A Fast Method for the Alignment of the Displacement of Voxel Data

Denis SPELIC[1], Franc NOVAK[2], Borut ZALIK[1]
[1]*Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia*
[2]*Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia*
denis.spelic@uni-mb.si

***Abstract*—A fast algorithm for the alignment of the displacement of voxel data is presented. In contrast to the existing solutions, the proposed algorithm achieves a less accurate alignment, but in a much shorter time. The algorithm consists of two parts: a translation and a rotation. While the translation part of the alignment process is error free, the rotation part introduces a small inherent error due to the finite arithmetic and discretization. Experimental results based on three datasets are presented.**

***Index Terms*—data preprocessing, image matching, matching pursuit algorithm, optimal matching, biomedical image processing.**

I. INTRODUCTION

The investigation of an object's interior is an important issue in a variety of disciplines, for example, in the study of terrain erosion [1], geographic information systems [2], material analysis [3], hydrology [4], photorealistic scene reconstruction [5], and even during animation [6]. In the commonly used spatial-enumeration representation [7], the space of interest, a volume, is divided into discrete parts. The smallest discrete entity representing a value on a regular grid in three-dimensional space is referred to as a voxel. Each voxel has a unique position and an associated scalar value representing the characteristics of the material located in this voxel. In medicine, for example, this value is defined according to the Houndsfield scale [8]. The object's interior is usually obtained as a sequence of parallel cross-sections that form a set of 2D images. These images are, therefore, considered as slices. Then, to form a volume, the slices are combined according to the direction of the cross-sections. Various analyses can be performed on such volumes, including virtual operations [9]-[11], segmentations [12], or structure analyses [13]. In many cases, the volume of interest is investigated at different times so as to discover possible changes in its interior structure [14]-[15]. This is because, during a given time interval, changes in the position of the original volume may occur. The two volumes (i.e., the original and the translated and/or rotated) need to be aligned with each other in order to perform a subsequent analysis. In the literature [16], the following alignment approaches have been proposed:

- 3D volume registration, which is frequently applied in medical imaging, where the medical data obtained by different investigation methods have to be aligned (for example, 3D UZ and CT images) [17]. Firstly, a small number of characteristic voxels is identified [18], and after that the smooth deformation fields are computed in order to relate these voxels [19]. In [16] a method is described that can successfully register even very different objects in both volumes. The method finds the correspondence for each voxel in both volumes based on the voxels' intensity.
- 3D morphing, which is used in 3D graphics to gradually transform one shape into another [20]. Usually, a polyhedral representation of the objects' boundary is used. In practice, the characteristic points are selected manually [21], although an automatic method has also been reported [22]. However, the basic idea can be adapted to volume morphing [23].
- 3D shape matching, which has been applied for boundary-represented objects. The solution proposed in [24] determines the iso-contures and tries to match them using a graph. So-called alpha-shapes, which have been originally used for a surface reconstruction from scattered points [25], have been proposed in [26].

The volume-registration methods are without doubt the best methods available today for matching volume data. Unfortunately, however, they are computationally intensive, which represents a bottle-neck when processing large volume spaces.

It is this fact that fostered our attempts to develop a new method that achieves a less accurate alignment, but in a much shorter time. The method supposes that the actual objects in the two volumes to be aligned are similar, but they may differ in terms of position and/or orientation. The method first detects the characteristic voxels automatically, and after that calculates a set of geometric transformations to match the two volumes.

The remainder of the paper is organized as follows. In Section 2, the proposed algorithm is presented. In order to make it easier to understand, the principle of the algorithm is described in 2D. The experimental results based on three datasets are given in Section 3. Finally, in Section 4, the conclusions are drawn.

II. THE ALGORITHM

Let us consider volume *A* with the resolution *nx*, *ny*, and *nz* voxels in the *x*, *y*, and *z* directions, respectively. The object of interest stored in volume *A* is described by a set of voxels, and all the voxels are described by the same number of bits.

