开发文档

## 功能介绍

1. 读取.shp文件并显示在数据窗口。
2. 正确求出导入数据中的线实体。不同的线文件用不同的颜色表示（线段中的点由明显的表示），并满足图形放大缩小复原，单线文件显示等。
3. 可保存图形为.dxf文件，在.dxf中每条线实体有着自己的标注。
4. 可以选择不同的计算方法进行生成不同的报告，也可以一键生成4种方法。
5. 可生成两个线文件的Hausdroff和Frechet距离，并显示在报告窗口，同时可以人工提供阈值进行线实体之间的匹配。同时可对不同的匹配结果进行精度评价
6. 可将报告报存为.txt文件。
7. 在软件的左下角有切换窗口的快捷按钮，可通过这里进行切换数据窗口，图形窗口，报告窗口。

## 算法设计

1.求出Hausdroff距离

取两个文件，file1由n个F1\_LINE，file2由m个F2\_LINE。

list<list<double>> HH; 两个线文件中的所有线实体之间的Hausdroff距离

list<double> H;

1. For i 🡨0 to n-1
2. For j🡨0 to m-1
3. H.add(Max( H\_line(F1\_LINE(i), F2\_LINE(j)) ,H\_line(F2\_LINE(j), F1\_LINE(1))))
4. End
5. HH.add(H)
6. end
7. Return HH;
8. Double H\_line(line1,line2) {
9. For i 🡨0 to line1的点数-1
10. Double Dmin🡨无穷大
11. For j🡨0 to line2的点数-1
12. Dmin🡨min(Dmin,line1(i)到line2(j)的欧式距离)
13. H.add(Dmin)
14. Return H.Max();
15. }

2.求出Frechrt距离

取两个文件，file1由n个F1\_LINE，file2由m个F2\_LINE。

list<list<double>> HH; 两个线文件中的所有线实体之间的Hausdroff距离

list<double> H;

D():表示欧式距离

1. For i 🡨0 to n-1
2. For j🡨0 to m-1
3. Double[F1\_LINE(i)的点数, F2\_LINE(j)的点数] ff={-1}
4. H.add(F\_line(F1\_LINE(i), F2\_LINE(j), F1\_LINE(i)的点数-1, F2\_LINE(j)的点 数-1,ff))
5. end
6. HH.add(H)
7. end
8. Return HH;
9. Double F\_line(line1,line2,n,m,ff){
10. If(ff[n,m]>-1)
11. Return ff[n,m]
12. Else If(n=0,m=0)
13. Return D(line1(0),line2(0));
14. Else If(n>0,m=0)
15. Return max(F\_line(line1,line2,n-1,m,ff),D(line1(n),line2(m))
16. Elxe If(n=0,m>0)
17. Return max(F\_line(line1,line2,n,m-1,ff),D(line1(n),line2(m))
18. Else If(n>0,m>0)
19. Return max(min(F\_line(line1,line2,n-1,m,ff), F\_line(line1,line2,n- 1,m-1,ff) ,F\_line(line1,line2,n,m-1,ff)),D(line1(n),line2(m))
20. Else
21. ff[n,m]=-1
22. Rerurn ff[n,m]
23. }

3.求出改进hausfroff距离

取两个文件，file1由n个F1\_LINE，file2由m个F2\_LINE。

list<list<double>> HH; 两个线文件中的所有线实体之间的Hausdroff距离

D():表示欧式距离

1. file1有n个线实体 ，file2有m个线实体
2. For i 🡨0 to n -1
3. For j🡨0 to m-1
4. HH.add(Max( H\_line(F1\_LINE(i), F2\_LINE(j)) ,H\_line(F2\_LINE(j), F1\_LINE(1))))
5. Return HH;
6. Double H\_line(line1,line2) {
7. 求出line1和line2的线实体长度,短的线实体放入line\_short,line\_long.
8. 求出line\_long有n个线段
9. 求line\_short的中点mid
10. 定义List<double> imp\_H\_line 存放mid到line\_long线实体每条线段的改进 hausdroff距离
11. For i 🡨0 to n-1
12. 做mid点到line\_long线实体第i+1条线段的垂足F
13. If( F在line\_long线实体第i+1条线段上)
14. {imp\_H\_line.add( D(mid,F))}
15. Else
16. {
17. imp\_H\_line.add( min (D(mid, 第i+1条线段起点), D(mid, 第i+1条线段终点))
18. }
19. end
20. return imp\_H\_line.min();
21. }

