Raster Data Coding

栅格数据编码

INFORMATION ENGINEERING UNIVERSITY

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

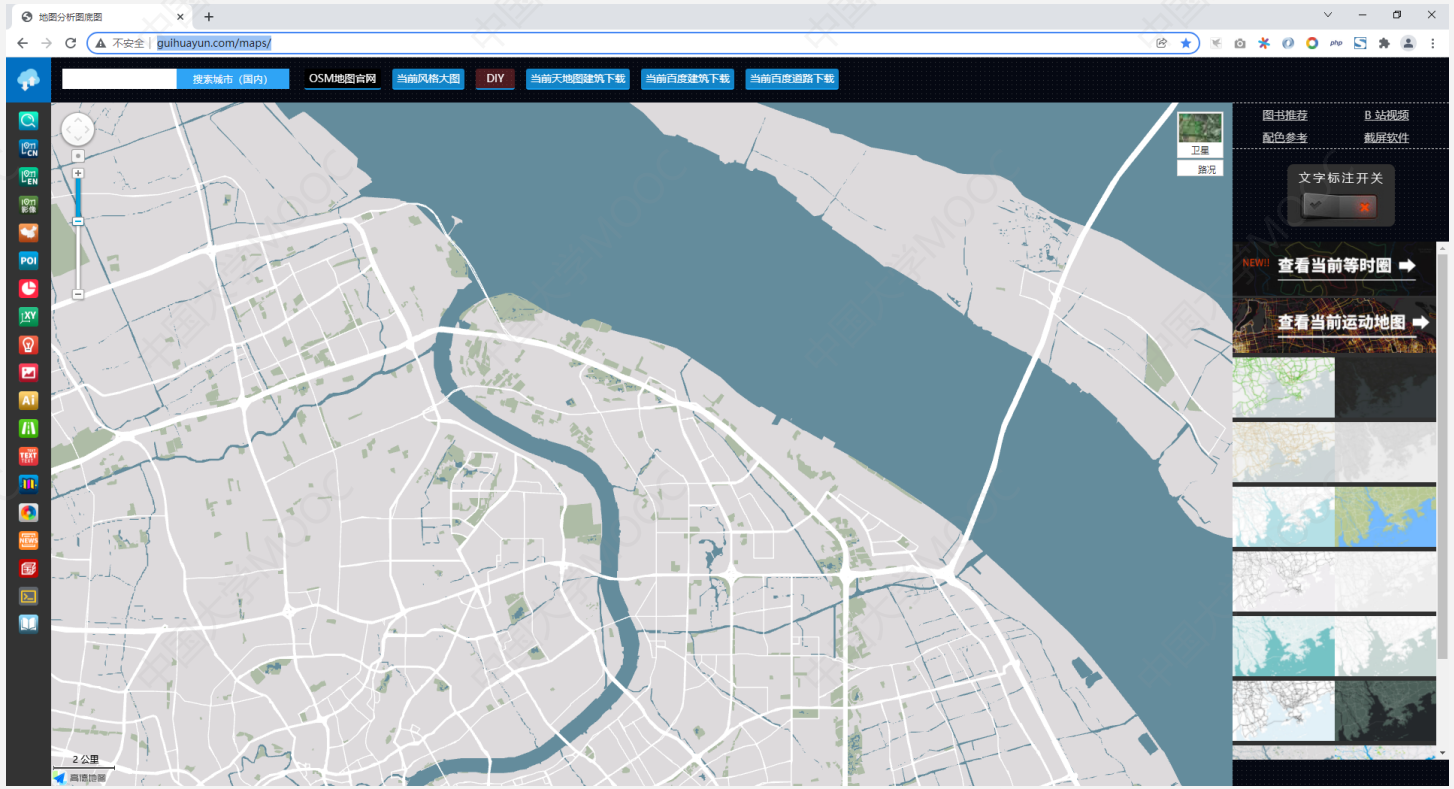
栅格数据常用的编码方式：

- 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。

