SIGRAD (2014)
M. Obaid, D. Sjölie and M. Fjeld (Editors)

GPU-based ray-casting of non-rigid deformations: a comparison between direct and indirect approaches

F. M. M. Marreiros[1,2] and Ö. Smedby[1,2,3]

[1]Center for Medical Image Science and Visualization (CMIV), Linköping University, Sweden
[2]Department of Science and Technology (ITN) - Media and Information Technology (MIT) , Linköping University, Sweden
[3]Department of Radiology (IMH), Linköping University, Sweden

Abstract

For ray-casting of non-rigid deformations, the direct approach (as opposed to the traditional indirect approach) does not require the computation of an intermediate volume to be used for the rendering step. The aim of this study was to compare the two approaches in terms of performance (speed) and accuracy (image quality).
The direct and the indirect approach were carefully implemented to benefit of the massive GPU parallel power, using CUDA. They were then tested with Computed Tomography (CT) datasets of varying sizes and with a synthetic image, the Marschner-Lobb function.
The results show that the direct approach is dependent on the ray sampling steps, number of landmarks and image resolution. The indirect approach is mainly affected by the number of landmarks, if the volume is large enough. These results exclude extreme cases, i.e. if the sampling steps are much smaller than the voxel size and if the image resolution is much higher than the ones used here. For a volume of size 512×512×512, using 100 landmarks and image resolution of 1280×960, the direct method performs better if the ray sampling steps are approximately above 1 voxel. Regarding accuracy, the direct method provides better results for multiple frequencies using the Marschner-Lobb function.
The conclusion is that the indirect method is superior in terms of performance, if the sampling along the rays is high, in comparison to the voxel grid, while the direct is superior otherwise. The accuracy analysis seems to point out that the direct method is superior, in particular when the implicit function is used.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Hardware Architecture; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling ; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism ;

1. Introduction

Since the introduction of the Graphics Processing Units (GPUs) with programming functionalities, we have seen an explosion of algorithms being ported to these devices. The NVIDIA's Compute Unified Device Architecture (CUDA) enables the usage of these devices. The necessary software programming interface to access the hardware is available in ”C for CUDA” [Cud10] (C with NVIDIA extensions and certain restrictions), but many language bindings are also available.

One algorithm that largely benefited from programmable GPU architectures is volume ray-casting. Volume ray-casting is a well known Direct Volume Rendering (DVR) technique used mainly to render regular grid volume data, but it also can be used to render implicit surfaces [Sig06]. Examples of such data are medical image modalities like Computed Tomography (CT) and Magnetic Resonance Imaging (MRI). Ray-casting operates by casting one ray per pixel, and each ray is computationally independent, thus benefiting greatly from highly parallel programmable GPU architectures, e.g. shaders or CUDA. In this paper, we will use mainly isosurface ray-casting, a special case of volume ray-casting also known as *first-hit ray-casting*, with regular grid and implicit surface data. The same conclusions can be generalized for other DVR types.

Ray-casting is mostly used to generate images of static objects; if the volume is deformed over time, the traditional approach (indirect) is to compute an entire new volume for each time step and render it. The intensities at the corresponding positions in the original volume have to be calculated for each voxel. In the literature there are several voxel-based non-rigid deformation methods, which can also be used for specific tasks like volume morphing or volume registration. In this paper we focus on the Radial Basis Functions (RBF) method, one of the most widely used non-rigid deformation methods in the medical field that also has fast GPU implementations.

We compare the direct and indirect methods exploiting in both approaches the benefits of the GPU hardware architecture. The comparisons are provided in terms of performance and accuracy.

The motivation is to study if it is possible to enable deformation interaction with a frame rate that allows visual feedback (at least 1 frame-per-second). There are two main application scenarios considered: first tracking of points in real-time to guide the deformation; second for artistic purposes where the artist/animator is controlling the position of the points to guide the deformation. In both cases visual feedback is critical, thus a proper frame rate is essential.

2. Related work

2.1. GPU-based ray-casting

Several GPU-based ray-casting techniques have been proposed in the literature. An overview of the major developments in this area, including acceleration techniques like early-ray termination or empty space skipping, can be found in [HLSR09]. From a historical perspective, some of the initial implementations of GPU-based ray-casting with acceleration techniques are from Krüger and Westermann [KW03] and Röttger et al. [RGW*03]. Both implementations use shaders and early-ray termination, but their empty space skipping implementations are different. Krüger and Westermann [KW03] use an octree for empty-space skipping. Röttger et al. [RGW*03] calculate the ray's intersections with the volume bounding box and use them as the starting and end points of the rays. Li et al. [LMK03] compared several empty space skipping and occlusion clipping techniques for texture-based volume rendering.

