

Raster Data Coding

栅格数据编码



中国人民解放军战略支援部队 信息工程大学—曹一冰讲师

PLA Strategic Support Force Information Engineering University——Lecturer, Yibing Cao

- 主要研究方向：地理空间建模、地理信息系统平台及应用技术研究。
- 获省部级科技进步二等奖1项、三等奖1项。获第五届全国高校GIS青年教师讲课比赛一等奖，指导第九届全国大学生GIS应用技能大赛获特等奖。
- 近五年来，主持国家重点研发计划项目子课题2项，发表学术论文10篇，受理国家发明专利9项，获得计算机软件著作权7项。

栅格数据编码

Raster Data Coding



栅格数据常用的编码方式：

- 直接栅格编码

- 游程长度编码

- 常规四叉树编码

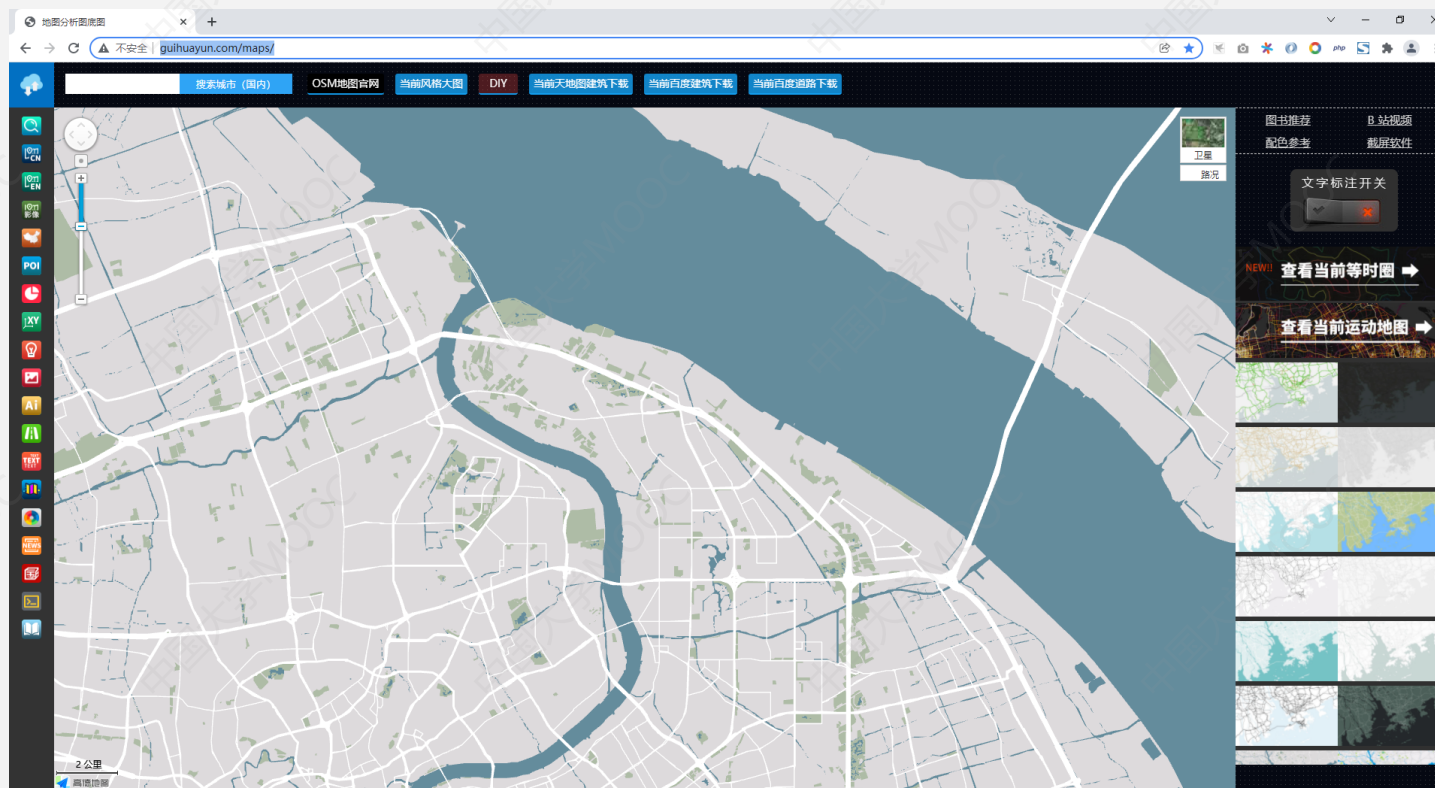
- 十进制线性四叉树编码

栅格数据编码

Raster Data Coding



比如打开“规划云”的首页。



<http://guihuayun.com/maps/>

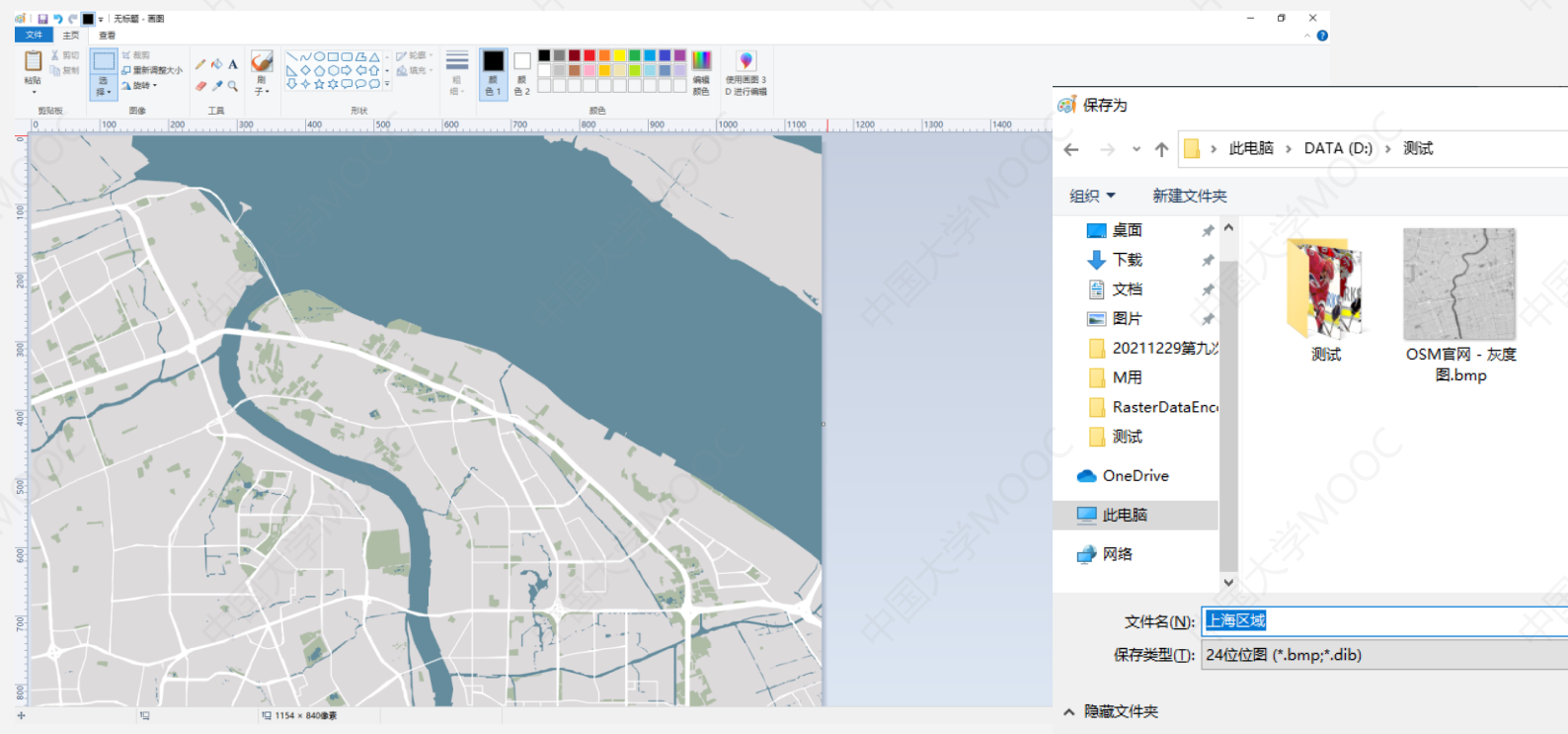


栅格数据编码

Raster Data Coding



采用系统的windows+shift+s快捷键，截取当前视口中的地图数据，在画图工具中，将栅格数据保存到本地路径中，保存的格式为24位位图。

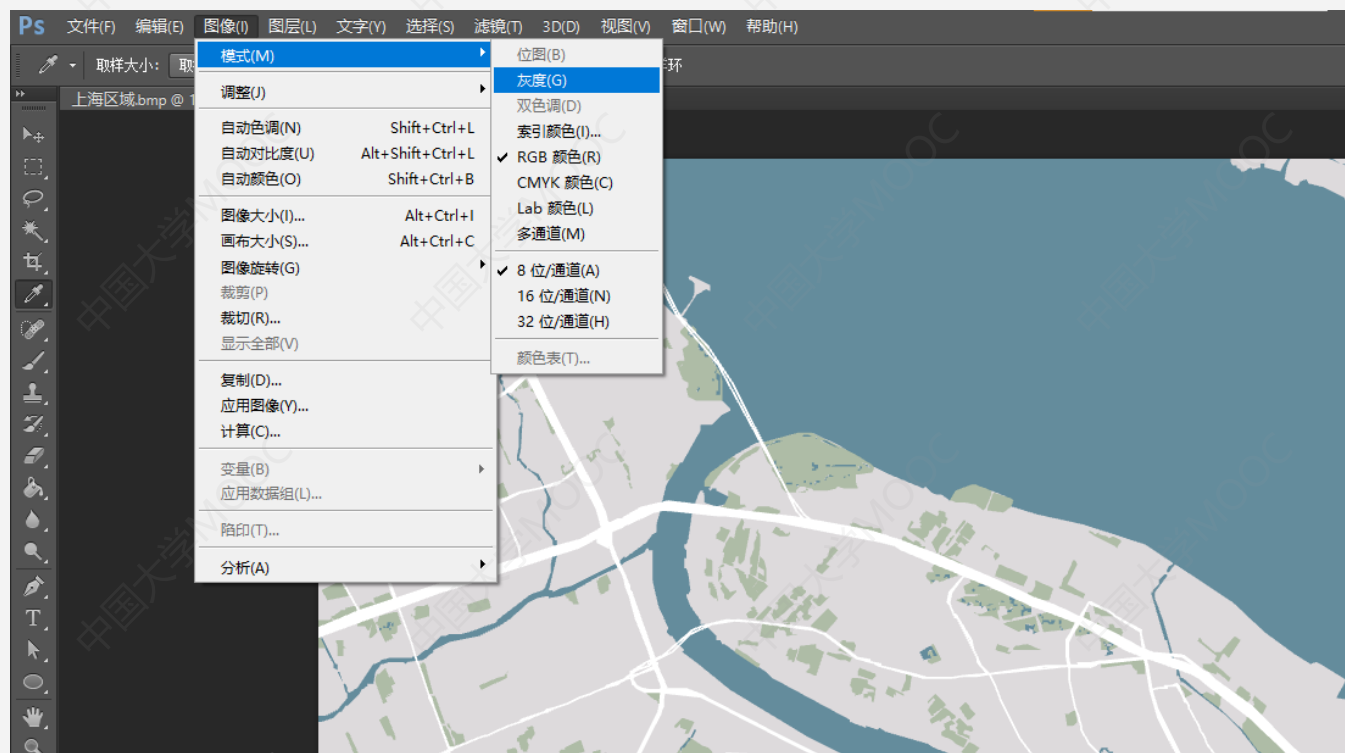


栅格数据编码

Raster Data Coding



用photoshop软件将24位的彩色位图转换为8位的灰度图。图像—模式—灰度，将地图转换为灰度图。



栅格数据编码

Raster Data Coding



文件内容包括四个方面的信息：

- 文件头

- 信息头

- 颜色表

- 位图数据

博文网址→

<https://www.cnblogs.com/wainiwann/p/7086844.html>



栅格数据编码

Raster Data Coding



这里从55-58四个字节是00 00 00 00，代表第一个颜色的RGBA都为0。

```
D:\测试\上海区域.bmp - Notepad++
文件(F) 编辑(E) 搜索(S) 格式(O) 编码(C) 语言(L) 设置(S) 工具(T) 宏(A) 运行(R) 插件(P) 窗口(W) 2
VFP_3SPForm.xml VFP_3SPForm_xiangshi.xml OSA.xml cshpit.xml horecompass.xml VFP_Main07.xml dingpinogram.py CSW窗口 - 宏图图.bmp VLF.bmp rasterest.bmp 上海区域.bmp