Let us suppose that we have an identical volume *B*, with respect to the resolution and the number of slices. Both volumes, *A* and *B*, store the same object, but the data may

This work was supported by the Slovenian Research Agency under grants for research programmes P2-0041 and P2-0098.

Digital Object Identifier 10.4316/AECE.2012.02007

vary due to some minor local modifications and orientation. Such changes may occur in practice since the data of the two volumes have been collected at different times. We consider volume *A* to be the reference volume and we try to align volume *B* with volume *A* by determining the corresponding translation and rotation factors.

The algorithm consists of two steps. In the first step, a representative sub-volume in volume *A* is identified and the algorithm searches for a similar representative sub-volume in volume *B*. After identifying the corresponding representative sub-volume in volume *B* the rotation procedure is performed in order to align the two volumes. In the following, the algorithm is described in more detail and its performance is demonstrated in a number of experimental results. In order to make it easier to understand the operation of the algorithm is described in 2D. The translation into 3D is trivial.

Let us now assume that we are dealing with two slices that represent the same object, but possibly slightly modified and rotated for the second slice. Our goal is to derive an algorithm that would be capable of aligning these two slices in a very short time, and with an accuracy sufficient for most practical applications. Notice, however, that we are not striving for a perfect alignment, which would be computationally intensive and might result in an excessive computation time. It should also be noticed that geometric transformations, like rotation or scaling, always result in some errors due the finite arithmetic and discrete voxel space.

The algorithm consists of two parts: translation and rotation.

A. *Translation*

The translation part is performed first and proceeds as follows:

1. A sub-space in slice *A*, denoted as *chunk* C_A, is selected. The chunk has the form of a square with the dimension $1 + 2k$, where k is a positive integer. The chunk has an odd integer dimension in order to facilitate a rotation around its central pixel.
2. Let $k = 1$, which means the size of the chunk C_A is equal to 3 x 3 = 27 pixels. Initially, the chunk is positioned in the center of slice *A* and its contents are inspected. If its pixel values diverge sufficiently (i.e., the number of pixels with a different pixel value is greater than the predefined percentage determined experimentally), the same starting point is positioned in slice *B*. If not, a new starting point in slice *A* is selected in the area with *x*, *y* coordinates within ± ¼ of the slice dimension from the center and its contents are inspected. Once the suitable chunk C_A is found, the counterpart of its central point P_A is located in slice *B* and denoted by P_B.
3. In slice *B*, the chunks C_B positioned in the area within ± ¼ of the slice dimension from P_B are selected. The chunks C_B have a dimension equal to $1 + 2(k + 1)$.
4. Repeat the following actions until there is only a single chunk C_B left:
   a. Inspect the contents of the chunks C_B and select those with the highest number of pixel values common to chunk C_A. The rest of the chunks C_B are deleted. If the number of remaining chunks C_B is equal to or less than 9, the distances between their central points are checked. The chunks C_B with central points lying within the area of 3 x 3 pixels are substituted by a single chunk with its central point positioned at the average position of chunk central points.
   b. If the size of the chunks C_B is equal to half of slice *B*, exit the repeat loop; otherwise, increase the size of chunk C_A and the chunks C_B (i.e., $k \leftarrow k + 1$).

According to the experiments, the percentage of different pixel values in the chunk C_A should be at least 75%. The background of the above estimation is to avoid the positioning of chunk C_A in the area where the pixel values are almost the same. In contrast to chunk C_A with the dimension $1 + 2k$, the chunks C_B have a dimension equal to $1 + 2(k + 1)$. This precaution is taken to ensure that the area in slice *A* selected by the chunk C_A corresponds to the area selected by C_B in slice *B*, considering the fact that the slices *A* and *B* could be arbitrarily rotated.

The final result of the translation is the difference in the coordinates of C_A determined in step 2 and the coordinates of C_B determined in step 4.

B. *Illustrative example*

So as to make it easier to understand, a demonstration video is available at [http://gemma.uni-mb.si/VoxelMatching/].