More recently, a combination of CUDA and OpenGL can be used to compute the images (CUDA side) and pass them directly to be rendered (OpenGL), bypassing the traditional rendering pipeline. This uses the OpenGL Pixel Buffer Object (PBO) to store the texture ensuring that the generated images reside in the GPU as described in the chapter OpenGL Interoperability of the CUDA Programming Guide and the example code [Cud10]. There are already advanced CUDA ray-casting implementations [MRH10]. To be noticed, in the CUDA SDK a simple ray-casting example can also be found. Although CUDA was used here the implementation could also be made using OpenCL or shaders, with similar results expected.

2.2. Non-rigid deformations

A considerable number of non-rigid deformations methods have been proposed. A good survey of physically based deformable models can be found in [NMK*05]. Most of these approaches are dependent on a mesh representation, but a subset of mesh-free methods exist. Besides the physically based methods, many others exist that in general try to optimize the transformation minimizing some constrains. In this work we will use a non physically based mesh-free method.

A distinguishing factor between them is the data they use to drive the deformation. They can thus be divided into point-based, surface-based, intensity-based, etc. In this paper, we will focus on the point-based approach, in particular the Radial Basis Function (RBF) method, and one specific RBF: the Thin Plate Spline (TPS). A comparison of some of these methods can be found in Fang et al. [FSRR00] who also describe an implementation that does not require the intermediate volume. Further methods that also do not require the intermediate volume and are tied to the rendering stage, include ray deflectors [KY97], free-form [CHM03], free-form and texture mapping [WRS01], spatial transfer functions [CSW*03], constrained illustrative [CSC10], and 3D chainmail algorithms [Gib97]. From the last set of methods the ones that can use homologous points to control the deformation in the same way as with the TPS are the free-form methods [WRS01], [CHM03]. These share many similarities with our work. In [WRS01] no intermediate volume is calculated but instead a shape model and an appearance model are used. The shape model is a tessellation of the surface enclosing the object and the appearance a 3D texture of the volume. The second free-form method [CHM03] computes a deformed ray in the original object space and approximates the ray path by polylines.

Recent, RBFs have been implemented in the GPU, for warping and non-linear registration. Levin et al. [LDS04], Rowland [Row07] and Lapeer et al. [LSR10] all use shaders or a shader/CPU combination for comparisons. One should notice that these implementations could be tuned to trade speed for accuracy, a feature essential in some application scenarios. All these approaches require the computation of a new volume (indirect), which can possibly be rendered in a subsequent step. The voxel positions in the new volume (Target) have to be evaluated, and the intensity of the corresponding position in the original volume (Source) fetched and assigned to the new voxel. This process is known as *backward mapping*.

We also implement the indirect method for comparison purposes using CUDA. For the direct approach, we evaluate and perform backward mapping at the ray-casting sampling positions. This has the drawback of being useful only for rendering purposes, but if rendering is the primary goal, then our approach may prove valuable. In addition, we can control image quality and frame rate using the traditional ray-casting parameters.

3. Materials and Methods

3.1. RBF

RBFs are well known for their smooth interpolation properties. They are able to smoothly interpolate point positions in 3D to generate a surface, as seen, e.g., in [CBC*01]. We use the RBF as a smooth mapping function, to determine corresponding positions in two 3D volume spaces. Each space has a set of points, also referenced in the literature as landmarks or centers, necessary to drive the mathematics of the method. Both spaces need to have the same number of landmarks and each landmark in one space (Template or Source) has a corresponding homologous landmark (e.g. the same anatomical location) in the mapped space (Target). An initial use of RBFs, in particular the thin-plate spline (TPS) for modeling of biological shape change, was proposed by Bookstein [Boo89]. Bookstein formulated the TPS algebra in 2D; later extensions by Lapeer and Prager [LP00] enabled the use in 3D. We can also find in [LP00] the concepts of forward and backward transformation.