Address 0 1 2 3 4 5 6 7 8 9 a b c d e f Dump
00000000 42 4d 58 d5 0e 00 00 00 00 00 36 04 00 00 28 00 BMX?.....6...(.
00000010 00 00 82 04 00 00 48 03 00 00 01 00 08 00 00 00 ..?.H.....
00000020 00 00 22 d1 0e 00 12 0b 00 00 12 0b 00 00 00 00 .."?.....
00000030 00 00 00 00 00 00 00 00 00 00 01 01 00 02 02 ..
00000040 02 00 03 03 03 00 04 04 04 00 05 05 05 00 06 06 .....
00000050 06 00 07 07 07 00 08 08 08 00 09 09 09 00 0a 0a .....
00000060 0a 00 0b 0b 0b 00 0c 0c 0c 00 0d 0d 0d 00 0e 0e .....
00000070 0e 00 0f 0f 0f 00 10 10 10 00 11 11 11 00 12 12 .....
00000080 12 00 13 13 13 00 14 14 14 00 15 15 15 00 16 16 .....
00000090 16 00 17 17 17 00 18 18 18 00 19 19 19 00 1a 1a .....
000000a0 1a 00 1b 1b 1b 00 1c 1c 1c 00 1d 1d 1d 00 1e 1e .....
000000b0 1e 00 1f 1f 1f 00 20 20 20 00 21 21 21 00 22 22 .....
000000c0 22 00 23 23 23 00 24 24 24 00 25 25 25 00 26 26 ".###.$$.%&
000000d0 26 00 27 27 27 00 28 28 28 00 29 29 29 00 2a 2a &.''.(((.)).**
000000e0 2a 00 2b 2b 2b 00 2c 2c 2c 00 2d 2d 2d 00 2e 2e *.+++,---...
000000f0 2e 00 2f 2f 2f 00 30 30 30 00 31 31 31 00 32 32 .///.000.111.22
00000100 32 00 33 33 33 00 34 34 34 00 35 35 35 00 36 36 2.333.444.555.66
