1. 简介

项目由来:在一个地理信息系统中进行地图的显示,底层的底图使用PDF渲染,原本的设计是调用现成的foxit渲染库实现PDF的渲染,但项目设计之初没有考虑到展示的时候要切换坐标系(实现高斯投影,外部墨卡托投影等转换),而现有的PDF库只支持渲染PDF文件,无法在渲染之前实现对矢量数据的投影转换



结果:红色箭头所示方式转换出的高斯投影,实质上仅仅是对位图进行了拉伸,结果是错误的,因为投影方式的变化对于栅格数据而言并不是均匀的,所以单纯的拉伸无法达到效果

解决方法:绿色箭头所示方式,使用引擎将矢量数据从PDF中提取出来,并对其进行投影转换,之后再渲染生成正确的位图

1. PDF文件内容流介绍

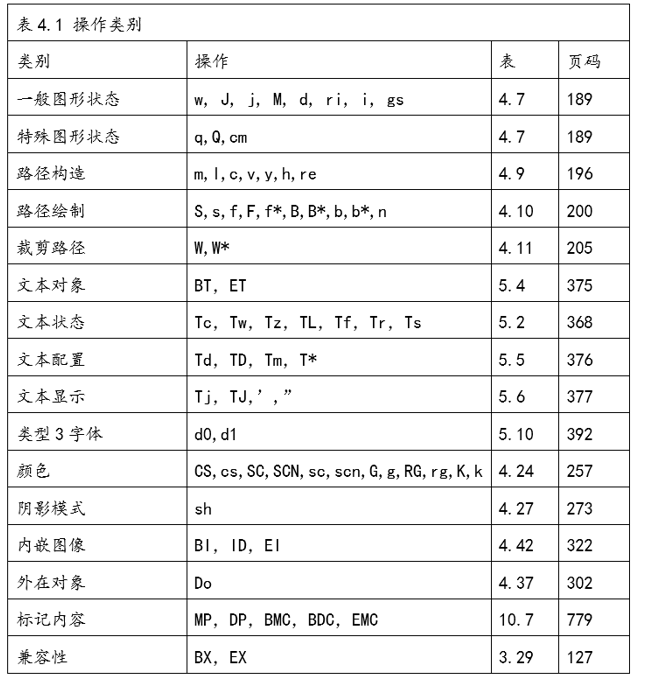
PDF文件具有严谨的文档结构,大体分为四部分:版本号,对象,交叉引用表,入口点

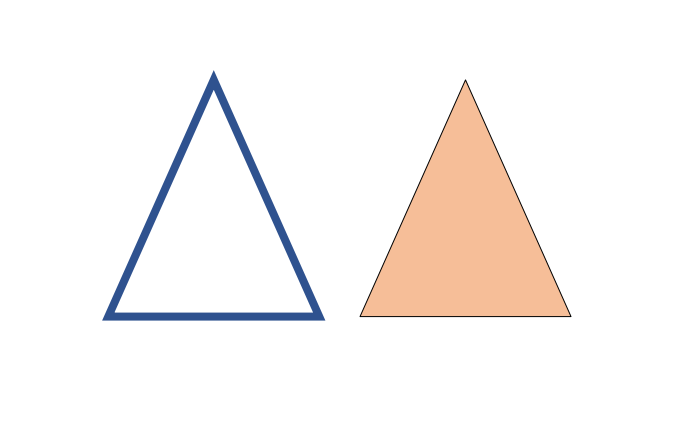
我们要处理的矢量数据隐藏在对象中,PDF文档采用对象树的结构存放信息,下图简要的展现了文档结构