In algebraic terms, the backward mapping can be described by the following equation:

$$(x_s, y_s, z_s) = f(x_t, y_t, z_t) \tag{1}$$

where: (x_t, y_t, z_t) are target point coordinates in Cartesian space;

(x_s, y_s, z_s) are source point coordinates in Cartesian space;

$f(x, y, z)$ is a function mapping points in the target space to the source space. This function needs to be decomposed, solved and evaluated separately per coordinate, in the 3D case:

$$f(x_t, y_t, z_t) = [f'_X(x_t, y_t, z_t), f'_Y(x_t, y_t, z_t), f'_Z(x_t, y_t, z_t)] \tag{2}$$

where: $f'_X(x_t, y_t, z_t) = x_s; f'_Y(x_t, y_t, z_t) = y_s; f'_Z(x_t, y_t, z_t) = z_s$. Per coordinate the function f' has the following equation:

$$f'_*(x, y, z) = a_{1*} + a_{2*}x + a_{3*}y + a_{4*}z + \sum_{i=1}^{n} \lambda_{i*}\phi(|P_i - (x, y, z)|) \tag{3}$$

where:

$*$ is an index for the individual coordinates, it should be replaced by X, Y or Z;

$a_{1*}, a_{2*}, a_{3*}, a_{4*}$ are the coefficients of an affine transformation;

n the number of landmarks;

λ_{i*} are the weights;

P_i a landmark point - in backward mapping the target landmarks, in forward mapping the source landmarks;

ϕ the radial basis function.

The TPS function in 3D is given by the following equation.

$$\phi(r) = r \tag{4}$$

Using the landmark values in equation (2), we obtain a linear system of equations that can be directly solved by LU decomposition. In this way, the unknown affine coefficients and weights can be obtained for each coordinate ([Boo89], [LP00]). To solve the linear system of equations we used a C++ linear algebra library named Armadillo [San10] in CPU, as the time needed to solve the system is very short. The greatest amount of time is spent in the evaluation of the new point positions performed in GPU.

Knowing the coefficients of the affine transformation and the weights, we can evaluate the point coordinates directly using equation (2). The purpose of this mapping is to obtain the corresponding image intensity values of the target points. Using the indirect method, we want to know the intensity values of the voxels positioned on a regular grid (target volume image). Since the target voxel positions are known, we can use the mapping function to obtain the corresponding position in the source volume (equation (2)).

$$I(x_{t_i}, y_{t_j}, z_{t_k}) = I(x_s^{'}, y_s^{'}, z_s^{'}); i = 1...n_x, j = 1...n_y, k = 1...n_z$$
$$(x_s^{'}, y_s^{'}, z_s^{'}) = f(x_{t_i}, y_{t_j}, z_{t_k}) \tag{5}$$

where: n_x,n_y,n_z are the dimensions of each volume coordinate.

The mapped point in the source image may lie between voxel positions; if this is the case, then trilinear interpolation is required to obtain the intensity value. The trilinear interpolation is performed by the GPU in hardware, considerably reducing computation time. Still, evaluating all voxels in the volume requires considerable computational power, since the RBF depends on the number of landmarks.

3.2. Ray-casting of non-rigid deformations

The traditional (indirect) way of using ray-casting and non-rigid deformations is to first calculate an entire new volume by backward mapping all the voxels and then render it. Our direct approach is different: we evaluate (backward mapping and check the intensity in source image) at the sampling points locations along the ray. The evaluation of the gradients, used for illumination, is also calculated in this way. In our case, a 3D Sobel filter is used to estimate the gradient. This filter requires 26 neighbor points, which are backward mapped to obtain the appropriate intensity value in the source image. The indirect method also uses the Sobel filter, but in the intermediate volume space. Furthermore early-ray termination is used to stop the ray when the maximum opacity level is reached and adaptive sampling to increase the precision of the isosurface position by increasing the sampling rate by eight times in positions where the intensity values approximate the isosurface threshold.

3.3. Volume bounding boxes

The position and size of the target bounding box is defined by the user. This is done to allow the user to select the appropriate dimensions. In cases where the source and target bounding boxes have the same size, an expansion of the target volume would possibly clip it. If this is the case, the user manually increases the size of the bounding box and the problem is solved.

The bounding boxes serve also to determine where the rays initiate and possibly terminate, if the surface was not hit. This is performed by computing where the ray intercepts the faces of the bounding boxes to determine the points closer and further from the center of projection.

