Eurographics Workshop on Visual Computing for Biomedicine (2008)
C. P. Botha, G. Kindlmann, W. J. Niessen, and B. Preim (Editors)

GPU Accelerated Normalized Mutual Information and B-Spline Transformation

Matthias Teßmann[1], Christian Eisenacher[1], Frank Enders[1,2], Marc Stamminger[1] and Peter Hastreiter[1,2]

[1]Computer Graphics Group, University of Erlangen-Nuremberg, Germany
[2]Neurocenter, Dept. of Neurosurgery, University Hospital Erlangen, Germany

Abstract

Visualization of multimodal images in medicine and other application areas requires correct and efficient registration. Optimally, the alignment operation is made an integral part of the rendering process. Voxel based approaches using mutual information ensure high quality similarity measurement. As a limiting factor, high computational load is caused since for every iteration of the optimization procedure one volume is transformed into the coordinate system of the other, a 2D histogram is generated and mutual information is computed. The expensive trilinear interpolation operations are well supported by 3D texture mapping hardware. However, existing strategies compute the histogram and mutual information on the CPU and thus require a cost intensive data transfer. Overcoming this considerable bottleneck, we introduce a new approach that efficiently supports all computations on modern graphics cards. This makes expensive data transfers from GPU to main memory dispensable. Due to its modularity, the presented approach can be integrated into general frameworks. As a major result, the speed improvement over existing GPU-CPU strategies amounts to a factor of 4 and over pure conventional CPU techniques to more than a factor of 10. Overall, the suggested strategy contributes considerably to integration of multimodal registration directly into interactive volume visualization.

Categories and Subject Descriptors (according to ACM CCS): I.4.3 [Image Processing And Computer Vision]: Registration

1. Introduction

The rapid developments in imaging systems and the ever increasing amount of acquired datasets reflects today's importance of advanced and fast processing. Based on new algorithmic approaches and latest developments in computer hardware, both the analysis and the understanding of complex relationships can be substantially improved.

Registration is an important task in image processing and computer vision if there are two or more datasets obtained at different times, from different sensors or with different viewports [Bro92]. It aims at calculating a transformation which optimally maps geometrically corresponding locations onto each other in order to allow for the correct overlay or fusion of the data. Application fields requiring image registration include target recognition systems, model acquisition systems and most importantly medical image processing. It supports diagnostic and therapeutical strategies as well as arising new applications [HHH01a].

Every registration problem needs an appropriate combination of choices of a feature space, a search space, a search strategy and a similarity metric [Bro92]. Given pixel or voxel data as input, the intensity information defines the feature space. As illustrated in figure 1, one dataset is defined as fixed image which means that it remains unchanged throughout the alignment process. The other one is denoted as moving image since it is transformed depending on the current parameters. The search space is represented by the class of transformations describing the misalignment of the two images. In order to find the optimal transformation parameters the search strategy is important. It decides on the update of the transformation and is briefly referred to as optimizer. Finally, the quality of alignment is evaluated by the similarity metric. In this context, voxel based approaches have proven

to provide the highest level of accuracy with mutual information (MI) being regarded as appropriate for many registration problems [WFW*99]. As an advantage, voxel based approaches work without pre-segmentation which could limit the accuracy due to delineation errors. However, they require a huge number of interpolation operations for the sampling of the moving image within the grid of the fixed image and a subsequent cost intensive evaluation of the similarity measure for every iteration of the optimization procedure.

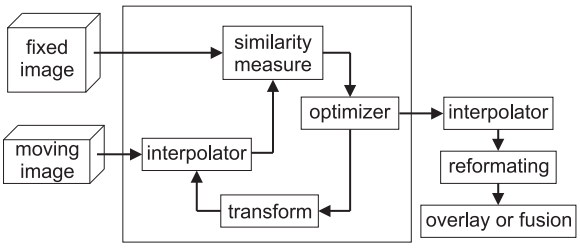

Figure 1: *Typical registration framework using intensity based similarity measures.*

In rigid body situations the performance scales with the size of the underlying image data and becomes critical for volume data. For nonlinear alignment problems the computational load increases dramatically due to the flexibility of the underlying transformations and the respective high number of optimization parameters. For example, the alignment of preoperative and nonlinearly deformed intraoperative image data during neurosurgical procedures is of major relevance in order to access information which is only preoperatively available [HRSS*04]. Since efficient calculation is an important issue, acceleration techniques are seen as urgent challenge in medicine and other application areas. In the optimal case, registration is made an integral part of the data generation and presentation process. This could considerably support the analysis of time critical problems and complex scenarios within a simulation-visualization cycle.

