GPU-ACCELERATED DIGITALLY RECONSTRUCTED RADIOGRAPHS

Daniel Ruijters
X-Ray Predevelopment
Philips Medical Systems
Best, the Netherlands
email: danny.ruijters@philips.com

Bart M. ter Haar Romeny
Biomedical Engineering,
Image Analysis and Interpretation
Technische Universiteit Eindhoven
Eindhoven, the Netherlands

Paul Suetens
Medical Image Computing,
ESAT/Radiologie
Katholieke Universiteit Leuven
Leuven, Belgium

ABSTRACT
Intensity based two-dimensional to three-dimensional (2D-3D) registration algorithms usually rely on the generation of a Digitally Reconstructed Radiograph (DRR) in each iteration of the registration algorithm. The vast majority of the computation time of such registration algorithms can be contributed to the calculations needed to produce DRRs. Since 2D-3D registration often is applied during clinical interventions, it is of great importance to obtain the result of the registration within limited time. We present an approach that is both very fast and accurate, harvesting the parallel processing power of today's mainstream graphics hardware. Especially we use the high precision z-buffer as intermediate buffer, in order to produce accurate results.

KEY WORDS
Data Representation and Visualization, Digitally Reconstructed Radiograph, 2D-3D Registration, Radiotherapy, GPU Acceleration

1 Introduction

The objective of a 2D-3D registration algorithm is to find a spatial mapping between the 2D and the 3D image. Typically a registration algorithm consists of a similarity measure, indicating the quality of a given spatial mapping, and an optimization algorithm, which iteratively searches the optimum (maximum or minimum, depending on the measure) of the similarity measure. The search space consists of the multi-dimensional control variables of the spatial mapping. For intensity based 2D-3D registration algorithms, generally a Digitally Reconstructed Radiograph (DRR) has to be generated in each iteration of the optimization algorithm, in order to calculate the similarity measure. The generation of such a DRR is computationally expensive, and the largest part of the computation time of the 2D-3D registration algorithm can be accounted to the generation of the DRRs. Therefore, it is our objective to accelerate its computation by using off-the-shelf graphics hardware, without loss of accuracy.

The purpose of a DRR is the extraction of a 2D image from a 3D CT dataset, which can be correlated with an X-ray image. The intensities on the X-ray image are the result of the X-ray absorption along the paths of the rays:

$$I = I_0 \, e^{-\int \mu(x) dx} \tag{1}$$

Whereby I and I_0 are the output and input X-ray intensity and $\mu(x)$ expresses the attenuation coefficient along the path of the ray. The Hounsfield scale is used in CT datasets to quantitatively describe the radiodensity per volume element. Since there is a linear correspondence between the X-ray attenuation coefficients, and the Hounsfield units, the DRR can be based on casting virtual rays through the CT volume, and integrating the encountered Hounsfield values [1].

The evaluation of a line integral, along the path of a given ray through the discrete grid of the CT dataset, can be expressed as:

$$L = \sum_i \sum_j \sum_k l(i,j,k) \cdot p(i,j,k) \tag{2}$$

Whereby $p(i,j,k)$ is the intensity of the voxel at index i,j,k, and $l(i,j,k)$ is the contribution of that particular voxel to the given ray. $l(i,j,k)$ is determined by the path of the ray and the interpolation scheme. The brute force execution of equation 2 would be rather inefficient, since for most interpolation schemes, such as nearest neighbour and tri-linear, $l(i,j,k)$ is zero for the vast majority of the voxels.

