Augmented reality approaches in intelligent health technologies and brain lesion detection

Tomasz Hachaj[1], Marek R. Ogiela[2]

[1] Pedagogical University of Krakow, Institute of Computer Science and Computer Methods, 2 Podchorazych Ave, 30-084 Krakow, Poland, tomekhachaj@o2.pl
[2] AGH University of Science and Technology
30 Mickiewicza Ave, 30-059 Krakow, Poland, mogiela@agh.edu.pl

Abstract. In this paper authors present their new proposition of system for cognitive analysis of dynamic computer tomography perfusion maps (dpCT). The novel contribution of this article is introducing an augmented reality visualization module that supports real time volume rendering (VR) of derived data. Authors also presents the results of their researches on optimization of VR algorithm memory usage by dynamic computation of volume gradient instead of pre-generation of gradient Authors compare five different discrete gradient computation schemas taking into account image quality and processing speed on two VR algorithms: volume ray casting and texture based visualization with view aligned slices.

Keywords: Pattern recognition, cognitive analysis, dynamic brain perfusion, volume rendering, gradient estimation, augmented reality.

1 Introduction

The ensuring of patient security during the health care is one of the basic duties of medical personnel [1]. Nowadays many dedicated diagnosis support systems (DSS) emerge on purpose to help physicians in everyday practice. The modern DSS has to satisfy many requirements in order to be accepted in medical society. It has to be not only a reliable tool (low total error rate coefficient) but also quickly generates the support information and has intuitive interface.
In this article authors present their new proposition of system for cognitive analysis of dynamic computer tomography perfusion maps (dpCT) that satisfies all of those requirements.
The proposed solution is the extension of previous works [2], [3], [4]. The novel contribution of this article is introducing an augmented reality visualization module that supports real time volume rendering (VR) of derived data.
Authors also present the results of their researches on optimization of VR algorithm memory usage. The huge amount of GPU memory may be saved by dynamic computation of volume gradient instead of pre-generation of gradient. Authors compare five different discrete gradient computation schemas taking into account image quality and processing speed on two VR algorithms: volume ray casting and

texture based visualization with view aligned slices. Many similar researches on the field of gradient reconstruction for VR were previously reported (for example in [5], [6], [7]). Authors intention was to compare the speed of two most popular VR algorithms with pre-generated and dynamically computed volume gradient on clinical data using different transfer function [8] in order to justify using one (or both) of them in their diagnosis support program. Despite the fact that transfer function is the factor that might highly affect the speed of the VR (especially when rendering algorithms utilizes acceleration techniques like early ray termination or empty-space skipping [9]) we did not found papers in which transfer function was taken into account during dynamic gradient computation. Because of that the presented comparison is also important contribution to state of art of VR visualization.

2 Diagnosis support system architecture

The authors system enables the quantitative and quality analysis of visualized symptoms like head injuries, epilepsy, brain vascular disease, ischemic and hemorrhagic stroke that changes blood perfusion. The new implementation of the DMD (detection measure and description system [2]) also includes intuitive augmented reality visualization module derived from [10]. The schema of the system is presented in Figure 1.

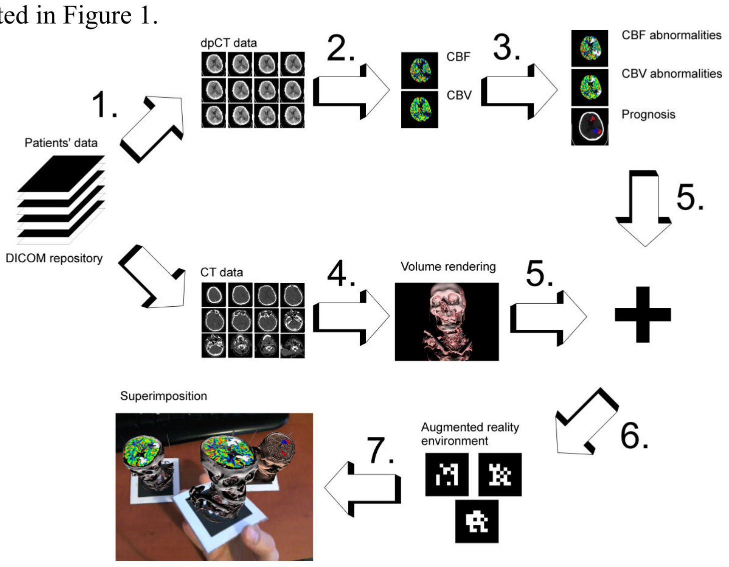


Figure 1. Data flow in DMD system with augmented reality visualization module. Detailed description in text.