In this paper, an innovative strategy exploiting the 3D power and technical features of modern PC graphics cards is introduced which efficiently extends our previous work. Focusing on MI, the entire calculation of the similarity measure is offloaded onto the graphics processor unit (GPU). The approach uses the trilinear interpolation capabilities available with texture mapping subsystems and overcomes the crucial problem requiring a cost intensive read back operation from frame buffer to main memory.

The paper has the following structure: In section 2, previous work is outlined. Section 3 presents the mathematical background of MI. Important issues of the GPU implementation are explained in section 4. The respective calculation of the 2D histogram and the evaluation of MI are introduced in sections 5 and 6. Further on, section 7 shows how this approach allows for free form deformations using B-splines. Finally, results are presented in section 8.

2. Previous Work

The idea to accelerate voxel based registration using 3D texture mapping and to integrate it with 3D visualization was introduced by Hastreiter et al. [HE98]. This was an extension to pure software alignment using MI [SW49] which used hierarchical subsampling [CMD*95] and stochastic approximation [WVK95] for acceleration. Considering hardware based registration, FPGA based techniques [JLR02] and multiprocessor environments [RM03] also proved to be a valuable assistance to accelerate the evaluation of a similarity measure. In contrast, a considerable speed-up was reported by Glocker et al. with a novel CPU approach using discrete labeling and linear programming [GPK*07].

The class of GPU accelerated transformations was extended to piecewise linear transformations by Rezk-Salama et al. [RSSSG01]. They were applied by Hastreiter et al. [HRSS*04] for deformable voxel-based registration to examine intraoperative brain deformations denoted as brain shift. Further improved non-linear alignment was achieved in the work of Soza et al. [SBH*02] by using 3D Bézier functions implemented on the GPU for the non-linear alignment of volume data. This strategy also proved to be adequate for parameter estimation in brain shift simulation [SGN*04] and correction of diffusion tensor imaging data [MHS*04, MSS*07].

A non-linear GPU based regularized gradient flow method for the registration of 2D images was presented by Strzodka et al. [SDR04]. This approach was extended to 3D and fully offloaded to the GPU in the work of Köhn et al. [KDRMK06]. However, limited performance was assessed as a result of memory bottlenecks. Exploiting vertex shaders and hardware-accelerated 3D texture mapping Levin et al. [LDS05] suggested an approach of volume warping based on thin plate splines for iterative image registration. In a more general scope, Sigg et al. [SH05] presented a fast strategy for the evaluation of B-spline transformations on modern graphics cards which allows for a reduction of texture lookups. Using MI and the Kullback-Leibler divergence Vetter et al. [VGXW07] presented a GPU based non-rigid registration for multi-modal medical image data. As a drawback, this approach uses 2D-textures only and omits exploiting the full 3D power of graphics cards. The important issue of efficient optimization using MI and B-splines was investigated by Klein et al. [KSP07]. In a recent work of Muyan-Ozcelik et al. [MOOXS08], the Demons registration was efficiently mapped to the GPU using the Compute Unified Device Architecture (CUDA) programming environment.

3. Mutual Information

Voxel-based registration takes into account the entire grey value information of the image data. However, the choice of an appropriate similarity metric is important for the evaluation of the registration quality during the optimization pro-

cess. For both mono- and multimodal image data MI has shown to be adequate [WFW*99].

Defined in information theory, MI describes the statistical dependence of two images or the amount of information that one image contains about the other. More formally, let

$$H(X) = -\sum_{x} p_x ln(p_x) \tag{1}$$

denote the marginal- and

$$H(X,Y) = -\sum_{x,y} p_{x,y} ln(p_{x,y}) \tag{2}$$

the joint entropy of two images X and Y. Then the MI of the two images ξ and ζ is defined as

$$C_{MI}(\xi,\zeta) = H(\xi) + H(\zeta) - H(\xi,\zeta) \quad . \tag{3}$$

Varying overlap of the fixed and the moving image can cause disproportional contributions to the MI. This happens predominantly in areas of very low intensities such as background noise [HHH01b]. In order to make the registration more insensitive to changes in image overlap, normalized MI (NMI) has been proposed which has been shown to be considerably more robust than standard MI [SHH99]. It is defined as

$$C_{NMI}(\xi,\zeta) = \frac{H(\xi) + H(\zeta)}{H(\xi,\zeta)} \quad . \tag{4}$$

With the similarity measure, an optimization term can be constructed that allows maximizing the NMI by numerical computations [RSH*99]. Direction set methods are frequently used if the explicit evaluation of partial derivatives must be avoided. There are numerous strategies which mainly differ in the set of direction vectors they use. Since the focus of this work does not cover optimization strategies, a readily available implementation of Powell's algorithm was chosen, which generates a set of mutually conjugate direction vectors for the iterations [PFTV88].