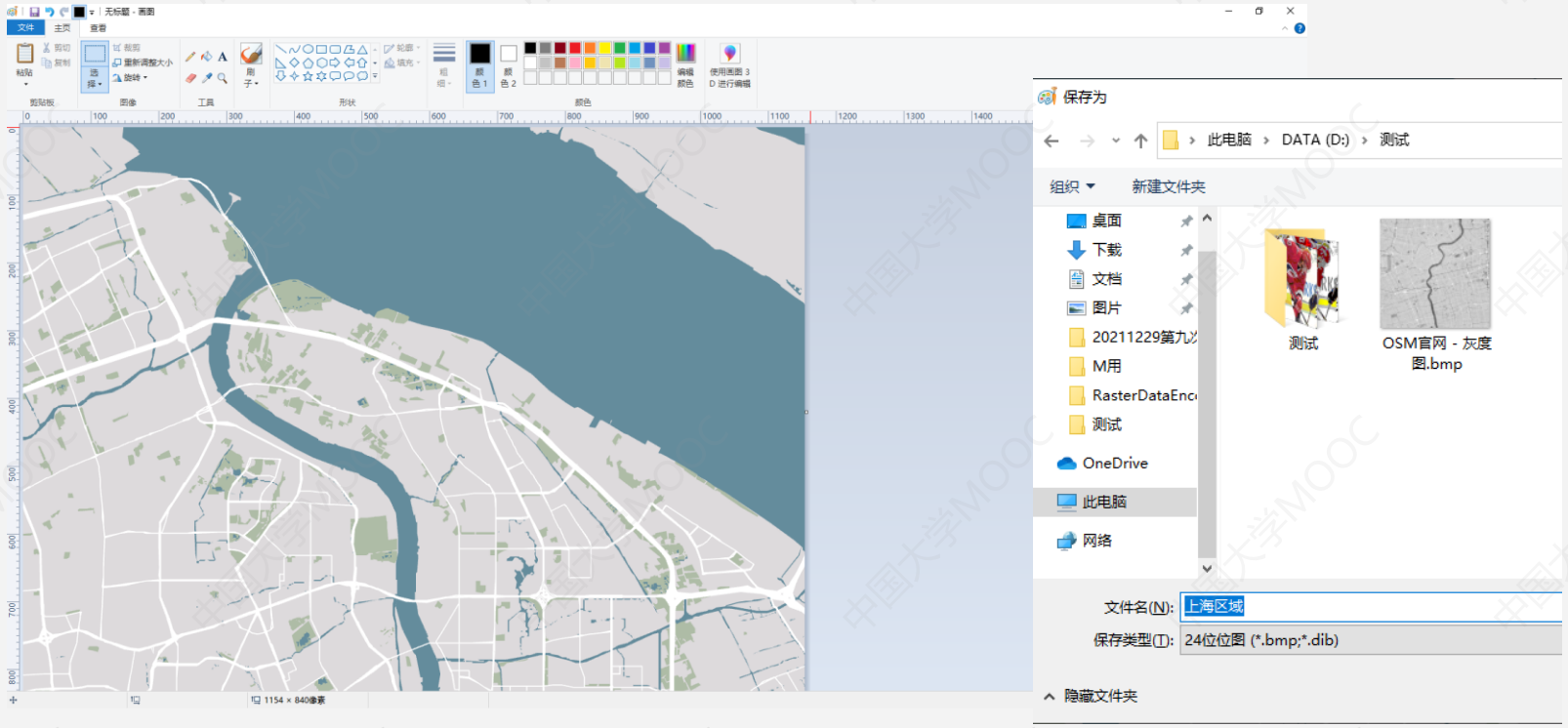


http://guihuayun.com/maps/

栅格数据编码

Raster Data Coding

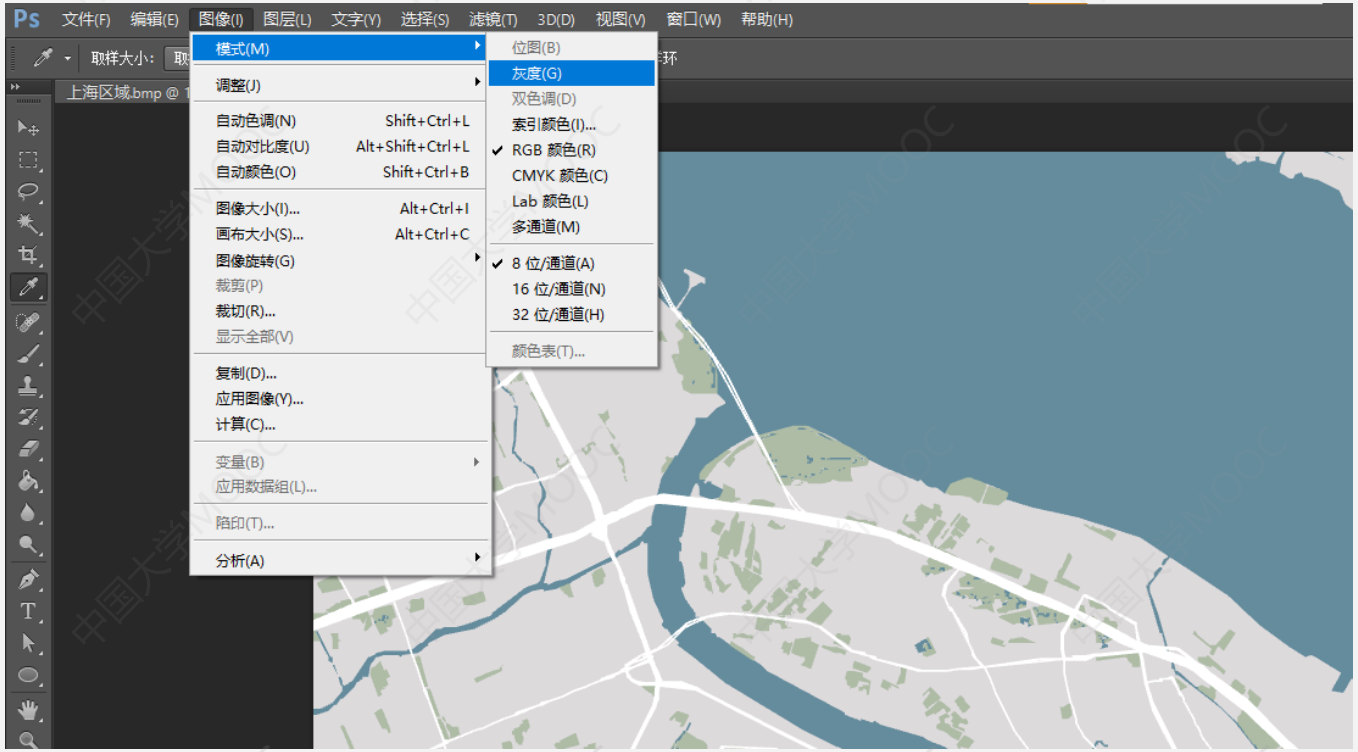
采用系统的windows+shift+s快捷键，截取当前视口中的地图数据，在画图工具中，将栅格数据保存到本地路径中，保存的格式为24位位图。



