An Algorithm for Visualization of Patient-specific CT-based Vascular Data for the Model of 1D Hemodynamics

V. Karlov, S. Simakov

Sechenov University, Moscow, Russia
e-mail: simakovss@ya.ru

Abstract. This paper accounts for the problem of the postprocess data visualisation coming from the patient-specific CT-based 1D blood flow modelling. Two approaches and their modifications are studied and analysed. Finally, the own method of interactive data representation is developed.

1 Introduction

Nowadays, cardiovascular diseases are a severe problem of all humankind. In many cases, such diseases develop for years without symptoms with unexpected acute termination. Recent achievements of mathematics and mathematical modelling allow advanced diagnosis and treatment of such conditions [1-4]. With the help of numerical calculations and simulations, it is possible to predict the practical outcome of surgical operations on the vessels, to optimise the shape and position of endovascular implants, to study their effect on hemodynamics, to study the impact of graduated stocking compression of veins [1, 3]. The most advanced mathematical models use patient-specific data, which clinicians measure in regular practice [1,2,4,5]. Such data include CT-scans, ultrasound velocity measurements, heart rate, systolic and diastolic arterial pressure, the velocity of the pulse wave propagation, etc. In many practical cases, this information is sufficient for the quantitative prognosis of patient conditions basing on the computational experiments without real surgical procedure and other types of medical treatment, such as noninvasive assessment of fractional flow reserve of coronary flow [1, 2]. In this connection, especially useful, are the reduced order models [6-8].

Along with sophisticated mathematical models, a task of the data visualisation is very difficult important. The vascular network of a human has a complex 3D structure with a significant number of elements. Several algorithms of patient-specific CT data segmentation has been proposed. They successfully produce a 3D structure of the local vascular region. Algorithms dealing with centerline extraction and 1D core graph generation for reduced order modelling are less common [9].

This article concerns the problem of algorithm development, which provides a user-friendly, interactive, cross-platform tool for the interaction with computed data, which are mapped to the virtual geometric model of the individual vascular structure. This tool is intended both for the experienced IT users with excellent programming skills, who perform the simulations, and for the medical doctors, who need fast, and

In: P. Sosnin, V. Maklaev, E. Sosnina (eds.): Proceedings of the IS-2019 Conference, Ulyanovsk, Russia, 24-27 September 2019, published at http://ceur-ws.org

reliable processing of the data from the patient. The proposed algorithm is an extension of CT data processing. As an input, it takes the centerlines and average diameters of the vascular segments, which were generated by the other algorithm [9,10]. It produces a mapping and visualises the data from a computational module of 1D hemodynamics. Thus, it is a postprocessing algorithm of the general workflow, which provides flexible frontend with possible access thorugh the Internet or local network.

2 Related works

The generation of individualised geometrical domains based on medical imaging is a well-known task. The assumed 1D hemodynamic model [6,8] uses a 1D network of a patient□specific vascular region with a large number of vessels. The 1D network is the graph with nodes in the 3D space, which keeps geometric data such as length and average diameter of the vessel segments as parameters. 1D network generation algorithm, includes reconstruction, correction, and local adaptation with two modes of centerline representation: sets of connected voxels and parametric curves with assigned parametric diameters. The efficiency of this algorithm was tested on several examples. Also, it can be applied to other tubular structures, such as a network of lymphatic vessels or trachea-bronchial tree [11,12].

Several algorithms of skeletonisation and centerline extraction for tubular structures have been proposed. Voronoi diagram methods find centerlines as the paths in Voronoi diagrams that minimise the integral of the radii of the maximal inscribed spheres along the pathway [13]. Distance mapping methods are generally used to construct the shortest path between two points. It generates the distance from the source map (DFSM), which is the distance from the source point to each voxel inside the 3D object. The shortest path from the endpoint to the source point is found by descending through the gradient of the DFSM. The centredness problem is solved by adding a penalty to the distance cost at each node to keep the shortest paths away from the boundary [14].