00000110 36 00 37 37 37 00 38 38 38 00 39 39 39 00 3a 3a 6.777.888.999.:
00000120 3a 00 3b 3b 3b 00 3c 3c 3c 00 3d 3d 3d 00 3e 3e :.;;.<<<==>
00000130 3e 00 3f 3f 3f 00 40 40 40 00 41 41 41 00 42 42 >.???.@@@.AAA.BB
00000140 42 00 43 43 43 00 44 44 44 00 45 45 45 00 46 46 B.CCC.DDD.EEE.FF
00000150 46 00 47 47 47 00 48 48 48 00 49 49 49 00 4a 4a F.GGG.HHH.III.JJ
00000160 4a 00 4b 4b 4b 00 4c 4c 4c 00 4d 4d 4d 00 4e 4e J.KKK.LLL.MMM.NN
00000170 4e 00 4f 4f 4f 00 50 50 50 00 51 51 51 00 52 52 N.OOO.PPP.QQQ.RR
00000180 52 00 53 53 53 00 54 54 54 00 55 55 55 00 56 56 R.SSS.TTT.UUU.VV
00000190 56 00 57 57 57 00 58 58 58 00 59 59 59 00 5a 5a V.WWW.XXX.YYY.ZZ
000001a0 5a 00 5b 5b 5b 00 5c 5c 5c 00 5d 5d 5d 00 5e 5e Z.[[.\\].]^
000001b0 5e 00 5f 5f 5f 00 60 60 60 00 61 61 61 00 62 62 ^._``.aaa.bb
000001c0 62 00 63 63 63 00 64 64 64 00 65 65 65 00 66 66 b.ccc.ddd.eee.ff
000001d0 66 00 67 67 67 00 68 68 68 00 69 69 69 00 6a 6a f.ggg.hhh.iii.jj
000001e0 6a 00 6b 6b 6b 00 6c 6c 6c 00 6d 6d 6d 00 6e 6e j.kkk.lll.mmm.nn
000001f0 6e 00 6f 6f 6f 00 70 70 70 00 71 71 71 00 72 72 n.ooo.ppp.qqq.rr
00000200 72 00 73 73 73 00 74 74 74 00 75 75 75 00 76 76 r.sss.ttt.uuu.vv

Hex Edit View nb char : 972120 Ln : 1 Col : 1 Sel : 0 Hex BigEndian INS
```