3.4. Accelerating the RBF evaluation

As mentioned earlier, with the direct approach the RBF is evaluated at the points along the sampling ray and with the indirect method at the point locations of the voxels in the intermediate volume. The evaluation requires the computation of a sum that depends on the number of landmarks. In order for these operations to be performed as fast as possible, we make use of CUDA texture memory. We then need to store in texture memory the coefficients of the affine part, the target landmarks and the weights. Note that these variables are required for each ray/voxel, because they all are needed for the deformation computation of each sample point. This CUDA memory usage greatly improves performance since texture memory is cacheable; the alternative would be to get these variables from global memory every time we need to use them. This would introduce a great penalty due to memory latency and bandwidth.

4. Results

To compare consistently the indirect and direct methods in terms of performance, independently of the volume data we have used empty volumes with source and target landmarks with the same values (no deformation) and three volume sizes:

- $512\times512\times512$
- $512\times512\times256$
- $512\times512\times128$

with isotropic voxels of 0.5 mm per coordinate. The performance tests were dependent on:

- ray sampling step
- number of landmarks
- image resolution (number of rays)

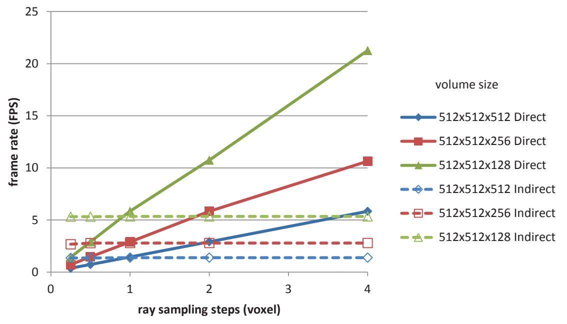

Figure 1: *Performance test: variation of ray sampling steps. The image resolution used was 1280×960 and the number of landmarks 100. The direct and indirect methods are compared.*

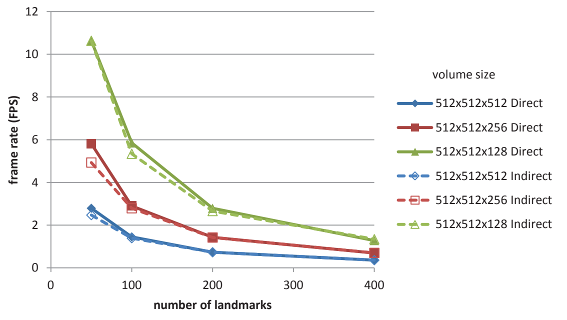

Figure 2: *Performance test: variation of number of landmarks. The image resolution used was 1280×960 and the ray sampling step 1.0 voxel. The direct and indirect methods are compared.*

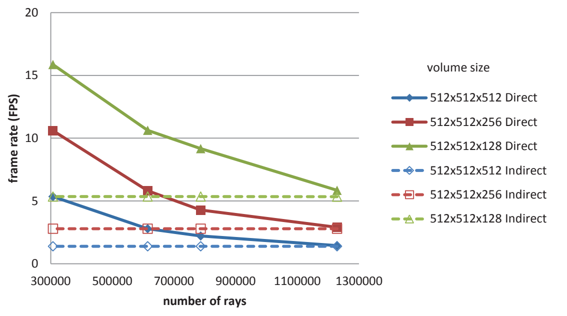

Figure 3: *Performance test: variation of number of rays. The resolutions used were: 1280×960 (1,228,800 rays), 1024×768 (786,432 rays), 960×640 (614,400 rays), 640×480 (307,200 rays). The ray sampling step used was 1.0 voxel and the number of landmarks 100. The direct and indirect methods are compared.*