Pictures of some of the individual steps are included in the paper in order to illustrate the operation of the algorithm. For this, the pixel size was artificially increased. The first set of pictures illustrates the translation part of the algorithm. Individual steps are briefly described below.

The red square in Figure 1 represents the area that is inspected in order to find the initial chunk C_A. (with the dimension 3 x 3 pixels) whose pixel values diverge sufficiently. The green square indicates the selected chunk C_A.

The corresponding search area in slice *B* is shown in Figure 2 together with the set of chunks C_B with the highest number of pixel values common to chunk C_A. Notice that their dimension is 5x5 pixels.

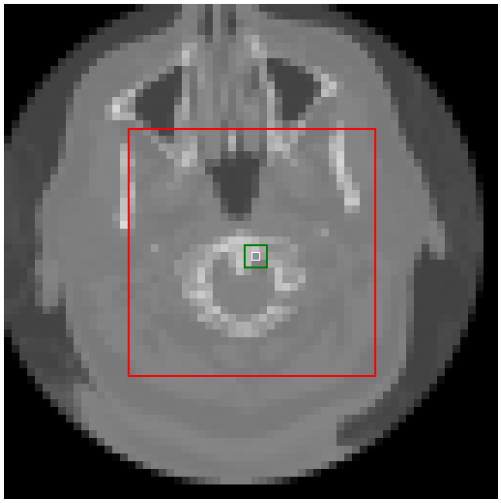

Figure 1. Inspected area and the selected chunk in slice A

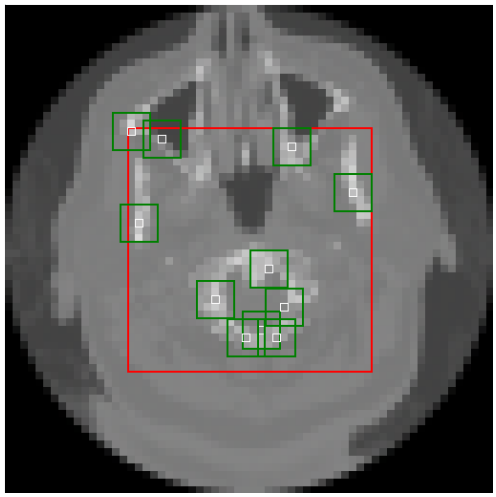

Figure 2. Search area and the set of chunks in slice B

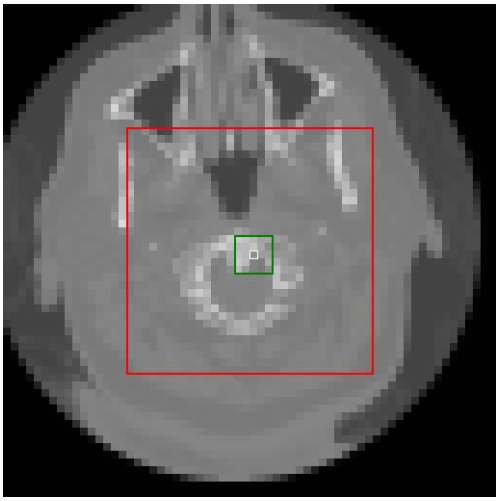

Figure 3. Increased chunk C_A in slice A

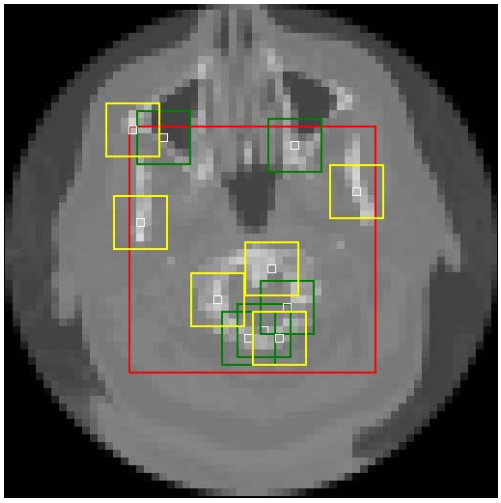

Figure 4. Increased chunks C_B in slice B

Next, the size of chunk C_A (Figure 3) and the chunks C_B (Figure 4) are increased and again the chunks C_B with the highest number of pixel values common to chunk C_A are selected (the yellow squares in Figure 4). The number of yellow squares is less than 9; hence, we check their central points. The central points are not positioned within the area of 3 x 3 pixels; therefore, the process continues.