The patient's data (1) is kept in DICOM repository. It is consisted of neuroradiology images like dpCT, CT (computer tomography) and MR (magnetic resonance). The dpCT is used for generation of perfusion CBF and CBV maps (2). The next step (3) is detection of perfusion abnormalities and generation of ischemia prognosis for infarcted tissues. The CT (or RM) data is processed and visualized with VR algorithms (4). The data from (3) is superimposed into the 3D object (5). The result date is superimposed with augmented reality environment (6) onto image captured by digital camera (7).

2.1 Detection of perfusion abnormalities

The process of an analysis of dpCT proposed by authors is a fusion of image processing, pattern recognition and image analysis procedures. All of these stages will be described in this paragraph. The input data for the algorithm is cerebral blood flow, cerebral blood volume and CT image.

Image processing. Image processing step is consisted of lesion detection algorithm and image registration algorithm. The algorithm used for detection of potential lesions is The Unified Algorithm detailed described in [2]. Lesion detection algorithm finds potentially pathologic tissues (regions of interests - ROI). Image registration algorithm is used for creating of deformable brain atlas in order to make detailed description of visible tissues. Potential lesions are compared with corresponding CT / MR image in order to check its presence there. This process enables proper treatment planning. For registration purpose authors have chosen free-from deformation algorithm proposed in [11]. The detailed (AA based) description of image in authors system has been presented elsewhere [12].

Image analysis. Defining the features of entire image after lesion detection step is an easy task. Algorithm measures some important from (medical point of view) features:

- Perfusion in ROI in left and right hemisphere.
- Relative perfusion (perfusion in ROI in left hemisphere divided by perfusion in ROI in right hemisphere and perfusion in ROI in right hemisphere divided by perfusion in ROI in left hemisphere).
- Size of ROI.

The scaling factors between perfusion map and “real brain” can be derived directly from DICOM files.

Pattern recognition. In pattern recognition step algorithm determinate what type of lesion was detected and in which hemisphere. In order to do it is necessary to gather medical knowledge about average perfusion values. After image processing step two symmetric regions are detected in left and right hemisphere. Authors’ algorithm compares perfusion in left and right (symmetrical) ROI with average perfusion norms and place potential lesion in hemisphere where modulus of difference between average and ROI value is greater. After this it is an easy task to determinate the type of lesion (hemorrhagic or ischemic) simply by checking if perfusion in ROI is greater or smaller than average.
The last step done by the algorithm is to state prognosis for lesion evolution in brain tissues. CBF and CBV have prognostic values in evaluation of ischemic evolution. In many cases simultaneous analysis of both CBF and CBV perfusion parameters enables accurate analysis of ischemia visualized brain tissues and predict its further

changes permitting a not only a quality (like CT angiography) but also quantitative evaluation of the degree of severity of the perfusion disturbance which results from the particular type of occlusion and collateral blood.
The algorithm analyze both perfusion maps simultaneously in order to detect:

- Tissues that can be salvaged (tissues are present on CBF and CBV asymmetry map and values of rCBF did not drop beyond 0.48 [13]).
- Tissues that will eventually become infracted (tissues are present on CBF and CBV asymmetry map and values of rCBF did drop beyond 0.48 [13]).
- Tissues with an auto regulation mechanism in ischemic region (decreased CBF with correct or increased CBV).

Summing up, the output data of the algorithm is consisted of: regions of perfusion abnormalities, AA description of brain tissues, measures of perfusion parameters and prognosis for infracted tissues. That information is superimpose onto volumetric CT data and displayed to radiologist.

2.2 GPU-based volume rendering

Volume rendering describes a wide range of techniques for generating images from three-dimensional scalar data [8]. These techniques are originally motivated by scientific visualization, where volume data (three dimension arrays of pixels) is acquired by measurement or numerical simulation of natural phenomena. Typical examples are medical data of the interior of the human body obtained by computed tomography (CT) or magnetic resonance imaging (MRI). The scalar data (often monochromatic) is mapped to color space by transfer function that is often implemented as lookup table.
The power of GPUs is currently increasing much faster than that of CPUs. That trend forces many computer programmers to move the burden of algorithm computation to GPU processors. Also the hardware support for interpolation of not only 2D but also 3D texture data enables the rendering of complex volumetric images in real time. Nowadays graphic programmers utilize high-level languages that support implementation of highly parallel algorithms that runs on programmable GPU's shaders. The main obstacle that must be overcome during creation of volume rendering algorithm is the fact that contemporary computer hardware still does not support direct rendering of volumetric data. There are two main groups of the algorithms that support fast visualization of volumetric data with hardware accelerated interpolation. The first group is ray-casting algorithms the second one texture-based algorithms. Both groups are capable to produce almost identical visualization results but they have quite different schemas. Moreover algorithms differs much in performance speed, which is important factor that must be taken into account in case of augmented reality environment [14]. All algorithms described below use tri-linear interpolation hardware support.