栅格数据编码

Raster Data Coding



BMP格式详解<转>

BMP格式详解

BMP文件格式详解 (BMP file format)

BMP文件格式，又称为Bitmap（位图）或是DIB(Device-Independent Device，设备无关位图)，是Windows系统中广泛使用的图像文件格式。由于它可以不作任何变换地保存图像像素域的数据，因此成为我们取得RAW数据的重要来源。Windows的图形用户界面（graphical user interfaces）也在它的内建图像子系统GDI中对BMP格式提供了支持。

下面以Notepad++为分析工具，结合Windows的位图数据结构对BMP文件格式进行一个深度的剖析。

BMP文件的数据按照从文件头开始的先后顺序分为四个部分：

- Ø **bmp文件头(bitmap file header)**：提供文件的格式、大小等信息
- Ø **位图信息头(bitmap information)**：提供图像数据的尺寸、位平面数、压缩方式、颜色索引等信息
- Ø **调色板(color palette)**：可选，如使用索引来表示图像，调色板就是索引与其对应的颜色的映射表
- Ø **位图数据(bitmap data)**：就是图像数据啦^_^

下面结合Windows结构体的定义，通过一个表来分析这四个部分。

数据段名称	对应的Windows结构体定义	大小(Byte)
bmp文件头	BITMAPFILEHEADER	14
位图信息头	BITMAPINFOHEADER	40
调色板		由颜色索引数决定
位图数据		由图像尺寸决定

我们一般见到的图像以24位图像为主，即R、G、B三种颜色各用8个bit来表示，这样的图像我们称为真彩色，这种情况下是不需要调色板的，也就是所位图信息头后面紧跟的就是位图数据了。因此，我们常见到有这样一种说法：位图文件从文件头开始偏移54个字节就是位图数据了，这其实说的是24或32位图的情况。这也就解释了我们按照这种程序写出来的程序为什么对某些位图文件没用了。

2021年12月

2021年12月(1)

2021年5月(1)

2020年10月(1)

2020年8月(1)

2020年5月(1)

2020年3月(1)

2020年2月(1)

2020年1月(2)

2019年11月(7)

2019年10月(14)

随笔 - 630 文章 - 0 评论 - 160 阅读 - 267万

昵称：阿波伦
园龄：10年11个月
粉丝：276
关注：144
+加关注

随笔档案 (630)

Raster Data Codingg

[illegible]

栅格数据编码

Raster Data Coding



`m_hbm`用于存储位图句柄，`m_dib`用于存储位图数据，`m_memdc`用于存储当前设备上下文。

```
29     BOOL EndOutline(INT pos), // 判断位图数据是否已经写完
30
31     private:
32         // 图片必须是标准的Windows位图
33         HBITMAP m_hbm; // 位图句柄
34         DIBSECTION m_dib; // 位图的数据
35         HDC m_memdc; // 当前设备上下文
36     };
```


栅格数据编码

Raster Data Coding



LoadBitmapFile函数用于加载当前位图，采用的是Windows底层相关函数。LoadImage获取位图句柄，SelectObject获取当前设备上下文，GetObject获取位图数据。

```
17 //加载当前位图数据
18 BOOL RLE_Encoding::LoadBitmapFile(string filepath)
19 {
20     //删除原有位图句柄
21     if (m_hbm != NULL)
22     {
23         DeleteObject(m_hbm);
24     }
25
26     m_hbm = (HBITMAP)LoadImage(NULL, filepath.c_str(), IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE | LR_CREATEDIBSECTION);
27
28     //文件打开错误
29     if (m_hbm == NULL)
30         return FALSE;
31
32     //获取当前设备上下文
33     SelectObject(m_memdc, m_hbm);
34     //获取位图数据
35     GetObject(m_hbm, sizeof(DIBSECTION), &m_dib);
36
37     return TRUE;
38 }
```

栅格数据编码

Raster Data Coding



SaveBitmapFile函数用于保存压缩后的位图文件。
CompressInRLE8函数用于实现位图数据的压缩，GetDIBColorTable用于获取位图的颜色表。