The Graphics Processing Unit (GPU) can be regarded as a Single Instruction Multiple Data (SIMD) processor, meaning that it can perform the same instruction on multiple data simultaneously. It has been demonstrated that volume rendering can be performed very efficiently, using the GPU [2, 3]. There is, however, a serious limitation that has to be overcome, when employing the GPU for DRR generation; intermediate results stored in buffer arrays, are typically of 8-bit integer point precision. This is especially a problem in summations: when a small number is added to a large number, the small number simply 'disappears', due to the lack of precision. That may not be a problem for a single addition, but in the addition of a lot of small numbers, the error becomes noticeable. It should be pointed out that this effect still persists with 16-bit floating point numbers, since they only have a 10-bit mantissa. The usage of floating point buffers was not an option for us anyway, since they are only available in the most recent graphics

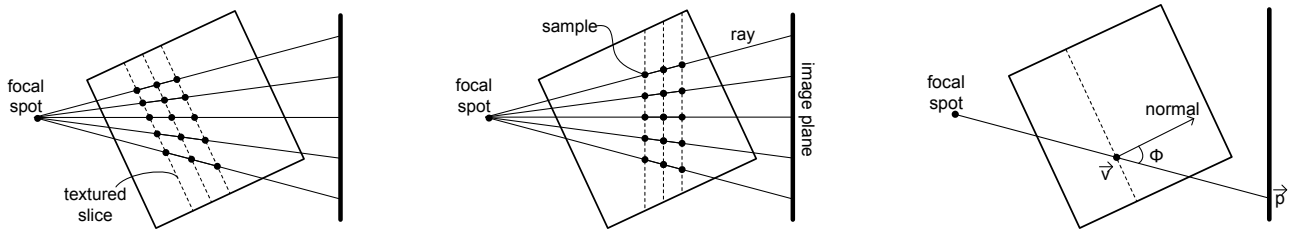


Figure 1. (a) Axis-aligned, and (b) view port-aligned orientation of the textured slices through the CT dataset volume. (c) The sample in the volume at position $\vec{v}$ is projected on the detector grid at position $\vec{p}$.

hardware implementations, and we have to support a large installed base. Therefore, we present an approach that uses the higher precision of the graphics card z-buffer as intermediate buffer, which is a standard feature on almost any graphics card.

2 State of the art

The registration of 2D X-ray images (such as fluoroscopy images) and 3D CT images has a number of clinical applications, such as radiotherapy planning and verification [4, 5], surgery planning and guidance [6, 7], and minimal invasive treatment in cardiac [8], vascular [7, 9, 10] and neuro-interventions [11, 12].

Intensity-based 2D-3D registration methods [6, 7, 10, 12, 13, 14, 15, 16, 17] directly use the pixel and voxel values to calculate a similarity measure, and require no or little segmentation. They are usually based on the generation of a DRR in each iteration.

DRR generating algorithms can be regarded as a subset of the class of Volume Rendering algorithms. Therefore, various methods available for efficient Volume Rendering can also be applied to DRR generation, such as raycasting [18, 19, 20], splatting [21, 22], shear-warp [23] and frequency domain rendering [24, 25]. Rusakoff *et al.* [26] have presented a method for accelerating the generation of DRRs, based on attenuation fields. Though this method is fast, it is preceded by a relatively time-consuming preprocessing step, and demands a considerable amount of memory.

Employing the texture capabilities of modern graphics hardware for producing DRRs has already been published by Cullip and Neumann [2] as early as 1993. They also pointed out the low precision issues that occur when using their approach.

3 Method

3.1 Textured slices

A well established approach to garner the parallel processing power of modern graphics hardware in Volume Rendering, is the usage of textured slices [2, 3]. This means that a set of parallel planes (slices) are defined through the volume space, and the CT information, contained by a 3D texture, is interpolated on these slices. In order to preserve the full dynamic range of the CT data, we use 16-bit textures. Typically the slices are placed at a constant interval. The orientation of the slices can be either aligned with the view port, or with one of the dataset's main axes (x-, y-, or z-axis), see figure 1. In the case of axis-aligned rendering the main axis with the smallest angle to the view port normal is chosen to serve as normal for the slices. The slices are then projected consecutively on a image plane.

3.2 Projecting the slices

The generation of a DRR comprises the calculation of line integrals over the Hounsfield values along the rays through the voxel volume. The rays are defined by the focal spot of the (virtual) X-ray source, and a discrete point on the (virtual) detector grid.