The centerlines also accurately produced by level set methods [5]. Skeletons of 3D regions can be generated by voxel□based algorithms [16]. A comparison of different skeletonisation methods can be found in [17]. It also exists a method of direct centerline extraction without segmentation [18]. The topological thinning process removes voxels on the boundary of the shape with preserving connectivity and topology [19].

Review and more details on these algorithms can be found in [9,17], which also proposes fully automatic coronary artery segmentation, automated reconstruction of 1D network using the skeleton extraction from a segmented image, automatic 1D network extraction from a set of centerlines with a common root, two 1D network graph postprocessing algorithms for graph correction, and local adaptation.

Our algorithm takes the output of some of these algorithms for backward reconstruction of the 3D tubular domain with assigned data from the computational module of 1D haemodynamics and interactive features allowing a visual analysis of the flow in patient vascular network in different virtually generated situations.

Currently, this unusual but essential task for successful development of monolithic technology covering all stages from patient data collecting to final personalised analysis and recommendations. This work is focused on two approaches. The first approach is based on the PyMesh library developed by Qingnan Zhou as part of his research at New York University [20, 21]. The method uses client-server architecture involving the generation of a 3D model on the server and its subsequent transfer to the client's side. The second approach uses Babylon library.js, which contains a large number of possibilities for generating and displaying 2D and 3D objects [22-24]. The advantage of this approach is fully client-oriented algorithms for constructing 3D models without a server. Further, in this work, we will give details on both of these approaches with the analysis of the advantages and disadvantages relative to our specific task.

3 PyMesh library approach

The first approach is the use of implementation-based PyMesh library [20, 21]. Zindani Zhou developed this library in the framework of his research at New York University. PyMesh is a rapid prototyping platform for processing geometrical data. PyMesh is developed using both C++ and Python. C++ is used for the complex parts of the code. Python is used as a high-level wrapper, which was designed to provide a minimalistic and straightforward interface. One of the purposes of this library is to generate meshes for elementary shapes (primitives), such as a cube, sphere and cylinder. A mesh is a collection of vertices, edges, and faces, which defines the shape of a multi-faceted object in 3D graphics and volume modelling. Faces are typically triangles, quadrilaterals or other convex polygons. The meshes may also include nonconvex polygons and polygons with the holes. The interface allows specifying the number of vertices in the generated objects, thereby enabling variation of quality of the 3D image and required memory. The consumed memory is especially important for network applications. Figure 1 shows the results of the sphere rendering with a different number of polygons. The first (left) case uses 2 KB memory. The second refined case uses 52 KB.

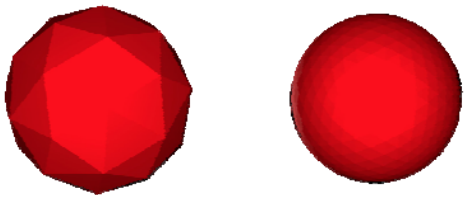

Figure 1. Comparison of memory cost to the quality of rendering.

A script generates the 3D tubular structure. First, we divide centerlines into segments and substitute them with the cylinders (see Figure 2, top). At the joints of the cylinders, the spheres were added. The radius of each sphere is calculated as the maximum radius of the adjacent cylinders (see Figure 2, bottom). It makes this approach difficult for the applications, which deals with a wide tubular network

rendering. It requires the construction of a lot of primitives for every tubular segment, which results in increased memory consumption, which is critical for the network-based applications.

This approach can be improved by a Display Model Concept (DMC). This concept combines PyMesh as a mechanism for generating a 3D model and some other stack for the rendering on the terminal device. We assume the target platform as a web browser. The suitable library for final rendering is a Three.js [25]. Three.js is a lightweight cross-browser JavaScript library for creating and displaying animated 3D computer graphics in web applications. Thanks to the WebGL technology [24] Three.js allows developing GPU-accelerated 3D graphics using JavaScript as a part of the site without proprietary plugins.

