**一、概述**

    遥感影像和地理坐标进行关联的方式一般有好几种，一种是直接给出了仿射变换系数，即6个参数，左上角地理坐标，纵横方向上的分辨率，以及旋转系数。在这种情况下，求出某一像素点的地理坐标非常容易，直接用公式可以求出，具体代码如下：

void CPL\_STDCALL GDALApplyGeoTransform(double \*padfGeoTransform,

                            double dfPixel, double dfLine,

                            double \*pdfGeoX, double \*pdfGeoY )

{

   \*pdfGeoX = padfGeoTransform[0]+ dfPixel \* padfGeoTransform[1]

                                   + dfLine  \* padfGeoTransform[2];

   \*pdfGeoY = padfGeoTransform[3]+ dfPixel \* padfGeoTransform[4]

                                   + dfLine  \* padfGeoTransform[5];

}

可 能有的读者额看到这个觉得非常熟悉，这个是GDAL里面实现的由像素点转换到地理坐标点的转换函数，这个函数的声明在gdal.h中，定义在 gdaltransformer.cpp文件中，注意，这个函数计算的是影像像元的左上角的地理坐标，也可以计算中心点坐标，具体看需求。

另外一种方式是记录影像控制点坐标的数据，一般控制点都记录着控制点自身的地理坐标，还有控制点在影像中的像素坐标点，还有控制点必须是在某一特定的地理坐标系下，在GDAL中控制点的结构体为：

/\*\* Ground Control Point \*/

typedef struct

{

   /\*\* Unique identifier, often numeric \*/

   char        \*pszId;

   /\*\* Informational message or "" \*/

   char        \*pszInfo;

   /\*\* Pixel (x) location of GCP on raster \*/

   double      dfGCPPixel;

   /\*\* Line (y) location of GCP on raster \*/

   double      dfGCPLine;

   /\*\* X position of GCP in georeferenced space\*/

   double      dfGCPX;

   /\*\* Y position of GCP in georeferenced space\*/

   double      dfGCPY;

   /\*\* Elevation of GCP, or zero if not known \*/

   double      dfGCPZ;

} GDAL\_GCP;

那读者会有疑问了，这个控制点的数据怎么获得呢，GDAL里面有函数接口可以获得，具体如下；

char\* pszWkt1 = (char\*)poDatasetchangsha->GetProjectionRef();

    if (0 == strlen(pszWkt1))

    {

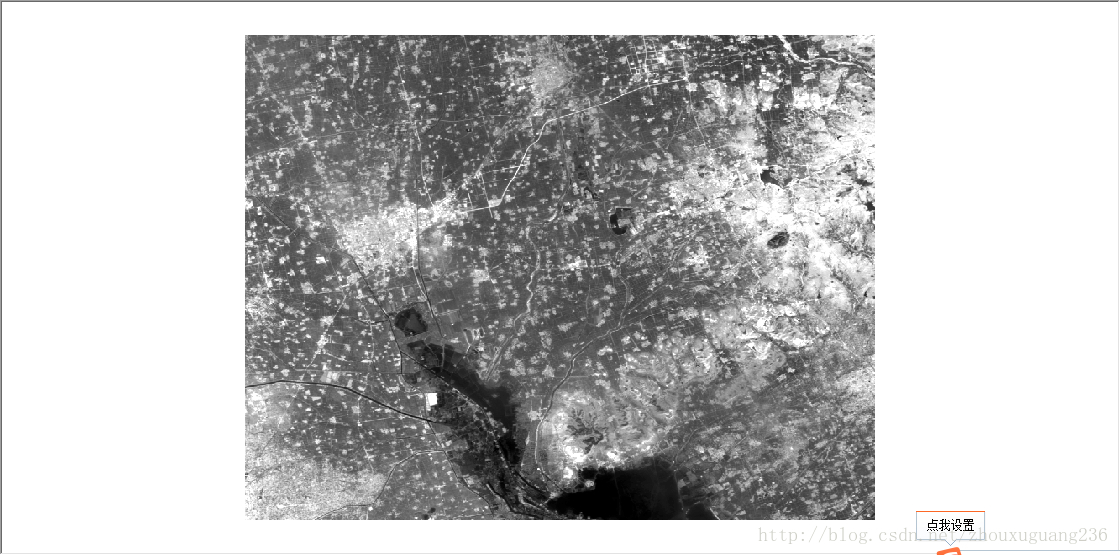
        pszWkt1= (char\*)poDatasetchangsha->GetGCPProjection();

