1 Web-Based 3D Visualisation of Biological and Medical Data

Ciril Bohak, Žiga Lesar, Primož Lavric, and Matija Marolt

Abstract

In this chapter we present an overview of web-based frameworks for visualisation of medical and biological data, with emphasis on visualisation of volumetric data such as radiological data (e.g. magnetic resonance imaging, computed tomography or positron emission tomography) and microscopy data (e.g. focused ion beam scanning electron microscopy). We compare web-based frameworks with state-of-the-art standalone visualisation tools and point out the advantages and disadvantages of both. We also present our open-source web-based visualisation environment Med3D.

Keywords

Volumetric data · Collaborative visualisation · Web-based visualisation · WebGL 2.0 · Volumetric rendering

C. Bohak (✉) · Ž. Lesar · M. Marolt
Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia
e-mail: ciril.bohak@fri.uni-lj.si; ziga.lesar@fri.uni-lj.si; matija.marolt@fri.uni-lj.si

P. Lavric
Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia

Cosylab d.d., Ljubljana, Slovenia
e-mail: primoz.lavric@lgm.fri.uni-lj.si

1.1 Introduction

In recent years, the advances in computer graphics hardware and software, as well as the development of novel visualisation techniques, have allowed the implementation of real-time visualisation systems for 3D medical radiological data obtained with magnetic resonance imaging – MRI (Rinck et al. 1990), computed tomography – CT (Crawford and King 1990; Kalender et al. 1990), positron emission tomography – PET (Ollinger and Fessler 1997) and 3D ultrasound (Krakow et al. 2003). It is also possible to visualise the biological 3D microscopy data obtained with electronic microscopy such as focused ion beam scanning electron microscopy – FIB-SEM (Briggman and Bock 2012). This is possible due to the larger system memory size which allows storing large amounts of data obtained with the above-mentioned techniques, due to the increased computational performance of graphical processing units – GPUs and due to the improved software systems which enable the development of highly optimised visualisation software products.

Such data is typically represented in the form of 3D scalar or vector fields where the values represent different properties of the scanned object. Depending on the scanning technique the values could represent various tissues inside the human body or inside the biological sample. In

P. M. Rea (ed.), *Biomedical Visualisation* , Advances in Experimental Medicine and Biology 1235,
https://doi.org/10.1007/978-3-030-37639-0_1

some cases the desired tissues can be additionally expressed using a contrast agent.

The first attempt of visualising volumetric data was developed in the late 1980s by Levoy (1988). The paper presents a direct surface rendering technique for volumetric data, which is briefly presented in Sect. 1.2.3.2. An in-depth review of volumetric rendering related work from the beginning until 2007 is presented in Šrámek (2006). Most notable techniques of that time were:

- **Volume Ray Casting** (Levoy 1988; Ke and Chang 1993), which is derived directly from the rendering equation (Kajiya 1986);
- **Splatting** (Westover 1991), a technique where all the elements of the volume are splatted on the canvas in back to front order as disks with diametrically varied properties (e.g. colour and/or transparency) according to normal distribution;
- **Shear Warp** (Cameron and Undrill 1992; Lacroute and Levoy 1994), a technique which decomposes the viewing transformation into a 3D shear component parallel to the data slices, projects the data to form an intermediate distorted image and finally uses a 2D warp to create an undistorted final image;
- **Texture-Based Rendering** (Hibbard and Santek 1989), a technique which exploits the functionality of dedicated hardware (later GPUs) for mapping the textures (in this case slices of volumetric data) onto parallel planes.

In the following years many improvements of the above techniques were presented and the methods were integrated into the end-user applications presented in the following subsections. Further development of some methods have also resulted in special purpose hardware implementations for achieving real-time performance. Such example is a family of Cube products (Bakalash et al. 1992; Pfister et al. 1994; Kanus et al. 1997) designed for real-time volume data visualisation which in the latest edition enabled users to interactively visualise volumes of sizes up to 128^3. The development of custom hardware has ended when support for general purpose computing (GPGPU) was brought to the GPUs in 2001. Afterwards, the visualisation systems were implemented for the GPU hardware using OpenCL[1] or CUDA[2] APIs for computation and OpenGL[3] or DirectX[4] for visualisation purposes.

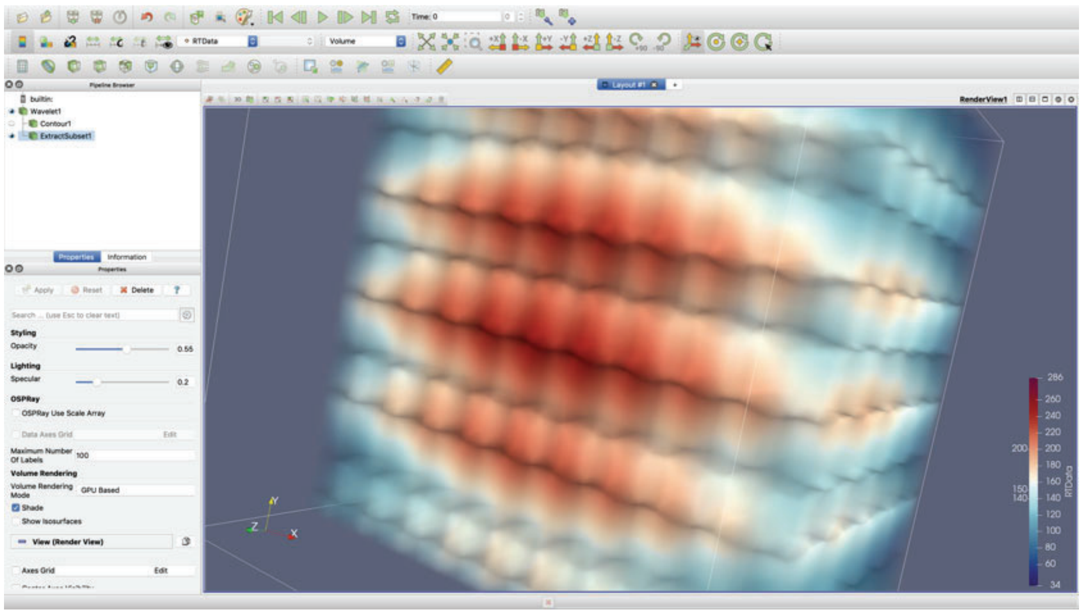
1.1.1 Stand-Alone Visualisation Applications

With development of GPGPUs the implementation of newly developed visualisation methods into an end-user application became more viable and, due to broad access and affordability of the hardware, visualisation applications could be used on high-end desktop computers without the need for dedicated hardware. The selection of such software products, which are still being developed and maintained, is presented below.

1.1.1.1 VTK

The Visualisation Toolkit[5] – VTK (Schroeder et al. 2000) is a collection of software tools that allows development of customised visualisation applications on top of the implemented visualisation pipeline. While the toolkit is not intended for end-use it simplifies the development of end-user application, such as ParaView and 3D Slicer presented in the following subsections. It is one of the most wide-spread toolkits used in numerous commercial applications, since it is independent of the operating system. Unfortunately, it is based on an outdated version of the OpenGL standard (version 2.1), which means that many features cannot be optimised and the interactive rendering methods cannot be significantly improved performance-wise. Moreover, the toolkit does not support any physically-based volumetric rendering and thus does not offer state-of-the-art volumetric rendering capabilities.

[1] https://www.khronos.org/opencl/
[2] https://www.nvidia.com/cuda/
[3] https://www.opengl.org/
[4] http://msdn.microsoft.com/directx
[5] https://vtk.org/



Styling
Opacity 0.55
Lighting
Specular 0.2
OSPRay
OSPRay Use Scale Array
Data Axes Grid Edit
Maximum Number Of Labels 100
Volume Rendering
Volume Rendering Mode GPU Based
Shade ✓
Show Isosurfaces
View (Render View)
Axes Grid Edit

Fig. 1.1 ParaView application window displaying sample volumetric data

1.1.1.2 ParaView

Built on top of VTK, ParaView[6] (Ahrens et al. 2005) is a general data analysis and visualisation application used in a variety of research fields such as engineering, geology, climate science, astrophysics, etc. The application is open-source and runs on all major platforms. It is presented in Fig. 1.1 for a sample dataset. Its main purpose is to support large-scale datasets (up to terascale) and to exploit the power of distributed computing. It is designed as an application framework as well as an end-user application, and can be modified by developers for specific use cases. While the application can be used for visualisation of data from the medical and biological domains, it does not offer state-of-the-art volumetric rendering capabilities since it is built on top of VTK.

1.1.1.3 Voreen and Inviwo

Voreen[7] (Meyer-Spradow et al. 2009) is an open source application development framework for visualisation of multi-modal volumetric datasets. It supports GPU accelerated volume rendering and data analysis and allows users high flexibility with the development of custom data analysis and visualisation workflows. The development of the framework split into two branches: Voreen and Inviwo[8] (Jönsson et al. 2018). While both current versions offer better volumetric rendering capabilities implementing ray casting with global illumination, it is not defined how global illumination is implemented and none of the application is using volumetric path tracing or equivalent state-of-the-art volumetric rendering techniques. A sample data visualisations using both applications are presented in Fig. 1.2.