Volume ray-casting algorithms. In ray-casting process for each pixel in the image to render, algorithm casts a single ray from the eye through the pixel's center into the volume, and integrates the optical properties obtained from the encountered volume densities along the ray. Algorithm uses standard front to back blending equations in order to find color and opacity of rendered pixels:

$$C_{dst} = C_{dst} + (1-\alpha_{dst})\alpha_{src}C_{src} \tag{1}$$

$$\alpha_{dst} = \alpha_{dst} + (1-\alpha_{dst})\alpha_{src} \tag{2}$$

Where C_{dst} and α_{dst} are the color and opacity values of the rendered pixels and C_{src} and α_{src} are the color and opacity values of the incoming fragment.
The popular approach proposed in [9] includes standard acceleration techniques for volume ray casting like early ray termination and empty-space skipping. By means of these acceleration techniques, the framework is capable of efficiently rendering large volumetric data sets including opaque structures with occlusions effects and empty regions.

Texture-based algorithms. The ray casting approach is a classical image-order approach, because it divides the resulting image into pixels and then computes the contribution of the entire volume to each pixel [8]. Image-order approaches, however, are contrary to the way GPU hardware generates images. Graphics hardware usually uses an object-order approach, which divides the object into primitives and then calculates which set of pixels primitive influences.
In order to perform volume rendering in an object-order approach, the resampling locations are generated by rendering proxy geometry with interpolated texture coordinates.
The most basic proxy geometry is a stack of planar slices, all of which are required to be aligned with one of the major axes of the volume (either the x, y, or z axis) [8]. During rendering, the stack with slices most parallel to the viewing direction is chosen. In order to minimize the switching artifacts inter-slice interpolation ought to be included [8].
The more complex proxy geometry is slices aligned with the viewport [8]. Such slices closely mimic the sampling used by the ray-casting algorithm. The sufficient number of slices required for accurate visualization can easily be adjusted during algorithm performance.

2.3 Augmented reality

Augmented reality (AR) is a technology that allows the real time fusion of computer generated digital content with the real world. Unlike virtual reality (VR), that completely immerses users inside a synthetic environment, augmented reality allows the user to see three-dimensional virtual objects superimposed upon the real word [15].
Augmented reality shows its usefulness especially in the field of the medicine [16]. The most notable examples are deformable body atlases, AR surgical navigation systems, interfaces and visualization systems. Pre, intra and post – operative visualization of clinical data is a major source of information for decision making.

Augmented Reality aim at lifting this support to a new level by presenting more informative and realistic three-dimensional visualizations [17]. In our system AR is used as intuitive interface that enable easy and reliable manipulation of visualized objects.
Augmented reality environment used by authors utilizes size - known square markers. The transformation matrices from these marker coordinates to the camera coordinates are estimated by image analysis. The details of the algorithm can be found in [15].

3 Gradient reconstructions for VR

Illumination and shading within volume rendering refers to the same illumination models and shading techniques used in polygon rendering. The goal is to enhance the appearance of rendered objects, especially to emphasize their shape and structure, by simulating the effects of light interacting with the object [18].
The most common local illumination model, which calculates light per pixel, is the Phong [19] lighting model. The Phong model is made up of three parts, ambient, diffuse and specular.

$$L = L_{ambient} + L_{diffuse} + L_{specular} \tag{3}$$

$$L_{ambient} = I_{ambient} \cdot K_{ambient} \tag{4}$$

Where $I_{ambient}$ is the constant intensity of the ambient light and $K_{ambient}$ is the coefficient of the ambient reflection of the surface. The ambient term is a constant, which simulates indirect light on the parts of geometry where there is no direct light, they would otherwise be entirely black.

$$L_{diffuse} = \begin{cases} I_{diffuse} \cdot K_{diffuse} \cdot \cos(\alpha), & |\alpha| < 90^\circ \\ 0 \quad otherwise \end{cases} \tag{5}$$

Where $I_{diffuse}$ is light source intensity, $K_{diffuse}$ is a material constant describing color and α is the angle between light source and surface normal. The diffuse term calculates the reflection of light from a surface without shininess, called a matte surface. The light is reflected at the same angle as it is striking the surface relative to the surface normal.