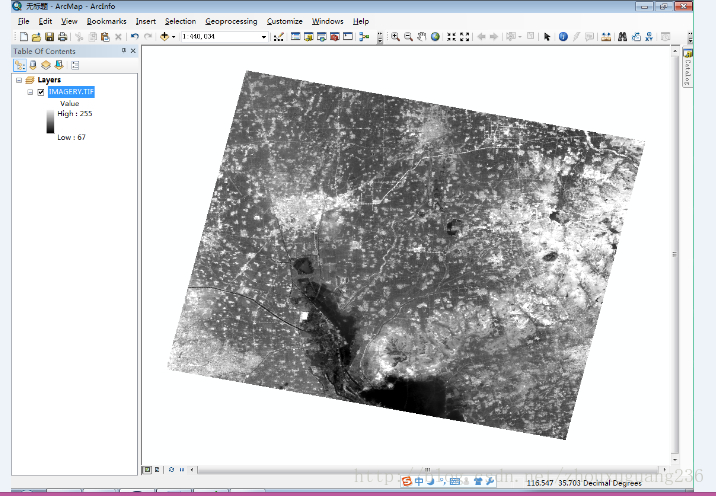
    }

    int nGCPCount = poDatasetchangsha->GetGCPCount();

    const GDAL\_GCP \*pGCPList= poDatasetchangsha->GetGCPs();

我首先给出一个例子先有一个比较直观的感受，我手上有一个spot传感器的数据，它是带控制点信息的，控制点刚好是图像的四个角点。比如，用arcgis打开就已经是纠正显示了。没有实施纠正显示的情况和ARCGIS实现了纠正显示的情况如下图：





Arcgis显示的时候，边缘的一些像素由于在原始影像上找不到对应的采样点，所以就用白色填充。

**二、几何校正方法**

关于影像几何校正，一般可以采用下面的几种方法：

1、几何多项式校正模型

  这个模型比较简单，回避了遥感影像的成像过程，适用于覆盖面积不大喝地形比较平坦的区域。

2、有理函数数学模型

   建立像素和地面位置对应关系的简单数学模型。理论上可以达到跟严格卫星传感器模型相当的定位精度。其优点在于多项式包含高程信息,可以提高校正精度。这个模型自带了高程信息，一般需要影像提供RPC文件，或者用户自己选择地面控制点。

3、局部区域校正模型

   这个模型的基本思想是利用控制点建立不规则三角网，然后分区域利用几何多项式校正，但是这种模型需要很多的控制点，对于地形起伏很大的区域需要的控制点更多，往往实施难度比较大。

4、卫星传感器模型

   这种模型是依据卫星遥感成像的几何关系，利用摄影测量学中成像瞬间的地面点、透视中心以及相应的像素点三点共线建立起来的一种模型，但是此模型需要DEM数据，优点是精度比较高。

由于本文主要关注几何多项式模型，所以只利用GDAL中几何多项式的模型来校正。

   可以根据地面控制点求算出仿射变换系数，可以调用GDAL的函数，其函数是：

int CPL\_DLL CPL\_STDCALL

GDALGCPsToGeoTransform(int nGCPCount,const GDAL\_GCP\*pasGCPs,

                        double \*padfGeoTransform, int bApproxOK );

这个函数的定义在gdal\_misc.cpp文件中，改天可以去了解下。求出仿射变换系数后可以求出图像的大致范围和分辨率，这样就可以目标图像所占的行数和列数。

下面的代码是求出影像的envelope，大概的范围

double dbX[4];

            double dbY[4];

            dbX[0] = dblGeoTransform[0]+ 0\*dblGeoTransform[1] + 0\*dblGeoTransform[2];

            dbY[0] = dblGeoTransform[3]+ 0\*dblGeoTransform[4] + 0\*dblGeoTransform[5];

            //右上角坐标

            dbX[1] = dblGeoTransform[0]+ (lWidth/\*-0.5\*/)\*dblGeoTransform[1] + 0\*dblGeoTransform[2];

            dbY[1] = dblGeoTransform[3]+ (lWidth/\*-0.5\*/)\*dblGeoTransform[4] + 0\*dblGeoTransform[5];