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：

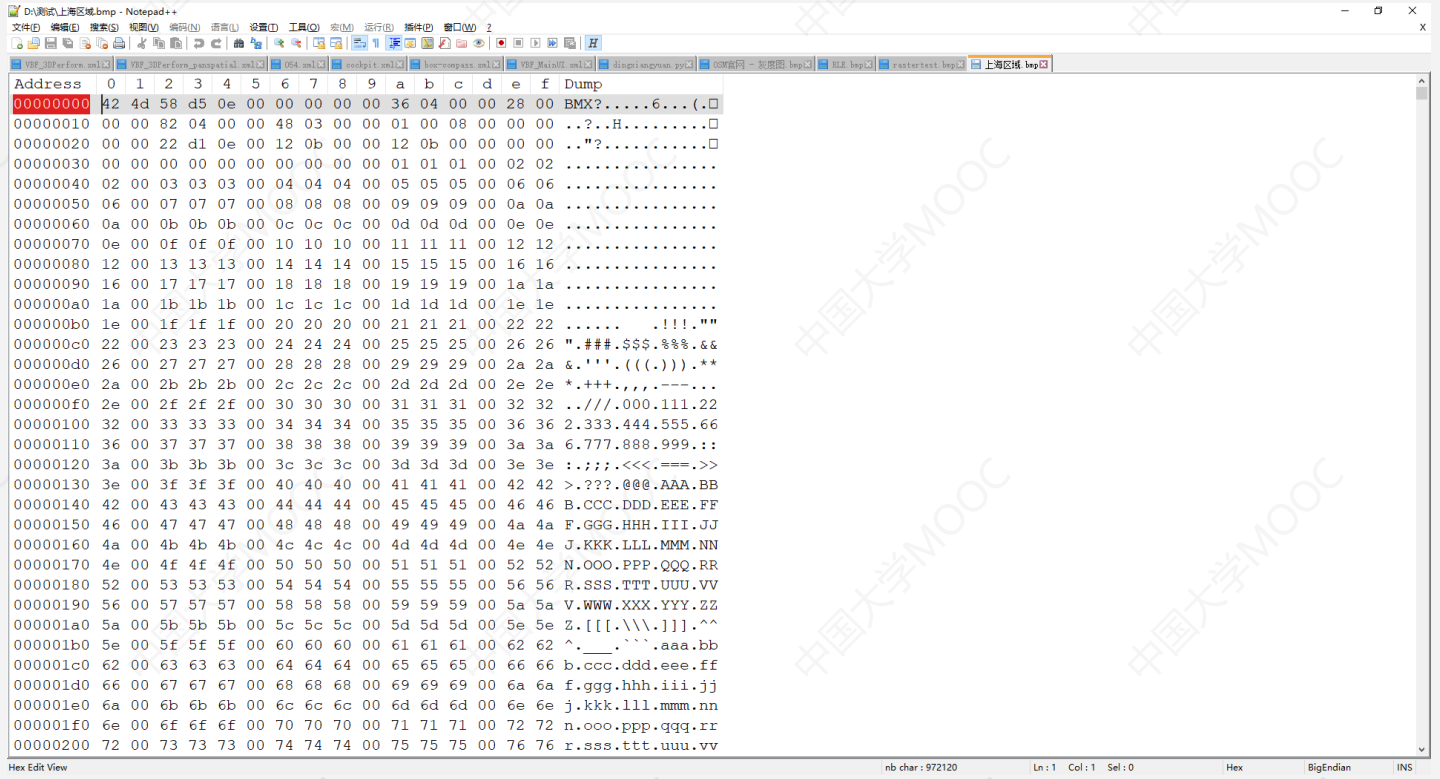
- 文件头
- 信息头
- 颜色表
- 位图数据

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。



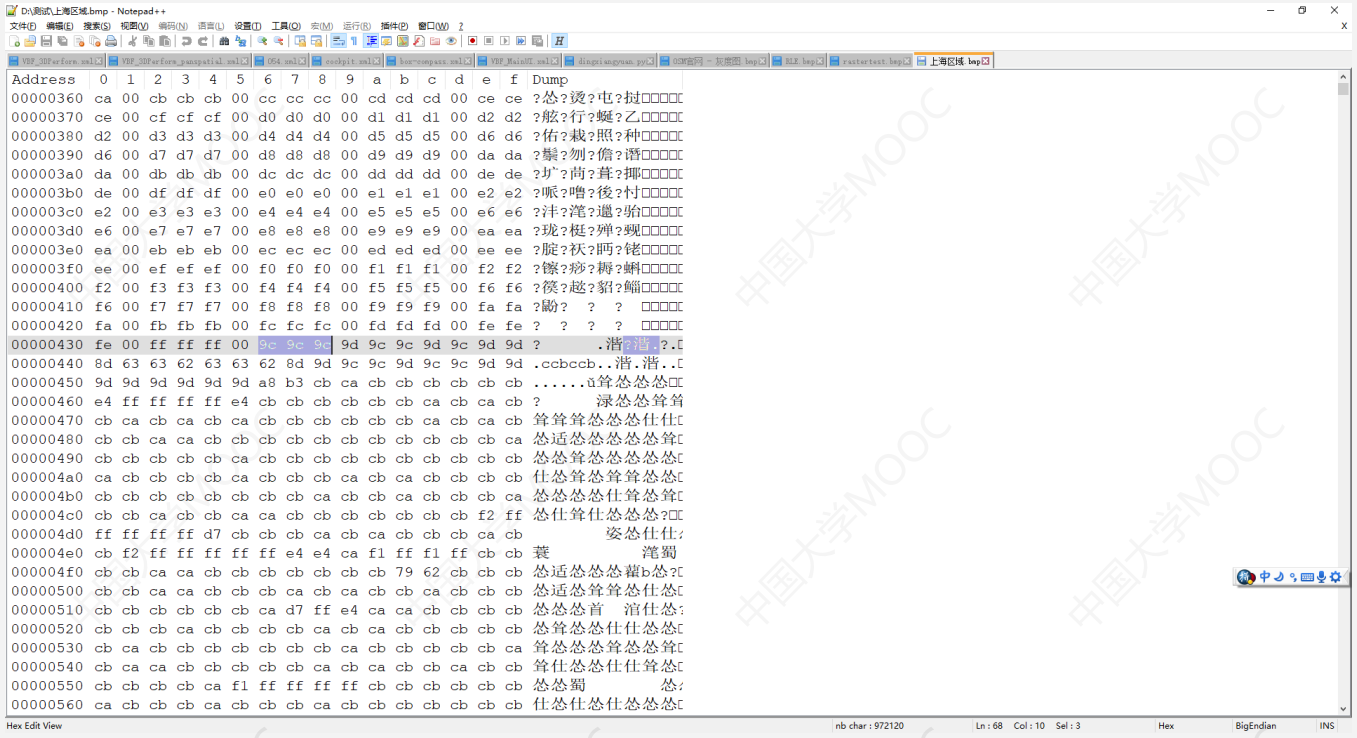
栅格数据编码

Raster Data Coding

栅格数据编码

Raster Data Coding

位图数据按照从左至右，从下至上的顺序，存储每个像素的灰度值信息，每个像素占一个字节。



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 EndOfLine(int pos); //判断位图数据当前行是不是最后一行
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。

```
//加载当前位图数据
BOOL RLE_Encoding::LoadBitmapFile(string filepath)
{
    //删除原有位图句柄
    if (m_hbm != NULL)
    {
        DeleteObject(m_hbm);
    }

    m_hbm = (HBITMAP)LoadImage(NULL, filepath.c_str(), IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE | LR_CREATEDIBSECTION);

    //文件打开错误
    if (m_hbm == NULL)
        return FALSE;

    //获取当前设备上下文
    SelectObject(m_memdc, m_hbm);
    //获取位图数据
    GetObject(m_hbm, sizeof(DIBSECTION), &m_dib);

    return TRUE;
}
```

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。

```
//保存编码压缩后的位图
BOOL RLE_Encoding::SaveBitmapFile(string filename)