$$L_{specular} = \begin{cases} I_{specular} \cdot K_{specular} \cdot \cos^m(\beta), & |\beta| < 90^\circ \\ 0 \quad otherwise \end{cases} \tag{6}$$

where $I_{specular}$ is the intensity of the incident light, $K_{specular}$ is the coefficient of specular reflection for the material, β is the angle between the reflection vector and the viewing vector and m controls the extension of the highlight. The specular term is

added to simulate the shininess of surfaces, it creates highlights, which give the viewer cues of light source positions and geometry details.
An essential part of the illumination model is the surface normal. In VR case a surface normal is replaced by the gradient vector calculated at all points of interest [20].

$$\nabla f(x,y,z) = \left(\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz} \right) \tag{7}$$

There are several methods for calculating the gradient vector. The most common are (see Appendix: 3D discrete gradient operators):

- Intermediate difference operator (8) [20].
- Central difference operator (9) [20].
- Neumann gradient operator (10) [20].
- Zucker-Hummel operator (11) [21].
- Sobel operator (12) [18], [22].

Intermediate difference gradient takes as input four neighboring voxels, Central difference gradient takes as input six neighboring voxels, the rest of operators take as input 26 or 27 neighboring voxels. The 26/27 neighbors give usually a better estimation of the gradient, but take more time to calculate. Another disadvantage is that additional smoothing might be introduced [23].
There are two basic methods of gradient computation strategy [7]: pre-generation of volume gradient and passing it to GPU memory by 3D texture or dynamic computation during rendering process. The first method requires large amount of memory (for each volume voxel the three gradient coefficients must be stored). The second method does not require additional memory (that means it uses even four times less memory then pre-generation) but is more expensive computationally as the components are calculated separately. Capacity of GPU memory is still bottleneck in VR algorithm even in visualization of typical medium-sized medical data. The question that we answered in the next paragraph is what is the performance speed of VR algorithms with pre-generated versus dynamically computed volume gradient on typical (off-the-shelf) graphical hardware and if it is sufficient for our needs.

4 Results and discussion

The five gradients calculation methods from the previous paragraph where compared using three volume data (real CT images) of the size 256x256x221, 256x256x212 and 256x256x207 voxels. All of those models were rendered with two transfer function: transfer function with huge amount of semi transparent pixels (Figure 2, top row) and function without semi transparent pixels (Figure 2, bottom row). Those are two boundary cases that might be considered in medicine practice.

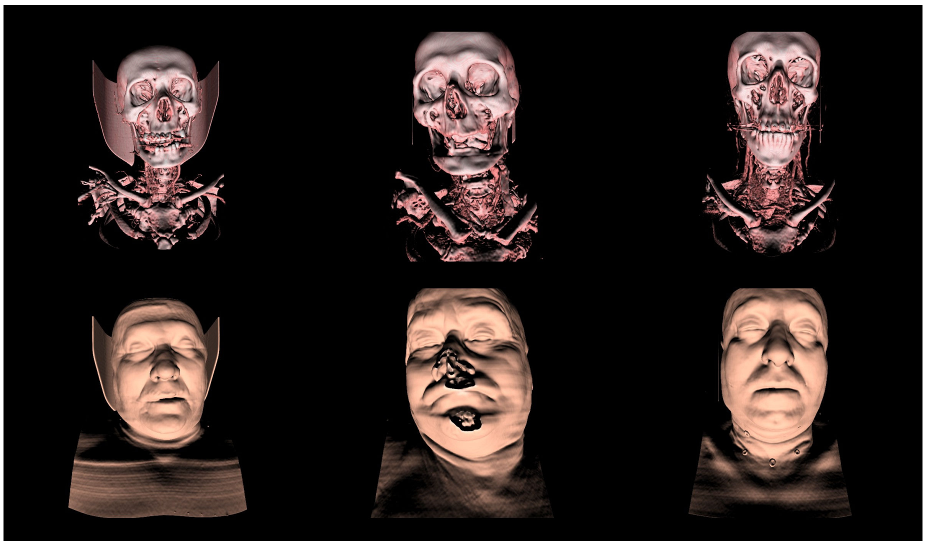


Figure 2. Three different models based on volume CT data. Top row – models with transfer function with semi transparent pixels. Bottom row – same models with transfer function without semi transparent pixels.

For rendering purpose authors used two VR algorithms: volume ray casting (with early ray termination and empty space skipping) and texture based algorithm with view-aligned slices. The algorithms were implemented in .NET C#, XNA Framework 2.0 (Direct X 9.0) with HLSL and executed on Intel Core 2 Duo CPU 3.00 GHz processor, 3.25 GB RAM, Nvidia GeForce 9600 GT graphic card with 32 – bit Windows XP Professional OS.
The performance speed of VR was computed for three volume datasets mentioned above and then averaged. The gradient calculation methods were pre-computation and dynamic computation with 4-points, 6-point and 27-point method. The results are presented in Table 1 and Figure 3.