4. Implementation

For image registration, it is necessary to load both datasets into memory and to determine their similarity based on the NMI. Then, the chosen transformation is applied to the dataset defined as moving image. Finally, the similarity of the images is recomputed and checked whether it has improved. In the registration process the transformation of the moving image is one major bottleneck due to the huge amount of interpolation operations. Since the memory of modern graphics cards has enormously increased so that medical volume data can be entirely stored in 3D textures memory, it became feasible to perform the interpolation of non grid positions in the graphics subsystem [HE98]. However, it is not reasonable to offload the transformation to the GPU exclusively and to calculate the NMI on the CPU. The necessary copying of the data to the graphics card and the cost intensive read-back operations of the transformed data would eliminate any performance gain [SBH*02].

Recent developments of GPUs made it possible to overcome this limitation. Framebuffer objects (FBO) make it practical to perform off-screen rendering without the additional cost of a context switch. Additionally, OpenGL 1.5 promoted the well known vertex buffer objects to general *buffer objects* which serve as a generic data storage in GPU memory. Version 2.0 then added shading support via shader objects and the GL shading language (GLSL) to the API. To date the final contribution important for the approach suggested in this paper, OpenGL 2.1 promoted the *ARB_pixel_buffer_object* extension to a generic target for buffer objects. Based on these advances a fully standard compilant and portable GPU-based registration implementation became achievable.

The *pixel buffer objects* are crucial for the presented algorithm. Applying them allows reusing framebuffer objects as vertex data, which is the core part of the histogram generation method. In the beginning, a buffer object is generated which is bound to the GL as PIXEL_PACK_BUFFER. This causes all subsequent calls to *ReadPixels*() to target at the on board memory of the graphics card to which the buffer object is assigned to. Hence, rendered framebuffer data can be read back directly to graphics memory without the necessity to make a detour to generic system RAM. The so filled buffer object can then be rebound as an ordinary vertex array, which in turn can be used for rendering using the *DrawArrays*() call.

The process of NMI calculation is divided into two parts: (1) Computation of the 2D histogram of the superimposed fixed and moving datasets on the GPU. (2) Evaluation of the NMI on the graphics card with the obtained histogram.

5. Histogram Generation

Following is the basic GPU procedure for the 2D histogram:

1. Create an off-screen buffer for sample generation.
2. Create an off-screen buffer containing the desired number of bins for the histogram.
3. Upload the image data to 3D texture memory and generate samples from the overlapping image area.
4. Interpret the sample values of the overlapping area as geometric coordinates to the histogram area.
5. Count the number of samples into the histogram.

Sampling from Overlapping Image Areas Sampling of volume data is effectively carried out by the GPU, since it is highly optimized for trilinear interpolation. Hence, the fixed and the moving image is uploaded to the graphics card and stored as 3D luminance textures. The fixed volume is placed at the origin, with a width, height and depth equal to 1.0 respectively. This maps the fixed volume onto its own texture

space. The rigid transformation of the moving image volume is stored in the current model-view matrix of OpenGL. To obtain samples of the overlapping area, the volumes are rendered slice by slice into an off-screen buffer with multitexturing enabled. Thereby, both datasets can be accessed as textures during the rendering pass. A simple vertex shader passes the current vertex coordinate of the fixed volume to the fragment stage. For the moving volume, the shader multiplies the vertex coordinate with the inverse model-view matrix to transform it into the texture space of the fixed volume.

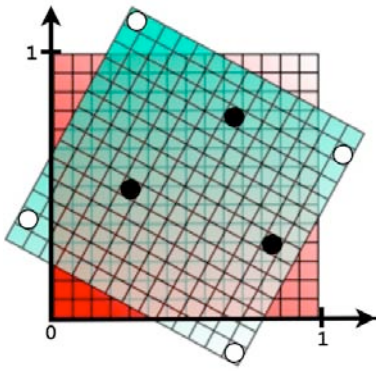

Figure 2: *Determination of valid overlapping samples. Samples drawn in black are valid, since the are located within the overlapping area of the images. Samples drawn in white are invalid, since they are located outside the valid texture space range of* $[0,1]$.

Since a unit cube mapping of the datasets is used in texture space , the corresponding fragment shader just needs to determine whether the generated texture coordinate is in the range of $[0,1]$. If the coordinate is inside this range, it is a valid sample located in the overlapping area of the two datasets. Consequently, the sample value of the fixed volume is written to the red channel of the output buffer and the corresponding sample of the moving volume is written to the green channel. If the coordinate value falls outside the range of $[0,1]$, the associated sample value cannot lie within the overlapping area. Hence, it is discarded and the value of -1 is written to the respective output channels, indicating that the value is invalid (Fig. 2).