{
    //RLE目前只支持256色位图或者灰度图
    if ( m_dib.dsBmih.biBitCount != 8)
        return FALSE;

    int size;
    QByteArray RLE8_bitmaps_bits; //定义字节数组，用于存储压缩后的位图数据
    RLE8_bitmaps_bits.fill(char(0), 4194304); //初始化为4MB，所有字节位都为0

    //使用Run-Legth-Encoding(8 bit)压缩位图数据
    CompressInRLE8( (BYTE*)m_dib.dsBm.bmBits, RLE8_bitmaps_bits, size);

    _8Bit_BITMAPINFO _8bit_bmi;
    _8bit_bmi.bmi.bmiHeader = m_dib.dsBmih; //拷贝完整的BITMAPINFOHEADER结构信息
    _8bit_bmi.bmi.bmiHeader.biCompression = BI_RLE8; //设置为8位RLE编码
    _8bit_bmi.bmi.bmiHeader.biSizeImage = size ; //压缩后的位图大小

    //从当前设备上下文中获取颜色表信息，存储在_8bit_bmi中
    GetDIBColorTable(m_memdc, 0, _8bit_bmi.bmi.bmiHeader.biClrUsed, _8bit_bmi.bmi.bmiColors );

    HANDLE hf;                  //定义文件句柄
    BITMAPFILEHEADER hdr;       //定义位图文件头
    PBITMAPINFOHEADER pbih;     //定义位图信息头
    DWORD dwTmp;
    //强制转换获取位图信息头
    pbih = (PBITMAPINFOHEADER) &_8bit_bmi;

    //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
155      return 0 == ((pos+1)% m_dib.dsBmih.biWidth);
156  }
```