            //右下角点坐标

            dbX[2] = dblGeoTransform[0]+ (lWidth/\*-0.5\*/)\*dblGeoTransform[1] + (lHeight/\*-0.5\*/)\*dblGeoTransform[2];

            dbY[2] = dblGeoTransform[3]+ (lWidth/\*-0.5\*/)\*dblGeoTransform[4] + (lHeight/\*-0.5\*/)\*dblGeoTransform[5];

            //左下角坐标

            dbX[3] = dblGeoTransform[0]+ 0\*dblGeoTransform[1] + (lHeight/\*-0.5\*/)\*dblGeoTransform[2];

            dbY[3] = dblGeoTransform[3]+ 0\*dblGeoTransform[4] + (lHeight/\*-0.5\*/)\*dblGeoTransform[5];

            double dbMinx = 0;

            double dbMaxx = 0;

            double dbMiny = 0;

            double dbMaxy = 0;

            dbMinx = min(min(min(dbX[0],dbX[1]),dbX[2]),dbX[3]);

            dbMaxx = max(max(max(dbX[0],dbX[1]),dbX[2]),dbX[3]);

            dbMiny = min(min(min(dbY[0],dbY[1]),dbY[2]),dbY[3]);

            dbMaxy = max(max(max(dbY[0],dbY[1]),dbY[2]),dbY[3]);

但是上面的方法计算出来的范围有可能不是很准确，所以这里要寻找一种比较精确的方法，那就是求出每个边缘像素的边缘对应的地理坐标，然后逐个比较大小求出比较精确的范围。这种方法可以采用GDAL算法模块里面GDALSuggestedWarpOutput2函数。

CPLErr CPL\_STDCALL

GDALSuggestedWarpOutput2(GDALDatasetH hSrcDS,

                          GDALTransformerFunc pfnTransformer,

                          void \*pTransformArg,

                          double \*padfGeoTransformOut,

                          int \*pnPixels, int \*pnLines,

                          double \*padfExtent,int nOptions)；

这个函数里面的参数的意义解释可以参考GDAL官方文档，也可以直接看GDAL源代码，下面我就给出各个参数的意思。

第一个参数是数据源，第二个参数是转换的回调函数，比如是GCP控制点转换的函数GDALGCPTransform，或者是RPC校正的函数等等。第三个参数是代表转换的关系，这里的转换关系是控制点几何校正的转换关系，此转换关系可以由void CPL\_DLL \*

GDALCreateGCPTransformer(int nGCPCount,const GDAL\_GCP\*pasGCPList,

                          int nReqOrder, int bReversed );函数生成。第四个参数是图像纠正后的仿射变换的6个参数，pnPixels是纠正后图像的列数，pnLines是纠正后图像的行数，padfExtent就是图像纠正后的地理范围，nOptions是相关选项的设置，默认情况下可以设为空。

**三、重采样及其校正实现**

到 此，我们已经完成了原始图像的像素坐标到校正后的地理空间坐标建立了关联关系，并且输出影像的大小、地理范围、分辨率、像素位深度都已经知道，这样我们就 可以创建一个空的图像了，但是校正后的图像的各个像素的值还没确定，接下来就是如何确定校正后图像中各个像素的值，也就是图像处理中的重采样技术，一般大 家听的比较多的重采样技术可能是最邻近方法、双线性内插重采样以及三次卷积内插重采样等方法。还有一些其他的重采样方法，比如平均值法，中值法等。

重采样技术我们可以直接调用GDAL的函数去实现，但是我这里决定自己写重采样的过程，以便今后做一些改进，这里我先用最邻近采样方法实现重采样。

整个过程的代码如下：

**[cpp]** [view plaincopy](http://blog.csdn.net/zhouxuguang236/article/details/27965877)[在CODE上查看代码片](https://code.csdn.net/snippets/372396)