Creation of the Histogram In order to access the data of every rendered slice a buffer object is created. The buffer is then used as a PIXEL_PACK_BUFFER and the contents of the sample slice is read back into graphics card memory via *ReadPixels*(). Finally, the buffer is rebound as vertex array.

For the subsequent generation of the 2D histogram on the GPU an additional off-screen render buffer with the desired number of histogram bins is allocated. This buffer is initialized with zeros and a parallel projection is set up. The fragment values of the generated sample slices located in the buffer object are interpreted as geometric coordinates of the histogram slice, i.e. the vertex array is drawn. The intensity values of the red and the green channel of the sample slice which are in the range $[0,1]$, are mapped to the range between zero and the maximum number of histogram bins in $x-$, and $y-$directions, respectively. For all value pairs, a point with an intensity value of 1 is drawn at the derived location (Fig. 3).

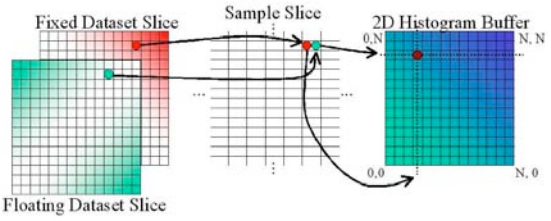

Figure 3: *Generation of the 2D histogram from the overlapping area of the rendered sample slices.*

Additive blending is enabled during the drawing process, hence the corresponding histogram bin value is increased by one each time a point gets drawn. Since sample values outside the overlapping area were set to -1, the respective points lie outside the rendering area and are thus clipped automatically.

6. Calculation of NMI

After the computation of the 2D histogram, the NMI is also evaluated in graphics hardware. In order to take advantage of the GPU processing capabilities it is necessary to reformulate the equations for the marginal- and joint entropies $H(\xi)$, $H(\zeta)$ and $H(\xi,\zeta)$ from section 3. Based on the obtained histogram, the probabilities of individual sample values can be calculated by simply taking the sample count $N_{\xi,\zeta}(i,j)$ stored in each histogram bin and dividing it by the total number $A_{\xi,\zeta}$ of all samples within the histogram.

More formally, the total number of all samples is

$$A_{\xi,\zeta} = \sum_{i\in\Omega_\xi}\sum_{j\in\Omega_\zeta} N_{\xi,\zeta}(i,j) \quad . \tag{5}$$

Inserting $A_{\xi,\zeta}$ into equation 2 and defining

$$DS_{\xi,\zeta} = \sum_{i\in\Omega_\xi}\sum_{j\in\Omega_\zeta} N_{\xi,\zeta}(i,j) log\left(N_{\xi,\zeta}(i,j)\right) \quad , \tag{6}$$

the joint entropy can be written as

$$H(\xi,\zeta) = -\frac{1}{A_{\xi,\zeta}} DS_{\xi,\zeta} + log\left(A_{\xi,\zeta}\right) \quad . \tag{7}$$

Note that the summands of $DS_{\xi,\zeta}$ and $A_{\xi,\zeta}$ are only dependent on $N_{\xi,\zeta}(i,j)$. The same can be done for the marginal

entropies. If we denote the sum of all samples from a single image as $A_\xi = \sum_{i \in \Omega_\xi} N_\xi(i)$, we can accordingly define

$$SS_\xi = \sum_{i \in \Omega_\xi} N_\xi(i) log\left(N_\xi(i)\right) \quad . \tag{8}$$

The marginal entropy of a single image then results to

$$H(\xi) = -\frac{1}{A_\xi} SS_\xi + log(A_\xi) \quad . \tag{9}$$

This reformulation expresses the entropy formulae as sums of sample values already contained in the histogram and logarithms thereof. This provides the basis for an efficient GPU implementation.

6.1. Evaluation with Graphics Hardware

To evaluate the NMI with the adapted formulae on the GPU an off-screen buffer with 32 bits floating point precision is created. The existing histogram buffer from the preceding step is bound as a texture input for the upcoming calculations. In a first render pass, the sample values in all four color channels of the histogram buffer are added to get the total sample count of the corresponding histogram bin. Furthermore, the fragment shader that performs the addition operation also calculates the logarithm of this number and multiplies it with the total number of samples in the corresponding bin. Hence, the GPU computes $N_\xi(i)$ and $N_\xi(i) log(N_\xi(i))$. The values are written to the red and the green channel of the output buffer. To compute $N_\zeta(i)$ and $N_\zeta(i) log(N_\zeta(i))$ in the same step, the x and y texture coordinates for the histogram lookup are simply swapped. Then, a column-wise lookup of the histogram bins is achieved. The results are written to the blue and the alpha channel of the output buffer.

