# 基于 WebGL 的医学图像三维可视化研究①

方路平, 李国鹏, 洪文杰, 万铮结

(浙江工业大学 信息工程学院, 杭州 3100023)

**摘 要**: 现今医学图像往往与大型的硬件设备和复杂的软件联系在一起, 然而随着互联网技术的发展, 越来越多互联网应用的出现改变了人们对传统本地软件的依赖, 现今医学图像在互联网领域才刚刚起步, 提出了一种在浏览器中实现医学图像的三维可视化的方法, 能够通过成熟的本地医学图像平台(比如 3DSlicer)获取医学图像数据, 结合 HTML5 以及 WebGL(Web Graphics Library)来实现医学图像的三维可视化.

**关键词**: 医学图像; 可视化; 浏览器; WebGL; HTML5

## 3-D Visualization of Medical Images Based on WebGL

FANG Lu-Ping, LI Guo-Peng, HONG Wen-Jie, WAN Zheng-Jie

(Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China)

**Abstract**: Medical images are usually big data and manipulated by large-scale hardware and complex software platform. With the development of Internet technology, people tends to use more and more online applications instead of heavy-weighted traditional softwares installed locally. Researches on online medical images processing has just started. In this article, a light-weighted solution of 3-D visualization of medical images is presented. Combining of HTML5 features and WebGL technique, web browser can be used as 3-D visulization framework for medical images data of specific formats (such as vtk, nrrd etc) , which are produced from the traditional imaging platforms (such as 3D-Slicer).

**Key words**: medical images; visualization; browser; WebGL; HTML5

当今世界, 人们对医疗领域的关注不断增长, 医学工作者如何更快地分享医学信息也已经成为一个重要的课题. 随着互联网技术的发展, 很多医院都建立起了影像归档和通信系统 PACS(Picture Archiving and Communication Systems), 这使得医院的各个部门能够共享患者的图像信息, 比如计算机断层扫描图像(CT)或者磁共振图像(MRI)等. 这些图像信息都是以二维图像的方式传输的. 如果要对图像信息进行三维可视化或者其他图像处理, 往往需要在计算机上安装特定软件, 复杂的软件需求决定了在大型医院的影像科才有专门的设备.

而在一个基于网络的系统中, 不同的人就能够在不同的时间和地点同时通过综合信息平台来访问患者医学信息以及使用相关工具. 生物医学图像的在线处理和可视化不但能够提高远程本地工具的可达性, 也能够维持不同用户处理的一致性[1].

随着 HTML5 以及 WebGL 技术的问世和发展, 浏览器中的三维可视化在游戏和动画领域已经成为热门研究方向, 与此同时, 利用本地计算机的显卡直接在浏览器上实现医学图像的三维显示已经成为可能. Google 基于 WebGL 开发了 Google body browser, 详细地展现了三维人体解剖模型. 与此同时, 目前已经广泛使用的三维显示工具库 VTK(Visualization ToolKit)以及 Teem 组织设计的 NRRD(Nearly Raw Raster Data)标准能够很好为本课题提供数据支持.

本文主要的工作有以下几个方面:

① 分析了主流的医学图像数据结构并设计了提取图像数据的方法;

② 体数据三维显示方法研究;

③ 基于 WebGL 实现医学图像三维可视化.

① 收稿时间:2013-03-09;收到修改稿时间:2013-05-06

## 1 医学图像数据研究

### 1.1 DICOM

DICOM 即数字影像和通信标准(Digital Imaging and Communications in Medicine). 在医学图像的发展过程中, 由于医疗设备厂商的不同, 相关设备的医学图像存储、传输方式也都不尽相同. 这使得医学图像的信息交换与共享受到阻碍. 因此, 美国放射协会(ACR)和全美电子厂商联合会(NEMA)认识到急需建立一种规范医学图像的标准, DICOM 在这种背景下产生. 目前 DICOM 图像已经发展到 3.0 标准. 按照 DICOM 存储的文件即 DICOM 文件, DICOM 文件通常由文件头和数据集组成, 通过对 DICOM 标准的分析, 我们能查询并提取 DICOM 文件中的图像信息, 进而进行三维显示. 然而 DICOM 文件往往包含大量与图像无关的信息, 且由一系列的 DICOM 序列文件组成, 不仅占用大量存储空间, 在数据的传输和读取上也极不便利, 为此, 有些公司及组织为便于对医学图像进行可视化, 设计了不同的软件系统和标准, 自定义了一系列的文件格式用于存储医学图像信息, 比如 VTK 格式, NRRD 格式等.