Table 1. Average performance speed (fps) of 3D models as a function of rendering algorithm type, gradient computation method and transfer function.

Rendering algorithm	**Gradient computation method**	**Transfer function**	
		Semi-transparent pixels	No semi-transparent pixels
View-aligned slices	Pre-computation	38.00	38.00
	4 - points	28.33	15.67
	6 - points	24.67	12.33
	27 - points	8.33	2.67
Volume ray casting	Pre-computation	29.67	54.00
	4 - points	12.67	32.00
	6 - points	11.00	28.67
	27 - points	2.67	9.67

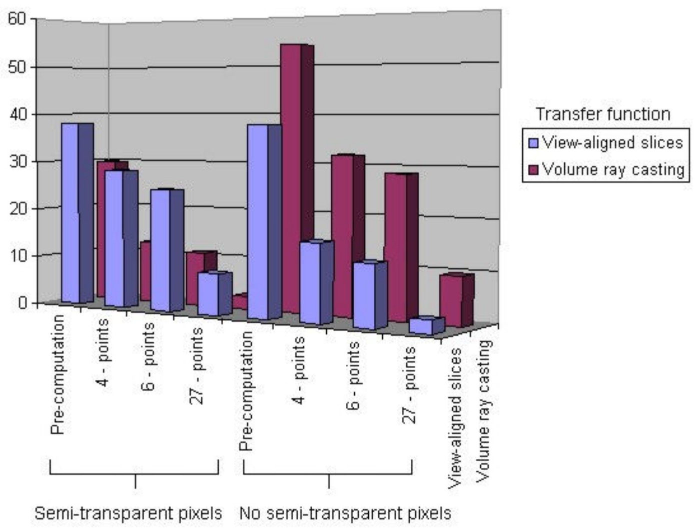


Figure 3. Data from Table 1 presented in the chart. Dependents of average performance speed (fps) of 3D models as a function of rendering algorithm type, gradient computation method and transfer function.

The most time demanding operation is retrieving image data (voxel values) from 3D texture in GPU for computing of a gradient. This is the reason why algorithms with dynamic computed gradients are slower than with their pre-computed versions. The more points are needed for gradient calculation the slower the whole algorithm is. The very important factor that highly affects performance is a transfer function that is used for volume rendering. Volume ray casting is the fastest algorithm if the transfer function do not have many semi-transparent pixels. It is caused by the fact that in that case early ray termination quickly finishes the ray lookup loop. The texture-based algorithm has to render all slices no matter if there are visible or not. On the other hand if there are many semi-transparent pixels the empty space skipping and ray termination will not help the ray casting to improve program speed. The number of computation and references to 3D texture memory is higher than in texture-based algorithm, which becomes quicker.
The second factor that we took into account during comparison of gradient computation methods is a quality of gradient reconstruction (Figure 4). The results was similar to those described in [5]. The intermediate difference and central difference operator are good low-pass filters. There are also highly visible quality differences between those two methods: intermediate difference generates much more lighting artifact then the 6-point methods. The Sobel gradient estimation method, as well as other 26/27-pixels gradient estimation approaches, produces less fringing artifacts than other methods. The differences between Sobel, Neumann, Zucker-Hummel are nearly invisible and does not affect the image analysis process. What is more important the differences of quality (when taking into account fringing artifacts) between central difference and the high order estimation methods do not disqualify

the 6-points method. Also visualization with central difference operator is 3 to 4 times faster than with high quality kernels.

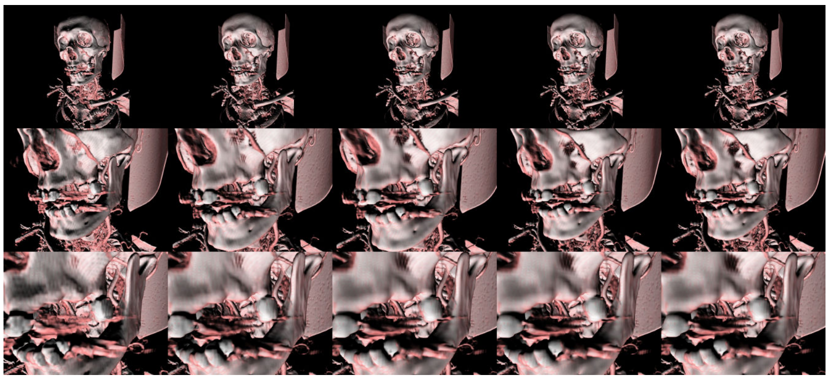
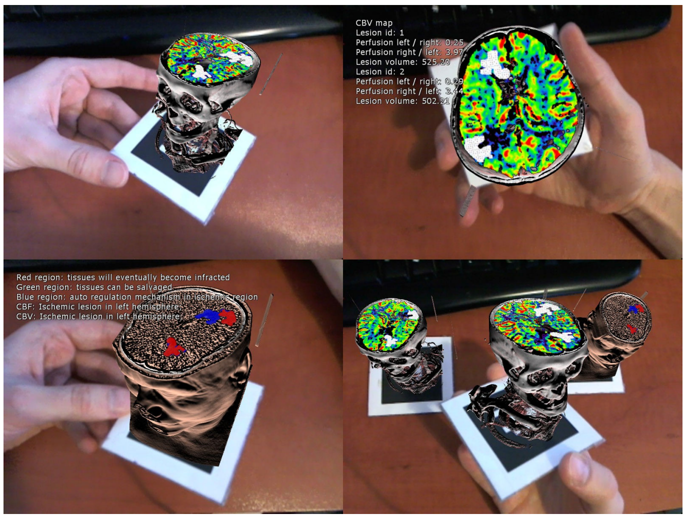