In the last step (Figure 5 and Figure 6) the yellow square has the highest number of pixel values common to chunk C_A and represents the result of the translation.

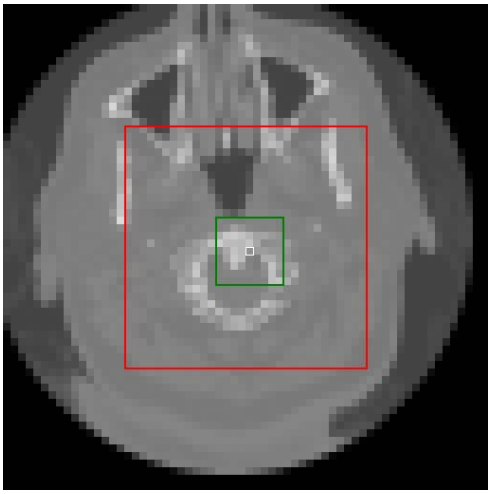

Figure 5. Increased chunk C_A in slice A

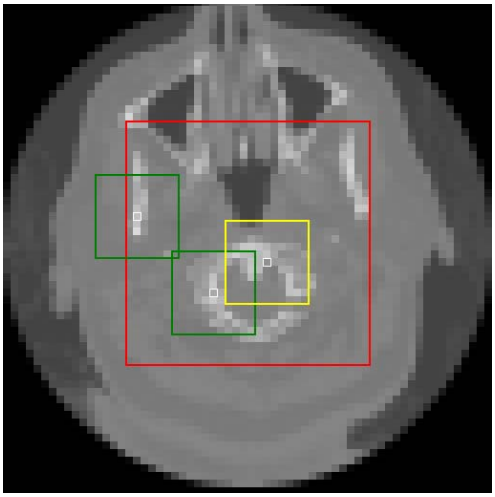

Figure 6. Increased chunks C_B in slice B, the yellow chunk represents the result of the translation

C. *Rotation*

The rotation part is performed next and proceeds as follows:

1. Compute the shortest distance from P_A to the edge of the slice. Generate a circle with its center at P_A and a radius r equal to 80% of the shortest distance to the edge of the examined object. Like in the translation part, chunks with the dimension $1 + 2k$ are considered.
2. Let $k = 1$, which gives the size of the chunk C_A equal to 3 x 3 = 9 pixels. Inspect all the chunks C_A with the center points on the circle and select a chunk with pixel values that mostly diverge.
3. In the slice *B*, the chunks C_B with the center points positioned in the ring within $r \pm 1$ are selected. The chunks C_B have a dimension equal to $1 + 2(k + 1)$.
4. Repeat the following actions until there is only a single chunk C_B left:
    a. Inspect the contents of the chunks C_B and select those with the highest number of pixel values common to the chunk C_A. The rest of the chunks C_B are deleted. If the number of remaining chunks C_B is equal to or less than 9, the distances between their central points are checked. The chunks C_B with central points lying within the area of 3 x 3 are substituted by

a single chunk with its central point positioned at the average position of chunk central points.

b. If the size of the chunks C_B is equal to half of the slice *B*, exit the repeat loop; otherwise, increase the size of chunk C_A and the chunks C_B (i.e., $k \leftarrow k + 1$). For those chunks C_B that overlap the slice margin the values of the pixels lying outside are set to zero.

The result of the rotation is the angle between the vectors P_AC_A and P_BC_B. Vector P_AC_A is constructed from the central point P_A and C_A, determined in step 2, and the vector P_BC_B from the central point P_B to C_B determined in step 4.