1.1.1.4 3D Slicer

3D Slicer[9] (Fedorov et al. 2012) is a visualisation and analysis software platform for medical image informatics, image processing and 3D visualisation. It is an open source cross-platform tool for physicians, and other researchers connected to biological and medical domains. The tool supports plugin development in Python using the provided API. It is build on top of VTK which is also the reason why it does not offer any state-of-the-art volumetric rendering capabilities. It is

[6]https://www.paraview.org/
[7]https://www.uni-muenster.de/Voreen
[8]http://www.inviwo.org
[9]https://www.slicer.org/

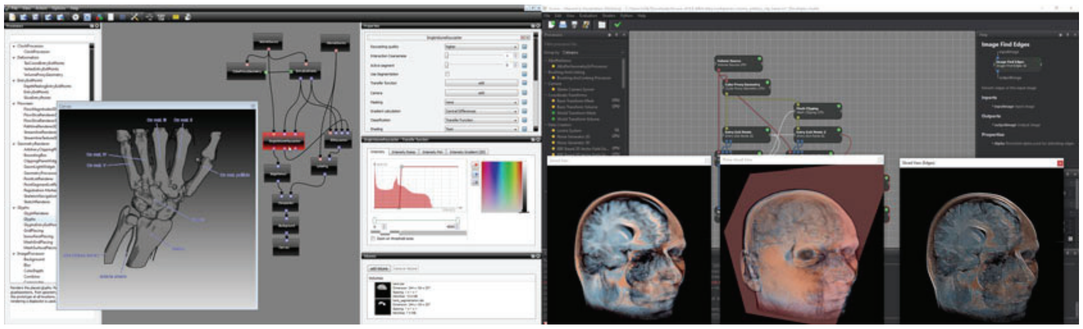


Fig. 1.2 Voreen application (left) and Inviwo application (right) displaying sample volumetric data visualisation

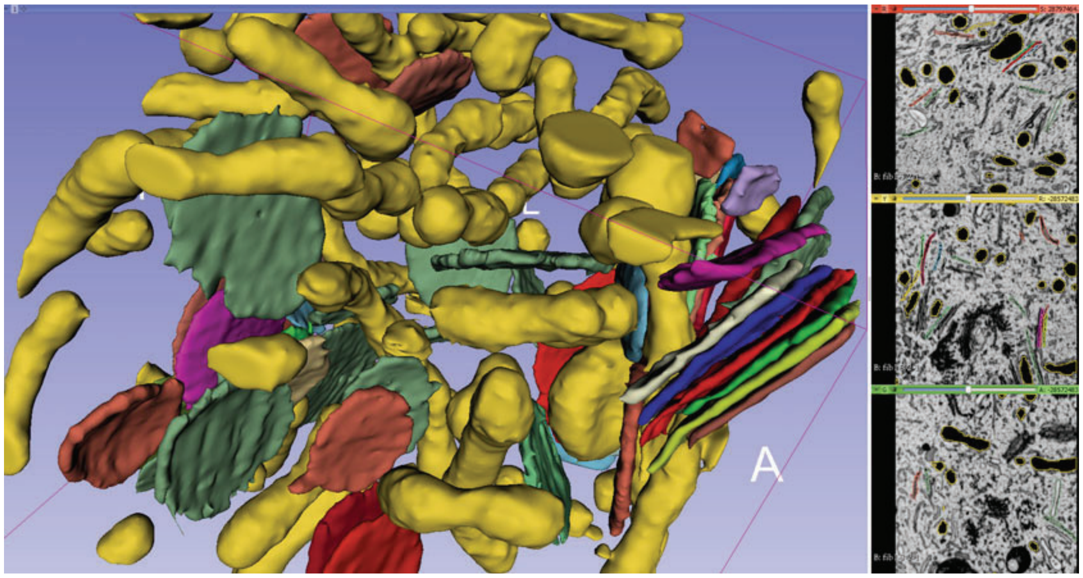


Fig. 1.3 3D Slicer application displaying segmented electronic microscopy volumetric data visualisation

broadly accepted analysis tool and has very active developer community. An example data visualisation of segmented electronic microscopy data is displayed in Fig. 1.3.

1.1.1.5 SimVascular

An example of very specialised application for simulation and visualisation of blood flow in vessels SimVascular[10] (Updegrove et al. 2017) was designed for specific purpose: providing a complete pipeline from segmented medical images to patient specific blood flow simulation, analysis and visualisation. It is build on top of VTK and has same limitations regarding the volumetric rendering, but offers good composed visualisation of the vessel data and simulation results. An example use-case is presented in Fig. 1.4.

1.1.1.6 NeckVeins

A specialised, platform-independent framework, developed in Java, is intended for fast volume-to-mesh conversion (Bohak et al. 2014) and indirect visualisation of volumetric data converted to mesh geometry for the interactive vessels visualisation. The tool supports visualisation of complex mesh geometry obtained form volume data using Marching Cubes (Lorensen and Cline 1987) or MPUI (Ohtake et al. 2003). The application with sample data is presented in Fig. 1.5.