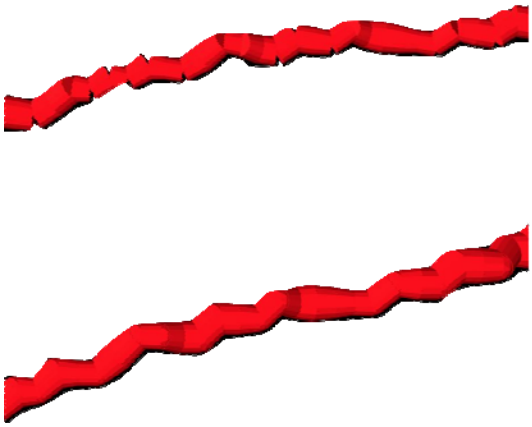

Figure 2. Primitive-based rendering of the 3D tubular structure with PyMesh.

Three.js allows loading generated meshes and performing various transformations on them, including change of the colour, texture, size, etc. The DMC with Three.js requires splitting 3D model into parts. After that, every component can be transformed separately, which enables the interactive behaviour of the 3D model. From one hand, this splitting is a natural consequence of using PyMesh. From the other hand, it decreases the performance of the model. Figure 3 shows the result of model construction for the two coronary networks [1,2] using DMC implementation. The surface is sufficiently rugged due to the presence of a large number of cylindrical and spherical primitives, especially in the high-quality rendering. The memory, which is used by these models, is more than 50Mb, which is unacceptable.

We conclude, that, since the PyMesh library is a python language library and contains source code written in C++ it is challenging to use it as a cross-platform and/or browser-based solution. The only available solution is to delegate the generation of 3D models to a remote server. This solution has several side effects:

- the need for a permanent Internet connection from the user,
- the need to maintain the server infrastructure,
- related difficulties in scaling the service,
- the complexity of the service architecture, which often prevents further support and development,

- the need to maintain the user authorisation, and the security of the system.

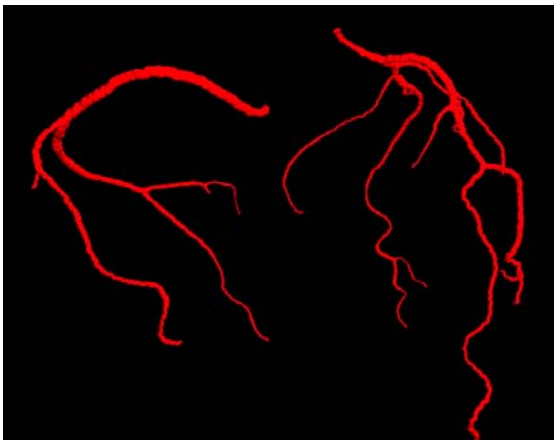

Figure 3. Coronary network generated by implementation of Display Model Concept.

When using PyMesh, difficulties may arise at the time of the library installation. The easiest way is to install the docker image, although this solution allows using the library in sandbox mode, it is not suitable for full deployment on the server. The only possible option is to build the entire library using the CMake utility. C++ is not a cross-platform language, so if after changing the server infrastructure, the build process should be repeated. This method, although acceptable, in practice causes a lot of difficulties and problems, and also makes it impossible to maintain the versions of all necessary libraries up to date.

4 Babylon library approach

The use of PyMesh based approach gave us an understanding of the significant problems and bottlenecks, which have to be solved for developing a quality and convenient service. The further work was concentrated on

- increasing the smoothness of the model with the memory limitations,
- improving the service/model performance by reducing the limit of consumed resources and transferring the mesh generation to the client side within a thick-client architecture.

In this connection, the Babylon library [22-24] was examined. Babylon.js is a 3D real-time engine that uses JavaScript library to display 3D graphics in a web browser using HTML5 technology. The source code is available on GitHub and is licensed under the Apache License 2.0. David Katuhe and David Russe with the support of artist Michel Rousseau developed and released this library in 2013 as an engine for 3D games. The source code is written in TypeScript and then compiled into a JavaScript version. The JavaScript version is available to end users via NPM or CDN, who then encode their projects in JavaScript, accessing the engine API. 3D engine and custom Babylon code are interpreted in its way by all web browsers, which

support HTML5 and WebGL. The library interface makes it possible to generate and display objects with the help of a browser. Thus, this approach allows for avoiding problems related to the support of the remote server infrastructure.