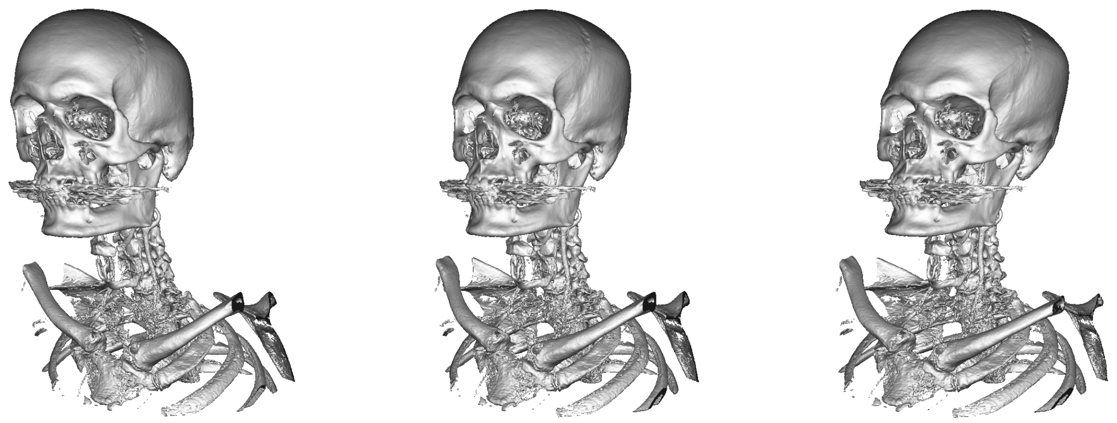

Figure 4: *Examples of a TPS non-rigid deformation, using a CT of a head and partially chest (512×512×512) dataset. The images on the left and right are deformed and the center one is not deformed.*

Care was taken to place the volumes projections (by controlling the camera position) fully within the image space. The graphics card used was an NVIDIA Geforce GTX 680 with 1536 CUDA cores, 2GB of dedicated video memory and CUDA architecture 2.0. Figures 1, 2, 3 present the results found for the several tests. To analyze the performance we look at the frame rate in frames per second (FPS). Figure 1 shows that the direct method is linearly dependent on the ray sampling steps: the frame rate varies linearly for all datasets, but with different slopes. For the indirect method, there are no significant differences if the volume is large enough. These results exclude extreme cases, i.e. if the sampling steps are much smaller than the voxel size. In these cases it is expected that the time to compute the deformation is neglectable in comparison to the ray traversal producing very low framerates. Figure 2 shows that both methods are dependent on the number of landmarks: the frame rate decays exponentially with the increase of landmarks. Finally, in Figure 3 we can see that the frame rate using the direct method is exponentially decaying, while for the indirect method, if the volume is large enough, the frame rate is constant. Extreme cases where the image resolution is much higher than the ones used here are not considered. The same consideration as in the ray sampling steps apply here.

For the direct method, we perform an extra test. The performance with and without early-ray termination was compared, using a CT dataset of the head and part of the chest (512×512×512), presented in Figure 4, where the central image corresponds to the volume without deformation and the remaining with a deformation applied. The results are presented in Figure 5.

In Figure 5 we can see that the increase in performance with early termination is small. In fact, it may even take more time to produce the rendering with the early-ray termination than to traverse the entire empty volume. This is due to adaptive sampling: the sampling rate increases in regions with values near the isosurface threshold, but if the ray does not hit the surface, then the performance penalty is high.

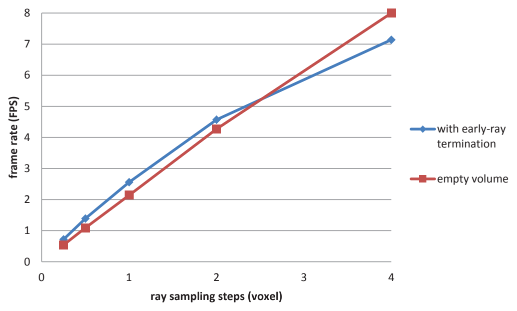

Figure 5: *Performance test: early-ray termination versus an empty volume of the same size, using the direct method and multiple ray sampling steps. The image resolution used was 1280×960 and the number of landmarks 100.*

Besides the performance tests we also tested image quality. A synthetic volume was created using the Marschner-Lobb function [ML94], with a resolution of 512×512×256. To generate the ground truth approximation, we used the direct method and in the source space tested the voxel values with the analytical function with a translation of 1.25 voxel in each coordinate, rendering the values larger or equal to 0.5. To compute the gradient, we used also the Sobel filter, but again testing the values with the analytical function. The offsets of the sampling positions were in all three coordinates 0.5 voxel, and the ray sampling steps 0.05 voxel. The comparison was made using two frequencies: $f_M = 10$ and $f_M = 20$ and $\alpha = 0.25$. The two rendering approaches had as input an implicit function (Marschner-Lobb) or a sampled volume regular grid. The values of the voxels of the

Table 1: *PSNR of the direct and indirect method (with implicit or volume regular grid as input) for the Marschner-Lobb functions with frequencies ($f_M = 10$, $f_M = 20$) and* $\alpha = 0.25$.

PSNR	direct implicit	indirect implicit
$f_M = 10$	+57.4130 dB	+25.3260 dB
$f_M = 20$	+57.1295 dB	+23.5643 dB
	direct regular grid	indirect regular grid
$f_M = 10$	+32.8225 dB	+25.5789 dB
$f_M = 20$	+24.5623 dB	+23.3019 dB

Table 2: *SSIM of the direct and indirect method (with implicit or volume regular grid as input) for the Marschner-Lobb functions with frequencies ($f_M = 10$, $f_M = 20$) and* $\alpha = 0.25$.