The technique of textured slices can also be applied to obtain the line integrals [19, 2]. The line integral is then calculated by summating the samples at the intersection points of the ray and the slices. The samples are defined by the interpolated attenuation values at the intersection points. Usually bi-linear or tri-linear interpolation is used. Since the distance between the sample points depends on the direction of a ray, see figure 1, the summation has to be weighted with the sample distance. The sample distance corresponds to $(1/\cos\phi)$, whereby ϕ is the angle between the normal of the slices and the vector that describes the direction of a particular ray.

In order to interpolate the slices on the discrete grid of the detector, a 4×4 matrix M is defined, such that $\vec{p} = M \cdot \vec{v}$, whereby $\vec{p}$ and $\vec{v}$ are homogenous coordinates. Vector $\vec{v}$ is then a coordinate in the voxel space, and vector $\vec{p}$ a coordinate on the detector grid (the z value of $\vec{p}$ is simply disregarded). Matrix M can be decomposed into two matrices: $M = P \cdot T$, whereby P is the perspective transformation defined by the position of the focal spot (f_x, f_y, f_z), the position of the center of the detector (c_x, c_y, c_z), and the detector dimensions (d_x, d_y). All co-

ordinates are assumed in the detector coordinate system:

$$P = \begin{pmatrix} 2/d_x & 0 & 0 & 2(f_x - c_x)/d_x \\ 0 & 2/d_y & 0 & 2(f_y - c_y)/d_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/(f_z - c_z) & 1 \end{pmatrix} \quad (3)$$

Matrix T is defined by the viewing incidence on the CT dataset, and can be expressed as:

$$T = \begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} \quad (4)$$

Whereby R is a 3 by 3 rotational matrix, and the three components of vector $\vec{t}$ express the translation. In the case of an X-ray angiography C-arm system, the translation can be set to zero. R is determined by the L-arm angle (γ), the rotation (β) and the angulation (α) of the C-arm. Note that the order of the matrix multiplications is given by the mechanics of the C-arm system.

$$R = R_x \cdot R_y \cdot R_z = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \cdot \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (5)$$

Now that matrix M is established, expressing the relation between the voxel space and the detector space, we can project any given position immediately on the detector grid. Thus we can define a textured polygon through a slice in the volume, and project the sampled values on that slice immediately on the detector grid. This is rather trivial task, using the GPU, since only the four polygon corners need to be specified, and the GPU will perform the interpolation of all voxel values that lie on the polygon.

3.3 Accurate summation

By summating the slices in the frame buffer, a DRR can be created. There is only one problem: by default the frame buffer is in 8-bit integer format on the PC platform. Therefore, here an alternative approach taken: we render to the z-buffer. This buffer, meant for occlusion tests, can be in 16-bit, 24-bit or 32-bit *integer* precision. In order to use the z-buffer as render target, we create two intermediate z-buffer textures. In each render pass we render a slice to one z-buffer texture as output, and we bind the other one as input texture. (A single texture cannot serve as input and output simultaneously.) The attenuation values sampled from the polygon are then added to the values already present in the input z-buffer texture, by a fragment program. In the next render pass the roles of both textures are swapped, and the next slice is processed.

A variation of the above would be rendering to the z-buffer, which is associated with a frame buffer, and to copying the result to a z-buffer texture, which is then bound as input texture for the next render pass. The advantage of this method is the fact that there is only one intermediate z-buffer texture, and it is somewhat simpler to implement. On the other hand, however, this alternative way is slower, since the entire z-buffer needs to be copied every pass.