In a second render pass, the computed values are added. This is done by a technique called *parallel reduction*, described in [BFH*04]. For this, a second floating point texture buffer is needed. This buffer is bound as the output framebuffer and the sum buffer from the previous step is bound as an input texture. The sum is computed by a *2:1 reduction*, i.e. rendering the upper half of the input image to the output buffer while a fragment shader is adding one row of the upper half of the input texture to its corresponding row of the lower half. Subsequently, the roles of the two buffers are reversed (*ping-ponging*) and the step is repeated until only one row is left. Finally, the last rendered row contains the partial sums of SS_ξ, SS_ζ, A_ξ and A_ζ for each sample value i. This *sum-row* is read back to main memory and the remaining computation is most efficiently carried out in software. This includes addition of the partial sums contained in the row, a division and evaluation of a logarithm, which yields $H(\xi,\zeta)$, $H(\xi)$ and $H(\zeta)$. What is left is only one addition and one division to evaluate C_{NMI}.

Having the evaluation of the NMI on the GPU, a rigid alignment is straightforward. Due to the transformation of the second texture coordinate by the inverse model-view matrix, the rigid transformation is already included in the sample generation process. The optimizer just needs to adjust the six parameters. Transformation and NMI evaluation occur in the same rendering step on the GPU.

7. Non-Rigid Registration on the GPU

For non-rigid image registration, free-form deformations (FFD) using a cubic B-spline transformation has proven to deliver high quality results [RSH*99]. Therefore, it is desirable to integrate such mapping into the presented GPU framework. Recall that a cubic B-spline based FFD of 3D volume data can be expressed as 3D tensor product of 1D cubic B-splines.

$$\mathbf{T}(\vec{p}) = \sum_{l=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} N_l^3(s) N_m^3(t) N_n^3(u) \vec{d}_{i+l,j+m,k+n} \quad . \tag{10}$$

Here, $\vec{p}$ is a position within the image domain $\Omega = \{\vec{p} = (x,y,z) | x \leq 0 < X, y \leq 0 < Y, z \leq 0 < Z\}$. $\vec{d}_{i,j,k}$ are the control points of a $N_x \times N_y \times N_z$ control point mesh of spacing $\delta_x \times \delta_y \times \delta_z$. Then, $u = x \,/\, \delta_x - \lfloor x \,/\, \delta_x \rfloor$, $v = y \,/\, \delta_y - \lfloor y \,/\, \delta_y \rfloor$, $w = z \,/\, \delta_z - \lfloor z \,/\, \delta_z \rfloor$, $i = \lfloor x \,/\, \delta_x \rfloor - 1$, $j = \lfloor y \,/\, \delta_y \rfloor - 1$ and $k = \lfloor z \,/\, \delta_z \rfloor - 1$.

This definition provides an intuitive behavior of the deformation and inherent elasticity, which matches normal soft tissue movement. Special care has to be taken when the control point mesh is generated. For the initial transformation a uniform, regularly spaced control point mesh is used. This ensures that no transformation happens. Additionally, extra control points have to be added on the boundary of the 3D grid. This is necessary since at the image border partition of unity would be violated. The extra control points circumvent this. Since they stay constant during the registration process, these additional points do not influence the calculation time significantly.

To achieve a correct deformation of the dataset, the B-spline evaluation has to be performed per voxel. This means that a neighborhood of 64 control points which affect a voxel is involved and has to be considered for each image element, i.e. each fragment. Using the programmability of GPUs this procedure is supported by traversing the datasets slice by slice. The required computations are performed per fragment. While the contributing value of the fixed image is obtained by a single trilinear texture lookup, the value of the moving image requires the expensive B-spline evaluation.

An essential problem is to handle all 64 control points per fragment. Common general purpose approaches use textures in order to store input values. However, in our case, the calculation itself is comparatively simple. Since the texture fetches for the input values are the major bottleneck, they were omitted and uniform variables were used instead. In consequence, the complexity of the fragment program was reduced resulting in a much higher fragment throughput.

8. Results

The suggested approach was evaluated with 11 datasets of tumor patients consisting of pre- and intraoperative MR volumes with $512 \times 512 \times 256$ voxels and a voxel size of $0.49 \times 0.49 \times 1.0mm^3$ acquired with a Siemens MR Magnetom Sonata Maestro Class 1.5 Tesla scanner. Additionally, corresponding CT data were used to test multi-modal registration. For the calculations, a PC equipped with an Intel Core 2 Duo, 3.0 GHz processor, 2 GBytes RAM and a Nvidia 8800 Ultra was used.

The evaluation focused on time measurements of the NMI calculation (rigid and non-rigid transformation) at different resolution levels of the volumes. The results are summarized in table 1 and serve as a basis for different optimization strategies.