[10]http://simvascular.github.io/

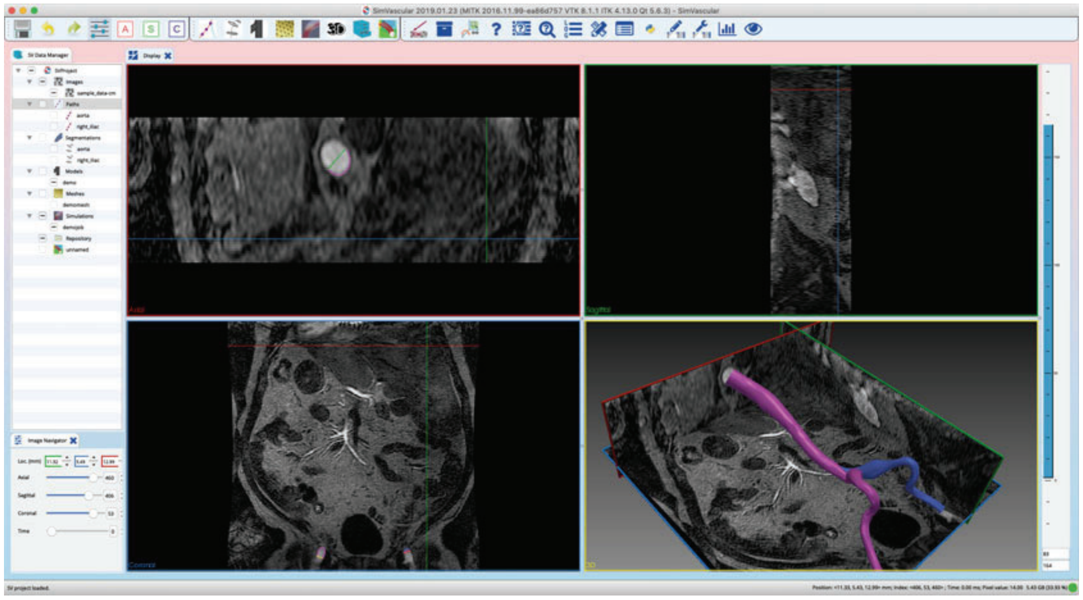


Fig. 1.4 SimVacular application displaying sample data visualisation

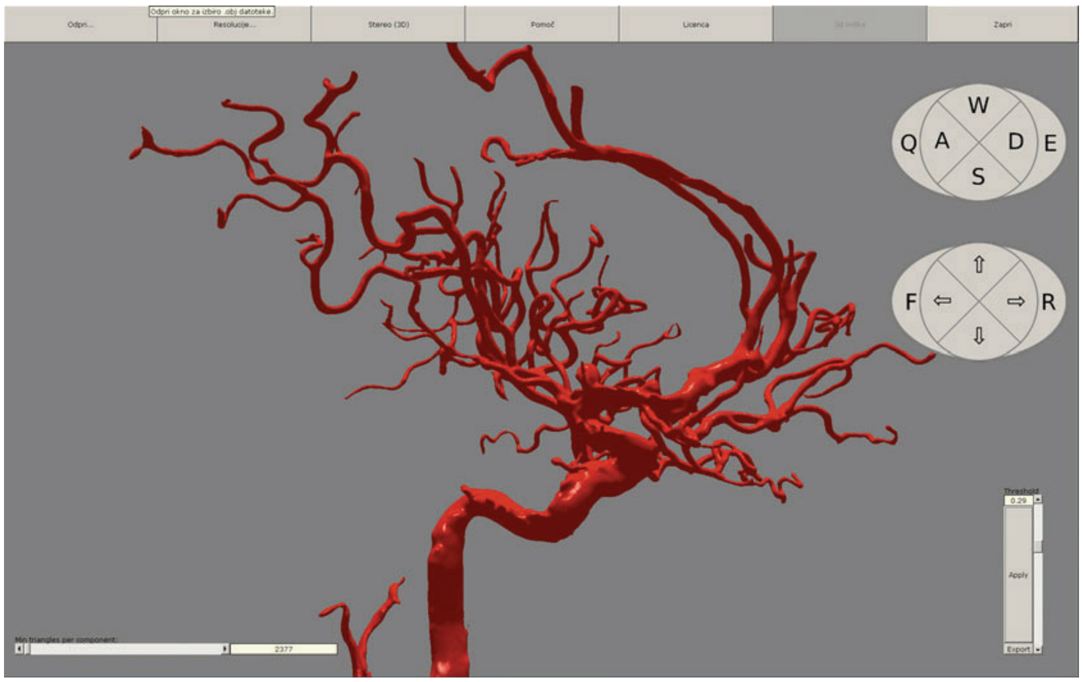
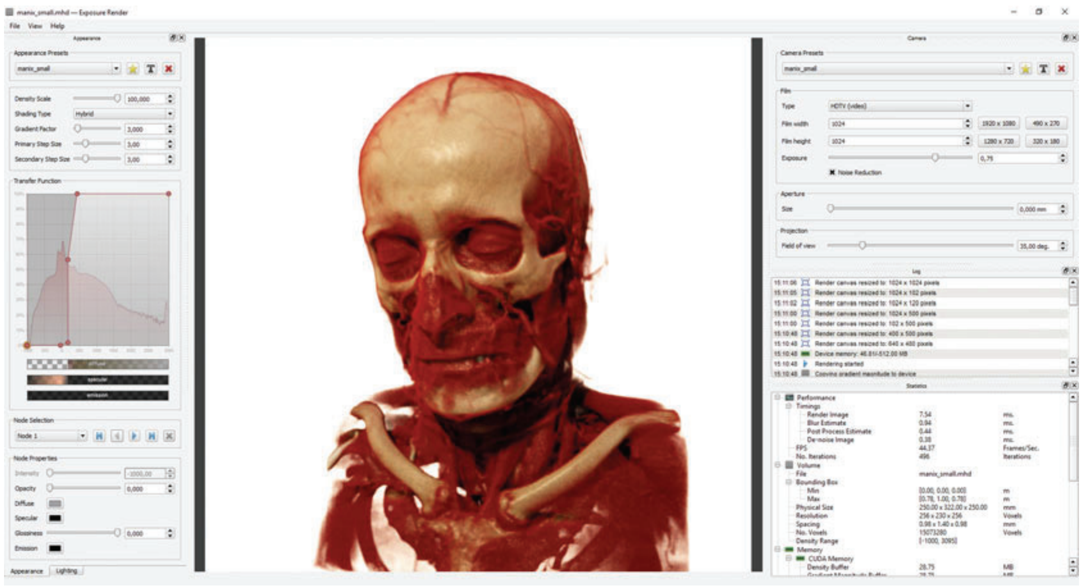


Fig. 1.5 Neckveins application displaying sample data visualisation



```
Appearance Presets: manix_small
Density Scale        100,000
Shading Type         Hybrid
Gradient Factor      3,000
Primary Step Size    3,00
Secondary Step Size  3,00
Node Selection: Node 1
Intensity   -1000,00
Opacity     0,000
Diffuse
Specular
Glossiness  0,000
Emission

Camera Presets: manix_small
Type         HDTV (video)
Film width   1024     1920 x 1080   490 x 270
Film height  1024     1280 x 720    320 x 180
Exposure     0,75
[✓ Noise Reduction]
Size           0,000 mm
Field of view  35,00 deg.
```

```
15:11:06  Render canvas resized to: 1024 x 1024 pixels
15:11:05  Render canvas resized to: 1024 x 102 pixels
15:11:02  Render canvas resized to: 1024 x 120 pixels
15:11:00  Render canvas resized to: 1024 x 500 pixels
15:11:00  Render canvas resized to: 102 x 500 pixels
15:10:48  Render canvas resized to: 400 x 500 pixels
15:10:48  Render canvas resized to: 640 x 480 pixels
15:10:48  Device memory: 46.81/-512.00 MB
15:10:48  Rendering started
15:10:48  Copying gradient magnitude to device
```

```
Performance
  Timings
    Render Image             7.54                       ms.
    Blur Estimate            0.94                       ms.
    Post Process Estimate    0.44                       ms.
    De-noise Image           0.38                       ms.
  FPS                        44.37                      Frames/Sec.
  No. Iterations             496                        Iterations
Volume
  File                       manix_small.mhd
  Bounding Box
    Min                      [0.00, 0.00, 0.00]         m
    Max                      [0.78, 1.00, 0.78]         m
  Physical Size              250.00 x 322.00 x 250.00   mm
  Resolution                 256 x 230 x 256            Voxels
  Spacing                    0.98 x 1.40 x 0.98         mm
  No. Voxels                 15073280                   Voxels
  Density Range              [-1000, 3095]
Memory
  CUDA Memory
    Density Buffer           28.75                      MB
```

Fig. 1.6 Exposure Render application displaying sample data visualisation

1.1.1.7 Exposure Render

Exposure Render[11] (Kroes et al. 2012) is the first open source physically based volumetric ray tracing renderer for medical domain which combines volumetric and surface rendering capabilities in a single solution. The application focuses on increasing the visual realism using light occlusion, depth of field and realistic lighting. This also has a positive effect on user perception. The presented work was basis for development of many commercial products used in clinical domain (e.g. Cinematic rendering (Eid et al. 2018)). An example of rendering output using Exposure Renderer is presented in Fig. 1.6. The downside is that the application is platform dependant (Microsoft Windows only) and requires Nivida CUDA support, meaning that end-users have to buy the specific hardware and software.

All the presented applications are open-sourced and available to users for free of charge. While VTK and ParaView are developed for very broad range of use-cases and domains, others are more specialised for medical use-cases. The realistic and even hyper-realistic rendering is now also being used by radiologists (Eid et al. 2018).

There are also numerous commercial applications developed for similar purposes (e.g. Amira,[12] MeVisLab,[13] and ScanIp[14]).

While the presented applications are useful for visualisation of medical and biological volumetric data, they are still a stand-alone applications where users are limited to their use on the machine(s) where they are installed. Nowadays, this problem is mostly tackled by developing web-based applications which can run on almost any platform, including tablets and mobile phones and from any location with adequate internet connection. Users only need the web-browser, internet connection and the access to the data. In this book chapter we present such web-based solutions which are being made possible due to the fast development of accessibility of hardware acceleration in the web browsers using WebGL

[11] https://graphics.tudelft.nl/exposure-render/

[12] https://www.thermofisher.com/si/en/home/industrial/electron-microscopy/electron-microscopy-instruments-workflow-solutions/3d-visualization-analysis-software/amira-life-sciences-biomedical.html

[13] https://www.mevislab.de/

[14] https://www.synopsys.com/simpleware/products/software/scanip.html

standard.[15] With modern graphical APIs for web-browsers the computational power of the GPUs is accessible from the web applications allowing the developers to implement solutions similar to the stand-alone ones presented above.

In the following sections we first present background methods. Next we present the web-based implementations of volume data visualisation in Sect. 1.3, we discuss their use-cases in Sect. 1.4, and in the conclusions we present the upsides and downsides of existing approaches. We give some pointers for what can be expected in the future and which directions should be pursued.

1.2 Background

In this section we give a basic background of the volumetric data, the techniques and the methods used for their visualisation.

1.2.1 Volumetric Data

Volumetric data is a 3D regular grid representation (see Fig. 1.7) of a desired object or part of the object, expressing its specific property. Such data can be obtained using different techniques as was already presented in the Introduction section and thus the data represents different properties. In most cases the data is scalar data, where individual element (voxel) represents the property value at the specific location in the object. In some cases we can even obtain multiple properties per element making the data multi-modal (e.g. CT value and segmentation annotations).

The resolution of the data depends on the acquisition hardware, the acquisition process and the technique used, and can exceed the sizes of 2048^3 elements. Even for a 8-bit precision scalar data of such resolution, this results in more than 8.5 GB of data. While this is not a problem for a modern workstation systems, where working memory (RAM) reaches 64 or in some cases even 512 GB, it still poses a big challenge for mobile devices where high-end devices offer up to 12 GB of RAM, which is party already used by system, leaving hardly enough space for storing such large volumetric data for processing.