1. /\*\*
2. \* @brief 几何多项式校正
3. \* @param pszSrcFile         输入文件路径
4. \* @param pszDstFile         输出文件路径
5. \* @param nGCPCount          GCP点对个数
6. \* @param pasGCPList         GCP点对列表
7. \* @param pszDstWKT          输出文件投影
8. \* @param iOrder             多项式次数
9. \* @param dResX              输出图像X方向分辨率
10. \* @param dResY              输出图像Y方向分辨率
11. \* @param eResampleMethod    重采样方式
12. \* @param pszFormat          输出文件格式
13. \* @return  返回值，目前只返回true或者false
14. \*/
15. int ImageWarpByGCP(const char \* pszSrcFile,
16. const char \* pszDstFile,
17. int nGCPCount,
18. const GDAL\_GCP \*pasGCPList,
19. const char \* pszDstWKT,
20. int iOrder,
21. double dResX,
22. double dResY,
23. GDALResampleAlg eResampleMethod,
24. const char \* pszFormat)
25. {
27. GDALAllRegister();
28. CPLSetConfigOption("GDAL\_FILENAME\_IS\_UTF8", "NO");
30. // 打开原始图像
31. GDALDatasetH hSrcDS = GDALOpen(pszSrcFile, GA\_ReadOnly);
32. if (NULL == hSrcDS)
33. {
34. return 0;
35. }
37. GDALDataType eDataType = GDALGetRasterDataType(GDALGetRasterBand(hSrcDS, 1));
38. int nBandCount = GDALGetRasterCount(hSrcDS);
40. // 创建几何多项式坐标转换关系
41. void \*hTransform = GDALCreateGCPTransformer(nGCPCount, pasGCPList, iOrder, FALSE );
42. if( NULL == hTransform )
43. {
44. GDALClose( hSrcDS );
45. return 0;
46. }
48. // 计算输出图像四至范围、大小、仿射变换六参数等信息
49. double adfGeoTransform[6] = {0};
50. double adfExtent[4] = {0};
51. int    nPixels = 0, nLines = 0;
53. if( GDALSuggestedWarpOutput2( hSrcDS, GDALGCPTransform, hTransform,
54. adfGeoTransform, &nPixels, &nLines, adfExtent, 0 ) != CE\_None )
55. {
56. GDALClose( hSrcDS );
57. return 0;
58. }
60. // 下面开始根据用户指定的分辨率来反算输出图像的大小和六参数等信息
61. double dResXSize = dResX;
62. double dResYSize = dResY;
64. //如果为0，则默认为原始影像的分辨率
65. if(dResXSize == 0.0 && dResYSize == 0.0 )
66. {
67. double dbGeoTran[6] = {0};
68. GDALGCPsToGeoTransform(nGCPCount,pasGCPList,dbGeoTran,0);
69. dResXSize = fabs(dbGeoTran[1]);
70. dResYSize = fabs(dbGeoTran[5]);
71. }
73. // 如果用户指定了输出图像的分辨率
74. else if ( dResXSize != 0.0 || dResYSize != 0.0 )
75. {
76. if ( dResXSize == 0.0 ) dResXSize = adfGeoTransform[1];
77. if ( dResYSize == 0.0 ) dResYSize = adfGeoTransform[5];
78. }
80. if ( dResXSize < 0.0 ) dResXSize = -dResXSize;
81. if ( dResYSize > 0.0 ) dResYSize = -dResYSize;
83. // 计算输出图像的范围
84. double minX = adfGeoTransform[0];
85. double maxX = adfGeoTransform[0] + adfGeoTransform[1] \* nPixels;
86. double maxY = adfGeoTransform[3];
87. double minY = adfGeoTransform[3] + adfGeoTransform[5] \* nLines;
89. nPixels = ceil(( maxX - minX ) / dResXSize );
90. nLines  = ceil(( minY - maxY ) / dResYSize ) ;
91. adfGeoTransform[0] = minX;
92. adfGeoTransform[3] = maxY;
93. adfGeoTransform[1] = dResXSize;
94. adfGeoTransform[5] = dResYSize;
96. // 创建输出图像
97. GDALDriverH hDriver = GDALGetDriverByName( pszFormat );
98. if (NULL == hDriver)
99. {
100. return 0;
101. }
102. GDALDatasetH hDstDS = GDALCreate( hDriver, pszDstFile, nPixels, nLines, nBandCount, eDataType, NULL );
103. if (NULL == hDstDS)
104. {
105. return 0;
106. }
107. GDALSetProjection( hDstDS,  pszDstWKT);
108. GDALSetGeoTransform( hDstDS, adfGeoTransform );
110. //获得原始图像的行数和列数
111. int nXsize = GDALGetRasterXSize(hSrcDS);
112. int nYsize = GDALGetRasterYSize(hSrcDS);
114. //然后是图像重采样
115. int nFlag = 0;
116. float dfValue = 0;
117. CPLErr err = CE\_Failure;
118. //最邻近采样
119. for (int nBandIndex = 0; nBandIndex < nBandCount; nBandIndex ++)
120. {
121. GDALRasterBandH hSrcBand = GDALGetRasterBand(hSrcDS,nBandIndex+1);
122. GDALRasterBandH hDstBand = GDALGetRasterBand(hDstDS,nBandIndex+1);
123. for (int nRow = 0; nRow < nLines; nRow ++)
124. {
125. for (int nCol = 0; nCol < nPixels; nCol ++)
126. {
127. double dbX = adfGeoTransform[0] + nCol \* adfGeoTransform[1]
128. + nRow  \* adfGeoTransform[2];
129. double dbY = adfGeoTransform[3] + nCol \* adfGeoTransform[4]
130. + nRow  \* adfGeoTransform[5];
132. //由输出的图像地理坐标系变换到原始的像素坐标系
133. GDALGCPTransform(hTransform,TRUE,1,&dbX,&dbY,NULL,&nFlag);
134. int nXCol = (int)(dbX + 0.5);
135. int nYRow = (int)(dbY + 0.5);
137. //超出范围的用0填充
138. if (nXCol < 0 || nXCol >= nXsize || nYRow < 0 || nYRow >= nYsize)
139. {
140. dfValue = 0;
141. }
143. else
144. {
145. err = GDALRasterIO(hSrcBand,GF\_Read,nXCol,nYRow,1,1,&dfValue,1,1,eDataType,0,0);
147. }
148. err = GDALRasterIO(hDstBand,GF\_Write,nCol,nRow,1,1,&dfValue,1,1,eDataType,0,0);
150. }
151. }
153. }