Resolution	NMI + Rigid	NMI + Non-Rigid
$512^2 \times 256$	1.2 sec	2.3 sec
$256^2 \times 128$	0.4 sec	0.7 sec
$128^2 \times 64$	0.16 sec	0.2 sec
$64^2 \times 32$	0.08 sec	0.13 sec

Table 1: *Timings of NMI evaluation, rigid- and unoptimized non-rigid transformation.*

Though the presented non-linear deformation is regarded as an add-on of this work, it is noteworthy that the obtained calculation times are very low. Compared to a CPU implementation the speedup is significant. For example, evaluating the NMI and performing a B-spline transformation on a $512^2 \times 256$ dataset using an optimized software implementation took about 22 seconds.

For completeness, we performed fully rigid and non-rigid registrations using the implementation of Powell's algorithm from [PFTV88]. Depending the optimization parameters results of different quality and total registration times were achieved. The rigid registrations were obtained in about 20 seconds (Fig. 4). The non-linear registration result of figure 5 c) took 20 minutes to complete. The higher quality registration result of figure 5 d) was computed in 1 hours.

9. Conclusion and Future Work

An approach for the efficient calculation of the NMI on modern GPUs including rigid and non-rigid transformations has been presented. Due to its modularity it can be used within different registration frameworks. Considering the evaluation strategy of the similarity measure, the GPU implementation includes rigid transformations almost "for free". Additionally, a FFD approach using a cubic B-spline transformation has been presented which integrates seamlessly into the presented pipeline. Hence, for the example of medical image registration it is possible to combine initial rigid alignment and subsequent non-rigid registration of the data

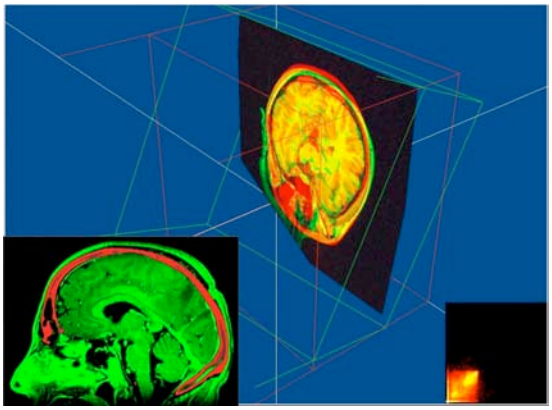

Figure 4: *Multi-modal rigid registration and simultaneous visualization including interactive adjustment of the 2D histogram (lower right) and fusion of the final result (lower left).*

which greatly reduces process times. Moreover, using the method within a framework permits easy integration of different GPU- and software-based methods which allows for convenient analysis.

Concerning the presented FFD evaluation on the GPU, it was shown that the 64 texture fetches can be replaced by just eight trilinear texture fetches [SH05]. Applying this method to the presented implementation might lead to a comparatively higher speed of non-rigid transformations.

Compared to todays more common general purpose GPU computing approaches such as CUDA, the presented approach has the advantage that volume visualization can be carried out while the registration is in progress. This allows analyzing the behavior of different registration and optimization algorithms in more detail. Additionally, the presented approach is flexible enough to permit on-line user interaction during the alignment process. First experiments with CUDA based registration have shown that a major bottleneck is the transfer of the computed results to a graphics context in which they can be prepared for visualization. With the evolution of CUDA, this limitation might be overcome in the future which will then allow a direct comparison of a CUDA and direct GPU implementations.

References

[BFH*04] BUCK I., FOLEY T., HORN D., SUGERMAN J., FATAHALIAN K., HOUSTON M., HANARAHAN P.: Brook for GPUs: Stream Computing on Graphics Hardware. *ACM Transactions on Graphics 23*, 3 (2004), 777–786.

[Bro92] BROWN L.: A Survey of Image Registration Techniques. *ACM Computing Surveys 24*, 4 (1992), 325–376.