- 创建一个空的BMP文件。
- 将文件头、信息头、颜色表和位图数据存储到文件中，实现压缩后文件的存储。

```
56 //保存编码压缩后的位图
57 BOOL RLE_Encoding::SaveBitmapFile(string filename)
58 {
59     //RLE目前只支持256色位图或者灰度图
60     if ( m_dib.dsBmih.biBitCount != 8)
61         return FALSE;
62
63     int size;
64     QByteArray RLE8_bitmaps_bits; //定义字节数组,用于存储压缩后的位图数据
65     RLE8_bitmaps_bits.fill(char(0), 4194304); //初始化为4MB,所有字节位都为0
66
67     //使用Run-Length-Encoding(8 bit)压缩位图数据
68     CompressInRLE8( (BYTE*)m_dib.dsBm.bmBits, RLE8_bitmaps_bits, size);
69
70     _8bit_BITMAPINFO _8bit_bmi;
71     _8bit_bmi.bmi.bmiHeader = m_dib.dsBmih; //拷贝完整的BITMAPINFOHEADER结构信息
72     _8bit_bmi.bmi.bmiHeader.biCompression = BI_RLE8; //设置为8位RLE编码
73     _8bit_bmi.bmi.bmiHeader.biSizeImage = size; //压缩后的位图大小
74
75     //从当前设备上下文中获取颜色表信息,存储在_8bit_bmi中
76     GetDIBColorTable(m_memdc, 0, _8bit_bmi.bmi.bmiHeader.biClrUsed, _8bit_bmi.bmi.bmiColors );
77
78     HANDLE hf; //定义文件句柄
79     BITMAPFILEHEADER hdr; //定义位图文件头
80     PBITMAPINFOHEADER pbih; //定义位图信息头
81     DWORD dwTmp;
82     //强制转换获取位图信息头
83     pbih = (PBITMAPINFOHEADER) &_8bit_bmi;
84
85     //创建另存的位图文件
```

栅格数据编码

Raster Data Coding



EndOfLine函数主要用于判断当前位置是不是每一行的最后一位。

```
145 //判断当前位置是不是每一行的最后一位
146 BOOL RLE_Encoding::EndOfLine(int pos)
147 {
148     if ( 0 != m_dib.dsBmih.biWidth % 4 )
149     {
150         int scanline_width = m_dib.dsBmih.biWidth;
151         scanline_width = scanline_width + ( 4 - (scanline_width%4));
152         pos %= scanline_width;
153     }
154     return 0 == ((pos+1)% m_dib.dsBmih.biWidth);
155 }
156 }
```


栅格数据编码

Raster Data Coding



CompressInRLE8函数实现具体的游程长度编码功能。

```
158 void RLE_Encoding::CompressInRLE8(BYTE* pSrcBits, QByteArray& pRLEBits, int& RLE_size)
159 {
160     int line;
161     int src_index = 0, dst_index = 0, counter, i;
162
163     //RLE8模式中，每个像素占8位
164     for ( line = 0; line < m_dib.dsBmih.biHeight; line++)
165     {
166
167         state_start: //数据块的开始
168
169         if ( EndOfLine(src_index)) //是不是每一行的最后一个像素
170         {
171             pRLEBits[dst_index++] = 1;
172             pRLEBits[dst_index++] = pSrcBits[src_index];
173
174             src_index++;
175             goto end_of_line;
176         }
177
178         //现在块长度至少为2，提前决定下一个状态，接着两个像素是相同的，输入压缩的模式
179         if (pSrcBits[src_index] == pSrcBits[src_index+1])
180             goto state_compress;
181
182         if ( EndOfLine(src_index+1)) //每一行的最后两个像素
183         {
184             //这两个像素不相同
185             pRLEBits[dst_index++] = 1;
186             pRLEBits[dst_index++] = pSrcBits[src_index++];
187             pRLEBits[dst_index++] = 1;
```