Figure 4. Volume rendering of example model with different methods of gradient estimation. From left to right: intermediate difference, central difference, Sobel, Neumann and Zucker-Hummel.

Knowing all of those results we decided to use in our solution both visualization algorithms with a dynamic central difference gradient computation method. After determining the transfer function an algorithm decides which of those method runs faster and uses it during session with program. That approach enables the visualization to run with the speed oscillating near 30 fps, which is an accepted value for our needs. The example visualization of superimposition of dpCT onto volume CT data in AR with our program is shown in Figure 5.


```
CBV map
Lesion id: 1
Perfusion left / right: 0.25
Perfusion right / left: 3.97
Lesion volume: 525.20
Lesion id: 2
Perfusion left / right: 0.29
Perfusion right / left: 3.44
Lesion volume: 502.21
```

```
Red region: tissues will eventually become infracted
Green region: tissues can be salvaged
Blue region: auto regulation mechanism in ischemic region
CBF: Ischemic lesion in left hemisphere;
CBV: Ischemic lesion in left hemisphere;
```

Figure 5. The example visualization of superimposition of dpCT onto volume CT data in AR. (A) CBF map with marked perfusion abnormalities. (B) The detailed view of CBV and prognostic image. In the left top side of each image description done by DMD system. (C) The detailed view of prognostic image. In the left top side of each image description done by DMD system. (D) From left to right: CBV, CBF and Prognostic image.

5 Conclusions and future work

This papers presents and summarizes the results of the integration of cognitive image analysis and patter recognition algorithm with a high quality hardware accelerated volume renderer into an augmented reality visualization environment. Augmented reality supplies the system with more informative and realistic three-dimensional visualizations offering very intuitive image manipulation interface. The performance of the presented techniques was demonstrated and evaluated in several experiments by rendering real human CT data with different rendering techniques. The authors also took into account the quality and speed of volume gradient estimation methods considering transfer function type.
There are many important aspects which future work should focus on. Superimposition the dpCT data onto volume CT / MR enables to perform more detailed diagnosis for example taking into account not only dynamic perfusion but also the state of the vascular system (brain angiography). The further researches should also be done in the field of real time large volume visualization that might be needed during superimposition of medical data of several modalities. Our augmented reality system should also be replaced by marker-less solution that would be more convenient for the end user.

Acknowledgments. This work has been supported by the Ministry of Science and Higher Education, Republic of Poland, under project number N N516 511939.

References

[1] Jara A. J., Zamora M. A., Skarmeta A. F. G., An Initial Approach to Support Mobility in Hospital Wireless Sensor Networks based on 6LoWPAN (HWSN6), Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, VOL. 2/3, 107-122, (2010)

[2] Hachaj, T., Ogiela, M.R.: CAD system for automatic analysis of CT perfusion maps, Opto-Electronics Review, 19(1), pp. 95-103, (2011), DOI: 10.2478/s11772-010-0071-2

[3] Hachaj T., Ogiela M.R.: Computer – assisted diagnosis system for detection, measure and description of dynamic computer tomography perfusion maps, in Ryszard S. Choraś, Antoni Zabłudowski (eds), Image Processing and Communication Challenges (2009)

[4] Hachaj T., Ogiela M. R., A system for detecting and describing pathological changes using dynamic perfusion computer tomography brain maps, Computers in Biology and Medicine 41, pp. 402-410 (2011)

[5] Bentum, M.J., Lichtenbelt, B.B.A., Malzbender T.: Frequency Analysis of Gradient Estimators in Volume Rendering, Journal IEEE Transactions on Visualization and Computer Graphics archive Volume 2 Issue 3, September (1996)