栅格数据编码

Raster Data Coding

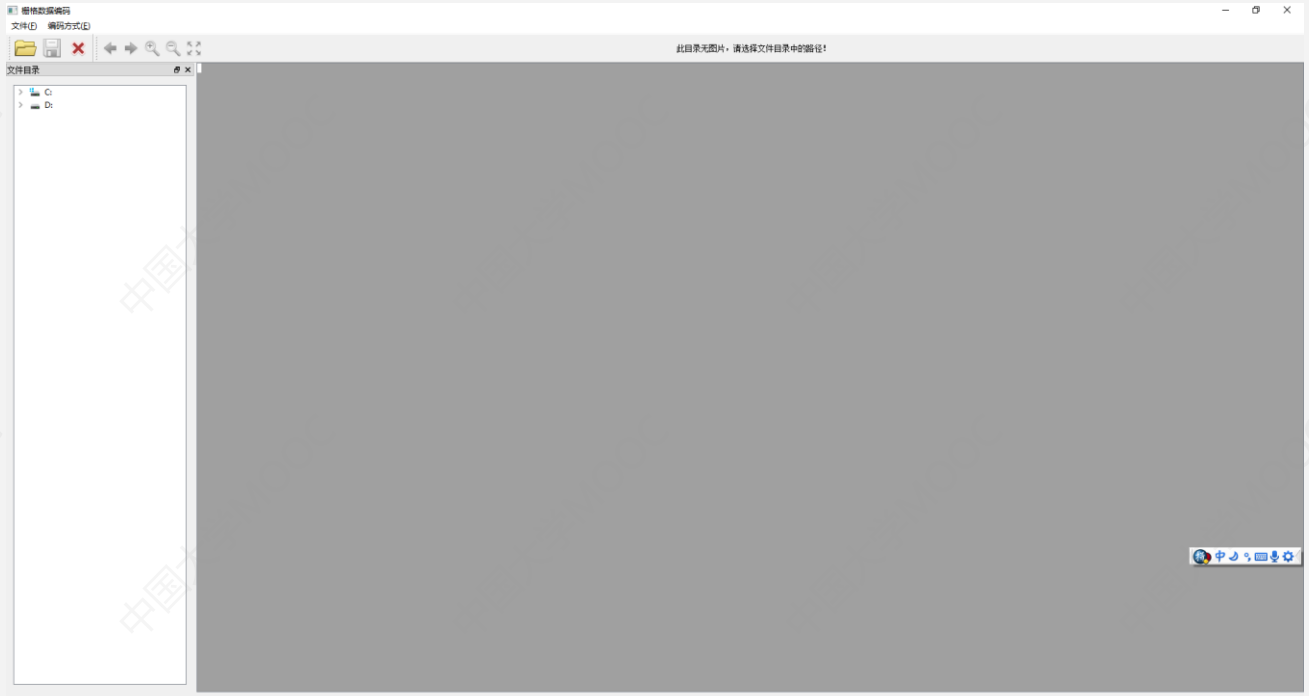
CompressInRLE8函数实现具体的游程长度编码功能。

```
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

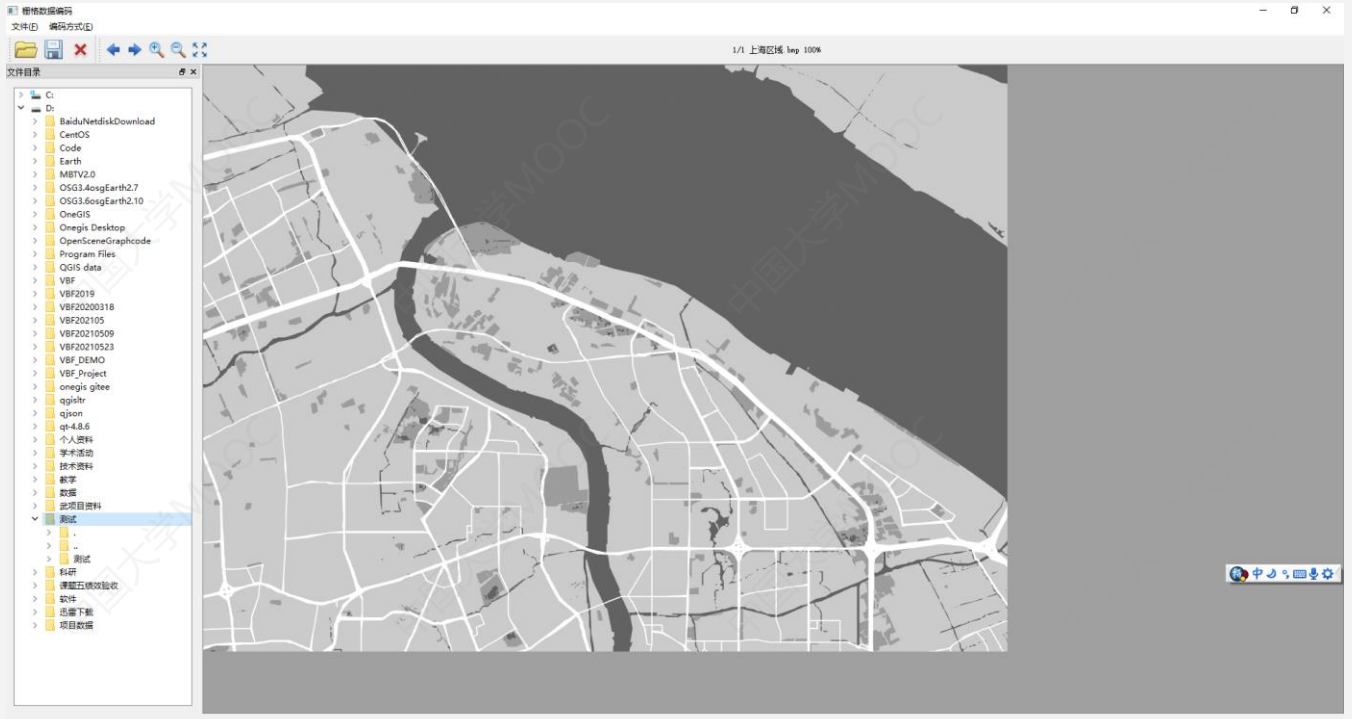
运行程序，打开系统主界面。



栅格数据编码

Raster Data Coding