SSIM	direct implicit	indirect implicit
$f_M = 10$	0.998835	0.834781
$f_M = 20$	0.998947	0.682028
	direct regular grid	indirect regular grid
$f_M = 10$	0.957126	0.847693
$f_M = 20$	0.718059	0.663997

regular grid were obtained using the analytical function. The translation applied to the ground truth was necessary in order to test the impact of the TPS transformation (the source landmarks and target landmarks have different positions). To produce equivalent results, the target landmarks need also to be translated 1.25 voxel in each coordinate. The results are presented in Figure 6.

To compare the images, we use the image quality metrics peak signal-to-noise ratio (PSNR) and the structural similarity measure (SSIM) [WBSS04]. SSIM takes into account human eye perception. Table 1 and 2 present the results obtained. The images used for these tests were cropped versions (boundaries were cropped) of the images in Figure 6.

5. Discussion

The major contribution of this paper is the comparison of direct and indirect TPS non-rigid deformations and making use of the graphics hardware (CUDA). Regarding direct methods, [FSRR00], [WRS01] and [CHM03] also have implementations that not require the intermediate volume and can use points to control the deformation. Although these seem conceptually similar the main differences are that they use intermediate data structures for acceleration purposes instead we rely on brute force GPU acceleration. Also, we expect a better deformed image quality due to the fact that the intermediate data structures are used for approximation purposes and can trade image quality for speed. The most similar method to ours is the free-form method [CHM03], but a key difference is that the free-form method deforms the rays in the original volume space and then traverse them. In our approach the rays are straight and traversed in the deformed volume space; the sampling positions are backward mapped to obtain the intensity values in the original volume space.

Comparing the two methods in terms of performance and accuracy (image quality), we note that the indirect method is less sensitive to changes in the ray-sampling steps and to image resolution, while both seem to be similarly affected by the number of landmarks. For larger volumes and greater sampling steps, the performance of the indirect method is worse than the direct method. There are also other drawbacks of the indirect method, for instance the inability to trade image quality for frame rate, because the main parameter used for this purpose is the ray sampling steps and for large volumes the indirect method presents almost no change (constant). In contrast, the direct method is very sensitive to the ray-sampling steps, thus allowing easy control of the frame rate. Furthermore, the indirect method requires that two volumes are loaded simultaneously to texture memory, which for larger volumes can present a severe limitation.

In this work, we did not explore faster deformation techniques, as we are interested in preserving image quality and keeping the method as general as possible. Using RBFs fast alternatives exist, that trade image quality for speed, as pointed out in the non-rigid deformations section [Row07], [LSR10]. We could also use the locally bounded Hardy method like in [FSRR00], and RBF with compact support [FRS01]. The choice of the method to be used should depend on the data and the purpose. Our implementation can easily be changed to use some of these methods, if needed.

The second set of tests performed in our study related to the accuracy of both methods. In Figure 6 we can see that differences exist in both methods, although, in some cases, they are rather hard to perceive with the naked eye. Using the image quality metrics PSNR and SSIM (the higher the better), the direct method gives superior results, in particular when the implicit function is used. The worse results of the indirect method can be explained by the double trilinear interpolation: first at the evaluation of each voxel value (backward mapping - generation of intermediate volume) and second in the reconstruction (at each sampling position). It also depends on the resolution with which the intermediate volume is sampled. Also to observe, the direct implicit method is less sensitive to frequency, because no volume grid sampling is performed to obtain the voxel intensities. In this case, the sources of errors reside in the sampling positions along the ray and in numerical rounding errors in the TPS calculation. However, as can be seen, these are rather small errors.