[6] Mihajlovic, Z., Budin, L., Radej, J.: Gradient of B-splines in volume rendering, Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean (2004)
[7] Csébfalvi, B., Domonkos, B.: Prefiltered Gradient Reconstruction for Volume Rendering, in WSCG: The 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Pilsen, Czech Republic, (2009)
[8] Engel, K., Hadwiger, M., Kniss, J., Rezk-Salama, C., Weiskopf, D.: Real-Time Volume Graphics, CRC Press, (2006)
[9] Krüuger, J., Westermann, R.: Acceleration Techniques for GPU-based Volume Rendering. IEEE Visualization (2003)
[10] Hachaj, T., Ogiela, M.R.: Augmented reality interface for visualization of volumetric medical data, Image Processing and Communications Challenges 2, Springer-Verlag Berlin Heidelberg (2010), pp. 271-277
[11] Parraga A., et all.: Non-rigid registration methods assessment of 3D CT images for head-neck radiotherapy, Proceedings of SPIE Medical Imaging, February, (2007)
[12] Hachaj T: The registration and atlas construction of noisy brain computer tomography images based on free form deformation technique, Bio-Algorithms and Med-Systems, Collegium Medicum - Jagiellonian University, Bio-Algorithms and Med-Systems, Vol. 7, (2008)
[13] Koenig, M., Kraus, M., Theek, C., Klotz, E., Gehlen, W., Heuser, L.: Quantitative assessment of the ischemic brain by means of perfusion-related parameters derived from perfusion CT., Stroke; a journal of cerebral circulation (2001);32(2):431-7.
[14] Kutter, O., Aichert, A., Bichlmeier, C. , Traub, J. , Heining, S.M. , Ockert, B. , Euler, E. , Navab N., Real-time Volume Rendering for High Quality Visualization in Augmented Reality International Workshop on Augmented environments for Medical Imaging including Augmented Reality in Computer-aided Surgery (AMI-ARCS 2008), USA, New York, September (2008)
[15] Haller, M., Billinghurst, M., Thomas, B.: Emerging Technologies of Augmented Reality: Interfaces and Design. Idea Group Publishing (2006)
[16] Yang, G., Jiang, T.: Medical Imaging and Augmented Reality. Second International Workshop, MIAR 2004 (2004)
[17] Denis, K. et al.: Integrated Medical Workflow for Augmented Reality Applications, International Workshop on Augmented environments for Medical Imaging and Computer-aided Surgery (AMI-ARCS) (2006)
[18] Grimm, S.: Real-Time Mono- and Multi-Volume Rendering of Large Medical Datasets on Standard PC Hardware. PhD thesis, Vienna University of Technology, Gaullachergasse 33/35, 1160 Vienna, Austria, February (2005)
[19] Phong, B.T.: Illumination for Computer Generated Pictures, Communications of the ACM, 18(6):311-317, June (1975)
[20] Jonsson, M.: Volume rendering, Master's Thesis in Computing Science, October 12, (2005), http://www8.cs.umu.se/education/examina/Rapporter/MarcusJonsson.pdf
[21] Ballard, D.H., Brown, C.M.: Confocal Volume Rendering: Fast Segmentation-Free Visualization of Internal Structures, Proceedings of SPIE Medical Imaging 2000 --- Image Display and Visualization, SPIE Vol. 3976, San Diego, California, Feb. 12-17, pp. 70-76, (2000)
[22] Chan, M-Y., Wu, Y., Mak, W-H., Chen, W., Qu, H.: Perception-Based Transparency Optimization for Direct Volume Rendering, IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2009), vol. 15, no. 6, Nov.-Dec. (2009)
[23] Pommert, A., Tiede, U., Wiebecke, G., Hohne, K.H.: Surface Shading in Tomographic Volume Visualization. In Proceedings of the First Conference on Visualization in Biomedical Computing, volume 1, pages 19 (1990)

Appendix: 3D discrete gradient operators

Intermediate difference operator:

$$\nabla f(x_i, y_j, z_k) = \begin{pmatrix} f(x_i+1, y_j, z_k) - f(x_i, y_j, z_k), \\ f(x_i, y_j+1, z_k) - f(x_i, y_j, z_k), \\ f(x_i, y_j, z_k+1) - f(x_i, y_j, z_k) \end{pmatrix} \tag{8}$$

Central difference operator:

$$\nabla f(x_i, y_j, z_k) = \frac{1}{2}\begin{pmatrix} f(x_i+1, y_j, z_k) - f(x_i-1, y_j, z_k), \\ f(x_i, y_j+1, z_k) - f(x_i, y_j-1, z_k), \\ f(x_i, y_j, z_k+1) - f(x_i, y_j, z_k-1) \end{pmatrix} \tag{9}$$

