基于 WebGL 的遥感影像滤镜模式设计与实现

Design and Implementation of Remote Sensing Image Filter Mode Based on WebGL

罗霄[1,2]，刘思言[1,2]，特日根[1,2,3]

（1.长光卫星技术有限公司，长春 130000；2.吉林省卫星遥感应用技术重点实验室，长春 130000；
3.中国科学院长春光学精密机械与物理研究所，长春 130000）

LUO Xiao[1,2], LIU Si-yan[1,2], TE Ri-gen[1,2,3]

(1.Chang Guang Satellite Technology Co., Ltd., Changchun 130000, China;
2.Jilin Key Laboratory of Satellite Remote Sensing Application Technology, Changchun 130000, China;
3.Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130000, China)

【摘 要】随着遥感卫星技术的不断发展，遥感数据被应用于不同的行业当中。论文基于对遥感影像进行非真实感绘制的问题，提出了一种使用 WebGL 的全新模式。通过结合 JavaScript 与 WebGL 技术，实现了通过前端生成遥感影像油画滤镜的业务逻辑。最后实验证明该模式可以有效利用用户端的 GPU 进行影像处理，既缩短了单幅影像的处理时间，又节省了网络传输所消耗的时间。因此，该模式适合为用户提供线上遥感影像处理服务。

【Abstract】With the continuous development of remote sensing satellite technology, remote sensing data is applied in different industries. Based on the problem of non photorealistic rendering of remote sensing image, this paper proposes a new mode of using WebGL. Through the combination of JavaScript and WebGL technology, this mode realizes the business logic of generating remote sensing image oil painting filter through the front end. Finally, experiments show that this mode can effectively use the GPU of user-end for image processing, which not only shortens the processing time of a single image, but also saves the time consumed in network transmission. Therefore, the mode is suitable for providing users with online remote sensing image processing service.

【关键词】WebGL；JavaScript；非真实感绘制；GPU；油画；滤镜

【Keywords】WebGL; JavaScript; non photorealistic rendering; GPU; oil painting; filter

【中图分类号】TP391　　【文献标志码】A　　【文章编号】1673-1069（2020）09-0194-03

1 引言

伴随计算机硬件性能发展和图像处理算法的不断进步，基于计算机算法实现的油画、水彩画、浮雕画等非真实感绘制（non photorealistic redering，NPR）的技术发展迅猛，其应用也越来越广泛[1,2]。近年来，随着遥感卫星的快速发展，遥感影像在数据量和分辨率上都有了很大的提高[3]。对遥感影像进行特殊风格的转换并最终生成装饰画也受到了卫星展馆、主题酒店、图书馆甚至个人用户的关注。遥感影像具有幅宽大、像素多[4]的特点，对遥感影像进行图像处理需要消耗大量的计算资源，因此，大多数算法都需要在后台进行计算。

随着 HTML5 的普及，前端可以承载客户端应用复杂的业务逻辑。WebGL[5]基于 OpenGL 扩展而来，将 OpenGL ES 2.0[6]与 JavaScript[7]结合到了一起，在不需要任何浏览器插件的情况下，完成基于 GPU 加速的图形绘制。通过这种处理方式，不但可以降低程序运行时对外部环境的依赖，也有效地减轻 CPU 的计算压力，提高了程序计算的并行程度。

本文通过使用 WebGL 技术在前端完成遥感影像油画滤镜处理的所有计算工作，将 B/S 端应用的计算压力从后台转移到了用户本地计算机。针对遥感影像的处理采用 WebGL 独有的硬件级加速方式[8]，降低程序运行时间。将该方法与传统 OpenCV 实现的方法进行对比，并从影像处理效率方面进行比较和分析，证明该方法的可行性及有效性。

2 油画滤镜的原理和算法分析

油画是一种色彩较重、色块十分明显的滤镜类型，其处理方式在一定程度上类似于对色彩信息的分类并聚集，处理的结果中会表现一些特定的色彩，而忽略和这些颜色相近的颜色[9]。根据油画算法的特点，设计算法步骤如下：

【作者简介】罗霄（1993-），男，吉林长春人，研究实习员，从事 WebGIS、人工智能研究。

①对每个像素 RGB 三个分量以不同的权值进行加权求和，按照式 (1) 得到像素的灰度值 $Q(x,y)$，其中 x、y 分别是像素在图像坐标系下的横、纵坐标。

$$Q(x,y)=0.299*(x,y)+0.578G(x,y)+0.114*B(x,y) \quad (1)$$

②设置模板半径为 r，以图像中每个像素为中心，按照式 (2) 统计其模板半径内的灰度直方图 Hist，其中 i 为像素的灰度值，card (x,y) 表示灰度值为 i 的元素个数。

$$\text{Hist}(i)=\text{card}(x,y)\,|\,Q(x,y)=i \quad (2)$$

③记录每个像素灰度直方图 Hist 的每个灰度值对象 i 的原始 RGB 值，并以 $C(i)$ 表示，其中 C 是一个三元素的向量，三元素分别表示 RGB 的值。$C(i)$ 的计算公式如式 (3) 所示。