D. Illustrative example(continued)

The next set of pictures illustrates the rotation part of the algorithm. The most significant steps of the selection of chunks in the slices *A* and *B* are presented.

In Figure 7 the central point P_A of the chunk C_A from Figure 5 is positioned. A circle with a radius of 80% of the shortest distance to the edge is shown in yellow. The most divergent chunk C_A (denoted by yellow) is selected from the inspected chunks with the central points on the circuit.

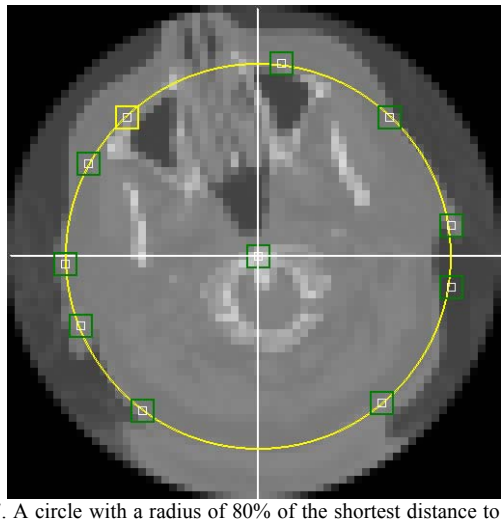

Figure 7. A circle with a radius of 80% of the shortest distance to the edge in slice A

In slice *B* we draw a circle from the central point of the chunk C_B, as positioned in Figure 6. The chunks C_B in the ring within $r \pm 1$ are inspected. Those with the largest number of pixel values common to chunk C_A are shown in yellow.

Similar steps are carried out until a single chunk C_B with the highest number of pixel values common to chunk C_A (shown in yellow in Figure 10) is left.

E. General remark

The actual implementation of the algorithm in 3D assumes voxels instead of pixels, and the chunks are cubes with dimensions $1 + 2k$ in the case of C_A, and $1 + 2(k + 1)$ in the case of C_B. The concept of the algorithm is basically the same as shown above.

Experimental results on 3D objects are presented in the following section.

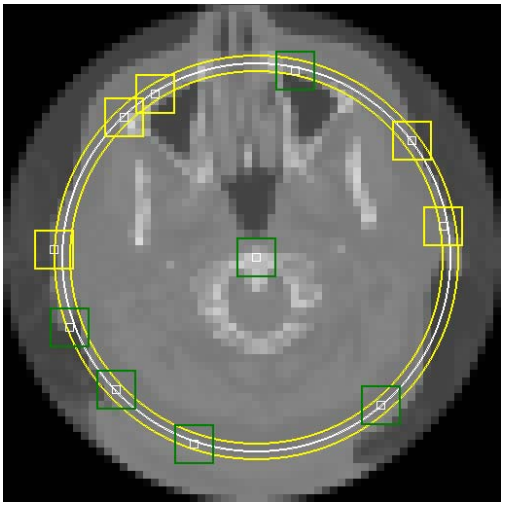

Figure 8. In slice B chunks C_B in the ring within $r \pm 1$ from the central point are inspected