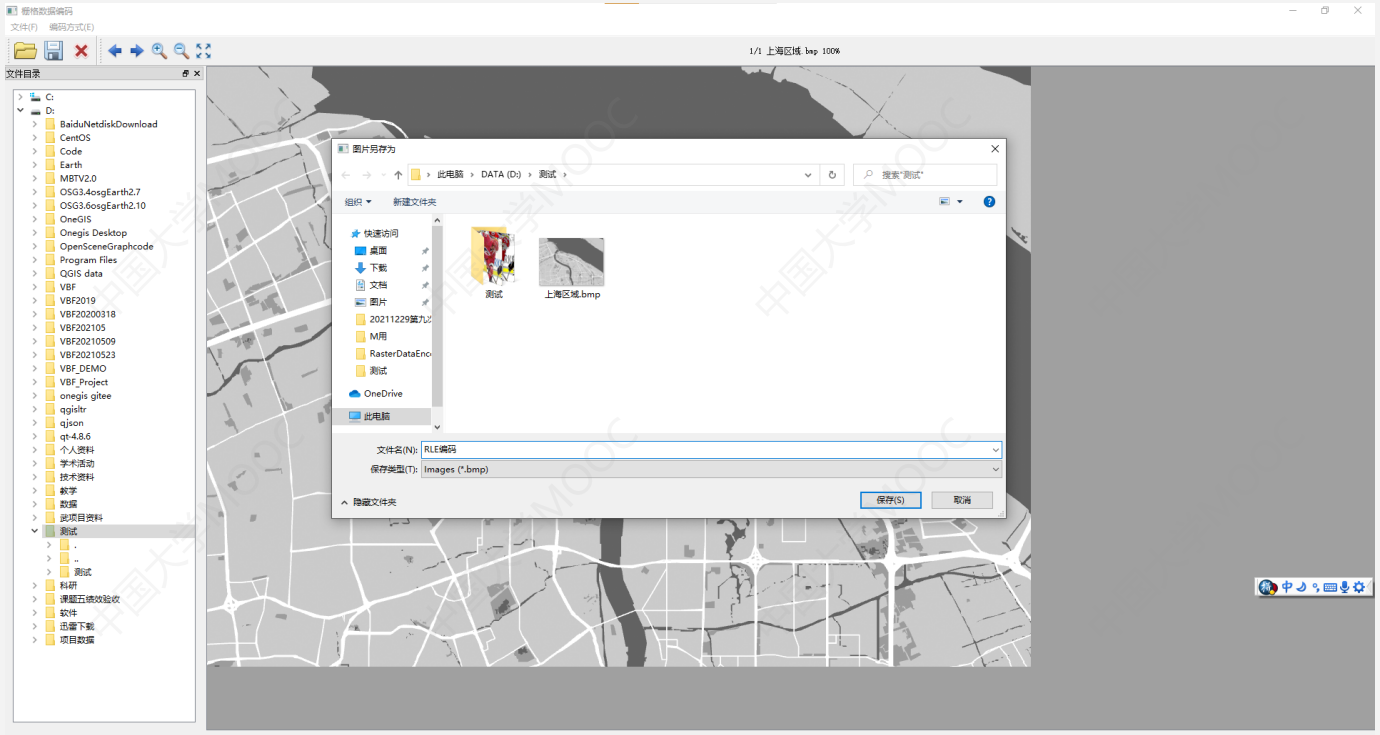
通过左侧目录，或者“打开文件夹”工具，定位到刚才位图的存储路径，打开位图文件。



栅格数据编码

Raster Data Coding

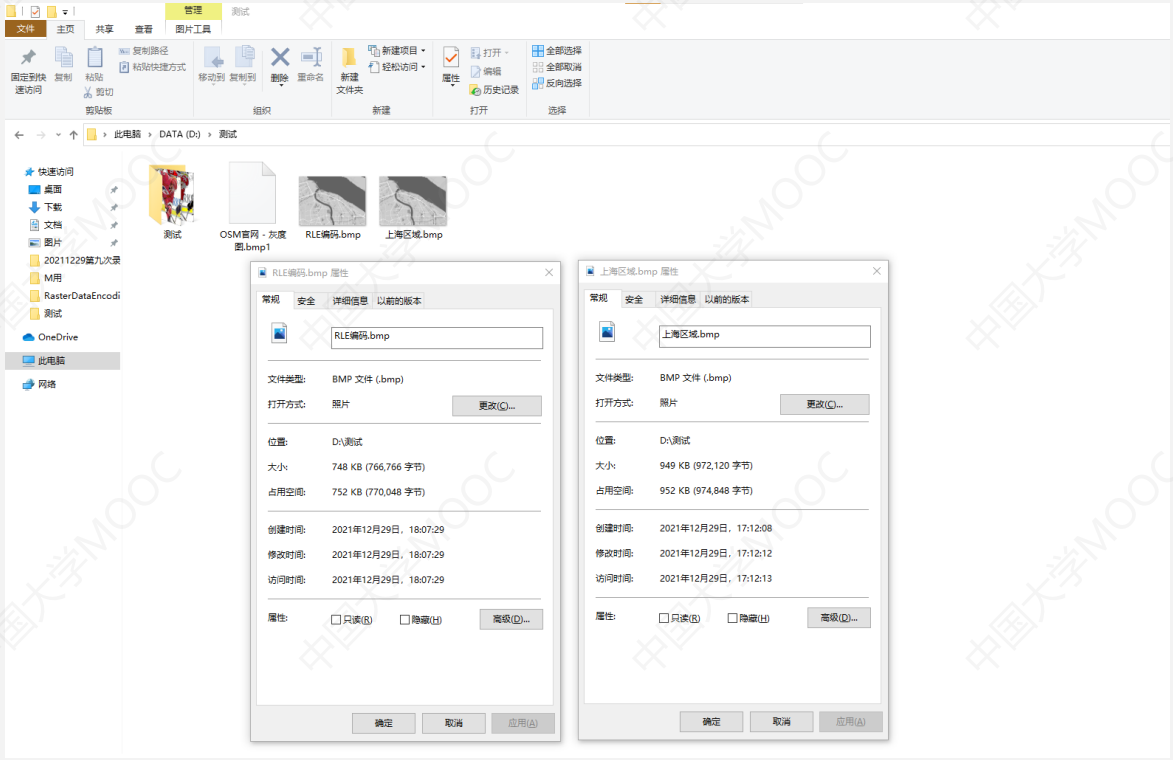
点击工具栏中的“另存为”按钮，在弹出的对话框中选择压缩后位图的存储路径，输入文件名，点击保存，完成位图的游程长度编码。



栅格数据编码

Raster Data Coding