4. 求出平均Frechrt距离

取两个文件，file1由n个F1\_LINE，file2由m个F2\_LINE。

list<list<double>> HH; 两个线文件中的所有线实体之间的Hausdroff距离

list<double> H;

D():表示欧式距离

(a,b)<(c,d) 的比较规则（即 min(CD[i,j], CD[i-1,j]，CD[i,j-1])）按照题目要求说明文档的规则

1. For i 🡨0 to n
2. For j🡨0 to m
3. H.add(F\_line(F1\_LINE(i), F2\_LINE(j), F1\_LINE(i)的点数-1, F2\_LINE(j)的点 数-1,ff))
4. end
5. HH.add(H)
6. end
7. Return HH;
8. Double F\_line(line1,line2,n,m){
9. 得到line1各点到line2各点的离散frechet距离double[n,m] FD
10. 得到line1各点到line2各点的欧式距离double[n,m] MD
11. 定义新数组 double[n,m,2] CD
12. For i🡨0 to n
13. For j🡨0 to m
14. CD[i,j,0]=FD[i,j]
15. CD[i,j,1]=MD[i,j]
16. End
17. End
18. 定义list<double> short\_path\_MD
19. Search\_short\_path(CD n-1,m-1, short\_path\_MD)
20. return short\_path\_MD.sum/ short\_path\_MD.count
21. }
22. Void Search\_short\_path(CD i,j, short\_path\_MD)
23. {
24. short\_path\_MD.add(CD[i,j,1])
25. If( i==0&&j==0) { return }
26. If(i==0){ Search\_short\_path(CD i,j-1, short\_path\_MD) }
27. If(j==0){ Search\_short\_path(CD i-1,j, short\_path\_MD) }
28. 记录min(CD[i-1,j-1] , CD[i,j-1], CD[i-1,j])的最小点的坐标(可能是(i,j)，(i-1,j)，(I,j-1))给(ii,jj)
29. Search\_short\_path(CD ii,jj, short\_path\_MD)
30. }

## 主要函数和变量说明

1. 主要变量

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变量名 | 类型 | 含义 | 变量名 | 类型 | 含义 |
| readnum | int | 读取的是几个线文件 | All\_rangeX | List<double> | 文件中X坐标的范围 |
| Shp1 | shpfile | 第一个线文件 | All\_rangeY | List<double> | 文件中Y坐标的范围 |
| Shp2 | shpfile | 第二个线文件 | Hausdroff\_D | List<list<double>> | 两个线文件各个线实体之间的Hausdroff距离 |
| checked\_status | Int | 图形显示的状态 | Frechet\_D | List<list<double>> | 两个线文件各个线实体之间的Frechet距离 |
| filelength; | int | 文件长度 | shapeType | int | 文件的存储信息 |
|  |  |  |  |  |  |

1. 信息类

class shpfile //定义.shp文件的信息格式

{ public int bigtolitter(int num) //对数据进行大端小端的转化（数据存储时的内存地址顺序的不同）}

class shpline //.shp文件中的实体信息（线信息）

public struct PointD {public string name; public double X; public double Y;} //定义文件中的点（点序号，坐标X，坐标Y）

1. 类及主要函数

public partial class 基于Hausdroff\_Frechet距离的线匹配 : Form

{ class discriminant\_result //目视解译信息

private void open\_shpfile\_Click(object sender, EventArgs e)// 打开shpfile文件

void readshp(string shppath ,shpfile shp) //读取shpfile文件

void DATAshow(DataGridView DATA\_shpLine ,shpfile shp) //在数据窗口显示数据

private void 退出程序\_Click(object sender, EventArgs e) //结束程序

private void 绘制图形\_Click(object sender, EventArgs e) //绘制两个.shp文件的信息

private void image\_redraw\_Click(object sender, EventArgs e) //图形重绘

private void save\_image\_Click(object sender, EventArgs e) //保存图形

private void report\_Click(object sender, EventArgs e) // 生成报告

private void save\_report\_Click(object sender, EventArgs e) //保存报告

private void Help\_Click(object sender, EventArgs e) //程序说明及帮助

private void 评价\_Click(object sender, EventArgs e) //生成匹配评价精度结果

}

class CREAT\_REPORT //计算距离的类

{ public List<List<double>> Hausdroff(shpfile shp1,shpfile shp2) //求出两个.shp文件的各个线实体的hausdroof距离