157. if (hTransform != NULL)
158. {
159. GDALDestroyGCPTransformer(hTransform);
160. hTransform = NULL;
161. }
163. GDALClose(hSrcDS );
164. GDALClose(hDstDS );

167. return 1;
168. }

外部调用的实例

**[cpp]** [view plaincopy](http://blog.csdn.net/zhouxuguang236/article/details/27965877)[在CODE上查看代码片](https://code.csdn.net/snippets/372396)

1. GDALDataset\* poDataset = (GDALDataset\*)GDALOpen("E\\248-279\_5\\SCENE01\\IMAGERY.TIF",GA\_ReadOnly);
3. char\* pszWkt1 = (char\*)poDataset->GetProjectionRef();
4. if (0 == strlen(pszWkt1))
5. {
6. pszWkt1 = (char\*)poDataset->GetGCPProjection();
7. }
9. int nGCPCount = poDataset->GetGCPCount();
10. const GDAL\_GCP \*pGCPList = poDataset->GetGCPs();
12. GDAL\_GCP \*pGCPs = new GDAL\_GCP[nGCPCount];
13. memcpy(pGCPs,pGCPList,sizeof(GDAL\_GCP)\*nGCPCount);
15. GDALClose(poDataset);
17. const char\* pszInFile = "E:\\248-279\_5\\SCENE01\\IMAGERY.TIF";
18. const char\* pszOutFile = "E:\\248-279\_5\\SCENE01\\IMAGERY-校正.TIF";
19. ImageWarpByGCP(pszInFile,pszOutFile,nGCPCount,pGCPs,pszWkt1,0,0,0,GRA\_NearestNeighbour,"GTiff");

对 于上述算法的函数，有些读者可能看出来了，算法的瓶颈就是在那个多重循环那，逐点计算是不是效率太低了，的确，这里可以用分块技术来优化一下，甚至有可能 的话，用GPU加速也是不错的选择，我这边只是向大家分享一下怎么去校正带控制点的影像，算是抛砖引玉吧。如果有什么不妥的地方也可以指出来，大家一起讨 论。

**四、后续研究**

1.  如何进行校正显示，不生成新到的校正影像。

2.      如何分块处理

3.      如何用硬件加速，GPU加速

**参考文献**

张永生,刘军. 高分辨率遥感卫星立体影像RPC模型定位的算法及其优化测绘工程, 2004,

刘军,王冬红,毛国苗. 基于RPC模型的IKONOS卫星影像高精度立体定位. 测绘通报, 2004

刘军,张永生,范永弘. 基于通用成像模型———有理函数模型的摄影测量定位方法. 测绘通报, 2003

巩丹超,邓雪清,张云彬. 新型遥感卫星传感器几何模型—有理函数模型. 海洋测绘, 2003

GDAL官方文档以及源码

阮秋琦、数字图像处理（第三版）

来源： <<http://blog.csdn.net/zhouxuguang236/article/details/27965877>>