栅格数据编码

Raster Data Coding



运行程序，打开系统主界面。



栅格数据编码

Raster Data Coding



通过左侧目录，或者“打开文件夹”工具，定位到刚才位图的存储路径，打开位图文件。

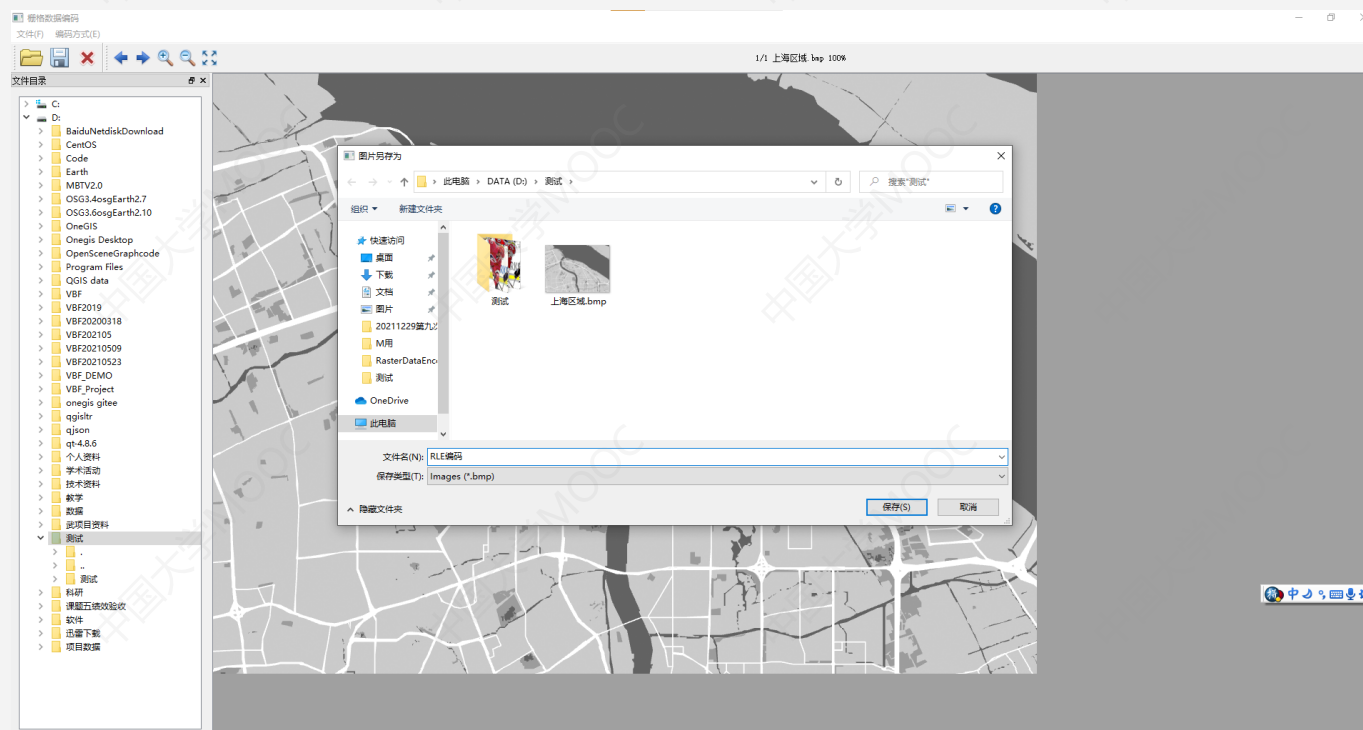


栅格数据编码

Raster Data Codingg



点击工具栏中的“另存为”按钮，在弹出的对话框中选择压缩后位图的存储路径，输入文件名，点击保存，完成位图的游程长度编码。

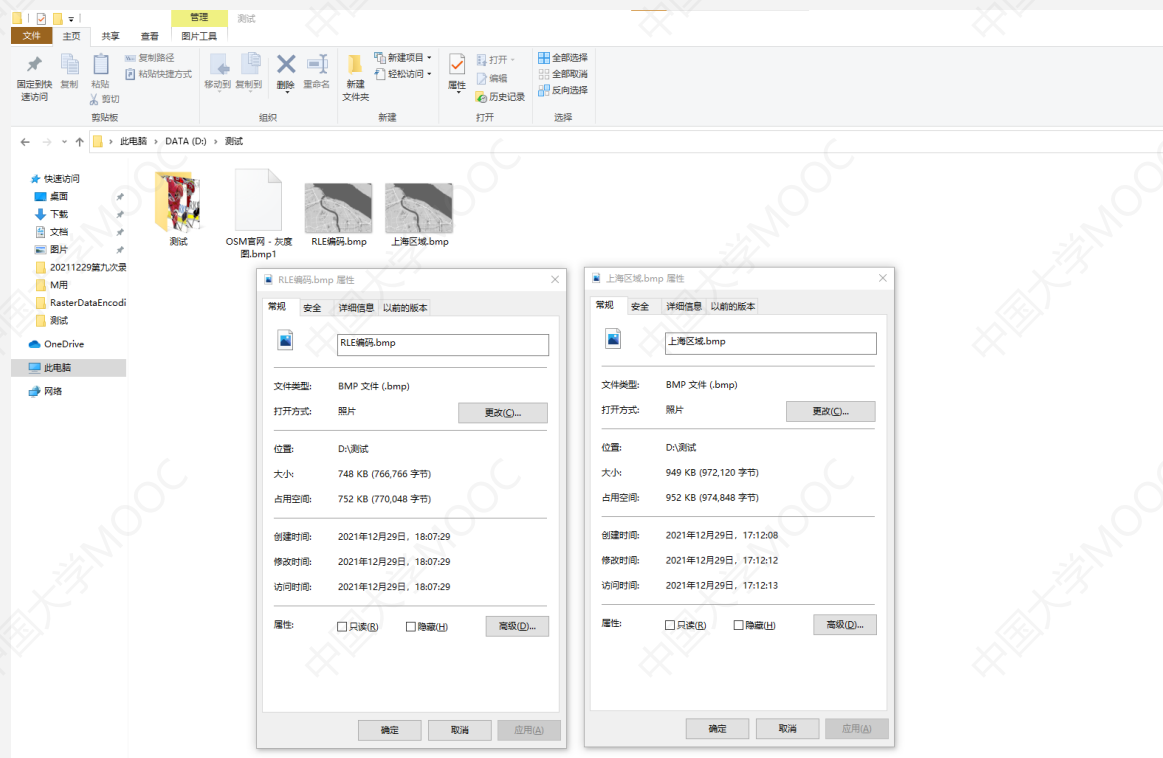