The primitives of the Babylon library are custom defined shell models. Polygons perform the 3D modelling of shell models with triangular faces, which are combined to a predefined set (shell model). The library allows the use of structural block geometry methods to construct the union, difference, and the intersection of shell models. Once the objects are built, they are displayed on the HTML5 canvas element using a Shader program that determines the position of pixels and colours on the canvas with polygon models, textures applied to each model, scene camera, light sources, and 4x4 world matrices for each object that stores their position, rotation angle, and scale.

The Structurally Solid Geometry (SSG) allows decreasing the memory and performance requirements. SSG is a technology, which used in solid modelling. This technology allows creating complex surfaces or objects from basic primitives (cube, cylinder, prism, pyramid, sphere, cone) using logical operators (logical union, logical subtraction, and logical intersection). Thus, one may develop models of nontrivial objects without storing vertices and edges, which will never be visible.

We use Bezier curves to smooth the image of the graph [26]. The drawback of this approach is the loss of information about the centerlines. But it allows producing a more visually attractive and user-friendly rendering. The useful properties of the Bezier parametrisation are:

- the segment is filled continuously from start to endpoint,
- the curve does not go beyond the shape specified by the lines connecting the terminal points,
- the Bezier curve is symmetric, that is, the order of the points forming it does not affect the shape of the curve,
- scaling and changing the proportions of the curve does not break its, stability, as it is an affine invariant.

A Bezier curve is a parametric curve, which cand be defined as

$$B(t) = \sum P_k F_{k,n}(t), 0 \le t \le 1,$$

where P is the vector of the support vertices and F is the basis functions of the Bezier curve, also called Bernstein polynomial

$$F_{k,n}(t) = \frac{n!}{k!(n-k)!} t^k (1-t)^{n-k},$$

where n is the order of the curve, k is the order of the vertex. 3D curves allow construction of a model of the vessels. The following equation gives the Cubic Bezier curve ($n = 3$)

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2 (1-t) P_2 + t^3 P_3.$$

This method of interpolation is used in the final version of our application. Figures 4 and 5 shows the result of the developed algorithm for the coronary [1,2] and cerebral [4,5] vessels, which are initially based on the patient-specific CT data. One may see smooth visual representation, which is also stable for interactive

transformations (rotations, scaling, etc) and allows manual selection of the small element of the model.

The developed algorithm of visualization of the system of centerlines should be extended with some additional functionality. It includes

- uploading and downloading files with computational data,
- parsing input data and convert tham to a suitable format for displaying,
- implementing interface of chart drawing for the data, which are associated with the selected region of the network.

We use the Vue framework [27] for the implementation of all these functions. In contrast to monolithic frameworks, Vue is intended for gradual development and implementation of web projects. This framework primarily solves the problems of the application view layer and allows reusing other libraries and existing projects. One of the main features of this framework is the component approach. Components help extend basic HTML elements and connect reusable code.

During the design phase, we break our application into independent parts and obtain a tree structure of the components. Having described the application in the architecture of this framework, we get a set of separate modules that can be worked with as a constructor. Each module remains independent and could be changed without effect to the other modules.

Figure 4. Model of the coronary arteries produced by using Babylon library.

The final pipeline of our solution is as follows

- user uploads data files to the service,
- service parses files and converts them into the internal format,
- Bezier curves are generated based on the centre lines,
- the obtained Bezier curves are used as a skeleton for constructing meshes,
- event listeners are added to the meshes,
- a camera, light sources, built meshes are added to the empty scene,

- listeners of keystrokes are bounded to the meshes,
- the graphics display module is activated.