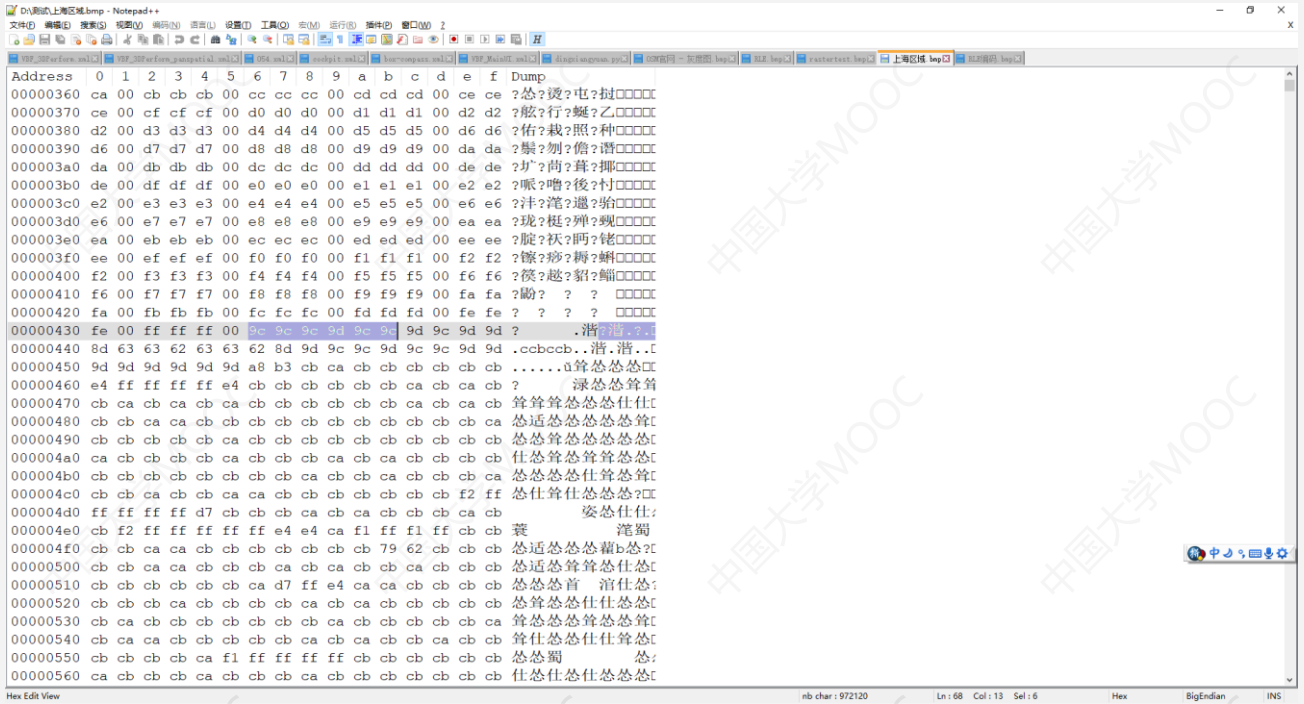
资源管理器打开位图存储路径。



栅格数据编码

Raster Data Coding

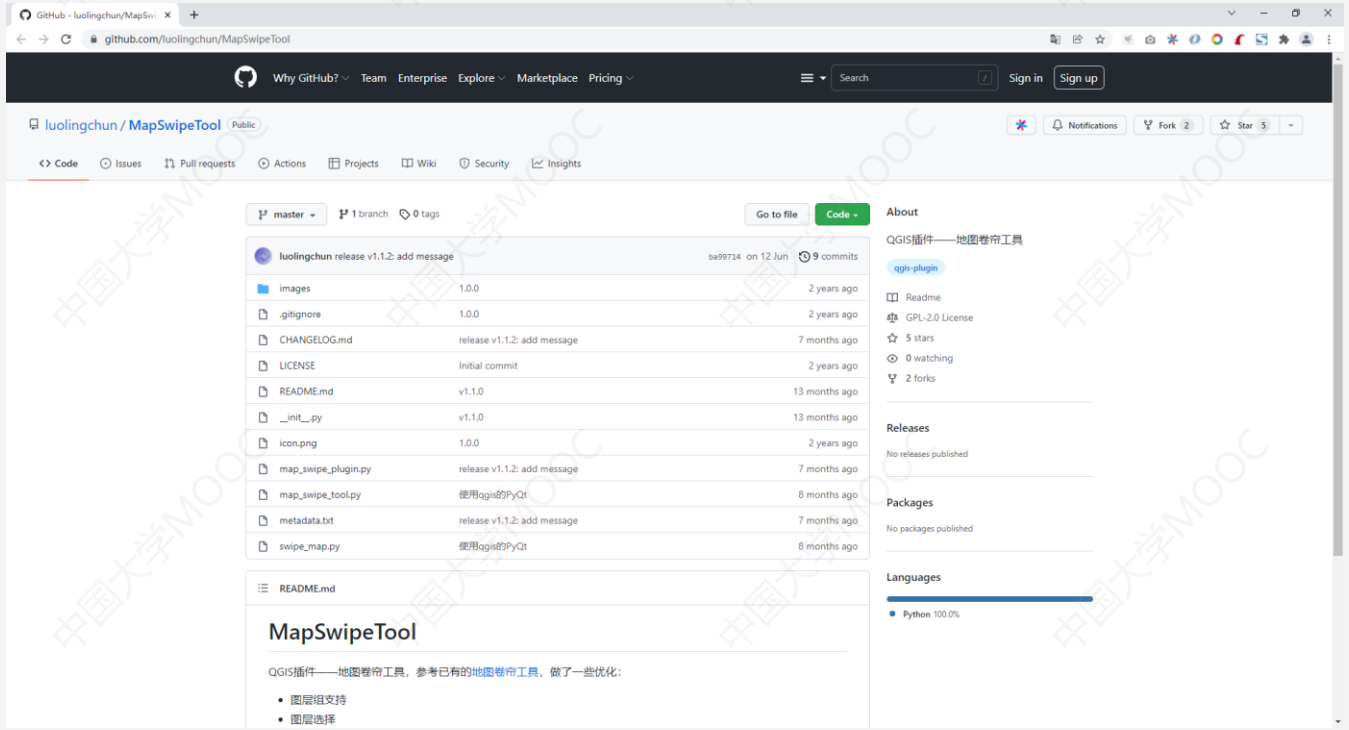
通过Notepad打开压缩前和压缩后的位图文件，定位到两幅位图的“位图数据”数据块，压缩前和压缩后的数据内容和我们理论部分讲到的记录方式是一致的。



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。