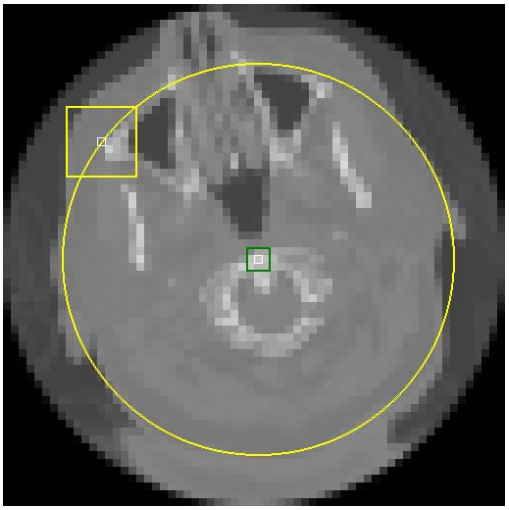

Figure 9. Repeated step in slice A

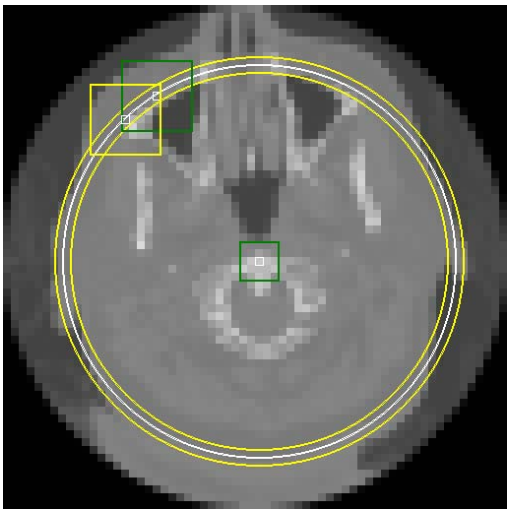

Figure 10. A single remaining chunk C_B (shown in yellow) with the highest number of pixel values common to chunk C_A

III. RESULTS

The algorithm was tested on three datasets (head, foot, bonsai) of resolution 256 × 256 × 256 voxels, downloaded from [27]. Each voxel space was rotated by five different angles and the required CPU time for the alignment was measured. While a short computation time is an important issue, an analysis of the associated errors should be

performed in order to establish the actual validity of the developed solutions. For this reason we first explored the inherent error due to the finite arithmetic and discretization. In the next step, the computation error of the proposed algorithm was addressed.

TABLE I. MEAN-SQUARED-ERROR (MSE) AFTER ROTATING THE VOXEL SPACE

α^O	MSE(Head)	MSE(Foot)	MSE(Bonsai)
±1	1.58	2.28	2.87
±2	2.21	3.22	4.13
±3	2.75	3.99	5.08
±4	3.12	4.49	5.80
±5	3.48	4.95	6.42
±6	3.77	5.32	6.95
±7	4.05	5.69	7.42
±8	4.28	5.96	7.75
±9	4.48	6.23	8.12
±10	4.69	6.44	8.37

Table 1 refers to the case where the voxel space was rotated around the *z* coordinate axis for an angle α followed by a rotation of – α. The content of the obtained voxel space was compared with the original voxel space and the mean-squared-error MSE was calculated.

Next, we performed operations in which the computation error of the proposed algorithm is demonstrated. We rotated the original voxel space for a given angle α and applied our method to determine the angle of rotation and to align the voxel space. The calculated angles of rotation are presented in Table 2.

TABLE II. CALCULATED ANGLE OF ROTATION

α^O	α (Head)	α (Foot)	α (Bonsai)
1	1.28	1.14	0.91
2	2.01	2.11	2.04
3	3.03	2.86	3.13
4	4.11	4.01	4.07
5	5.03	5.15	5.01
6	5.86	5.84	6.11
7	7.28	6.96	7.01
8	8.22	8.11	8.09
9	9.03	9.08	9.06
10	10.21	9.97	10.06

In Table 3, the resulting MSE_M of the voxel values between the original and the rotated voxel spaces are given.

TABLE III. THE RESULTING MSE_M

α^O	MSE_M(Head)	MSE_M(Foot)	MSE_M(Bonsai)
1	4.53	4.66	3.96
2	2.24	5.18	4.52
3	3.03	5.74	6.06
4	4.01	4.54	6.22
5	3.62	6.47	6.42
6	4.66	6.78	7.5
7	5.67	5.86	7.4
8	5.5	6.79	8.07
9	4.54	6.71	8.25
10	5.71	6.51	8.47

The ratio between the mean-squared-error caused exclusively by the finite arithmetic and the one obtained with the proposed algorithm is given in Table 4.