private double H\_line\_D(shpline line1,shpline line2)//线line1到line2的hausdroff距离

private double D(PointD p0,PointD p1)//两点之间的欧式距离

public List<List<double>> improve\_Hausdroff(shpfile shp1, shpfile shp2) //基于改进的Hausdroff\_SMHD距离

private double im\_H\_Line\_D(shpline line1,shpline line2) //计算两条线实体之间改进的hausdroff距离

private List<double> im\_H\_line\_length(shpline line) //计算一条线实体的长度

private PointD line\_midpoint(shpline line, List<double> line\_D) //求线实体的中点

private double perpendicular\_length(PointD p0, PointD p1, PointD p2) //计算点p0到线(p1\_p2)的垂线距离

public List<List<double>> Feachet(shpfile shp1, shpfile shp2)//求出两个.shp文件的各个线实体的Frechet距离

private double Feachet\_line(shpline line1,shpline line2)//线line1到line2的frechet距离

private double P\_P\_D(shpline line1, shpline line2, double[,] p\_p\_fc\_D, int i, int j) //利用递归求frechet距离

public List<List<double>> aver\_Feachet(shpfile shp1, shpfile shp2)//求出两个.shp文件的各个线实体的平均Frechet距离

private double aver\_Feachet\_line(shpline line1, shpline line2) //线line1到line2的平均frechet距离

private double aver\_P\_P\_D(shpline line1, shpline line2, double[,,] p\_p\_fc\_D, int i, int j) //利用递归求平均frechet距离

private void min\_path(double[,,] aver\_p\_p\_fc\_D, int i, int j, List<point\_int> p\_int) //递归寻找最短路径

private point\_int min\_point\_int(double[,,] aver\_p\_p\_fc\_D, point\_int p1, point\_int p2) //(a,b)<=(c,d)的比较规则

}

class DRAW //图形处理的类

{public Bitmap get\_picture(List<double>rangeX,List<double>rangeY, shpfile shp1, shpfile shp2) //在画布上画出图形，并得到画布

public PointF fun1(PointF base\_p,PointD p1, PointF p0,double sc)//实际点坐标转换到画布上的点坐标

public void save\_dxf(string path, List<double> rangeX, List<double> rangeY, shpfile shp1,shpfile shp2)// 保存.dxf文件

void point(StreamWriter sw, float ax, float ay, string layer\_name)//.dxf文件中画点

void text(StreamWriter sw, float ax, float ay, string layer\_name, string txt, float size)//.dxf文件中写文字

void line(StreamWriter sw, float ax, float ay, float bx, float by, string layer\_name)//.dxf文件中画线