$$C(i)=(R(x,y),G(x,y),B(x,y)\,|\,Q(x,y)=i \quad (3)$$

④统计灰度直方图 Hist 中最大 card 值对应灰度值对象 i 的原始 RGB 值，并将该值赋给中心点像素。

3 技术实现

3.1 WebGL 绘图的实现

WebGL 程序运行流程如图 1 所示，具体步骤如下：

①创建 DOM 作为容纳 WebGL 的容器，并根据容器获取 WebGL 的上下文。

②根据功能编写 GLSL 着色器程序，包括顶点着色器 (Vertex Shader) 与片段着色器 (Fragment Shader)。顶点着色器负责控制将顶点的大小、位置等属性输入屏幕上，片段着色器负责控制顶点的颜色信息。

③编译着色器程序，并将着色器程序绑定给 WebGL 上下文，链接生成 WebGL program。

④将数组及矩阵绑定到着色器中，WebGL 中一共有两种变量，顶点属性变量 (Attribute) 与常量 (Uniform)。其中，顶点属性变量传递那些与顶点相关的数据，包括位置、颜色、纹理信息等；而常量则处理那些与顶点数据都相同的数据，如环境光、投影矩阵、变换矩阵。

⑤在创建着色器程序并绑定数据之后，调用 WebGL 绘制接口进行绘制操作。

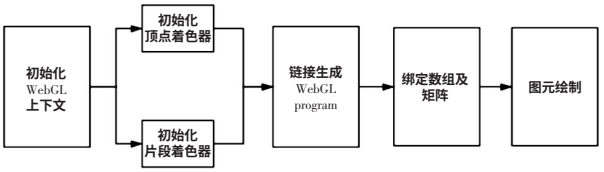


图 1 WebGL 绘制流程

3.2 油画滤镜的实现过程

纹素定义 纹素 (texture pixel，Texel) 是纹理元素的简称，它是构成计算机图形纹理空间的最小基本单位。

整体的油画算法实现流程如图 2 所示，主要包含如下几个步骤：

①在标准 WebGL 坐标系下，计算影像一个像素所占的大小。

②提取影像中一个像素对应纹素的 RGB 值。

③根据定义的加权求和公式计算每个纹素的灰度值。

④设置纹素的模板半径为 r，同时，以纹素为中心点，以 r 为计算半径，统计其直方图。

⑤根据统计结果，以直方图最大值对应 RGB 值作为纹素 RGB 值。

⑥重复上述四步，直到计算全部纹素的 RGB 值后结束该过程。

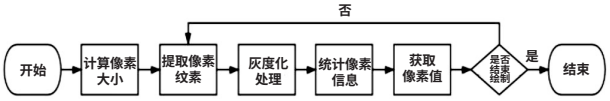


图 2 油画算法实现流程

4 实验与结果

4.1 软硬件平台

实验相关硬件条件如表 1 所示。

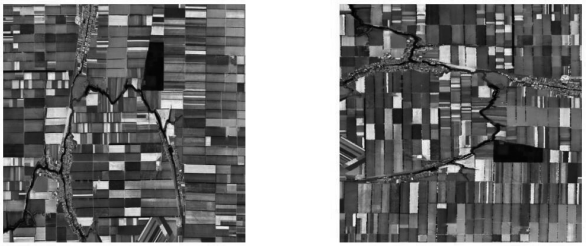
表 1 实验环境介绍

环境	版本/类型
CPU 处理器	Intel (R) Core (TM) i7-8700 CPU @3.20GHZ 3.19GHZ
显卡	NVIDIA GeFroce GT 1030
操作系统	Windows 10 专业版 64 位
编程环境	JetBrains WebStorm 2019.2 x64
浏览器运行环境	Google Chrome 80.0.3987.132

4.2 浏览器性能分析

本文实验中采用吉林一号拍摄的俄罗斯梅克列塔农田遥感影像为输入，图片大小为 4762*2678 像素，水平分辨率与垂直分辨率均为 72dpi，位深度为 24，原始输入影像如图 3a 所示。设置模板半径分为 4、8、16，其处理结果如图 3b、3c、3d 所示。通过实验结果可以发现，使用 WebGL 进行前端图像油画处理可以得到人们视觉感知中模糊效果，图像的纹理轮廓也得到了保留。由于显卡性能过低，当模板半径大于 16 时，算法计算量过大，造成浏览器崩溃。本文对比了选取不同模板半径算法在 Chrome 浏览器上的时间消耗，结果如图 4 所示。