### 1.2 VTK 格式

VTK 是由 Kitware 公司开发的可视化工具. VTK 设计了一系列的对象来读写主流的数据文件(比如 DICOM 文件). 此外, VTK 还提供了自己的文件格式(即 VTK 格式), 旨在为不同的数据集类型提供一种统一的描述方案, 使数据在软件之间的传输变得简单.

现阶段有两种不同的 VTK 文件格式. 普通格式和 XML 格式.

普通的 VTK 文件包含 5 个部分:

① 第一部分是文件的版本和标识符;

② 第二部分是文件头. 文件头是一组以换行符结尾的字符串, 最多包含 256 个字符, 用来描述与该 VTK 相关的信息;

③ 第三部分为文件格式. 文件格式描述了该文件为 ASCII 或者二进制;

④ 第四部分是数据集结构;

⑤ 第五部分描述了数据集的属性.

通过成熟的本地医学图像平台及软件对医学图像进行处理, 将分析结果存储为 VTK 格式后能够更加方便、快速、直观地在浏览器上进行可视化.

### 1.3 NRRD 格式

NRRD 格式是由 Teem 为实现多维光栅数据的科学可视化以及图像处理而设计的一种文件格式和库集合. 除了维度, NRRD 在数据类型、文件编码、字节顺序等方面都具有很好的灵活性.

## 2 医学图像三维可视化方法

医学图像文件中的图像信息往往以体数据的方式存储, 故根据所要可视化的内容, 可以将医学图像的可视化划分为面绘制(surface rendering)和直接体绘制(directed volume rendering)[2].

面绘制根据需要可视化的组织结构的数据阈值, 在体数据场中找到等值面来生成三维图像. 代表性的方法为 MarchingCubes.

直接体绘制则将体数据集合中的每个体素看做一个具有一定属性(颜色、不透明度)的个体, 通过累加每个体素对光线的作用来产生二维图像. 由于直接体绘制需要涉及所有的体数据, 故计算量大, 但它能够保体数据的完整性.

## 3 浏览器中三维可视化的实现

### 3.1 HTML5 和 WebGL

HTML5 是一种最新的 HTML 标准. HTML5 所提供的 canvas 元素, 使浏览器能够直接在浏览器上实现二维以及三维绘图. WebGL 是由 Khronos Group 提出的一种 3D 图形 API 网络标准, 跨平台且免版权. 在 WebGL 之前, 大部分的 3D 环境仅能够通过插件的形式存在于浏览器中, 这常常引起兼容性问题. 是 HTML5 浏览器的 OpenGL ES 2.0 图形库. 由于 WebGL 是基于 OpenGL ES 2.0(即 OpenGL 为移动设备设计的子集)设计的, 故它使用和桌面 OpenGL 同样的着色语言架构. 通过 WebGL, 应用程序能够已经定制好的 OpenGL ES API 直接访问显卡[3], 这样就能够为 canvas 提供硬件加速, 开发人员就能够更加快速流畅地显示三维场景和模型了.

目前主流浏览器如 Safari, Chrome, Firefox 以及 Opera 都是 WebGL 工作组的成员. 开发人员可以在网页的 canvas 元素中添加 JavaScript 脚本来完成 HTML5 的绘图过程. 在开发基于 WebGL 的网络应用时, 通过设置 canvas 的 WebGL 环境后, 开发人员就能够调用

WebGL 的图形接口, 这样就能够在网页编程时像使用 C/C++进行本地 OpenGL 编程一样, 用 JavaScript 编写浏览器中的图形渲染代码.

### 3.2 WebGL 的绘图流程

WebGL 沿用了 OpenGL ES2.0 的图形渲染流程[4], 如图 1 所示. 三维空间中的每个顶点都有一个或者多个属性, 比如坐标、法向量、颜色、纹理坐标等. 在 WebGL 中, 这些与顶点相关的数据称为顶点属性, 这些顶点属性组成的数据流从 Web 应用程序输入依次经过顶点着色器, 片元着色器的处理(期间经过元图组装、光栅化等处理), 最后到达图形帧缓存, 用以在视图(即 canvas 元素)中显示.