6. Conclusion

A comparison of direct and indirect ray-casting TPS non-rigid deformations was performed. Both methods were im-

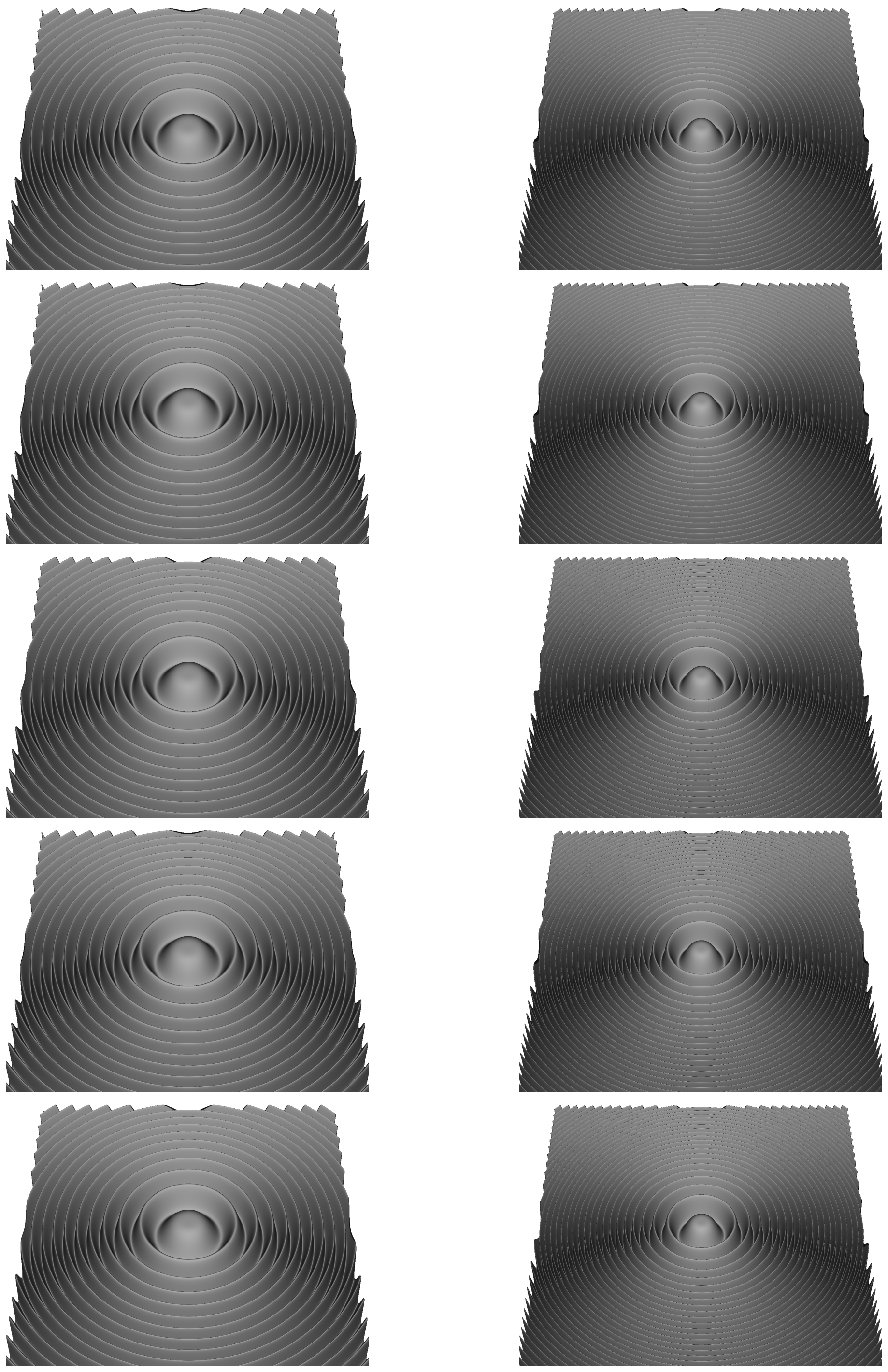

Figure 6: *Marschner-Lobb function comparison, ground truth (top row), direct method with implicit function (second row), direct method with regular grid (third row), indirect method with implicit function (fourth row), indirect method with regular grid (fifth row). The ground truth approximation was generated using the direct method and evaluated in source space with the analytical function with a 1.25 voxel translation, threshold 0.5 and using a Sobel filter for the gradient with offsets of the sampling positions in all tree coordinates 0.5 voxel. The ray sampling steps 0.05 voxel. Two frequencies were used:* $f_M = 10$ *(Left) and* $f_M = 20$ *(Right) and* $\alpha = 0.25$.

plemented in the GPU using the NVIDIA CUDA architecture for acceleration purposes. The comparisons of both methods were made in terms of performance and accuracy. Regarding performance, the indirect method is superior if the sampling along the rays is high, in comparison to the voxel grid, while the direct is superior otherwise. The accuracy analysis seems to point out that the direct method is superior, in particular when the implicit function is used.