[15] https://www.khronos.org/registry/webgl/specs/latest/2.0/

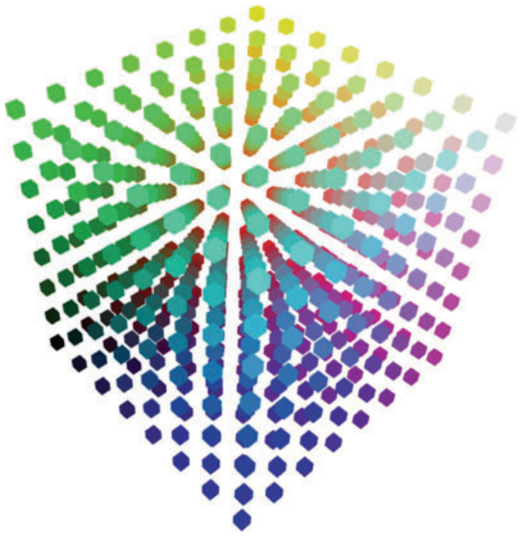


Fig. 1.7 Example volumetric data

1.2.2 Indirect Volume Rendering

First techniques for visualising volumetric data were designed so that the data was first converted into more appropriate form and then rendered. Most popular was the transformation into mesh geometry using different surface extraction techniques such as already mentioned Marching Cubes or its derivatives (e.g. Marching tetrahedra (Doi and Koide 1991; Bagley et al. 2016)) or more prominent methods such as multi-level partition of unity implicits (Ohtake et al. 2003; Berger et al. 2017).

The advantage of using such methods is the possibility of using existing hardware for accelerated rendering. The downside, is that such methods can be usually used only to visualise the surfaces, which is not always what we want to achieve. In some cases there is also the need of visualising translucent/transparent materials with its non-uniform structure. This is typically not possible using indirect rendering techniques and is the main reason for using direct volume rendering techniques. An example of indirect rendering is implemented in NeckVeins and SimVascular applications (some other applications also offer such a visualisation option – e.g. 3D Slicer and ParaView).

1.2.3 Direct Volume Rendering

Direct volume rendering (DVR) takes every sample value and maps it to opacity and colour. This is usually done with a so-called *transfer function* which defines how the values of the volumetric data remap to colour and opacity values. The final image is a projection and composition of these values for each pixel of the output image. We already presented basic methods in the Introduction section. In the following subsections we present the most common methods used in direct volume rendering which can be implemented in different ways (e.g. using ray casting, splatting, shear warping or other direct rendering technique).

1.2.3.1 Maximum Intensity Projection

One of the simplest direct rendering methods is the Maximum intensity projection (MIP) (Wallis et al. 1989). The method projects the data so that only maximum values along the selected path is projected onto the canvas and all the other values are ignored. The data is projected using desired projection transformation (either perspective or orthographic). The main benefit of this method is that it is fast and can be easily implemented even for large volumetric data. The downside on the other hand is the fact that we only see the maximum values in the data. Sometimes this might not be good enough for specific use cases. The method also does not provide any shading information which means that 3D structures in the data are often hard to recognise without the need of rotating the data. The method also lacks the depth perception since all the values are treated the same regardless of their distance from the user (Fig. 1.8).

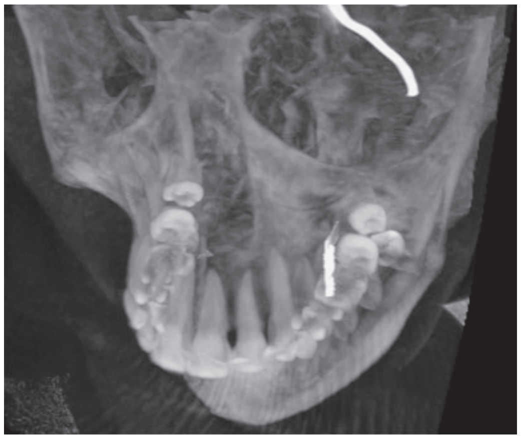


Fig. 1.8 An example of maximum intensity projection

1.2.3.2 ISO Surface Rendering

Extracting the surface from the volumetric data is what is mostly done in Indirect volumetric rendering. However, this was also one of the first DVR methods (Levoy 1988). The method can be implemented using different techniques the most common one being ray casting. The implementation using ray casting sends a ray through every pixel of the canvas and finds the first value in the volumetric data which exceeds the specified threshold. Several researchers have presented the improvements and different implementations of this method such as Parker et al. (1998) and Marmitt et al. (2004). The method can be further extended with surface normal calculation which are used in shading process, as can be see in an example in Fig. 1.9.

1.2.3.3 Emission-Absorption Model

MIP and ISO surface rendering only use single value of the volumetric data along the traced path through the volume for final rendering output and thus cannot present the transmittive media. Emission-Absorption Model (Max 1995b) is one of the most often used method that also supports the visualisation of transmittive media in the volume. It is based on the rendering equation (Kajiya 1986) but only uses the emission and absorption parts, neglecting in- and out-scattering contributions. This method is very useful for rendering continuous volumetric data. The data from medi-

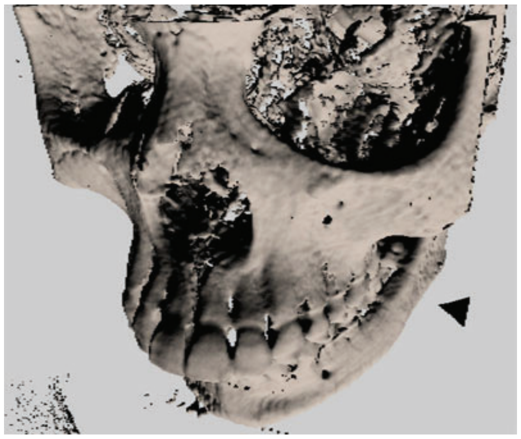


Fig. 1.9 An example of ISO surface rendering

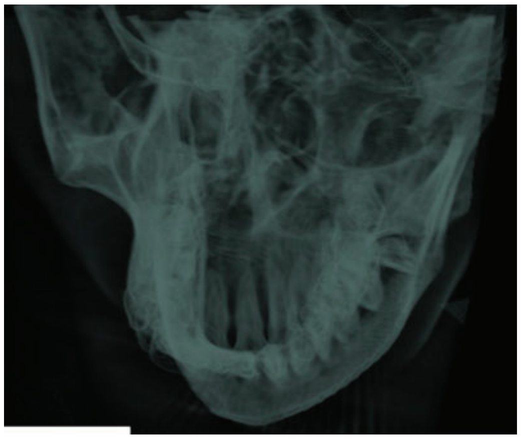


Fig. 1.10 An example of rendering using Emission-Absorption Model

cal and biological domain fit well in this category due to the nature of tissue structure. Users can define different emission and absorption weights for different value ranges and thus emphasise desired tissues. While the method can be implemented for fast computation and gives good insight into the data due to the absorption, it still lacks good depth perception due to the lack of phenomena such as shading and shadows, which can be seen in Fig. 1.10.

1.2.3.4 Multiple Scattering Model

Best state-of-the-art volumetric rendering techniques cover all aspects of the rendering

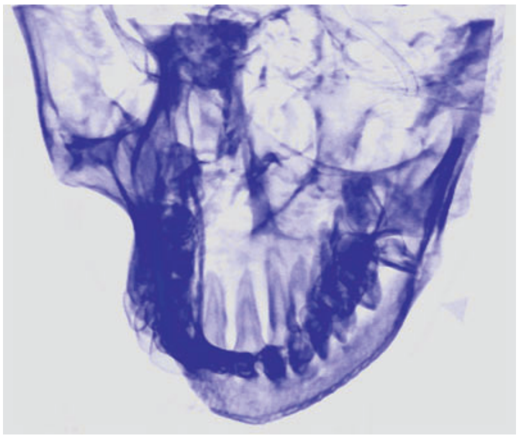


Fig. 1.11 An example of rendering using Multiple Scattering Model

equation: emission, absorption, in-scattering and out-scattering. While first attempts were developed (Kajiya and Von Herzen 1984) even before the rendering equation was published, many methods were developed as can be seen in an overview of early physically based volumetric rendering approaches (Max 1995a). The implementation of the rendering equation obsolete was implemented in many ways, such as: path-tracing (Kajiya 1986; Lafortune and Willems 1993), photon-mapping (Jarosz et al. 2008; Bitterli and Jarosz 2017) and Metropolis light transport simulation (Veach and Guibas 1997; Pauly et al. 2000).

The main problem of the multiple scattering models is their computational complexity. Most of them use approximation techniques such as Monte Carlo integration for calculating rendering equation integration estimates, which is computationally heavy. Still there is no implementation capable of real-time visualisation of a complex data using affordable hardware. There are some attempts of implementing interactive incremental visualisations such as already mentioned Exposure render (Kroes et al. 2012) and Volumetric path tracing framework (Lesar et al. 2018) presented in one of the following sections. The state of the art of physically-based volumetric rendering methods is presented in Novák et al. (2018a,b) (Fig. 1.11).