void writeline(StreamWriter sw, string a, string b) //.dxf文件中画线

void layer(StreamWriter sw, string layer\_name, int layer\_color) //.dxf文件中定义图层

}

## 主要程序运行界面

##### 4.1运行结果

-------------------基于传统Hausdroff距离线匹配----------------------------

基于传统Hausdroff距离 图层2线实体1 图层2线实体2 图层2线实体3 图层2线实体4 图层2线实体5 图层2线实体6 图层2线实体7

图层1线实体1 826.0004 1454.4323 1351.7509 1913.0175 1684.5120 123.2566 998.7330

图层1线实体2 826.0004 1454.4323 1351.7509 1376.7680 1511.8274 710.8288 657.6697

图层1线实体3 2266.9343 27.5983 499.6440 1845.3816 1953.4327 1477.0929 1010.8591

图层1线实体4 2266.9335 475.4969 412.7487 1845.3807 1783.5152 1477.0921 1010.8582

图层1线实体5 1843.8963 2254.6400 2112.9666 482.4554 952.0239 1964.8635 1749.9328

图层1线实体6 1746.7152 1745.3460 1651.2430 102.7268 976.7533 1881.7361 1640.3301

图层1线实体7 1475.5091 1008.2796 1008.2791 1689.6322 1953.4327 1003.9197 27.5983

--------------------------------------------------------

阈值：30

图层1 ---匹配--> 图层2 距离值

线实体3 ----> 线实体2 27.5983

线实体7 ----> 线实体7 27.5983

-------------------基于改进的Hausdroff\_SMHD距离线匹配----------------

基于改进的Hausdroff\_SMHD距离 图层2线实体1 图层2线实体2 图层线实体3 图层2线实体4 图层2线实体5 图层2线实体6 图层2线实体7

图层1线实体1 344.5015 576.4572 169.0221 255.5957 619.6293 5.7683 530.3502

图层1线实体2 6.4774 894.1446 171.2207 571.4431 924.5497 347.5743 201.1464

图层1线实体3 243.9600 2.5363 171.5753 1520.5424 1728.4781 590.9484 1010.8591

图层1线实体4 2.7767 202.0507 171.5753 1574.6258 1644.8005 346.5853 477.1534

图层1线实体5 4.8659 1591.5303 1393.1678 173.5808 248.0057 496.6841 722.2511

图层1线实体6 88.9565 1549.2147 1298.2031 3.9700 234.3848 96.4697 725.1329

图层1线实体7 203.5811 1008.2796 591.6575 637.0617 991.0534 532.7215 0.9653

--------------------------------------------------------