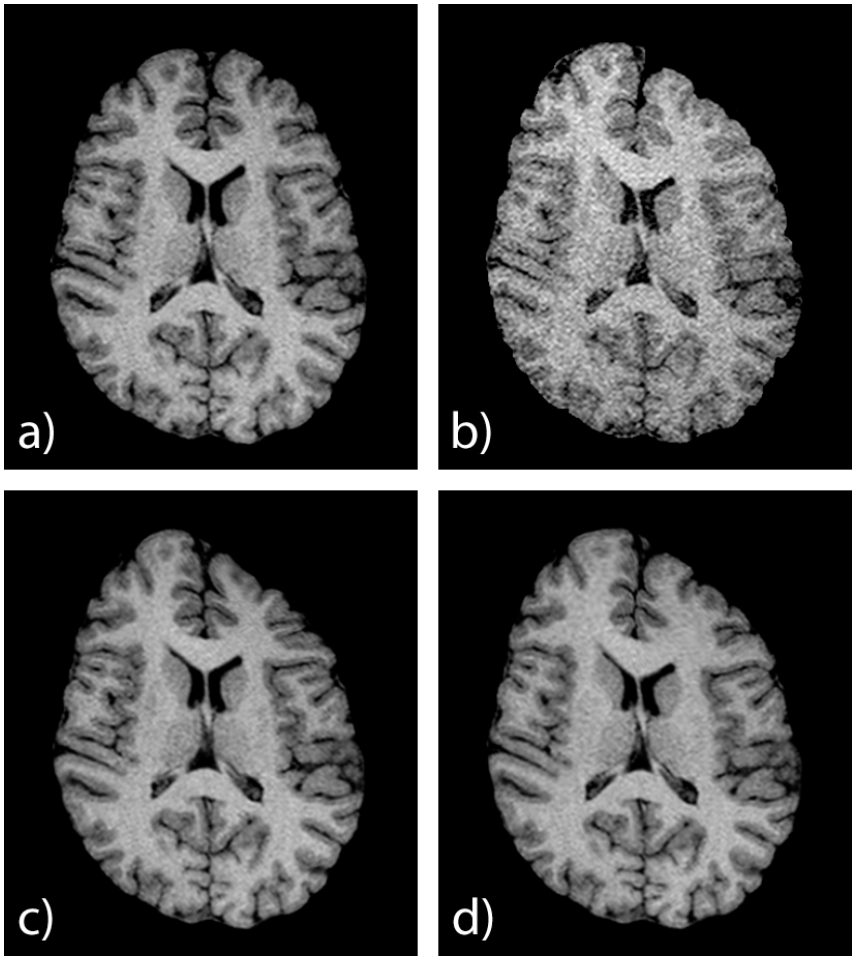

Figure 5: *Example of a patient dataset acquired before (a) and during (b) surgery and results of non-rigid registration with coarse (c) and optimal (d) parameter setting.*

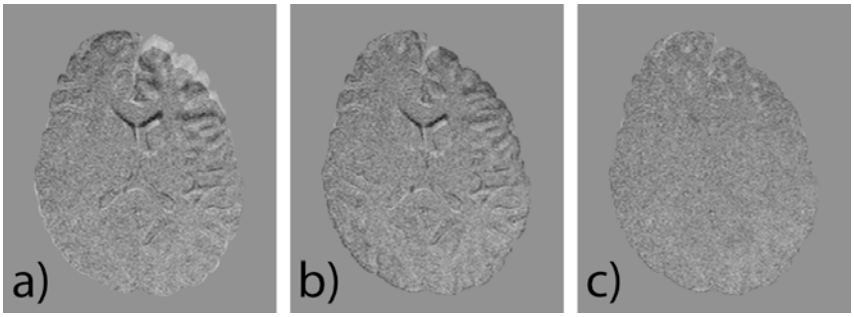

Figure 6: *Difference images of rigid registration (a) as well as non-rigid registration with coarse (b) and optimal (c) parameter setting.*

[CMD*95] COLLIGNON A., MAES F., DELAERE D., VANDERMEULEN D., SUETENS P., MARCHAL G.: Automated Multi-modality Image Registration Based on Information Theory. In *Information Processing in Medical Imaging* (1995), vol. 3, pp. 263–274.

[GPK*07] GLOCKER B., PARAGIOS N., KOMODAKIS N., TZIRITAS G., NAVAB N.: Inter and Intra-modal Deformable Registration: Continuous Deformations Meet Efficient Optimal Linear Programming. *Inf Process Med Imaging 20* (2007), 408–20.

[HE98] HASTREITER P., ERTL T.: Integrated Registration and Visualization of Medical Image Data. In *CGI* (Hannover, Germany, 1998), pp. 78–85.

[HHH01a] HAJNAL J., HAWKES D., HILL D.: *Medical Image Registration*. CRC Press, 2001.

[HHH01b] HAJNAL J., HILL D., HAWKES D.: *Medical Image Registration*. CRC Press, 2001.

[HRSS*04] HASTREITER P., REZK-SALAMA C., SOZA G., GREINER G., FAHLBUSCH R., GANSLANDT O., NIMSKY C.: Strategies for Brain Shift Evaluation. *Med Image Anal 8*, 4 (2004), 447–464.

[JLR02] JIANG J., LUK W., RUECKERT D.: FPGA-based Computation of Free-form Deformations. In *Proceedings of IEEE Int. Conf. on Field-Programmable Technology 2002* (16-18 Dec 2002), pp. 407–10.