TABLE IV. THE RATIO MSE/ MSE_M

α^O	MSE/MSE_M(Head)	MSE/MSE_M(Foot)	MSE/MSE_M(Bonsai)
1	0.348	0.489	0.724
2	0.986	0.621	0.913
3	0.907	0.695	0.838
4	0.778	0.988	0.932
5	0.961	0.765	1.000
6	0.809	0.784	0.926
7	0.714	0.970	1.002
8	0.778	0.877	0.960
9	0.986	0.928	0.984
10	0.821	0.989	0.988

It is clear that the additional error introduced by our method is small. Notice that the above error analysis has been intentionally performed on examples including only rotation, because the translation part of the alignment process is error free.

TABLE V. CPU RUN TIME

Dataset	Resolution	α°	CPU time (s)
Head	256 × 256 × 256	1	4.5
		3	4.5
		5	4.5
		8	4.5
		10	4.6
Foot	256 × 256 × 256	1	3.9
		3	4.0
		5	4.0
		8	3.9
		10	3.7
Bonsai	256 × 256 × 256	1	5.1
		3	5.3
		5	5.5
		8	5.3
		10	5.2

Table 5 summarizes the CPU run time when applying our method in the cases of the head, foot and bonsai datasets for aligning the voxel data at various angles. A system with an Intel Core2 Quad Q6600 2.40-GHz processor with 4.00 GB of RAM running under the Windows 7 64-bit operating system was employed. The prototype application was implemented in C#. The run time of the algorithm is two orders of magnitude smaller than the one reported in [16].

IV. CONCLUSION

This paper presents a new algorithm for the alignment of voxel data. In contrast to the existing solutions, we do not strive for perfect alignment, but prefer a short execution time at the expense of a reasonably small alignment error. The proposed approach is, by two orders of magnitude, faster than [16] and can be efficiently applied in practice in cases where precise adjustments are not imperative (for example, in the compression of time-varying voxel spaces [28]). Furthermore, the proposed algorithm can precede other more exact algorithms for a rough adjustment in order to cut down the time for a precise alignment.

REFERENCES

[1] B. Benes and R. Forsbach, "Layered Data Representation for Visual Simulation of Terrain Erosion," Proceedings of the 17th Spring conference on Computer graphics, 2001, pp. 80-86.

[2] J. Stoker, "Volumetric Visualization of Multiple-return Lidar Data: Using Voxels," Photogrammetric Engineering & Remote Sensing; 2009, pp. 109-112.

[3] T. Shinohara, J. Takayama, S. Ohyama and A. Kobayashi, "Analysis of Knit Fabric Structure with its Voxel Data," ICCAS2003, Gyeongju, KOREA, 2003.