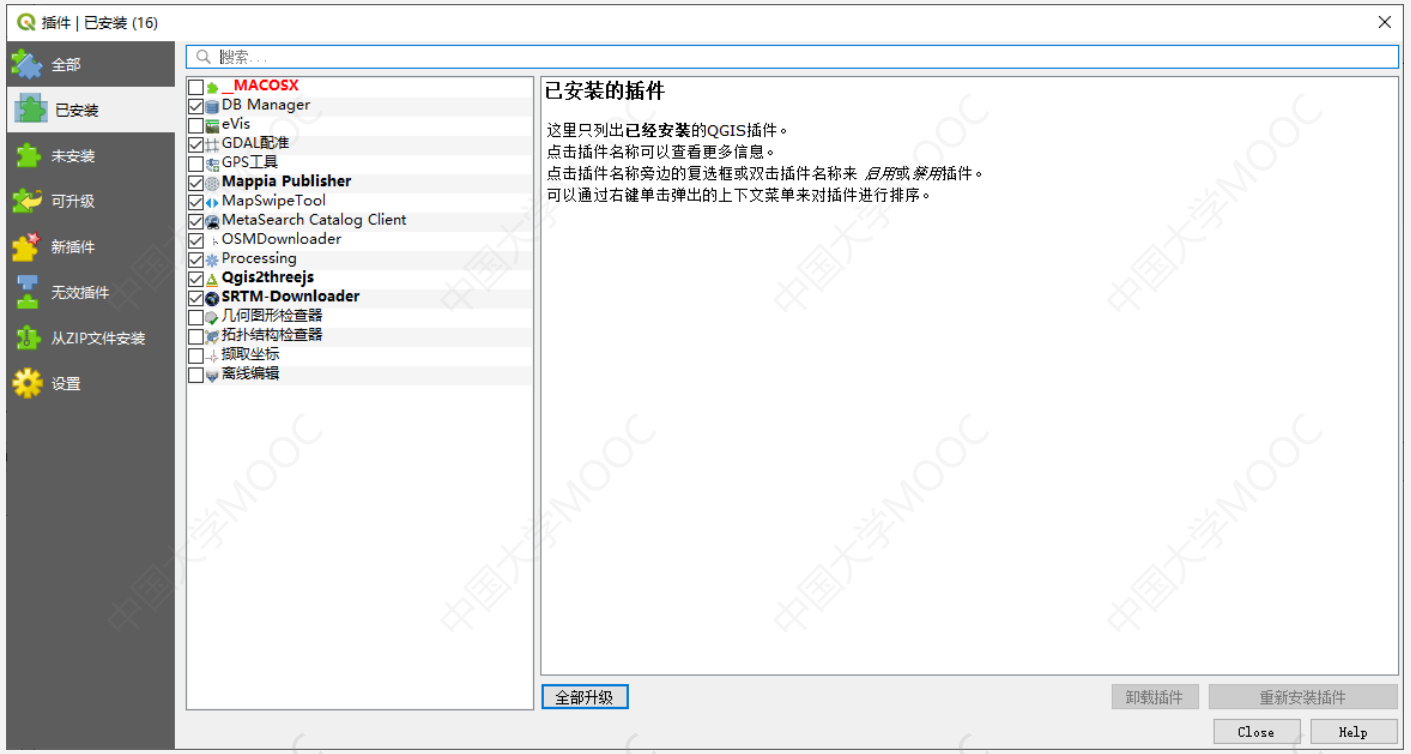


https://github.com/luolingchun/MapSwipeTool

栅格数据编码

Raster Data Coding

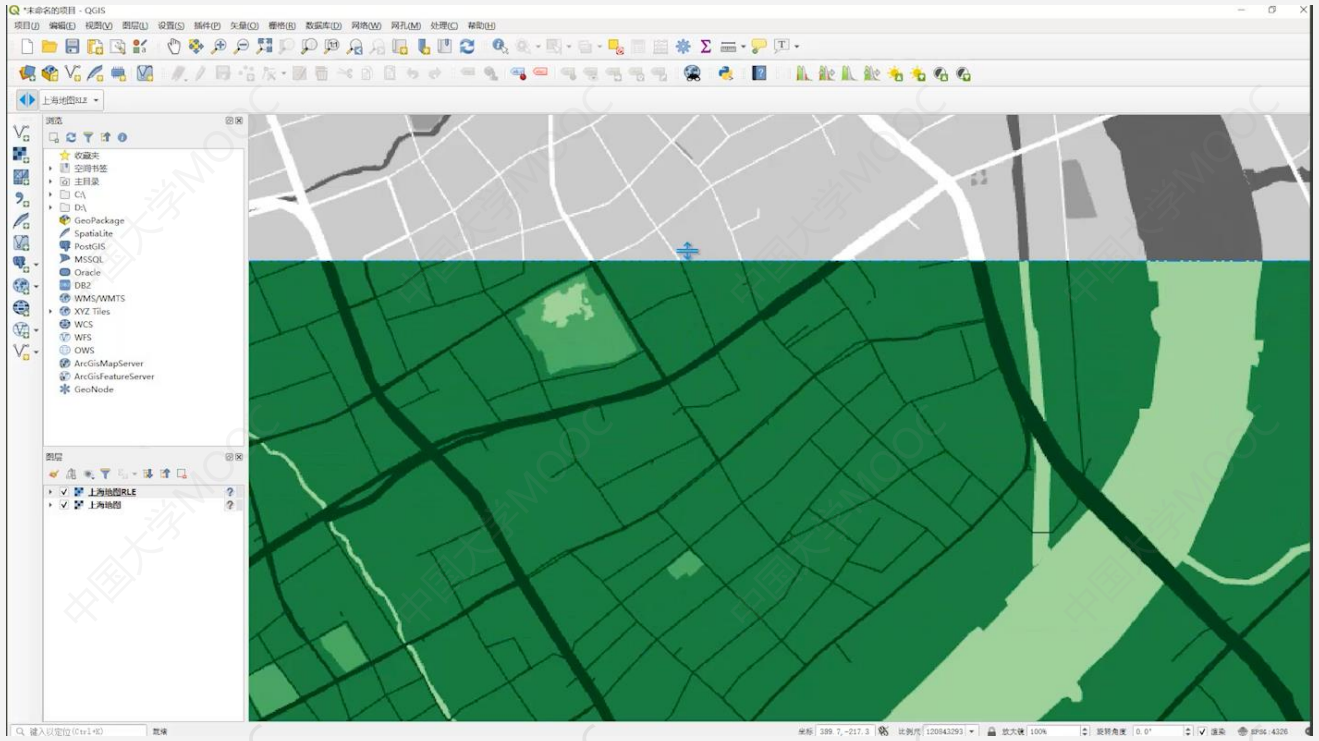
下载后，直接在QGIS软件中，打开插件——管理并安装插件。



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。

谢 谢 观 看