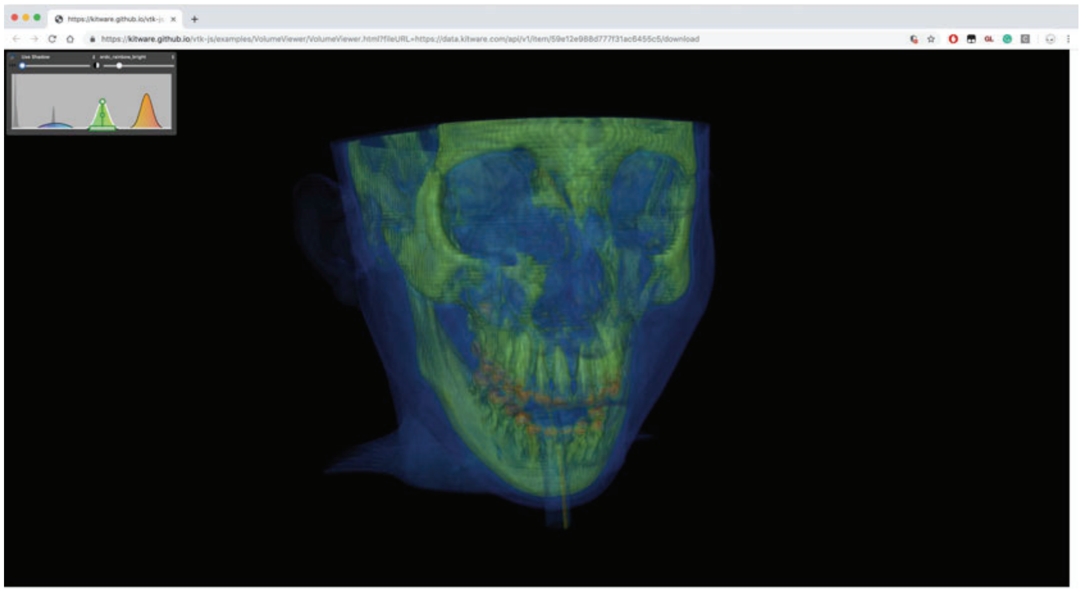


Fig. 1.12 An example of visualisation using VTK.js framework

1.3 Web Based Visualisation Systems

In the introduction section we presented a selection of stand-alone applications for visualisation of volumetric data for general or specialised use-cases. One of the main downsides of these application is their platform and hardware dependence. To overcome this problem several attempts were made to use the web platform for implementation of visualisation systems. In this section we present the selection of web-based visualisation systems for volumetric data. While some are the reimplementations of the existing stand-alone systems, other were build on top of general web-based visualisation systems and some were designed and developed from scratch.

1.3.1 VTK.js

VTK.js[16] – The Visualisation Toolkit developed in JavaScript, which is a subset of the original VTK C++ library and is intended for 3D graphics, volumetric rendering and visualisation. It includes several data processing algorithms and rendering techniques. VTK.js includes hardware accelerated volume rendering using ray casting developed in WebGL 1.0, which can be seen in Fig. 1.12. It also supports combining volumetric and mesh geometry rendering. The framework is not intended for end-users but for developers who can easily transition the application developed using original VTK framework on to the web platform. An example of such application is ParaViewWeb presented in the following section. Like original VTK, VTK.js is also missing state-of-the-art volumetric rendering methods. Since it is developed using WebGL 1.0 standard it does not exploit more advanced functionalities being introduced in WebGL 2.0 resulting in lower rendering performance.

1.3.2 ParaViewWeb

Built on top of VTK-js, ParaViewWeb,[17] a JavaScript Library, is not a reimplemntation of the original ParaView application, but a web

[16]https://kitware.github.io/vtk-js/

[17]https://www.paraview.org/web/

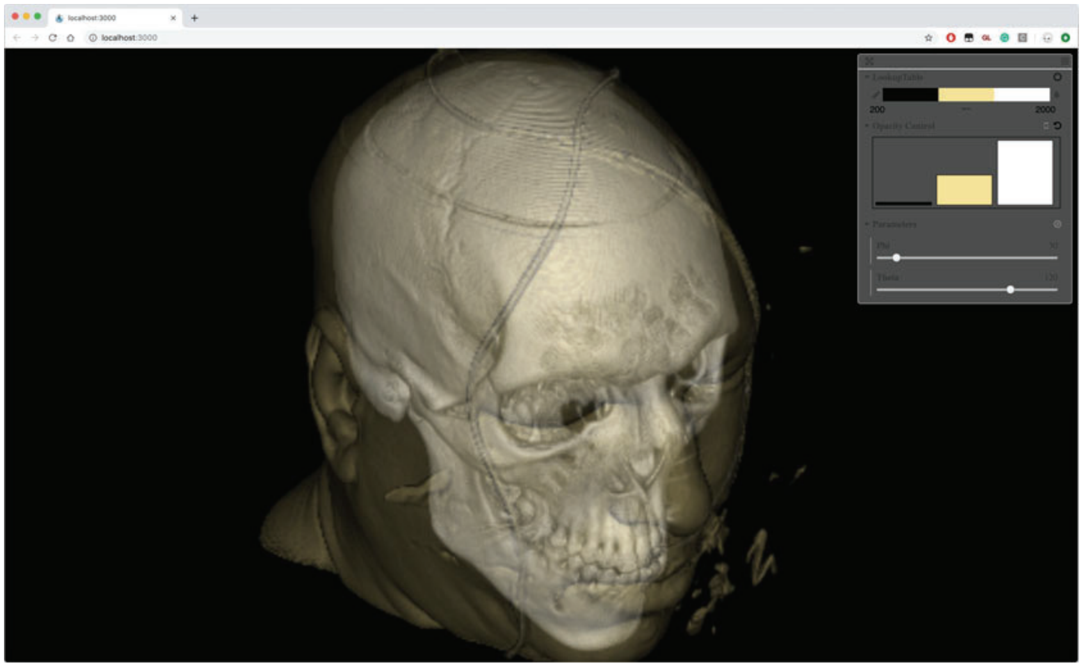


Fig. 1.13 An example application build using ParaViewWeb framework

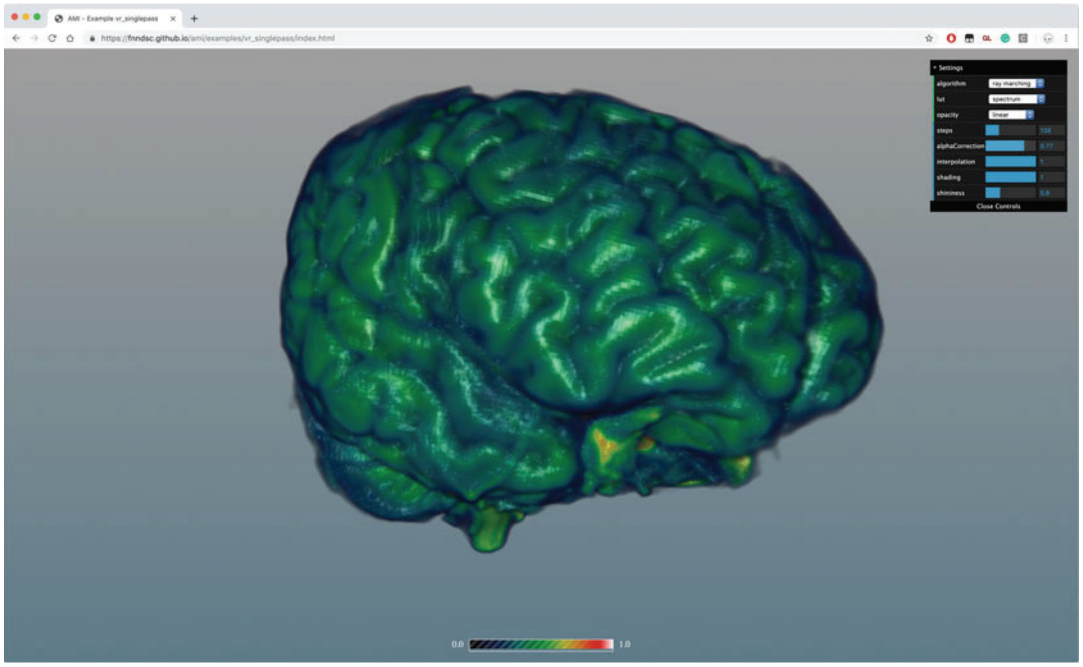
framework designed for building interactive scientific visualisations for the web platform. It extends the functionalities implemented in VTK.js with overlaid modules for user interaction and connection with the original ParaView application which can provide the visualisation data and acts as backend for data processing as well as rendering.