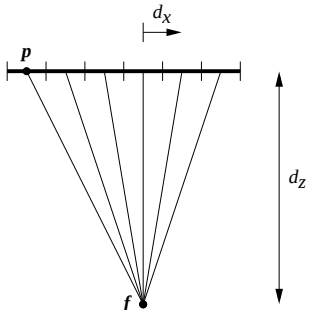


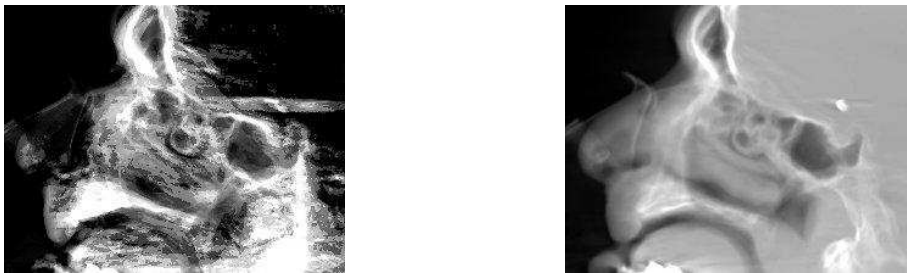
Figure 2. In order to perform an efficient weighting of the rays with the sample distance, we use tri-linear interpolation of vector $\vec{d'}$, which is based on the vector between the positions of the corner pixels $\vec{p}$ and the focal spot $\vec{f}$.

3.4 Weighting the integrals

The final summated result in the z-buffer texture still needs to be corrected for the different sampling distance in each line integral. Like mentioned before, this corresponds to weighting with $1/\cos\phi$, whereby ϕ is the angle between the normal of the slices and the ray direction. However, the direction of the individual rays is not available, since matrix M was used to directly project the textured slices on the virtual detector grid. Of course it is possible to calculate the line direction for every pixel in the bitmap. We, however, use a more efficient scheme: First the direction of the lines going through the four corner pixels of the detector grid is calculated. The position of a corner pixel is given by the $detectorCenter \pm (0.5 \cdot detectorSize - 0.5 \cdot pixelSize)$. Let $\vec{p}$ denote the position of a corner pixel, and $\vec{f}$ the position of the focal spot. The direction of the corresponding line is then given by: $\vec{d} = \vec{p} - \vec{f}$. All four corner directions $\vec{d}$ are divided by their respective z-component:

$$\vec{d'} = \left(\frac{d_x}{d_z}, \frac{d_y}{d_z}, 1 \right) \quad (6)$$

Since all the pixels of the line integral bitmap lie on a regular grid (i.e. the distance between the pixel position is always the same, see fig. 2), the direction of the lines of the other pixels can be obtained by a simple tri-linear



(a) (b)

Figure 3. A fraction of a DRR produced (a) using the 8-bit frame buffer, (b) using the 32-bit z-buffer.

interpolation of the direction $\vec{d'}$ of the corner pixels. This is only possible because we made sure that the z-value is the same for all corner directions. The weighting factor per pixel then is $w = 1/\cos\phi = 1/(\vec{n} \cdot \vec{d'_{norm}})$, whereby $\vec{n}$ is the normal of the slices, and $\vec{d'_{norm}}$ is the normalized interpolated $\vec{d'}$. This procedure is much faster since the trilinear interpolation can be very efficiently performed by the GPU.

4 Results

Figure 3 shows the same portion of a DRR generated using the 8-bit frame buffer and the 32-bit z-buffer. Using the 8-bit buffer leads to heavy saturation and quantization effects. We found that the deviation between a DRR produced by our z-buffer GPU implementation and by a similar CPU implementation was always less than 0.1%, using textures with 16 bit per voxel for the CT datasets. On a PC with a Intel Xeon 2.33 GHz Processor, and a nVidia QuadroFX 3500 graphics card with 256 MB of onboard memory, it took on average 47 ms to transfer a 4 MB dataset to the texture memory, 125 ms to transfer a 32 MB dataset, and 875 ms for a 200 MB dataset. It should be noted that this transaction only needs to be done once per dataset. The generation of a DRR on a 512^2 grid of the 4 MB dataset could be performed in 18 ms, and the 32 MB dataset took 29 ms, while the 200 MB dataset was rendered in 78 ms, which corresponds to respectively 56 fps, 35 fps and 13 frames per second.