Acknowledgment

This work was funded by the Visualization programme of the Swedish Foundation for Strategic Research, the KK Foundation, Vinnova, Invest in Sweden Agency, Vårdalstiftelsen (grant 2009/0079) and the Swedish Childhood Cancer Foundation (grant no. MT2013-0036). The authors are indebted to Petter Dyverfeldt for his help in the medical images acquisition and selection.

References

[Boo89] BOOKSTEIN F.: Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11* (1989), 567–585. 3

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 67–76. 3

[CHM03] CHEN H., HESSER J., MANNER R.: Raycasting free-form deformed-volume objects. *The Journal of Visualization and Computer Animation 14* (2003), 61–72. 2, 6

[CSC10] CORREA C. D., SILVER D., CHEN M.: Constrained illustrative volume deformation. *Computer & Graphics 34* (2010), 370–377. 2

[CSW*03] CHEN M., SILVER D., WINTER A. S., SINGH V., CORNEA N.: Spatial transfer functions - a unified approach to specifying deformation in volume modeling and animation. Proceedings of volumegraphics'03, pp. 35–44. 2

[Cud10] Nvidia cuda c programming guide, version 3.1, 2010. 1, 2

[FRS01] FORNEFETT M., ROHR K., STIEHL H.: Radial basis functions with compact support for elastic registration of medical images. *Image and Vision Computing 19* (2001), 87–96. 6

[FSRR00] FANG S., SRINIVASAN R., RAGHAVAN R., RICHTSMEIER J. T.: Volume morphing and rendering - an integrated approach. *Comput. Aided Geom. Des. 17* (January 2000), 59–81. 2, 6

[Gib97] GIBSON S. F.: 3d chainmail:a fast algorithm for deforming volumetric objects. Proceedings of the 1997 symposium on interactive 3D graphics. 2

[HLSR09] HADWIGER M., LJUNG P., SALAMA C. R., ROPINSKI T.: Eurographics 2009 course notes: Gpu-based volume raycasting with advanced illumination. In *Eurographics 2009 course* (2009), Eurographics 2009. 2

[KW03] KRÜGER J., WESTERMANN R.: Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings IEEE Visualization 2003* (2003). 2

[KY97] KURZION Y., YAGEL R.: Interactive space deformation with hardware-assisted rendering. *IEEE Computer Graphics and Applications 17*, 5 (1997), 66–77. 2

[LDS04] LEVIN D., DEY D., SLOMKA P. J.: Acceleration of 3d, nonlinear warping using standard video graphics hardware: implementation and initial validation. *Computerized Medical Imaging and Graphics 28*, 8 (2004), 471–83. 2

[LMK03] LI W., MUELLER K., KAUFMAN A.: Empty space skipping and occlusion clipping for texture-based volume rendering. In *Proc. IEEE Visualization 2003* (2003), pp. 317–324. 2

[LP00] LAPEER R., PRAGER R.: 3d shape recovery of a newborn skull using thin-plate splines. *Computerized Medical Imaging and Graphics 24* (2000), 193–204. 3

[LSR10] LAPEER R., SHAH S., R.S. R.: An optimised radial basis function algorithm for fast non-rigid registration of medical images. *Computers in Biology and Medicine 40*, 1 (2010), 1–7. 2, 6

[ML94] MARSCHNER S. R., LOBB R. J.: An evaluation of reconstruction filters for volume rendering. IEEE Visualization'94. 5

[MRH10] MENSMANN J., ROPINSKI T., HINRICHS K. H.: An advanced volume raycasting technique using gpu stream processing. In *GRAPP: International Conference on Computer Graphics Theory and Applications* (Angers, 2010), INSTICC Press, pp. 190–198. 2

[NMK*05] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics, 2005. 2

[RGW*03] RÖTTGER S., GUTHE S., WEISKOPF D., ERTL T., STRASSER W.: Smart hardware-accelerated volume rendering. In *VisSym* (2003). 2

[Row07] ROWLAND R. S.: *Fast Registration of Medical Imaging Data Using Optimised Radial Basis Functions :PhD Dissertation*. PhD thesis, University of East Anglia, 2007. 2, 6

[San10] SANDERSON C.: Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments, 2010. 3

[Sig06] SIGG C.: Representation and rendering of implicit surfaces, 2006. 1

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13* (2004), 600–612. 6

[WRS01] WESTERMANN R., REZK-SALAMA C.: Real-time volume deformations. *Computer Graphics Forum 20*, 3 (2001). 2, 6