It includes:

- **Visualisation components**, which is a set of interactive 2D visualisation tools for exploring the data including: a field selector, histograms, and parallel coordinates.
- **Interaction** support, which is a crucial property of all the implemented components and allows uses to interact with the data using standard inputs such as mouse, keyboard and touch input.
- **Data access**, which supports loading the data from the online sources using: (1) XHR (XML http request built into web browsers) for downloading any kind of data, (2) WebSockets for persistent communication with backend systems (e.g. ParaView server) and (3) Girder[18] – a consistent interface to KitWare Girder data management backend system.
- **UI widgets**, which is a collection of interactive widgets that allows the development of complex interactive UIs for interactive visualisation parameter settings (e.g. equaliser editor, light properties, editor, transfer function editor etc.).
- **Rendering viewers**, which is a set of dedicated viewing components such as: image viewer, WebGL image compositing viewer, and WebGL viewer for geometry. While originally ParaViewWeb was using Three.js[19] library for visualisation purposes, it migrated to VTK.js.

The authors have provided users/developers with a collection of example applications for different scientific domains (e.g. CT head sample visualisation presented in Fig. 1.13) covering variety of use-cases. The presented visualisation

[18]https://girder.readthedocs.io/en/stable/
[19]https://threejs.org/


AMI – Example vr_singlepass
https://fnndsc.github.io/ami/examples/vr_singlepass/index.html
Settings
Close Controls
0.0
1.0

algorithm: ray marching
lut: spectrum
opacity: linear
steps: 138
alphaCorrection: 0.77
interpolation: 1
shading: 1
shininess: 5.8

Fig. 1.14 An example visualisation of DICOM volumetric data using Ami.js toolkit

system offers many functionalities for data processing, analysis and visualisation, but lacks the support of modern visualisation standard and implementation of state-of-the-art volumetric rendering techniques. The downside is also the use of custom dataset format which is not compatible with other applications.

1.3.3 Ami.js

Ami.js – a Medical Imaging JavaScript Toolkit[20] (Rannou et al. 2017; Arbelaiz et al. 2017) includes 2D and 3D visualisation of medical data supporting most popular medical imaging formats (e.g. DICOM, NRRD, NIFTI and MHD). It is build on top of Three.js and integrates several libraries for parsing the data from different formats. It supports the composition of 2D images and 3D model visualisations in the same view. The toolkit also supports real-time interaction and provides a set of UI elements for visualisation system setting (Fig. 1.14).

While Ami.js is a lightweight visualisation toolkit build on top of Three.js framework it offers additional ray marching volumetric rendering with predefined transfer functions. Currently it does not support user-defined transfer functions, but they can be implemented during the development stage. It is not intended as a stand-alone tool, but as a basis for developing custom visualisation systems. It relies on existing libraries for loading volumetric data. Since it is build on top of Three.js it does not support the use of WebGL 2.0. Also, the UI is less adaptable than the one implemented in ParaViewWeb.

Ami.js has been used in several web-based visualisation systems and was also integrated into a collaborative web-based real-time neuroimage visualisation system (Bernal-Rusiel et al. 2017).

[20] https://github.com/FNNDSC/ami

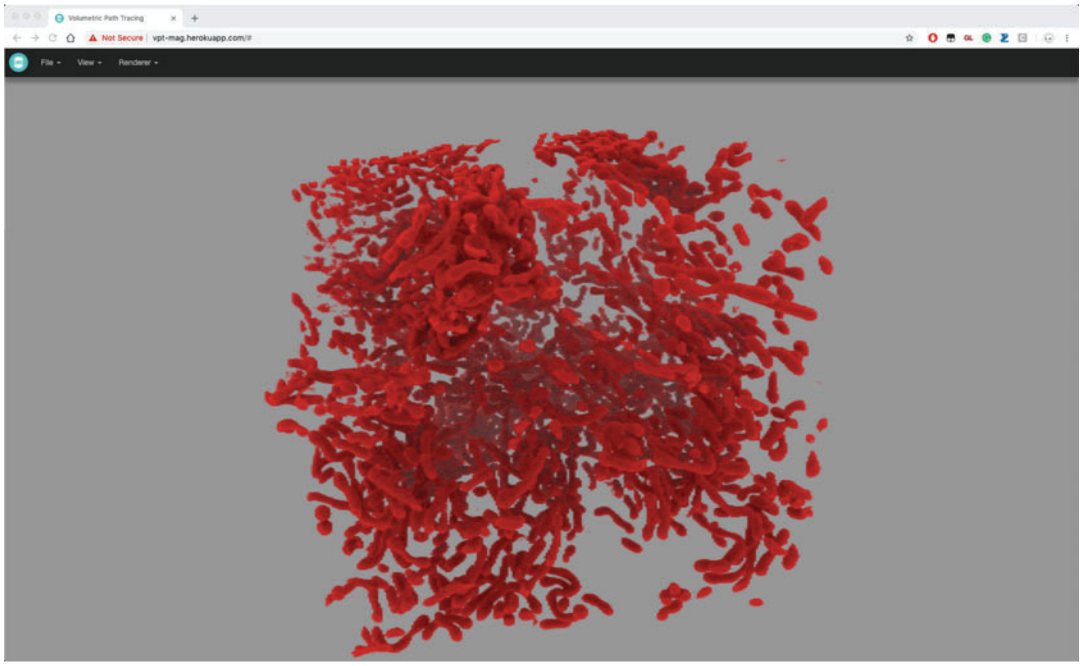


Fig. 1.15 Multiple scattering Monte Carlo path tracing of segmented biomedical microscopy data of mitochondria in mouse urinary bladder cell

1.3.4 VPT

VPT – Volumetric path tracing[21] (Lesar et al. 2018) is a web-based visualisation toolkit designed for visualisation of volumetric data supporting different techniques implemented with support for incremental rendering and is developed in JavaScript with WebGL 2.0 support. It supports visualisation using MIP, ISO surface rendering, EAM, single and multiple scattering Monte Carlo based volumetric path tracing, which is state-of-the-art web-based implementation. The path tracing implementation supports the use of user-defined 2D transfer functions and heterogeneous data. It was developed for interactive, real-time visualisation on mobile and desktop devices. An example of path traced biomedical microscopy data is presented in Fig. 1.15.

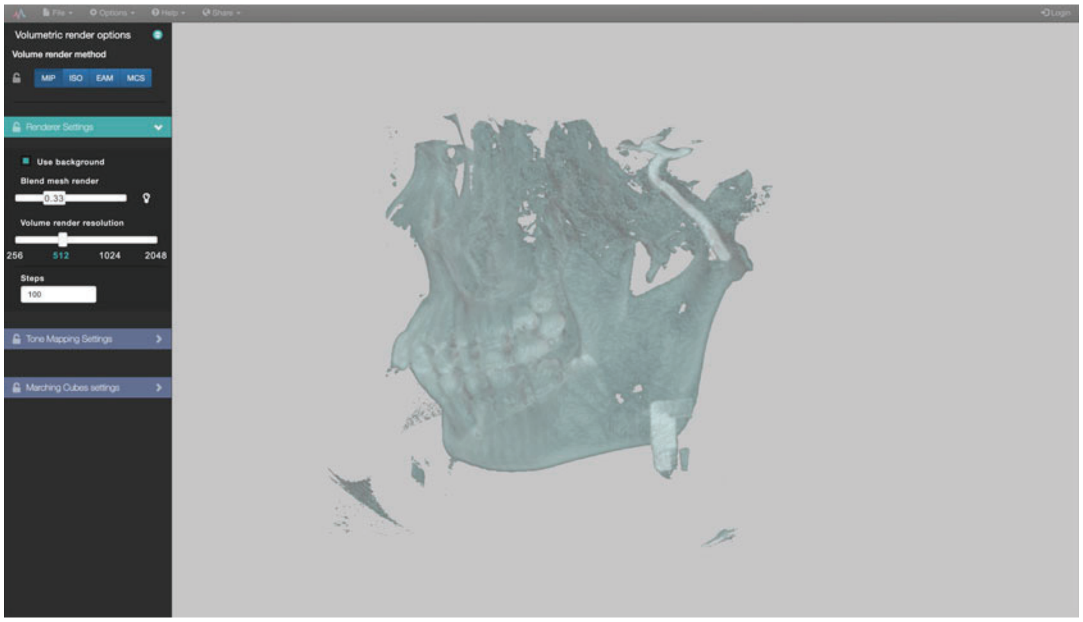
The visualisation techniques are implemented incrementally, taking the more complex methods to converge over multiple sequential frames. While simple techniques (e.g. MIP, ISO surface rendering and EAM) converge in few frames, more complex techniques need more time for converging, depending on the used hardware. The benefit of such implementation is that the visualisation is fully interactive even with use of complex rendering techniques. Once the user stops interacting (e.g. moving, rotation or zooming) the image converges to its final version. The system is extendable and can also be used as a plug-in in other visualisation systems, as is presented for the Med3D case in the following section.