5 Conclusions

We have described a method for implementing the generation of Digitally Reconstructed Radiographs (DRR), designated to use the GPU. Our method is based on mapping the Hounsfield values, found in a 3D CT dataset on a set of textured slices. The slices are then projected on a virtual detector, where the projected values are summated. The resulting summated values are finally corrected for the differences in sampling distance, by applying a weighting.

We have described how the algorithm can be implemented to run on the GPU, using only readily available features. We managed to achieve a very high accuracy, by using the high precision of the z-buffer. A very efficient weighting method was accomplished by calculating the direction of the rays of the corner pixels, and using the interpolation capabilities of the GPU to determine the directions of the rays through the other pixels.

The presented approach does not require any preprocessing, apart from the transfer of the CT data to graphics card, and the DRRs are generated very fast. This makes this approach highly suitable for application in intensity-based 2D-3D registration algorithms. Especially since 2D-3D registration often is applied during a clinical intervention, and the calculation time of the algorithm contributes to the time of the clinical procedure.

References

[1] R. M. Lewitt, "Reconstruction algorithms: Transform methods," *Proc. of the IEEE*, vol. 71, no. 3, pp. 390–408, 1983.

[2] T. J. Cullip and U. Neumann, "Accelerating volume reconstruction with 3D texture hardware," Tech. Rep. TR93-027, 1993.

[3] D. Ruijters and A. Vilanova, "Optimizing GPU Volume Rendering," *Journal of WSCG'06*, vol. 14, no. 1-3, pp. 9–16, 2006.

[4] K. G. A. Gilhuijs, P. J. H. van de Ven, and M. van Herk, "Automatic three-dimensional inspection of patient setup in radiation therapy using portal images, simulator images, and computed tomography data," *Medical Physics*, vol. 23, pp. 389–399, 1996.

[5] S. Clippe, D. Sarrut, C. Malet, S. Miguet, C. Ginestet, and C. Carrie, "Patient setup error measurement using 3-D intensity-based image registration techniques," *Int. J. Radiat. Oncol. Biol. Phys.*, vol. 56, no. 1, pp. 259–265, 2003.

[6] L. Lemieux, R. Jagoe, D. R. Fish, N. D. Kitchen, and D. G. T. Thomas, "A patient-to-computed-tomography image registration based on digitally re-

constructed radiographs," *Medical Physics*, vol. 21, pp. 1749–1760, 1994.

[7] J. Weese, G. P. Penney, P. Desmedt, T. M. Buzug, D. L. G. Hill, and D. J. Hawkes, "Voxel-based 2-D/3-D registration of fluoroscopy images and CT scans for image-guided surgery," *IEEE Trans. Inf. Technol. Biomed.*, vol. 1, no. 4, pp. 284–293, 1997.

[8] G. A. Turgeon, G. Lehmann, G. Guiraudon, M. Drangova, D. Holdsworth, and T. Peters, "2D-3D registration of coronary angiograms for cardiac planning and guidance," *Medical Physics*, vol. 32, no. 12, pp. 3737–3749, 2005.

[9] M. Breeuwer and et al., "The easi project-improving the effectiveness and quality of image-guided surgery.," *IEEE Transactions on Information Technology in Biomedicine*, vol. 2, no. 3, pp. 156–168, 1998.

[10] G. P. Penney, P. G. Batchelor, D. L. G. Hill, D. J. Hawkes, and J. Weese, "Validation of a two- to three-dimensional registration algorithm for aligning pre-operative CT images and intraoperative fluoroscopy images," *Medical Physics*, vol. 28, no. 6, pp. 1024–1032, 2001.

[11] Y. Kita, D. L. Wilson, and J. A. Noble, "Real-time registration of 3D cerebral vessels to X-ray angiograms," in *MICCAI '98: Proceedings of the First International Conference on Medical Image Computing and Computer-Assisted Intervention*, (London, UK), pp. 1125–1133, Springer-Verlag, 1998.