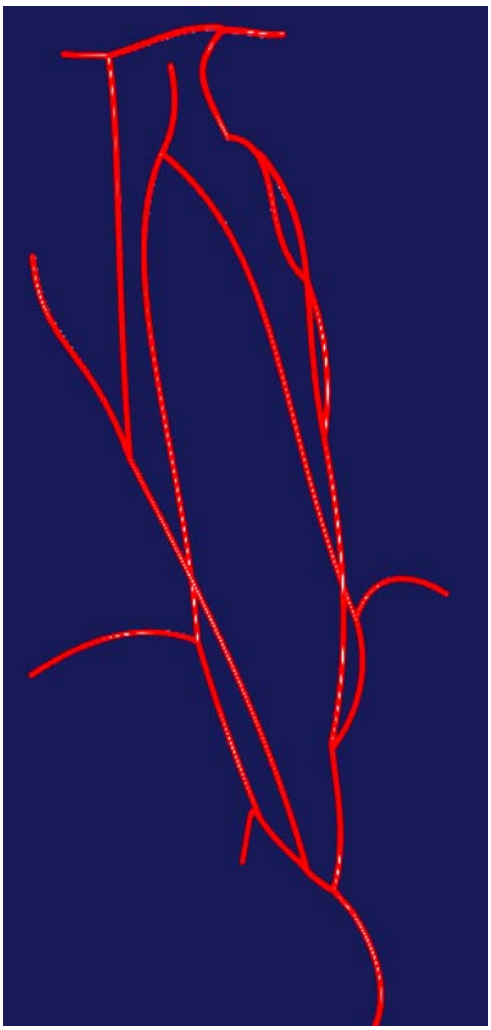

Figure 5. Scheme of the cerebral arteries produced by using Babylon library.

It should be mentioned, that the data parser does not contain complex logic. Its only task is to convert the input data into a format suitable for other modules. The graphs are implemented in such a way that it is possible to fix the constructed curves and compare them with the values at other points. The state machine uses a queue structure so one may reuse the free colours in the legend of the graphs.

5 Conclusion

This paper presents the solution, which allows attractive visualisation of the data computed using a 1D blood flow model. Review of existing algorithms and analysis of their advantages and disadvantages allows developing a cross-platform web service with acceptable network, memory and CPU performance. The performance and cross-platform nature make the solution suitable to most modern personal computers without any restrictions. Modular architecture enables to update individual components independently.

6 Acknowledgements

This work was supported by the Russian Foundation for Basic Research, Grants No 18-00-01524, 18-31-20048.

References:

1. Gognieva, D.G., Syrkin, A.L., Vassilevski, Yu.V.: Noninvasive Assessment of Fractional Flow Reserve Using Mathematical Modeling of Coronary Flow. Kardiologiya, Vol. 58, #12, pp. 8-92 (2018)

2. Ge, X., Liang, F., Vassilevski, Y., et.al.: Sensitivity of Coronary Flow Reserve to Cardiovascular Parameters: A Computational Model-Based Study. IECBES 2018 — Proceedings, pp. 32-35 (2019)

3. Wang, T., Liang, F., Liu, R., et.al.: Model-based Study on the Hemodynamic Effects of Graduated Compression Stockings in Supine and Standing Positions. 2018 IEEE EMBS Conference on Biomedical Engineering and Sciences, IECBES 2018 — Proceedings, pp. 27-31 (2019)

4. Kopylov, F.Yu., Bykova, A.A., Shchekochikhin, D.Yu., et.al.: Asymptomatic Atherosclerosis of the Brachiocephalic Arteries: Current Approaches to Diagnosis and Treatment. Terapevticheskii Arkhiv, Vol.89, #4, pp.95-100 (2017)

5. Simakov, S., Gamilov, T.: Computational Study of the Cerebral Circulation Accounting for the Patient-Specific Anatomical Features. Smart Innovation, Systems and Technologies, Vol.133, pp. 309-330 (2019)

6. Simakov, S.S.: Modern Methods of Mathematical Modeling of Blood Flow Using Reduced Order Methods. Computer Research and Modeling, Vol. 10, #5, pp. 581-604 (2018)