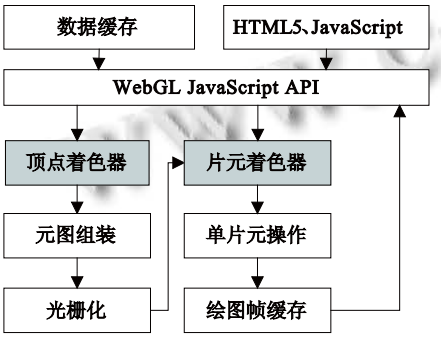



图 1 WebGL 图形渲染流程

基于 WebGL 的三维绘图的重点是输入物体的顶点信息以及由顶点构成的面信息, 一旦确定了顶点和面, 由计算机图形学知识, 就能结合模型视图矩阵和投影矩阵, 很好地实现三维绘图.

### 3.3 三维可视化系统框架

医学图像往往是以二维图像或者三维体数据存在, 其中三维体数据含有较大的信息量, 上文中提到, 医学图像的三维可视化方法一般分为面绘制和直接体绘制.

可以按照数据流将医学图像的三维可视化分为数据读取模块、数据处理模块以及可视化模块, 如图 2 所示.

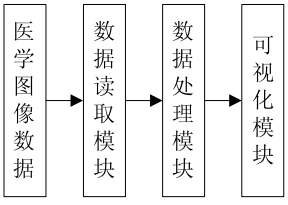



图 2 可视化系统框架

### 3.4 数据读取模块

Ajax 技术[5]能够使 JavaScript 在不重新载入网页的前提下通过与服务器进行数据交换. 目前所有浏览器都已经支持这个技术. 通过 XMLHttpRequest 对象就能够很好地实现医学图像文件的读取, 其算法流程如图 3 所示.

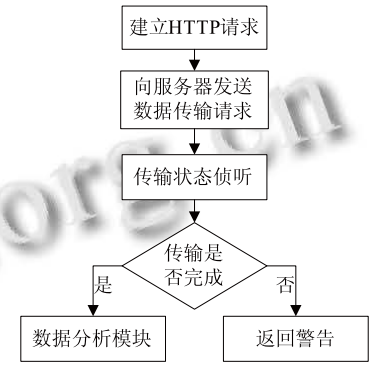



图 3 数据读取流程

### 3.5 数据处理模块

由于浏览器脚本语言的局限性, 在浏览器上直接进行 Marchingcubes 等面绘制算法效率比较低. 现在可以通过本地医学图像平台进行医学图像的面绘制运算, 将结果存储为 VTK 格式文件, 这样就能够通过直接加载 VTK 格式文件获取可视化所需的数据进行三维可视化, 而不需要重新执行面绘制算法. 下面以在 3D Slicer 中完成面绘制运算的人体颅骨为例, 分析 VTK 文件的处理流程. 通过查看文件内容我们可以发现颅骨 VTK 文件包含以下内容(省略号处表示数据区域):

DATASET POLYDATA

POINTS 131284 float

…

TRIANGLE_STRIPS 56141 418277

…

POINT_DATA 131284

…

NORMALS normals float

…

由 VTK 格式规范可知, 数据集的类型为多边形数据, 数据由 131284 个浮点类型的点组成, 而整个颅骨则是由这些点组成的 56141 个三角带构成. 提取数据中 WebGL 绘图所需的顶点坐标、法向量以及顶点之间的拼装(即三角带)信息, 将它们以数组形式存储到变

量中作为 WebGL 的输入数据, 就能够通过 WebGL 实现该颅骨在浏览器中的三维可视化. 数据的处理流程有以下几步:

① 逐行扫描获取到的 VTK 数据, 根据每行首个子字符串判断数据区域的数据类型;

② 若数据类型为顶点坐标或者法向量值, 则将读取到的顶点坐标和法向量分别存储到顶点数组和法向量数组中, 这样就能够获取颅骨上所有的顶点坐标及顶点对应的法向量值. 若数据类型为拓扑结构信息(颅骨文件中即三角带信息, 每个三角带信息描述了该三角带由定点数组中哪些顶点组成), 则存储拓扑结构信息;

③ 根据三角带信息从顶点数组中读取顶点坐标, 插入到新的顶点数组中, 就能够创建表示整个颅骨的顶点数组, 同时生成对应的法向量数组, 将定点数组、法向量数组作为 WebGL 的输入数组.