栅格数据编码

Raster Data Coding



资源管理器打开位图存储路径。

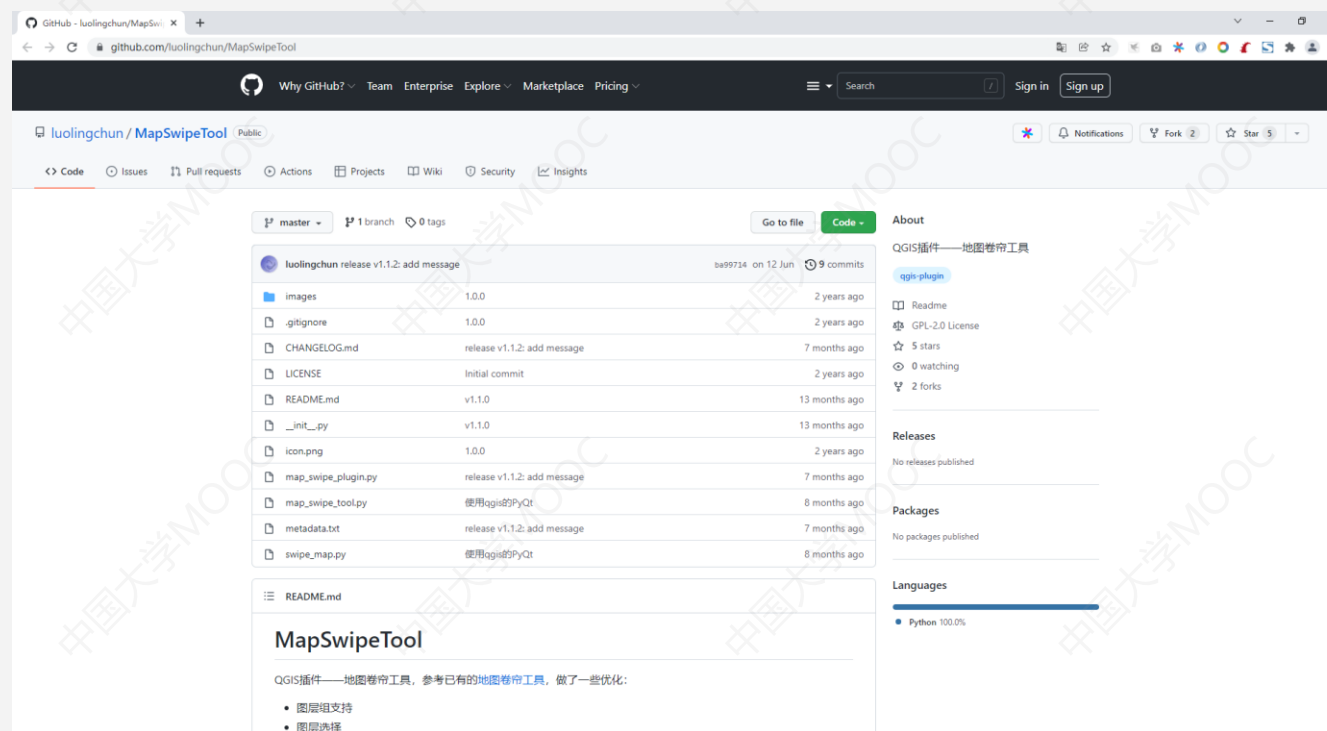


栅格数据编码

Raster Data Coding



在QGIS中打开压缩前后的位图，观察数据可视化以后的效果，这里需要用到QGIS的一个开源卷帘工具插件。



<https://github.com/luolingchun/MapSwipeTool>

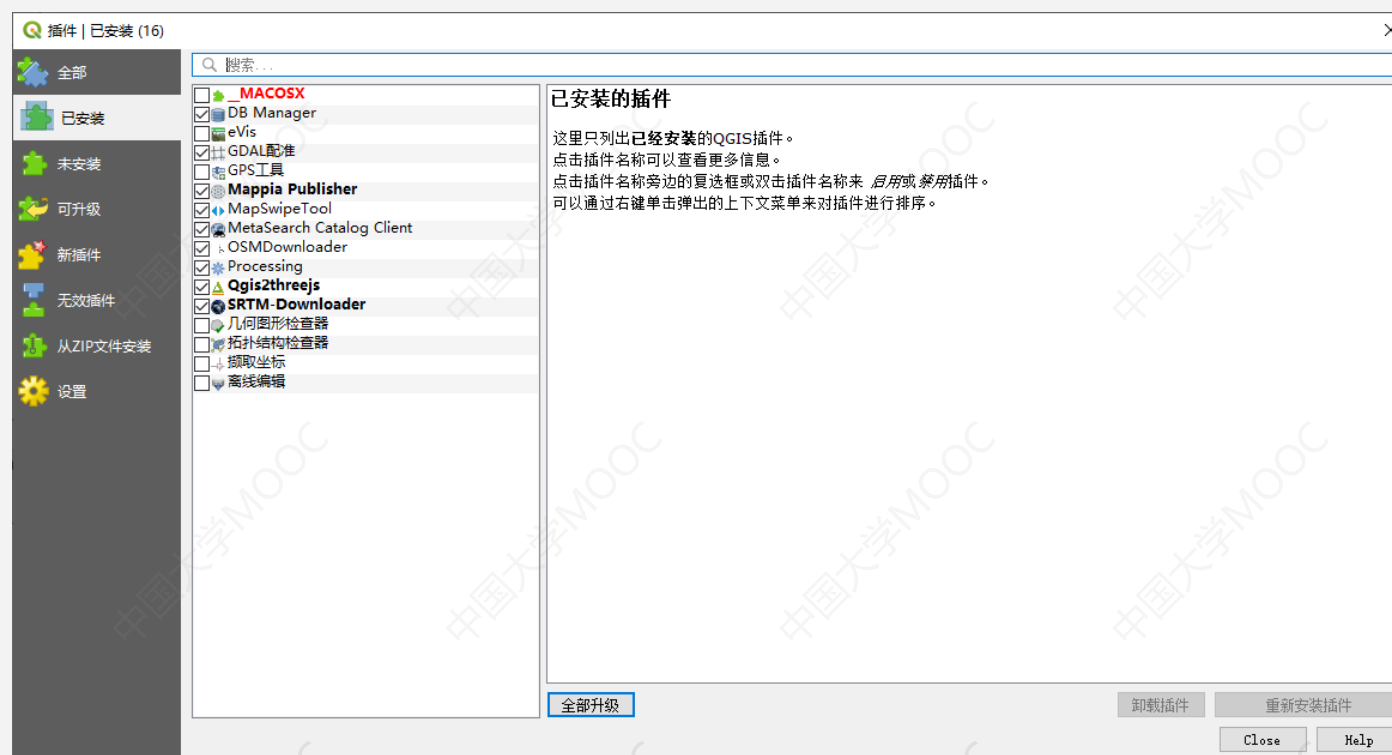


栅格数据编码

Raster Data Coding