[4] S. Venkataraman and K. O. Asante, "Voxel-Based Analysis and Visualization of Rainfall Data," Global Priorities in Land Remote Sensing, 2005.
[5] S. M. Seitz and C. R. Dyer, "Photorealistic Scene Reconstruction by Voxel Coloring," Int. Journal of Computer Vision, vol. 35, no. 2, 1999.
[6] Y. Sun, M. Bray, A. Thayananthan, B. Yuan and P.H.S. Torr, "Regression-Based Human Motion Capture From Voxel Data," Proceedings of BMVC06, 2006, pp 109-112.
[7] M. E. Mortenson, "Geometric Modeling,". John Wiley & Sons Inc, 1985.
[8] G. Dougherty, "Digital Image Processing for Medical Applications," New York: United States of America by Cambridge University Press, 2009.
[9] B. Li, Z. Wang, E. Smouha, D. Chen and Z. Liang, "Accelerating Virtual Surgery Simulation for Congenital Aural Atresia," Proceedings of SPIE Vol. 5367, Bellingham, 2004.
[10] F. Dong, G.J.Clapworthy, M. Krokos, "Volume rendering of fine details within medical data", Proceedings of the conference on Visualization'01, 2001, pp. 387-394.
[11] M. Zemek, J. Skála, I. Kolingerová, P. Medek, J. Sochor, "Fast Method for Computation of Channels in Dynamic Proteins", 13th International Fall Workshop Vision, Modeling and Visualization 2008, 2008, pp.333-342.
[12] A. S. M. Houston and S. Napel, "Fast Volume Segmentation With Simultaneous Visualization Using Programmable Graphics Hardware," IEEE Visualization, 2003, pp. 171-176.
[13] S. M. Smith, M. Jenkinson, M. W. Woolrich, C. F. Beckmann, T. E.J. Behrens, H. Johansen-Berg, P. R. Bannister, M. De Luca, I. Drobnjak, D. E. Flitney, R. K. Niazy, J. Saunders, J. Vickers, Y. Zhang, N. De Stefano, J. M. Brady and P. M. Matthews, "Advances in functional and structural MR image analysis and implementation as FSL," NeuroImage 23, 2004, pp. 208–219.
[14] K-L. Ma, "Visualizing time-varying volume data," Computing in Science and Engineering, vol. 5, no. 2, 2003, pp. 34–42.
[15] H. Akiba, K-L Ma and J. Clyne, "End-to-end data reduction and hardware accelerated rendering techniques for visualizing time-varying non-uniform grid volume data," Proceedings of the 4th international workshop volume graphics, 2005. pp. 31–39.
[16] G. Guetat, M. Maitre, L. Joly, S. Lai, T. Lee and Y. Shinagawa, "Automatic 3-D Grayscale Volume Matching and Shape Analysis," IEEE Transactions On Information Technology In Biomedicine, vol. 10, no. 2, 2006.
[17] A. Roche, X. Pennec, G. Malandain and N. Ayache, "Rigid registraion of 3-D ultrasound with MR images: A new approach combining intensity and gradient information," IEEE Trans.Med. Imag., vol. 20, no. 10, 2001, pp. 1038–1049.
[18] E. Guest, E. Berry, R. Baldock, M. Fidrich and M. Smith, "Robust point correspondence applied to two- and three-dimensional image registration," IEEE Trans. Pattern Anal.Mach. Intell., vol. 23, no. 2, 2001, pp. 165–179.
[19] R. Bajcsy and S. Kovacic, "Multiresolution elastic matching," Comput. Vis. Graph. Image Underst., vol. 46, no. 1, 1989, pp. 1–21.
[20] R. Urtasun, M. Salzmann and P. Fua, "3D Morphing without User Interaction," Eurographics Symposium on Geometry Processing, 2004.
[21] T-Y. Lee, C-H. Lin and H-Y. Lin, "Computer-aided prototype system for nose surgery (rhinoplasty)," IEEE Trans. Inf. Technol. Biomed., vol. 5, no. 4, 2001, pp. 271–278.
[22] J. Parus, I. Kolingerova and M. Malkova, "Multimorphing: A tool for shape synthesis and analysis," Advances in Engineering Software, vol.40, 2009, pp.323–333.
[23] T. He, S. Wang and A. Kaufman, "Wavelet-Based Volume Morphing," Proceedings of Visualization '94; Washington D.C., 1994, pp. 85–92.
[24] M. Hilaga,Y. Shinagawa, T. Komura and T. L. Kunii, "Topology matching for full automatic similarity estimation of 3D," Proc. SIGGRAPH 2001, Los Angeles, CA, 2001, pp. 203–212.
[25] N. Amenta, M. W. Bern, M. K. Kamvysselisand A.Crust, "A new Voronoi-based surface reconstruction algorithm," Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, 1998, pp. 415–422.
[26] H. Edelsbrunner, "The union of balls and its dual shape," Proc. 9th Annu. ACM Symp. Discrete and Computational Geometry, vol. 13, 1995, pp. 415–440.
[27] http://www9.informatik.uni-erlangen.de/External/vollib/
[28] K-L. Ma, D. Smith, M.-Y. Shih and H.-W. Shen, "Efficient Encoding and Rendering of Time-Varying Volume Data," ICASE report ; no. 98-22,1998.