### 3.6 可视化模块

结合 WebGL 的绘图流程, 数据的可视化包括四个步骤, 如图 4 所示.

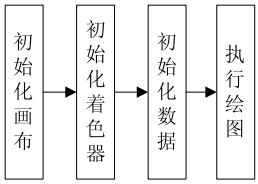



图 4　可视化流程

#### 3.6.1 初始化画布(canvas)

初始化画布来的目的是设置 WebGL 绘图环境并配置医学图像三维可视化的相关绘图参数, 其流程主要如下:

① 创建 canvas 并通过 getContext(“webgl”)获取 WebGL 绘图环境, 这样就能够使用 WebGL 的 API;

② 创建视图窗口, 定义其尺寸、在 canvas 元素中的位置并初始化视图窗口颜色;

③ 启用深度测试, 并设置深度测试参数;

④ 清空深度缓存以及颜色缓存, 开始图形绘制.

#### 3.6.2 初始化着色器

着色器决定了数据最终呈现在视图中的形式, 着色器采用 GLSL(图形库着色语言)编写. 初始化着色器有以下几个步骤:

① 编写顶点着色器. 顶点着色器的任务是在 GPU 中将输入的顶点坐标按一定规则转化成为屏幕中的坐标. 而这个规则是由模型视图矩阵以及投影矩阵决定的. 如图 5 所示, WebGL 将观察者的视觉空间定义为一个截头锥体, 将顶点数据所在的三维坐标系称为世界坐标系. 世界坐标系中顶点的变换矩阵称为模型矩阵, 将世界坐标系变换到以观察者为中心的视点坐标系的过程称为视图矩阵, 在计算机图形学中, 模型矩阵和视图矩阵统一在一起称为模型视图矩阵. 投影矩阵则决定了视点坐标系向屏幕坐标系的变换. 文献[6]介绍了变换的推导过程.

在应用过程中, 可以通过一些成熟的 JavaScript 图形库(如 Oak3D)提供的接口来定义变换矩阵, 作为着色器的输入矩阵;

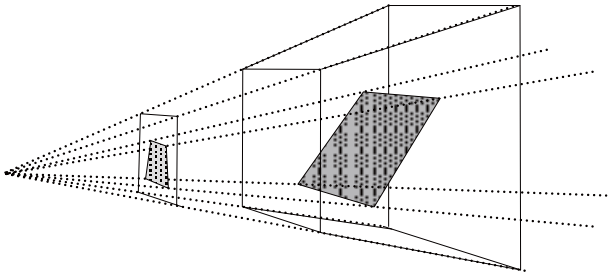
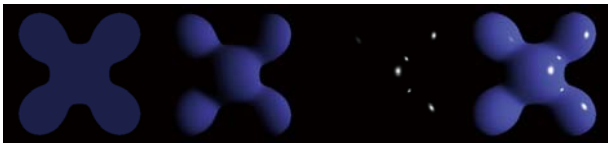


图 5　三维空间的透视投影

② 编写片元着色器. 经过顶点着色器后, 那些顶点位置上的像素则已经建立好了顶点, 为那些非顶点位置的像素调用片元着色器, 通过顶点颜色线性插值完成各个顶点间区域的填充过程. 在片元着色器的设计过程中, 除了顶点的颜色外, 还要模拟光照效果使二维图像更具三维立体效果. 根据冯氏反射模型[7], 将光线的作用分为三个部分, 如图 6 所示. 在顶点着色器中利用法向量和光线方向计算出漫反射强度、镜面反射强度, 再结合像素自身颜色, 最后加入环境光就能够得到最终的颜色.



