pp. 2887–2905, 2005.

[11] R. Strzodka, M. Droske, and M. Rumpf, "Image registration by a regularized gradient flow - a streaming implementation in DX9 graphics hardware," *Computing*, vol. 73, no. 4, pp. 373–389, 2004.

[12] G. E. Christensen, "Deformable shape models for anatomy," Doctoral thesis, Sener Institute, Washington University, 1994.

[13] G. E. Christensen, R. D. Rabbitt, and M. I. Miller, "Deformable templates using large deformation kinematics," *IEEE Transactions on Image Processing*, vol. vol. 5, October 1996.

[14] M. Bro-Nielsen, "Medical image registration and surgery simulation," Ph.D. dissertation, Institute of Mathematical Modelling, Technical University of Denmark, 1997.

[15] M. Bro-Nielsen and C. Gramkow, "Fast fluid registration of medical images," *Lecture Notes In Computer Science*, vol. vol. 1131, 1996.

[16] G. Wollny and F. Kruggel, "Computational cost of nonrigid registration algorithms based on fluid dynamics [mri time series application]," *Medical Imaging, IEEE Transactions on*, vol. 21, no. 8, pp. 946–952, Aug. 2002.

[17] W. R. Crum, C. Tanner, and D. J. Hawkes, "Anisotropic multi-scale fluid registration: evaluation in magnetic resonance breast imaging." *Phys Med Biol*, vol. 50, no. 21, pp. 5153–5174, Nov 2005. [Online]. Available: http://dx.doi.org/10.1088/0031-9155/50/21/014

[18] G. E. Christensen, M. W. Vannier, K. S. C. C. J. F. Dempsey, and J. F. Williamson, "Large-deformation image registration using fluid landmarks," *4th IEEE Southwest Symposium on Image Analysis and Interpretation*, p. p. 269, 2000.

[19] M. Pharr, *GPU Gems 2 - Programming Techniques for High-Performance Graphics and General-Purpose Computation*, 1st ed. Nvidia Corporation, 2004.

[20] K. Noe, K. Tanderup, C. Kirisits, J. Dimopoulos, T. Sørensen, J. Lindegaard, and C. Grau., "Accelerated deformable registration of repetitive mri during radiotherapy in cervical cancer," *Radiotherapy and Oncology*, vol. 81, pp. 210–211, 2006.

[21] T. Sørensen and J. Mosegaard, "An introduction to gpu accelerated surgical simulation," *3rd Symposium on Biomedical Simulation. Zurich, Switzerland*, pp. 93–104, 2006.