7. Petrov, I.B., Favorskaya, A.V., Favorskaya, M.N., et.al.: Development and Applications of Computational Methods. Smart Innovation, Systems and Technologies, Vol.133, pp. 3-7 (2019)

8. Bessonov, N., Sequeira, A., Simakov, S., et.al.: Methods of Blood Flow Modelling. Mathematical Modelling of Natural Phenomena, Vol.11, #1, pp. 1-25 (2016)

9. Danilov, A., Ivanov, Y., Pryamonosov, R., Vassilevski, Y.: Methods of Graph Network Reconstruction in Personalized Medicine. International Journal of Numerical Methods in Biomedical Engineering. Vol.32, #8, pp.e02754:1-20 (2016)

10. Pryamonosov, R., Danilov, A.: Robustness Analysis of Coronary Arteries Segmentation. Smart Innovation, Systems and Technologies, Vol.133, pp. 331-344 (2019)

11. Golov, A., Simakov, S., Soe, Y.N., et.al.: Multiscale CT-Based Computational Modeling of Alveolar Gas Exchange during Artificial Lung Ventilation, Cluster (Biot) and Periodic (Cheyne-Stokes) Breathings and Bronchial Asthma Attack. Computation, Vol. 5, #1, pp.11:1-18 (2017)

12. Golov, A.V., Simakov, S.S.: Mathematical Model of Respiratory Regulation During Hypoxia and Hypercapnia. Computer Research and Modeling, Vol.9, #2, pp. 297-310 (2017)

13. VMTK – the vascular modeling toolkit. (Available from: http://www.vmtk.org/) (2015)

14. Wan, M., Liang. Z., Ke, Q., et. al.: Automatic Centerline Extraction for Virtual Colonoscopy. IEEE Transactions on Medical Imaging, Vol. 21, pp. 1450-1460 (2002)

15. Hassouna, M., Farag, A.: Robust Centerline Extraction Framework Using Level Sets. Computer Vision and Pattern Recognition, IEEE Computer Society Conference, Vol. 1, pp. 458-465 (2005)

16. Reniers, D., van Wijk, J., Telea, A.: Computing Multiscale Curve and Surface Skeletons of Genus 0 Shapes Using a Global Importance Measure. IEEE Transactions on Visualization and Computer Graphics, Vol. 14, pp. 355-368 (2008)

17. Sobiecki, A., Yasan, H., Jalba, A., Telea, A.: Qualitative Comparison of Contraction-Based Curve Skeletonization Methods. In Mathematical Morphology and Its Applications to Signal and Image Processing, Hendriks C, BorgeforsG, Strand R (eds)., Lecture Notes in Computer Science. Springer: Berlin Heidelberg, Vol. 7883, pp. 425-439 (2013)

18. Gülsün, M., Tek, H.: Robust Vessel Tree Modeling in Medical Image Computing and Computer-Assisted Intervention. Lecture Notes in Computer Science, Metaxas, D., Axel, L., Fichtinger, G., Székely, G. (eds)., Vol. 5241, Springer: Berlin Heidelberg, pp. 602-611 (2008).

19. Pudney C.: Distance-Ordered Homotopic Thinning: a Skeletonization Algorithm for 3D Digital Images. Computer Vision and Image Understanding, Vol. 72, pp. 404-413 (1998)

20. Zhou, Q., Grinspun, E., Zorin, D., et.al.: Mesh Arrangements for Solid Geometry. ACM Siggraph, (2016).

21. URL https://github.com/PyMesh/PyMesh

22. Moreau-Mathis, J.: Babylon.Js Essentials. Packt Publishing (2016)

23. URL https://github.com/BabylonJS/Babylon.js

24. Catuhe, D.: WebGL Engine Design in Babylon.js. WebGL Insights, eds. Cozzi, P. CRC Press, (2015).

25. URL https://github.com/mrdoob/three.js

26. Rogers, D., Adams, J.: Mathematical fundamentals of the computer graphics. M.:Mir, (2001)

27. Macrae, C.: Vue.js: Up and Running. O'Reilly (2017)