阈值：30

图层1 ---匹配--> 图层2 距离值

线实体1 ----> 线实体6 5.7683

线实体2 ----> 线实体1 6.4774

线实体3 ----> 线实体2 2.5363

线实体4 ----> 线实体1 2.7767

线实体5 ----> 线实体1 4.8659

线实体6 ----> 线实体4 3.9700

线实体7 ----> 线实体7 0.9653

------------------基于离散Frechet距离线匹配----------------------------

基于离散Frechet距离 图层2线实体1 图层2线实体2 图层2线实体3 图层2线实体4 图层2线实体5 图层2线实体6 图层2线实体7

图层1线实体1 826.0004 1749.9328 1351.7509 1913.0175 2196.9851 1995.0950 1749.9328

图层1线实体2 2266.9342 1454.4323 1454.4320 1845.3816 1524.6636 710.8288 657.6697

图层1线实体3 2598.9629 27.5983 845.9568 2137.5387 2003.3195 1477.0929 1010.8591

图层1线实体4 2649.6749 562.8758 556.0720 2244.5069 1783.5152 1477.0921 1010.8582

图层1线实体5 1843.8963 2561.2647 2112.9666 482.4554 1277.2253 2704.4619 2561.2647

图层1线实体6 1746.7152 2018.4136 1651.2430 102.7268 1130.2350 2264.3270 2018.4135

图层1线实体7 2598.9629 1008.2796 1008.2791 2137.5387 2003.3195 1003.9197 27.5983

--------------------------------------------------------

阈值：30

图层1 ---匹配--> 图层2 距离值

线实体3 ----> 线实体2 27.5983

线实体7 ----> 线实体7 27.5983

------------------------基于平均Frechet距离线匹配--------------------------

基于平均Frechet距离 图层2线实体1 图层2线实体2 图层2线实体3 图层2线实体4 图层2线实体5 图层2线实体6 图层2线实体7

图层1线实体1 377.8940 756.6067 552.2862 903.7848 1232.5870 990.1564 1007.0548

图层1线实体2 941.8110 820.1637 798.5617 707.7797 848.9100 388.9718 395.9318

图层1线实体3 1123.4896 15.2458 424.4251 1736.3988 1743.1728 735.3673 600.6201

图层1线实体4 1127.7150 282.8847 279.4829 1772.0543 1669.9043 599.8209 734.4391

图层1线实体5 614.6403 1986.0602 1715.6212 300.2433 630.8863 1480.9653 1400.2747

图层1线实体6 660.1280 1685.7733 1423.1568 59.6507 733.9513 1238.0627 1060.0166

图层1线实体7 789.4280 607.4168 812.0061 1195.9251 1377.8057 793.6014 12.7543