1.3.5 Med3D

Med3D[22] (Bohak et al. 2016; Lavrič et al. 2017) is a lightweight web-based visualisation system developed for medical use-cases. It is developed in JavaScript with WebGL 2.0 and implements deferred rendering pipeline, unlike Three.js and VTK.js, which allows easy multipass rendering

[21] https://github.com/terier/vpt

[22] https://github.com/UL-FRI-LGM/Med3D



Use background

Blend mesh render 0.33

Volume render resolution 256 512 1024 2048

Steps 100

Fig. 1.16 A rendering composed of mesh geometry rendered using Med3D integrated renderer overlaid by maximum intensity projection rendered using VPT

and composition of several output rendering images into a seamless final image. This also allows the use of external rendering systems used in specified rendering pass (Bohak et al. 2019) as can be seen in Fig. 1.16, where basic mesh geometry is overlaid with maximum intensity projection obtained with VPT.

On top of fast and customisable rendering pipeline the framework offers an intuitive easy-to-use UI. It also implements the use of arbitrary input devices, which are handled by the underneath system (such as 3D mouse, Leap motion controller, game-pad, etc.). The framework supports the use of remote rendering system in a selected rendering pass of the pipeline (Lavrič et al. 2018). It supports loading of mesh geometry data and volumetric data. The volumetric data can be transformed into mesh geometry using Marching Cubes surface extraction algorithm for the user-defined ISO value or sent to volumetric render plug-in. The framework also implements two types of annotations: (1) 3D pinned annotations, which can be pinned to a selected region in 3D space on a mesh geometry and (2) view-aligned hand-drawn annotations which can be drawn on top of the current camera view. Both types of annotation can be seen in Fig. 1.17.

The framework was used for web-based vascular flow simulation visualisation (Oblak et al. 2018) and medical volumetric data visualisation. It offers remote user collaboration (Lavrič et al. 2017), where a user can share his current scene with other users. Collaborative functionality includes:

- **data sharing**, which allows users to share their own data (or data already stored on the server) with other users;
- **annotation sharing**, where a user can decide which annotations to share with other users (it works for 3D pinned as well as view-aligned annotations);
- **camera view sharing**, where a user can share his view of the data, which is very important when discussing the structure of the data or possible abnormalities;
- **visualisation properties sharing**, where a user can share their rendering setup (the selected rendering technique and its properties);

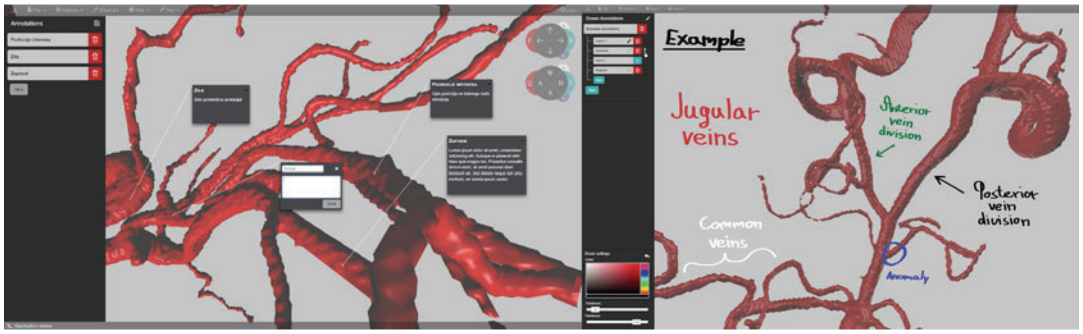


Fig. 1.17 Collaborative visualisation with annotations in Med3D visualisation framework

- **interactive chat**, where users can chat using text messages.

While the framework is very extendable and offers the use in many use-cases it also supports the integration with other rendering libraries (such as VPT) for providing state-of-the-art volumetric rendering on the web platform.

1.4 Discussion

Majority of the presented solutions are general purpose visualisation frameworks or toolkits, not suited for end-use. There are example applications available for specific use-cases with ParaViewWeb, Ami.js and Med3D, however the out-of-the-box solutions would increase the overall acceptance by the communities. The biggest community is behind the ParaViewWeb and VTK.js, due to their long lasting history in form of stand-alone application. On the other hand these are the solutions that do not offer state-of-the-art volumetric rendering implementations.

Most off the presented solutions support some kind of composed visualisations using mesh geometry and volumetric input data (apart from VPT which is designed for volume rendering only). VPT is the only solution offering a state-of-the-art physically based volumetric rendering support, which is more common in dedicated systems (mostly the commercial ones).

While the web platform offers great basis for implementation of collaborative visualisation environments only two solutions (Med3D and to some extend Ami.js) have these capabilities integrated into the system. Since visualisation in medical domain is often used for diagnostic purposes where more than single physician is involved, this poses a great advantage in the real-life scenarios (e.g. getting second opinion at the distance or use in distance learning environments).

Even though the web platform is well accepted in the everyday life, there are still a lot of opportunities where it can be used in professional scenarios, such as visualisation of medical and biological data. Such solutions allow users to access and visualise the data at the distance on the low-end laptops or even on mobile devices.

1.5 Conclusions

In this book chapter we have presented a short overview of current open source stand-alone visualisation systems used for medical and biological visualisations and have presented the overview of comparable web-based solutions. While some web-based solutions are reimplementations of the presented stand-alone solutions with a subset of features (e.g. VTK.js and ParaViewWeb), others were developed using existing web-based libraries and additionally adapted for medical domain specifics (Ami.js) or are developed from scratch and designed for such visualisation scenarios (VPT and Med3D). The later are also the only solutions that make use of the WebGL

2.0 standard for best GPU performance gain. Only one solution was developed with extensive support for collaborative scenarios in its core (Med3D).

The hardware accelerated rendering has made a new leap with Khronos' Vulkan,[23] Apple's Metal[24] and Microsoft's DirectX 12 graphics APIs. While this APIs offer even better exploitation of the GPUs processing power, they are not available for the web platform. This lead to the development of a new API by W3C group, called WebGPU. It is based on the above mentioned APIs for native development. This API will allow developers to create even more optimised implementations of visualisation systems making web-based real-time physically based volumetric rendering a step closer to the practical use.

References

Ahrens J, Geveci B, Law C (2005) 36 – paraview: an end-user tool for large-data visualization. In: Hansen CD, Johnson CR (eds) Visualization handbook. Butterworth-Heinemann, Burlington, pp 717–731. https://doi.org/10.1016/B978-012387582-2/50038-1

Arbelaiz A, Moreno A, Kabongo L, García-Alonso A (2017) Volume visualization tools for medical applications in ubiquitous platforms. In: ehealth 360°, pp 443–450. https://www.vicomtech.org/en/rdi-tangible/publications/publication/volume-visualization-tools-for-medical-applications-in-ubiquitous-platforms

Bagley B, Sastry SP, Whitaker RT (2016) A marching-tetrahedra algorithm for feature-preserving meshing of piecewise-smooth implicit surfaces. Proc Eng 163:162–174. (25th International Meshing Roundtable). https://doi.org/10.1016/j.proeng.2016.11.042

Bakalash R, Kaufman AE, Pacheco R, Pfister H (1992) An extended volume visualization system for arbitrary parallel projection. In: Eurographics workshop on graphics hardware, pp 64–69

Berger M, Tagliasacchi A, Seversky LM, Alliez P, Guennebaud G, Levine JA, … Silva CT (2017) A survey of surface reconstruction from point clouds. In: Comput Graph Forum 36:301–329

Bernal-Rusiel JL, Rannou N, Gollub RL, Pieper S, Murphy S, Robertson R, … Pienaar R (2017) Reusable client-side javascript modules for immersive web-based real-time collaborative neuroimage visualization. Front Neuroinform 11:32. https://doi.org/10.3389/fninf.2017.00032

Bitterli B, Jarosz W (2017) Beyond points and beams: higher-dimensional photon samples for volumetric light transport. ACM Trans Graph (Proc SIGGRAPH) 36(4). https://doi.org/10.1145/3072959.3073698

Bohak C, Sodja A, Marolt M, Mitrovič U, Pernuš F (2014) Fast segmentation, conversion and rendering of volumetric data using GPU. In: Iwssip 2014: proceedings, pp 239–242

Bohak C, Lavrič P, Marolt M (2016) Remote interaction in web-based medical visual application. In: Proceedings of the 19th international multi conference information society – is 2016, 11 oct 2016, Ljubljana, volume e, pp 5–8

Bohak C, Aleksandrov J, Marolt M (2019) Collaborative web-based merged volumetric and mesh rendering framework. In: Proceedings of augmented reality, virtual reality, and computer graphics 2019 (To appear)

Briggman KL, Bock DD (2012) Volume electron microscopy for neuronal circuit reconstruction. Curr Opin Neurobiol 22(1):154–161. (Neurotechnology). https://doi.org/10.1016/j.conb.2011.10.022

Cameron GG, Undrill PE (1992) Rendering volumetric medical image data on a simd-architecture computer. In: Proceedings of the third eurographics workshop on rendering, Bristol, pp 135–145