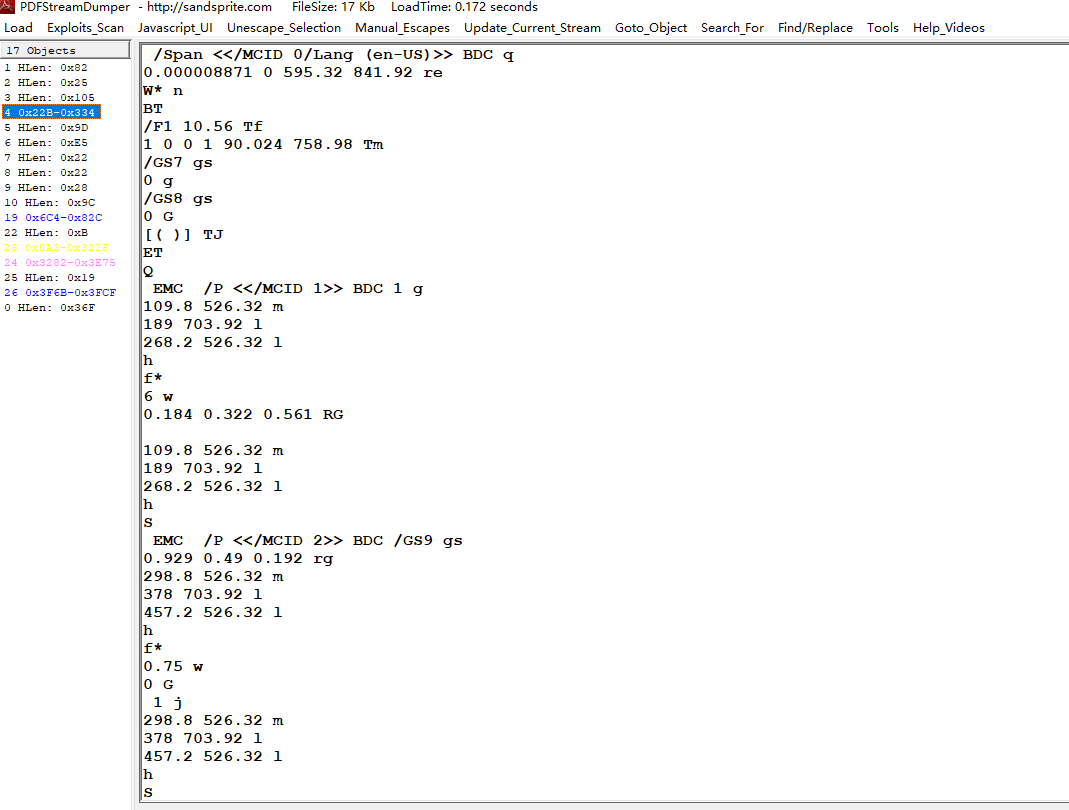


我们要提取的数据存放在流对象中

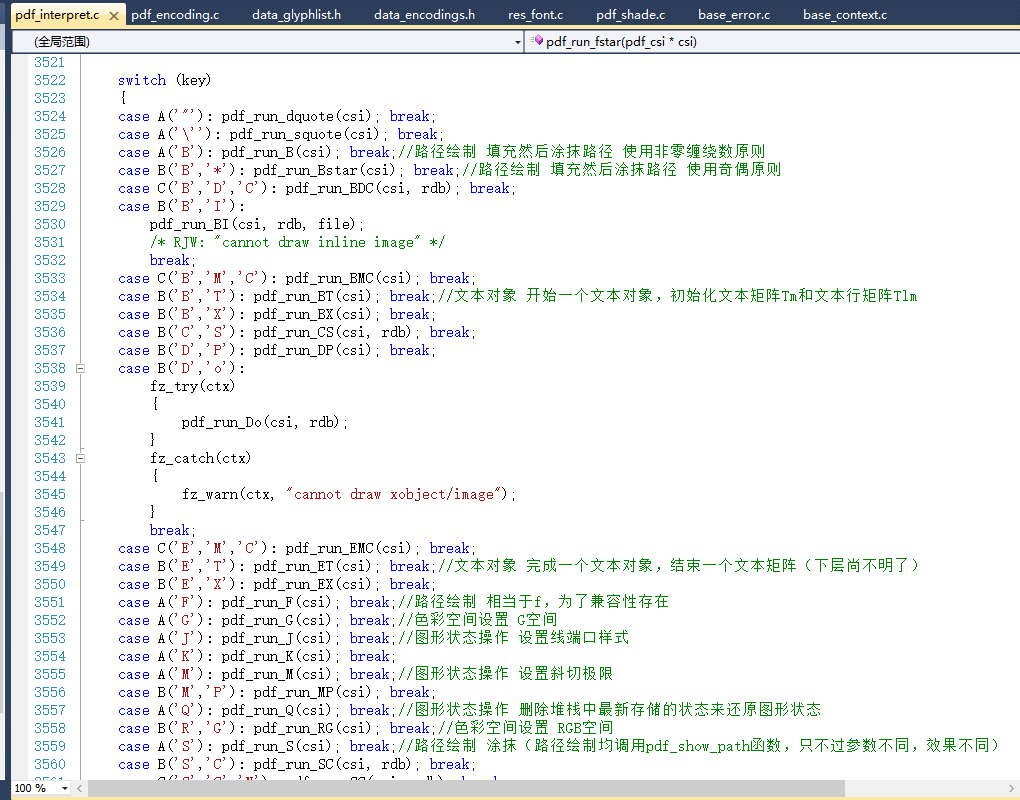
下表是PDF内容流的命令总表,PDF采用绘制命令的方式存储矢量数据



我们看到如下图所示的PDF简单图形,借助PDF Stream Dumper软件,我们可以看到这两个三角形在PDF内部的命令描述结构如下所示



可以看到,PDF里面是使用绘制命令序列的方式来实现对矢量数据的存储的,显然此种结构并不能够满足测绘领域的需求,所以我研究了MuPDF库的源码,核心部分如下图:



可以看到,其处理内容流的核心部分是一个庞大的switch对各个命令进行解析,我在这些解析命令中添加了代码,将提取到的指令流转化为点线面等结构体,并按照测绘所要求将其封装为C++对象