由图 4 可知，使用 WebGL 对影像进行油画处理，不仅实现了线上处理模式，同时，将处理压力分散到用户端，可以有效地提高服务响应效率。结合图 3 的实验结果对比可以发现，模板半径越大，处理时间越长，图像越模糊，边缘细节越少。但当模板半径为 8 时，图像边界模糊程度适中，较为敏感的小像素块能够完整地保留下来。因此，当算法模板半径为 8



a.原图　　b.WebGL 半径 4

c.WebGL 半径 8　　d.WebGL 半径 16

图 3　原始输入图像与油画算法生成图像

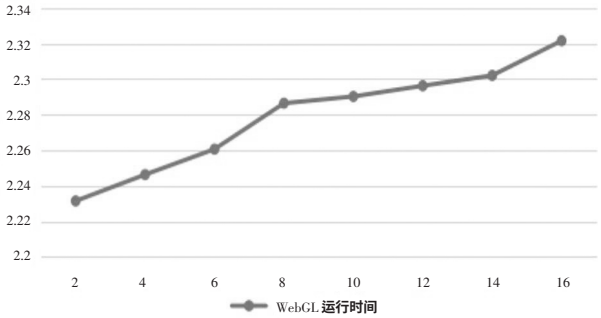
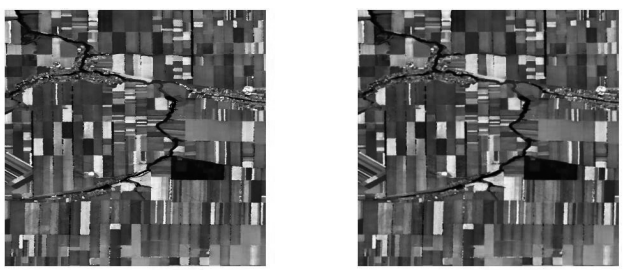


图 4　WebGL 油画算法运行时间

时，既保证了影像的处理效果，同时，响应效率能够满足线上业务需求。

4.3 与 OpenCV 对比实验

本文同时使用传统 OpenCV 在后端实现相同的油画算法并与 WebGL 进行对比，得到的结果图像对比如图 5 所示，时间对比如图 6 所示。从对比结果可以发现，两种处理方式得到的最终图像几乎相同，WebGL 技术使用 GPU 对算法进行加速，因此，在运行效率上较 OpenCV 有明显提升。选取半径越大，计算量越大，提升效果越明显。同时，使用 WebGL 技术，算法在用户本地计算机进行计算，减少了数据传输过程造成的时间消耗，节约了网络带宽。



a.WebGL 半径 8　　b.OpenCV 半径 8

图 5　OpenCV 与 WebGL 生成图像

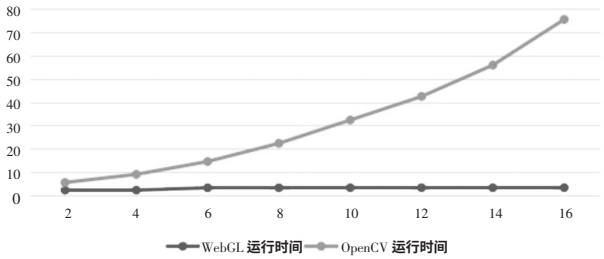


图 6　OpenCV 与 WebGL 油画算法运行时间对比

5 结语与展望

本文针对遥感影像非真实感绘制处理速度问题，提出了使用 WebGL 在前端完成图像处理任务，并利用该方法实现了油画滤镜算法，并通过遥感影像进行了实验验证。实验结果表明，通过引入 WebGL 技术，将遥感影像转换成油画与使用 OpenCV 方法效果相同，同时，因为 WebGL 自带的 GPU 加速，油画算法的实现速度较 OpenCV 有了明显的提高。通过 WebGL 技术来实现遥感影像处理工作，可以将数据处理工作从服务器端转移到用户端，可以分散影像处理压力，更适合于线上服务模式。

今后，我们将在保证应用运行速度、准确率的前提下，继续基于 WebGL 离屏渲染特性研究如水彩画、铅笔画等更加复杂的滤镜算法，同时，将尝试在视频处理功能上应用 WebGL 技术[10]在前端进行加速。

【参考文献】

【1】刘美芳,石春菊.浅谈计算机图像处理技术的发展[J].电脑知识与技术:学术交流,2016,12(32):241-242+250.

【2】卢少平,张松海.基于视觉重要性的图像油画风格化绘制算法[J].计算机辅助设计与图形学学报,2010,22(07):1120-1125.

【3】于梦华,王双亭,李英成,等.畸变差改正算法 OpenCL 并行加速研究[J].遥感信息,2019,34(3):88-92.

【4】鲁恒,李永树,何敬,等.无人机低空遥感影像数据的获取与处理[J].测绘工程,2011,20(01):51-54.

【5】Kouichi Matsuda,Rodger Lea.WebGL 编程指南[M].北京:电子工业出版社,2014.

【6】Aaftab Munshi,Dan Ginsburg,Dave Shreiner.OpenGL ES 2.0 Programming Guide[M].Hoboken:Addison-Wesley Professional,2008.

【7】Douglas Crockford .JavaScript:The Good Parts[M].Sunnyvale:Yahoo Press,2008.

【8】王星捷,卫守林.WebGL 技术的三维 WebGIS 平台研究与应用[J].遥感信息,2019,34(03):134-138.

【9】付庆军.利用 Photoshop 实现摄影作品的水彩画效果[J].照相机,2009(02):66-69.

【10】余亦豪,谭冲,周忠,等.虚实融合的实景视频 WebGIS 系统[J].系统仿真学报,2018,30(07):151-158.