[12] J. H. Hipwell, G. P. Penney, R. A. McLaughlin, K. Rhode, P. Summers, T. C. Cox, J. V. Byrne, J. A. Noble, and D. J. Hawkes, "Intensity-based 2-D-3-D registration of cerebral angiograms," *IEEE Trans. Med. Imag.*, vol. 22, no. 11, pp. 1417–1426, 2003.

[13] W. Birkfellner, J. Wirth, W. Burgstaller, B. Baumann, H. Staedele, B. Hammer, N. C. Gellrich, A. L. Jacob, P. Regazzoni, and P. Messmer, "A faster method for 3D/2D medical image registration - a simulation study," *Physics in Medicine and Biology*, vol. 48, no. 16, pp. 2665–2679, 2003.

[14] D. Knaan and L. Joskowicz, "Effective intensity-based 2D/3D rigid registration between fluoroscopic X-ray and CT," in *MICCAI 2003, 6th Int. Conf., Montréal, Canada, Part I*, vol. 2878 of *Lecture Notes in Computer Science*, pp. 351–358, Springer, 2003.

[15] G. P. Penney, J. Weese, J. A. Little, P. Desmedt, D. L. G. Hill, and D. J. Hawkes, "A comparison of similarity measures for use in 2D-3D medical image registration," *IEEE Trans. Med. Imag.*, vol. 17, no. 4, pp. 586–595, 1998.

[16] D. Tomaževič, B. Likar, T. Slivnik, and F. Pernuš, "3-D/2-D registration of CT and MR to X-ray images.," *IEEE Trans. Med. Imag.*, vol. 22, no. 11, pp. 1407–1416, 2003.

[17] L. Zöllei, W. E. L. Grimson, A. Norbash, and W. M. Wells III, "2D-3D rigid registration of X-ray fluoroscopy and CT images using mutual information and sparsely sampled histogram estimators," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, pp. 696–703, 2001.

[18] R. L. Siddon, "Fast calculation of the exact radiological path for a three-dimensional CT array," *Medical Physics*, vol. 12, no. 2, pp. 252–255, 1985.

[19] P. M. Joseph, "An improved algorithm for reprojecting rays through pixel images," *IEEE Trans. Med. Imag.*, vol. MI-1, no. 3, p. 192196, 1982.

[20] T. Köhler, H. Turbell, and M. Grass, "Efficient forward projection through discrete data sets using trilinear interpolation," in *IEEE Med. Imag. Conf. Abstracts*, (Lyon, France), Oct 2000.

[21] S. Matej and R. M. Lewitt, "Practical considerations for 3-D image reconstruction using spherically symmatric volume elements," *IEEE Trans. Med. Imag.*, vol. 15, no. 1, pp. 68–78, 1996.

[22] L. Westover, "Footprint evaluation for volume rendering," *SIGGRAPH Computer Graphics*, vol. 24, no. 4, pp. 367–376, 1990.

[23] P. Lacroute and M. Levoy, "Fast volume rendering using a shear-warp factorization of the viewing transformation," in *Proc. SIGGRAPH '94*, pp. 451–458, Jul 1994.

[24] T. Totsuka and M. Levoy, "Frequency domain volume rendering," in *Proc. SIGGRAPH '93*, (Anaheim, California), pp. 271–278, 1993.

[25] I. Viola, A. Kanitsar, and M. E. Gröller, "GPU-based frequency domain volume rendering," in *Proceedings of SCCG'04*, pp. 49–58, 2004.

[26] D. B. Russakoff, T. Rohlfing, K. Mori, D. Rueckert, A. Ho, J. R. Adler, Jr., and C. R. Maurer, "Fast generation of digitally reconstructed radiographs using attenuation fields with application to 2D-3D image registration," *IEEE Trans. Med. Imag.*, vol. 24, no. 11, pp. 1441–1454, 2005.