The kernel of Neumann gradient operator:

$$\nabla_x := \frac{1}{52}\left(\begin{pmatrix} -2 & 0 & 2 \\ -3 & 0 & 3 \\ -2 & 0 & 2 \end{pmatrix} \begin{pmatrix} -3 & 0 & 3 \\ -6 & 0 & 6 \\ -3 & 0 & 3 \end{pmatrix} \begin{pmatrix} -2 & 0 & 2 \\ -3 & 0 & 3 \\ -2 & 0 & 2 \end{pmatrix}\right) \tag{10}$$

$$\nabla_y := \frac{1}{52}\left(\begin{pmatrix} 2 & 3 & 2 \\ 0 & 0 & 0 \\ -2 & -3 & -2 \end{pmatrix} \begin{pmatrix} 3 & 6 & 3 \\ 0 & 0 & 0 \\ -3 & -6 & -3 \end{pmatrix} \begin{pmatrix} 2 & 3 & 2 \\ 0 & 0 & 0 \\ -2 & -3 & -2 \end{pmatrix}\right)$$

$$\nabla_z := \frac{1}{52}\left(\begin{pmatrix} -2 & -3 & -2 \\ -3 & -6 & -3 \\ -2 & -3 & -2 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2 & 3 & 2 \\ 3 & 6 & 3 \\ 2 & 3 & 2 \end{pmatrix}\right)$$

The kernel of Zucker-Hummel operator:

$$\nabla_x := \left(\begin{pmatrix} -\frac{\sqrt{3}}{3} & 0 & \frac{\sqrt{3}}{3} \\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{3}}{3} & 0 & \frac{\sqrt{3}}{3} \end{pmatrix} \begin{pmatrix} -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ -1 & 0 & 1 \\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{pmatrix} \begin{pmatrix} -\frac{\sqrt{3}}{3} & 0 & \frac{\sqrt{3}}{3} \\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{3}}{3} & 0 & \frac{\sqrt{3}}{3} \end{pmatrix}\right) \tag{11}$$

$$\nabla_y := \left(\begin{pmatrix} \frac{\sqrt{3}}{3} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \\ 0 & 0 & 0 \\ -\frac{\sqrt{3}}{3} & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{3}}{3} \end{pmatrix} \begin{pmatrix} \frac{\sqrt{2}}{2} & 1 & \frac{\sqrt{2}}{2} \\ 0 & 0 & 0 \\ -\frac{\sqrt{2}}{2} & -1 & -\frac{\sqrt{2}}{2} \end{pmatrix} \begin{pmatrix} \frac{\sqrt{3}}{3} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \\ 0 & 0 & 0 \\ -\frac{\sqrt{3}}{3} & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{3}}{3} \end{pmatrix}\right)$$

$$\nabla_z := \left(\begin{pmatrix} -\frac{\sqrt{3}}{\sqrt{3}} & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{3}}{\sqrt{3}} \\ -\frac{\sqrt{2}}{2} & -1 & -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{3}}{\sqrt{3}} & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{3}}{\sqrt{3}} \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\sqrt{3}}{\sqrt{3}} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{\sqrt{3}} \\ \frac{\sqrt{2}}{2} & 1 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{3}}{\sqrt{3}} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{\sqrt{3}} \end{pmatrix}\right)$$

The kernel of Sobel operator:

$$\nabla_x := \left(\begin{pmatrix} -1 & 0 & 1 \\ -3 & 0 & 3 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} -3 & 0 & 3 \\ -6 & 0 & 6 \\ -3 & 0 & 3 \end{pmatrix} \begin{pmatrix} -1 & 0 & 1 \\ -3 & 0 & 3 \\ -1 & 0 & 1 \end{pmatrix}\right) \tag{12}$$

$$\nabla_y := \left(\begin{pmatrix} 1 & 3 & 1 \\ 0 & 0 & 0 \\ -1 & -3 & -1 \end{pmatrix} \begin{pmatrix} 3 & 6 & 3 \\ 0 & 0 & 0 \\ -3 & -6 & -3 \end{pmatrix} \begin{pmatrix} 1 & 3 & 1 \\ 0 & 0 & 0 \\ -1 & -3 & -1 \end{pmatrix}\right)$$

$$\nabla_z := \left(\begin{pmatrix} -1 & -3 & -1 \\ -3 & -6 & -3 \\ -1 & -3 & -1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 3 & 1 \\ 3 & 6 & 3 \\ 1 & 3 & 1 \end{pmatrix}\right)$$