环境光+漫反射光+镜面反射=冯氏反射

图 6　冯氏反射模型

③ 着色器以及着色程序的创建. 着色器是在 GPU 中运行的程序, 三维场景的运算, 在 JavaScript 编

程过程中, 通过调用 WebGL 的 API 实现数据与 GPU 的通信, 调用 gl.createShader 创建着色器对象, 调用 gl.shaderSource 以字符串形式读取着色器代码并通过 gl.compileShader 编译成为显卡中可以运行的形式, 再通过 gl.attachShader 将顶点着色器和片元着色器添加到一个由 gl.createProgram 创建的着色程序中, 通过 gl.linkProgram 完成顶点着色器和片元着色器的链接, 并通过 gl.useProgram 启用该着色程序, 完成着色器程序与当前场景的对应. 最后通过 gl.getAttribLocation 以及 gl.getUniformLocation 完成着色程序数据对象与着色器中参数的对应.

3.6.3 初始化数据

通过初始化数据将创建数组对象来存储数据处理模块中提取的颅骨顶点坐标、法向量, 该数组对象位于 GPU 中. 以顶点坐标为例, 首先调用 gl.createBuffer 创建顶点坐标数组对象, 然后调用 gl.bindBuffer 绑定当前数组对象, 最后调用 gl.bufferData 将提取的顶点坐标填充到数组对象中, 这样就完成了顶点坐标的初始化, 同理可以实现法向量数据对象的初始化. 同时也能够定义顶点的其它信息, 如颜色、纹理坐标等.

3.6.4 执行绘图

完成了着色器和数据的初始化后, 还要将数据绑定到着色程序中, 这样就能通过着色程序在 GPU 中完成图形绘制. 调用 gl.bindBuffer 绑定当前数据, 调用 gl.vertexAttribPointer 将数据与着色程序中的数据对象对应起来. 调用 gl.uniform 将模型视图矩阵以及投影矩阵等其它绘图参数与着色程序中的数据对象对应. 最后通过调用 gl.drawArrays 就能够通过着色程序在 GPU 中实现图形绘制, 颅骨的试验结果如图 7 所示.

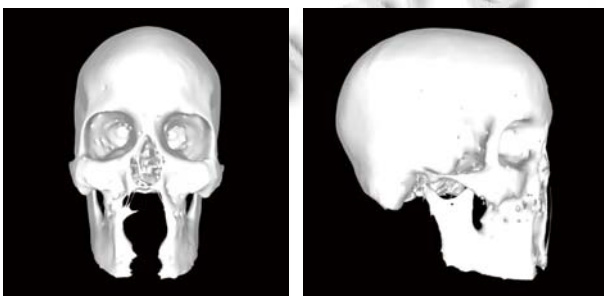


图 7 颅骨的三维显示

此外, 本文针对其它 VTK 数据进行了三维可视化, 如图 8, 图 9 所示.

肿瘤周围的神经纤维共包含 123751 个点及其组成的 757 条线. 膝盖模型共由 48 个 VTK 文件组成, 其中包各个骨骼、软骨、肌腱、神经、血管、肌肉等.

图 8 肿瘤周围神经纤维

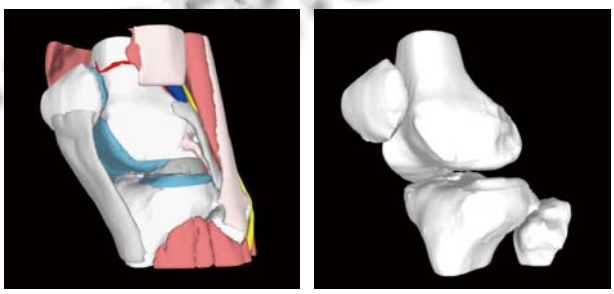


图 9 膝盖相关组织

### 3.7 简单体绘制的实现

WebGL 也能够通过医学图像的体数据信息进行直接体绘制. 体数据来源于 DICOM 序列文件中的图像信息, 故想要进行体绘制, 浏览器需要加载所有的 DICOM 文件并逐个分析提取其中的图像信息, 而医学影像(如 CT)往往都是由数百张甚至更多的断层扫描图像构成, 故直接使用 DICOM 文件进行体绘制是不便利的. 而将 DICOM 文件转换成 NRRD 文件后, 就能很好地直接读取体数据.