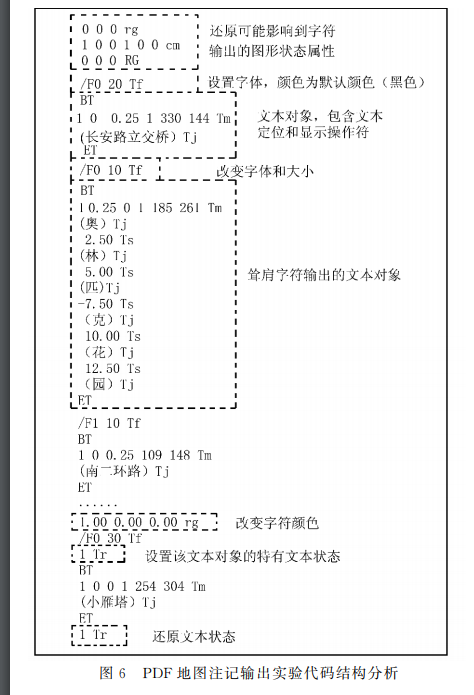
1. 引擎数据组织形式



总体而言,Ipe\_PdfDocument对象对应一个PDF文件,调用构造函数可以将PDF文件解析为我们设计的内部结构

Ipe\_PdfDocument对象: 持有一个Ipe\_Page的序列(当然对于地图我们就默认为只有一页)

Ipe\_Page对象: 对应PDF对象树中页对象的内容,包括三类:文本对象(PDF中我们看到的文字实际上有两种,一种是使用矢量数据来画出一个字,一种是给定坐标使用PDF指定的编码来写文字,如下图所示,此处的文本对象指第二种),XObject(一般是位图;PS:PDF虽然是矢量数据文件,但也可以通过XObject对象存储栅格数据,比如位图),以及我们重点关注的矢量数据



Ipe\_PDFElement对象: 是上述三种页对象的基类,我主要实现了对矢量数据的解析,其他两种对象在设计时留出类,暂未实现

Ipe\_PdfStack对象: 对应上文所说的矢量数据,图形状态栈这个名字取自于PDF reference官方文档,表示图形状态栈中的所有矢量数据共享一样的颜色,转置矩阵,裁剪路径等诸多属性

Ipe\_PdfPath对象: 是图形状态栈中的路径的集合,由一系列Ipe\_PdfPlane对象组成, Ipe\_PdfPlane对象表示一条路径,有参数来识别其是否封闭(PS:这样做的意义在于对封闭图形有时要进行填充颜色的操作)

Ipe\_PDFPlane对象: 一条一般意义上的路径,可能由若干直线与若干贝塞尔曲线序列组成

Ipe\_GraphicCell对象: 直线集与曲线集的基类对象

Ipe\_Lines对象: 连续的折线集,由一系列二维坐标点组成

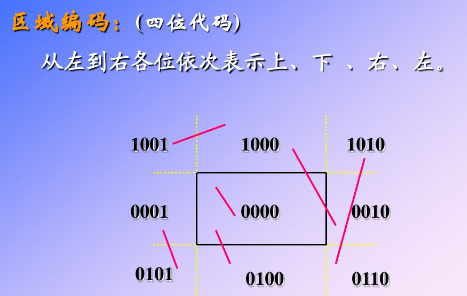
Ipe\_Bazeir对象: 连续的曲线集, 由一系列二维坐标点组成

Ipe\_Point2D对象: 二维坐标点x,y

1. 裁剪功能

依照测绘所要求,对矢量数据进行了裁剪的操作,具体实现:

直线的裁剪使用了编码裁剪法



曲线裁剪方法:

从起点开始进行遍历 (状态0)

若四个控制点都在矩形内部,则不进行修改;

若两个端点存在不在矩形内部的,则说明是跨越的线,保留这条线,并记住这个状态(状态1)

状态1下,下一部分若是完全在外部,则删除此条线,继续下去遍历,并维持状态

直到到达再次进入裁剪区域或结束,将状态置为零

若存在控制点不在裁剪区域内但是端点在裁剪区域内的,代入公式进行测试(取一

些点进行测试)之后是否在裁剪区域内的处理同上

两个端点的优先级大于两个控制点

面裁剪方法:

多边形/曲线部分不与裁剪框相交的面:从起点遍历,使用Sutherland\_Hodgman算法,将多边形与边界结合

多边形如果全部为直线组成,可以这样裁剪

如果多边形部分为曲线结构:遍历所有点只要该面存在部分在裁剪区域内,就进行全部保留

1. 效果

对真实地图数据进行解析后生成的SVG文件如图所示,与原图对比无误



裁剪之后效果



1. 总结

目前已经使用测绘所提供的军用地图进行了测试,解析出的数据基本无误(限于PDF本身内容流的松散结构可能有少数情况未能考虑到),也实现了裁剪功能,生成SVG格式文件的目的在于先确保数据正确,

待实现:将测绘所提供的投影变换方法集成到引擎中,以及渲染功能的实现

后续可能的开发:解析PDF中的文本对象以及栅格数据(XObject),将引擎处理的PDF文件由单纯的矢量PDF文件扩展到一般的PDF文件