[KDRMK06] KÖHN A., DREXL J., RITTER F., M. KÖNIG H.-O. P.: GPU Accelerated Image Registration in Two and Three Dimensions. In *Bildverarbeitung für die Medizin 2006* (2006), Handels H., Ehrhardt J., Horsch A., Meinzer H.-P., Tolxdorff T., (Eds.), Springer Verlag, pp. 261–65.

[KSP07] KLEIN S., STARING M., PLUIM J.: Evaluation of Optimization Methods for Nonrigid Medical Image Registration Using Mutual Information and B-Splines. *IEEE Trans Image Process 16*, 12 (Dec 2007), 2879–90.

[LDS05] LEVIN D., DEY D., SLOMKA P.: Efficient 3D Nonlinear Warping of Computed Tomography: Two High-performance Implementations Using OpenGL. In *Proc. SPIE Medical Imaging: Visualization, Image-Guided Procedures, and Display* (2005), Robert L. Galloway J., Cleary K. R., (Eds.), vol. 5744, pp. 34–42.

[MHS*04] MERHOF D., HASTREITER P., SOZA G., STAMMINGER M., NIMSKY C.: Non-linear Integration of DTI-based Fiber Tracts into Standard 3D MR Data. In *VMV (Vision, Modeling, and Visualization)* (2004), pp. 371–377.

[MOOXS08] MUYAN-OZCELIK P., OWENS J., XIA J., SAMANT S.: Fast Deformable Registration on the GPU: A CUDA Implementation of Demons. In *Int. Conf. on Comput. Sc. and its Appl. (ICCSA) 2008* (2008), IEEE Computer Society. in press.

[MSS*07] MERHOF D., SOZA G., STADLBAUER A., GREINER G., NIMSKY C.: Correction of Susceptibility Artifacts in Diffusion Tensor Data Using Non-linear Registration. *Med Image Anal 11*, 6 (2007), 588–603.

[PFTV88] PRESS W., FLANNERY B., TEUKOLSKY S., VETTERLING W.: *Numerical Recipes in C*. Cambridge University Press, 1988.

[RM03] ROHLFING T., MAURER JR. C.: Nonrigid Image Registration in Shared-memory Multiprocessor Environments with Application to Brains, Breasts, and Bees. *IEEE Trans Med Imaging 22*, 6 (2003), 730–41.

[RSH*99] RUECKERT D., SONODA L., HAYES C., HILL D., LEACH M., HAWKES D.: Non-rigid Registration Using Free-form Deformations: Application to Breast MR Images. *IEEE Trans Med Imaging 18*, 8 (1999), 712–721.

[RSSSG01] REZK-SALAMA C., SCHEUERING M., SOZA G., GREINER G.: Fast Volumetric Deformation on General Purpose Hardware. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2001).

[SBH*02] SOZA G., BAUER M., HASTREITER P., NIMSKY C., GREINER G.: Non-rigid Registration with Use of Hardware-Based 3D Bézier Functions. In *MICCAI* (2002), LNCS, Springer, pp. II,549–556.

[SDR04] STRZODKA R., DROSKE M., RUMPF M.: Image Registration by a Regularized Gradient Flow - A Streaming Implementation in DX9 Graphics Hardware. *Computing 73*, 4 (Nov. 2004), 373–389.

[SGN*04] SOZA G., GROSSO R., NIMSKY C., GREINER G., HASTREITER P.: Estimating Mechanical Brain Tissue Properties with Simulation and Registration. In *MICCAI* (2004), vol. 2 of *LNCS*, Springer, pp. 276–283.

[SH05] SIGG C., HADWIGER M.: Fast Third-Order Texture Filtering. In *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley, 2005, pp. 313–29.

[SHH99] STUDHOLME C., HILL D., HAWKES D.: An Overlap Invariant Entropy Measure of 3D Medical Image Alignment. *Pattern Recognition 32*, 1 (1999), 71–86.

[SW49] SHANNON C., WEAVER W.: *The Mathematical Theory of Communication*. Univ. of Illinois Press, 1949.

[VGXW07] VETTER C., GUETTER C., XU C., WESTERMANN R.: Non-rigid Multi-modal Registration on the GPU. In *Medical Imaging 2007: Image Processing* (2007), vol. 6512, SPIE, p. 651228.

[WFW*99] WEST J., FITZPATRICK J., WANG M., DAWANT B., C. MAURER J., KESSLER R., MACIUNAS R.: Retrospective Intermodality Registration Techniques for Images of the Head: Surface-Based Versus Volume-Based. *IEEE Trans Med. Imaging 18*, 2 (1999), 144–150.

[WVK95] WELLS III W., VIOLA P., KIKINIS R.: Multimodal Volume Registration by Maximization of Mutual Information. In *Medical Robotics and Computer Assisted Surgery* (1995), Wiley-Liss, New York, pp. 155–162.