Crawford CR, King KF (1990) Computed tomography scanning with simultaneous patient translation. Med Phys 17(6):967–982. https://doi.org/10.1118/1.596464

Doi A, Koide A (1991) An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. In: Ieice transactions of information and systems, vol E74-D

Eid M, De Cecco CN, Nance JW, Caruso D, Albrecht MH, Spandorfer AJ, De Santis D, Varga-Szemes A, Schoepf UJ (2018) Cinematic rendering in CT: a novel, lifelike 3D visualization technique. Am J Roentgenol 209(2):370–379. https://doi.org/10.2214/AJR.17.17850

Fedorov A, Beichel R, Kalpathy-Cramer J, Finet J, Fillion-Robin J-C, Pujol S, … Kikinis R (2012) 3D Slicer as an image computing platform for the quantitative imaging network. Magn Reson Imaging 30(9):1323–1341. https://doi.org/10.1016/j.mri.2012.05.001

Hibbard W, Santek D (1989) Interactivity is the key. In: Proceedings of the 1989 chapel hill workshop on volume visualization, New York, pp 39–43. https://doi.org/10.1145/329129.329356

Jarosz W, Zwicker M, Jensen HW (2008) The beam radiance estimate for volumetric photon mapping. Comput Graph Forum (Proc Eurograph) 27(2):557–566. https://doi.org/10.1111/j.1467-8659.2008.01153.x

Jönsson D, Steneteg P, Sundén E, Englund R, Kottravel S, Falk M, … Ropinski T (2018) Inviwo – a visualization system with usage abstraction levels. arXiv, Retrieved from http://arxiv.org/abs/1811.12517

Kajiya JT (1986) The rendering equation. SIGGRAPH Comput Graph 20(4):143–150. https://doi.org/10.1145/15886.15902

[23] https://www.khronos.org/vulkan/

[24] https://developer.apple.com/metal/

Kajiya JT, Von Herzen BP (1984) Ray tracing volume densities. In: Acm siggraph computer graphics, vol 18, pp 165–174

Kalender WA, Seissler W, Klotz E, Vock P (1990) Spiral volumetric ct with single-breath-hold technique, continuous transport, and continuous scanner rotation. Radiology 176(1):181–183. https://doi.org/10.1148/radiology.176.1.2353088

Kanus U, Meißner M, Straßer W, Pfister, H, Kaufman A, Amerson R, … Snider G (1997) Implementations of cube-4 on the teramac custom computing machine. Comput Graph 21(2):199–208. https://doi.org/10.1016/S0097-8493(96)00083-0

Ke H-R, Chang R-C (1993) Sample buffer: a progressive refinement ray-casting algorithm for volume rendering. Comput Graph 17(3):277–283. https://doi.org/10.1016/0097-8493(93)90076-L

Krakow D, Williams III J, Poehl M, Rimoin D, Platt L (2003) Use of three-dimensional ultrasound imaging in the diagnosis of prenatal-onset skeletal dysplasias. Ultrasound Obstet Gynecol 21(5):467–472. https://doi.org/10.1002/uog.111

Kroes T, Post FH, Botha CP (2012) Exposure render: an interactive photo-realistic volume rendering framework. PLOS ONE 7:1–10. https://doi.org/10.1371/journal.pone.0038586

Lacroute P, Levoy M (1994) Fast volume rendering using a shear-warp factorization of the viewing transformation. In: Proceedings of the 21st annual conference on computer graphics and interactive techniques, New York, pp 451–458. https://doi.org/10.1145/192161.192283

Lafortune EP, Willems YD (1993). Bi-directional path tracing. In: Proceedings of third international conference on computational graphics and visualization techniques (compugraphics'93), Alvor, pp 145–153

Lavrič P, Bohak C, Marolt M (2017) Collaborative view-aligned annotations in web-based 3d medical data visualization. In: Mipro 2017: 40th jubilee international convention, Proceedings, Opatija, 22–26 May 2017, pp 276–280

Lavrič P, Bohak C, Marolt M (2018) Vulkan abstraction layer for large data remote rendering system. In: Proceedings of augmented reality, virtual reality, and computer graphics 2018, pp 480–488

Lesar Ž, Bohak C, Marolt M (2018) Real-time interactive platform-agnostic volumetric path tracing in WebGL 2.0. In: Web3d 2018: proceedings, pp 1–7

Levoy M (1988) Display of surfaces from volume data. IEEE Comput Graph Appl 8(3):29–37. https://doi.org/10.1109/38.511

Lorensen WE, Cline HE (1987) Marching cubes: a high resolution 3d surface construction algorithm. SIGGRAPH Comput Graph 21(4):163–169. https://doi.org/10.1145/37402.37422

Marmitt G, Kleer A, Wald I, Friedrich H, Slusallek P (2004) Fast and accurate ray-voxel intersection techniques for iso-surface ray tracing. In: Proceedings of the vision, modeling, and visualization conference 2004 (VMV 2004), Stanford, 16–18 Nov 2004, pp 429–435

Max N (1995a) Efficient light propagation for multiple anisotropic volume scattering. In: Sakas G, Müller S, Shirley P (eds) Photorealistic rendering techniques. Springer, Berlin/Heidelberg, pp 87–104

Max N (1995b) Optical models for direct volume rendering. IEEE Trans Vis Comput Graph 1(2):99–108. https://doi.org/10.1109/2945.468400

Meyer-Spradow J, Ropinski T, Mensmann J, Hinrichs K (2009) Voreen: a rapid-prototyping environment for ray-casting-based volume visualizations. IEEE Comput Graph Appl 29(6):6–13. https://doi.org/10.1109/MCG.2009.130

Novák J, Georgiev I, Hanika J, Jarosz W (2018a) Monte Carlo methods for volumetric light transport simulation. Comput Graph Forum (Proc Eurographics – State of the Art Reports) 37(2). https://doi.org/10.1111/cgf.13383

Novák J, Georgiev I, Hanika J, Křivánek J, Jarosz W (2018b) Monte Carlo methods for physically based volume rendering. In: ACM siggraph courses. https://doi.org/10.1145/3214834.3214880

Oblak R, Bohak C, Marolt M (2018) Web-based vascular flow simulation visualization with lossy data compression for fast transmission. In: Proceedings of augmented reality, virtual reality, and computer graphics 2018, pp 3–17

Ohtake Y, Belyaev A, Alexa M, Alexa M, Turk G, Seidel H-P (2003) Multi-level partition of unity implicits. ACM Trans Graph 22(3):463–470. https://doi.org/10.1145/882262.882293

Ollinger JM, Fessler JA (1997) Positron-emission tomography. IEEE Signal Process Mag 14(1):43–55. https://doi.org/10.1109/79.560323

Parker S, Shirley P, Livnat Y, Hansen C, Sloan P-P (1998) Interactive ray tracing for isosurface rendering. In: Proceedings visualization'98 (cat. no.98cb36276), pp 233–238. https://doi.org/10.1109/VISUAL.1998.745713

Pauly M, Kollig T, Keller A (2000) Metropolis light transport for participating media. In: Rendering techniques 2000. Springer, Vienna, pp 11–22

Pfister H, Kaufman A, Chiueh T (1994) Cube-3: a real-time architecture for high-resolution volume visualization. In: Proceedings of the 1994 symposium on volume visualization, pp 75–83

Rannou N, Bernal-Rusiel JL, Haehn D, Grant PE, Pienaar R (2017) Medical imaging in the browser with the a* medical imaging (ami) toolkit. In: Proceedings of ESMRMB annual scientific meeting 2017

Rinck PA, Muller R, Peterson S (1990) An introduction to magnetic resonance in medicine: the basic textbook of the European workshop on magnetic resonance in medicine

Schroeder WJ, Avila LS, Hoffman W (2000) Visualizing with VTK: a tutorial. IEEE Comput Graph Appl 20(5):20–27. https://doi.org/10.1109/38.865875

Šrámek M (2006) 20 years of volume rendering. In: Proceedings of the 22nd spring conference on computer graphics, pp 7–16. https://doi.org/10.1145/2602161.2602162

Updegrove A, Wilson NM, Merkow J, Lan H, Marsden AL, Shadden SC (2017) Simvascular: an open source pipeline for cardiovascular simulation. Ann Biomed Eng 45(3):525–541. https://doi.org/10.1007/s10439-016-1762-8

Veach E, Guibas LJ (1997) Metropolis light transport. In: Proceedings of the 24th annual conference on computer graphics and interactive techniques, pp 65–76

Wallis JW, Miller TR, Lerner CA, Kleerup EC (1989) Three-dimensional display in nuclear medicine. IEEE Trans Med Imaging 8(4):297–230. https://doi.org/10.1109/42.41482

Westover LA (1991) Splatting: a parallel, feed-forward volume rendering algorithm. Unpublished doctoral dissertation, University of North Carolina at Chapel Hill, Chapel Hill (UMI Order No. GAX92-08005)