--------------------------------------------------------

阈值：30

图层1 ---匹配--> 图层2 距离值

线实体3 ----> 线实体2 15.2458

线实体7 ----> 线实体7 12.7543

匹配评价

---------------------------目视解译 线匹配---------------------------------

图层1 --> 图层2

线实体1 --> 线实体6

线实体2 --> 线实体1

线实体3 --> 线实体2

线实体4 --> 线实体1

线实体5 --> 线实体1

线实体6 --> 线实体4

线实体7 --> 线实体7

------------------------------线匹配结果评价---------------------------------

匹配模型：基于传统Hausdroff距离

图层1 --> 图层2

线实体3 --> 线实体2

线实体7 --> 线实体7

匹配精度（或叫匹配查准率）Matching precision（P）100.000%

匹配查全率 Matching recall（R）:28.571%

-------------------------------------------------------------------------------

匹配模型：基于改进的Hausdroff\_SMHD距离

图层1 --> 图层2

线实体1 --> 线实体6

线实体2 --> 线实体1

线实体3 --> 线实体2

线实体4 --> 线实体1

线实体5 --> 线实体1

线实体6 --> 线实体4

线实体7 --> 线实体7

匹配精度（或叫匹配查准率）Matching precision（P）100.000%

匹配查全率 Matching recall（R）:100.000%

-------------------------------------------------------------------------------

匹配模型：基于离散Frechet距离

图层1 --> 图层2

线实体3 --> 线实体2

线实体7 --> 线实体7

匹配精度（或叫匹配查准率）Matching precision（P）100.000%

匹配查全率 Matching recall（R）:28.571%

-------------------------------------------------------------------------------

匹配模型：基于平均Frechet距离

图层1 --> 图层2

线实体3 --> 线实体2

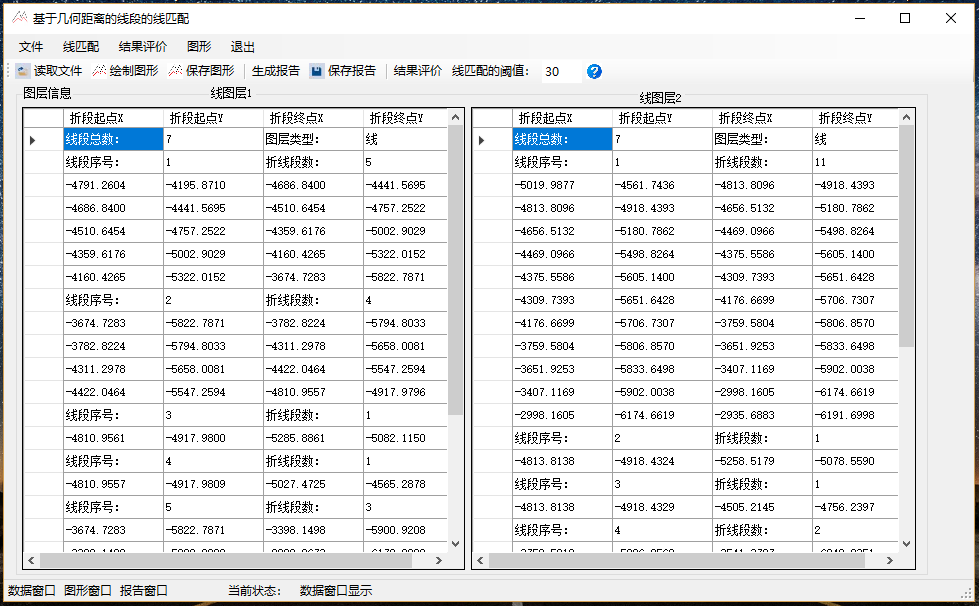
线实体7 --> 线实体7

匹配精度（或叫匹配查准率）Matching precision（P）100.000%

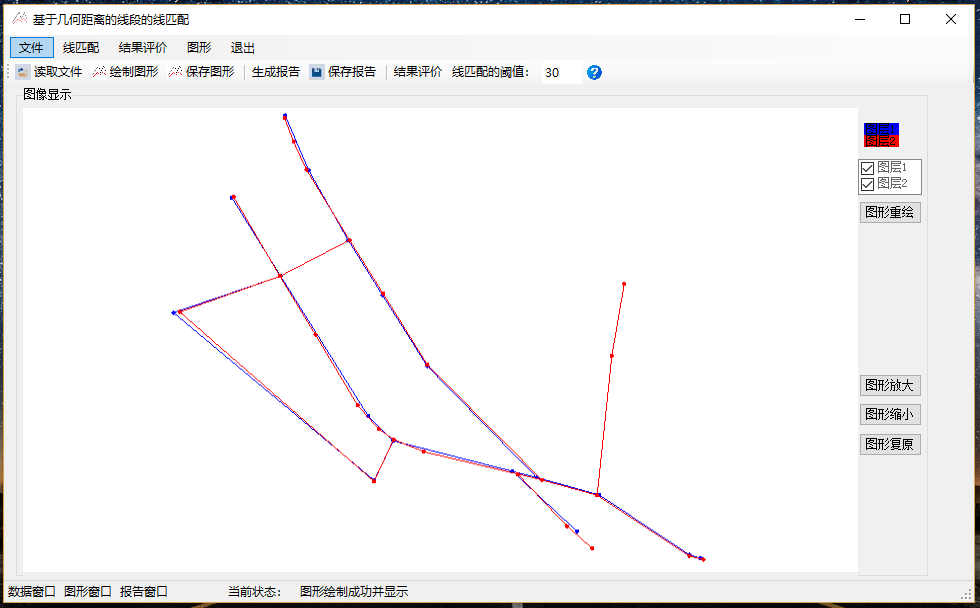
匹配查全率 Matching recall（R）:28.571%

-------------------------------------------------------------------------------

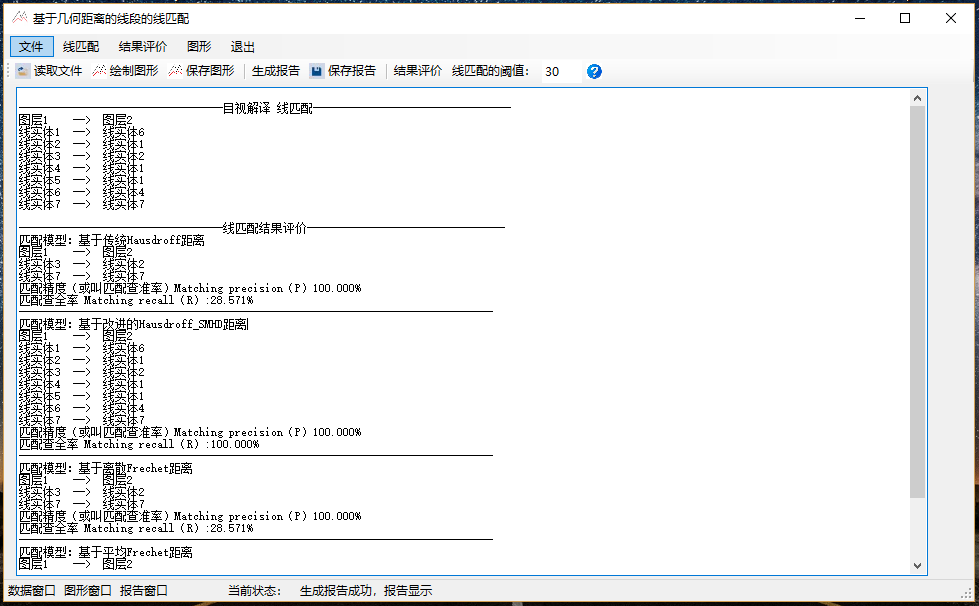
##### 4.2 用户界面



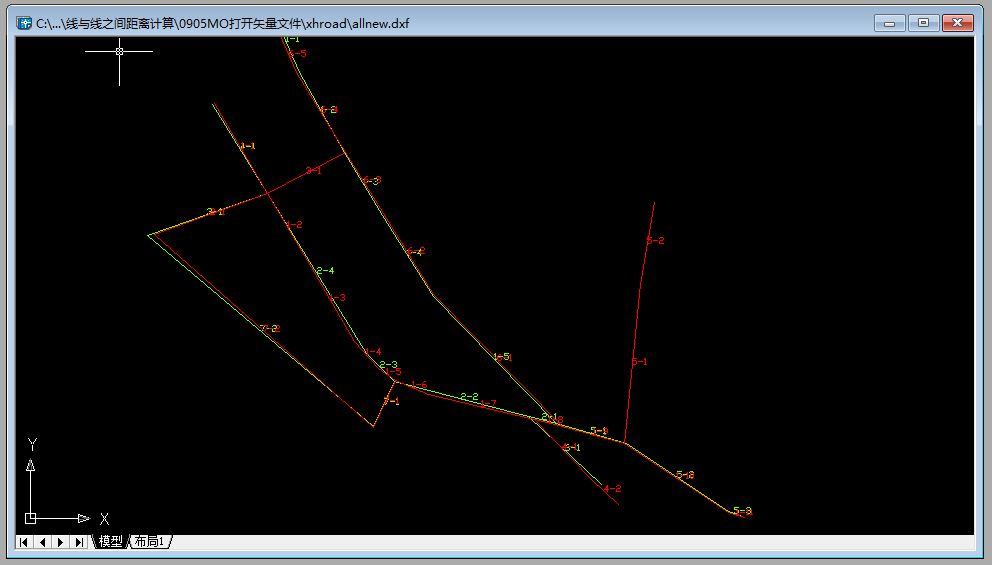
1. 图十六 数据显示



1. 图十七图形显示
2. 图十八 计算报告



1. 图十九 匹配结果评价



1. 图二十 在CAD打开保存的图