本文以头颅 CT 图像为例进行直接体绘制试验. 通过分析 NRRD 文件头对数据区域进行解析, 就能够获取三维体数据. 与面绘制方法不同, 体绘制技术并不生成如过渡图形(如三角带), 直接从三维体数据中产生二维图像. 本文采取了二维纹理映射的方式实现体数据的体绘制. 纹理数据对象创建流程如下:

① 读取体数据;

② 为每张切片建纹理数据对象;

③ 为每个纹理数据对象创建 RGBA 通道, 将体数据三维数组中对应的像素灰度值分别赋值给纹理数据 RGB 三通道, 第四通道为不透明度 A 定义为 255, 新的切片数据长度为原来的四倍;

④ 按 X、Y、Z 三个方向创建所有的纹理数据对象;

⑤ 计算视线方向(观察者指向数据坐标中心)与X、Y、Z 轴缩成的角度，当 X 方向角度最小时，则选用X方向纹理数据对象,Y方向角度最小时，选用Y方向纹理数据对象,Z 方向角度最小时，选用 Z 方向纹理数据对象.

确定纹理数据对象后，根据体数据的尺寸和间隔创建顶点数组和顶点纹理坐标，根据可视化的流程进行WebGL图形绘制，并在绘制过程中为片元添加纹理.实验结果如图 10 所示.

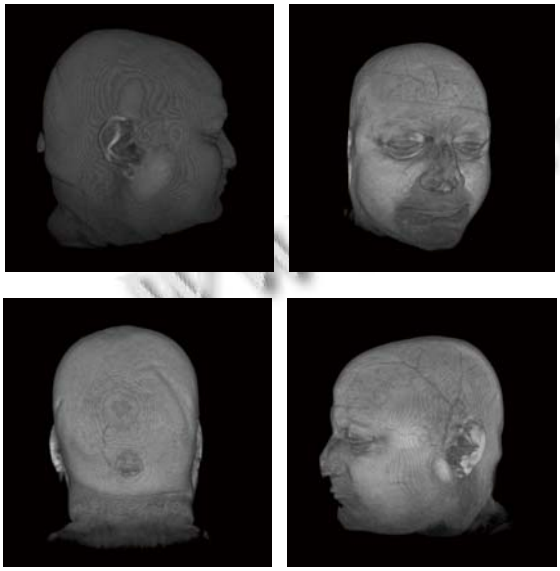


图 10 颅骨直接体绘制

### 3.8 实验结果分析

由面绘制和体绘制的实验结果可见，在VTK文件的支持下，浏览器能够很好地重现医学图像的面绘制，面绘制的效果取决于数据精度以及光照模型设计. 由图 10 可见，浏览器读取医学图像体数据后，能够实现直接体绘制，但限于本文使用二维纹理映射方法，各层纹理贴图之间间隔较为明显，与成熟的直接体绘制方法相比显示效果有较大的差距，此外，分别存储纹理序列会占用大量的内存空间.

## 4 结论及展望

本文基于 WebGL 从面绘制和直接体绘制两个方面实现了医学图像在浏览器中的三维可视化，可以看到，WebGL 使浏览器展现出了强大的三维绘图能力.现阶段关于WebGL的开发才刚刚兴起，微软并未参与WebGL 联盟，目前 IE 浏览器还不支持 WebGL. 随着浏览器的不断革新和WebGL自身的不断发展，在本地OpenGL 的基础之上的优秀方法都将在浏览器上得以实现.

## 参考文献

1 Kuleesha Y, Lin F. Biomedical Image Visualization as a Web Application. Journal of Next Generation Information Technology, 2012,3(1):17–27.

2 沈海戈,柯有安.医学体数据三维可视化方法的分类与评价.中国图像图形学报,2000,5(7):545–500.

3 谭文文,丁世勇,李桂英.基于 webGL 和 HTML5 的网页 3D 动画的设计与实现.电脑知识与技术,2011,07(28):6981–6983.

4 Munshi A.The OpenGL ES 2.0 programming guide. Massachusetts: Pearson Education,Inc,2009.

5 游丽贞,郭宇春,李纯喜.Ajax 引擎的原理和应用.微计算机信息,2006,22(6):205–207.

6 吴迪,黄文骞,王莹.计算机图形学中的透视投影变换矩阵.海军大连舰艇学院学报,2003,26(2):23–25.

7 Phong BT. Illumination for computer generated pictures. Communications of the ACM,1975,18(6):311–317.