下载后，直接在QGIS软件中，打开插件——管理并安装插件。

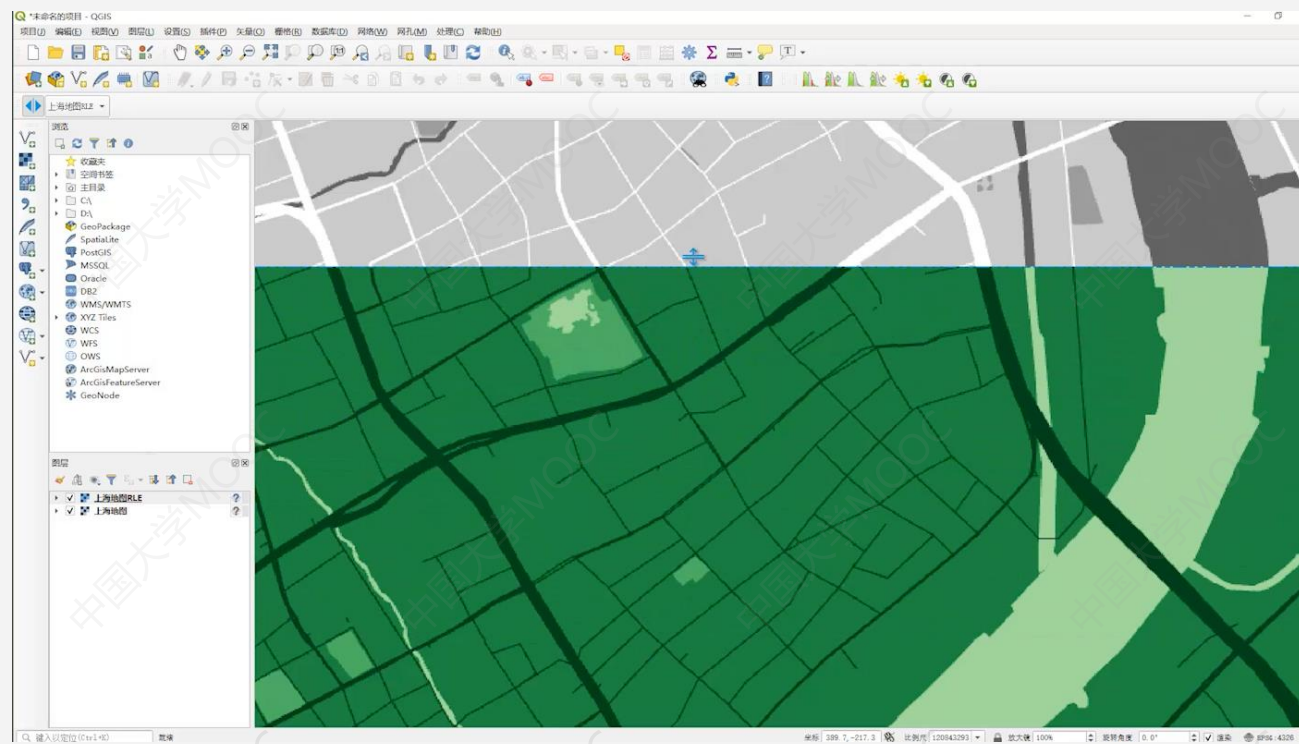


栅格数据编码

Raster Data Coding



为便于观察，给某一个图层调整符号样式，放大到一定级别，使用卷帘工具，发现压缩后的位图在精度、颜色和灰度信息方面能够还原，是无损压缩。



栅格数据编码

Raster Data Coding



作业题：

自己定义一种中间格式，记录压缩后的数据，然后在系统读取数据时，将自己的格式进行还原，还原为BMP的直接编码格式进行显示。

栅格数据编码

Raster Data Coding



谢谢观看