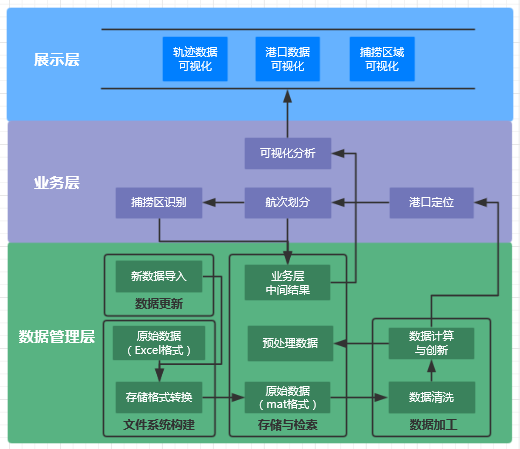
对于毕业设计的构想，目前只完成到捕捞区的识别部分。《OceanRoads》中的投票算法代码比较复杂，还在梳理，而且没有想好应该如何在界面中展示。

下面这是目前为止的体系架构图

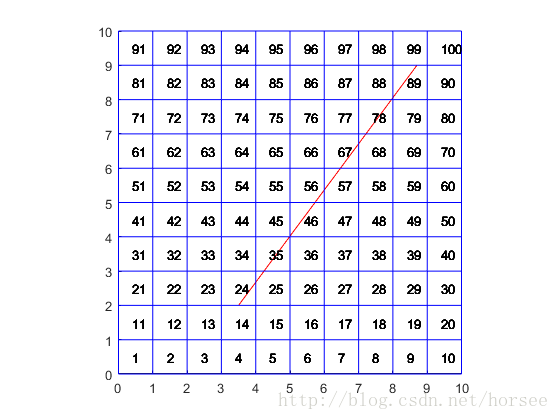


重新从头编写了过去的代码，具体工作如下：

1、数据格式转换与计算

2、港口定位

之前采用的方法是同时参考a.采样数据的网格点密度；b.<瞬时速度，船艏向>==<0，0>。这里存在三个问题：a.特征不是港口独有，海上停船同样存在这些特征；b.之前一直使用的是31艘船的数据集包含瞬时速度和船艏向字段，但之后黄海广师兄发来的175艘船的数据集中缺少瞬时速度和船艏向字段；c. 对于遗漏的部分港口进行了手动添加。

 在此基础上，考虑寻找港口独有的、仅基于经纬度坐标的、能囊括能多港口的特征。为此提出**五种思路**：

a.网格法-累计停留时间：计算在某一区域内的累计停留时间，而非单纯依靠数据点的数据量；不明白a和原来的方法的区别。

b.坐标重复：相邻两条记录的经纬度坐标值相同，表明船舶近似停泊，这一状态在港口发生的概率应远大于海上发生的概率或者异常数据的概率；

c.超低速：考虑到两个航次之间会关机（数据采集停止），将极大影响前两种方法对港口的判断，假设关机后船舶保持不动，利用关机前和开机后数据计算得到的平均速度应该极小，以此作为依据判断港口；

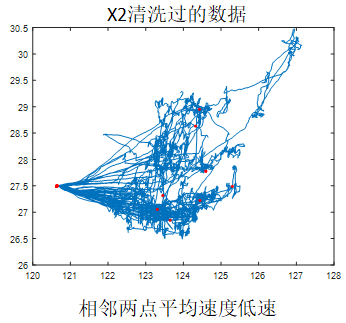
d.计算网次:根据渔船速度变化，识别网次，进而识别港口；什么叫网次呢？

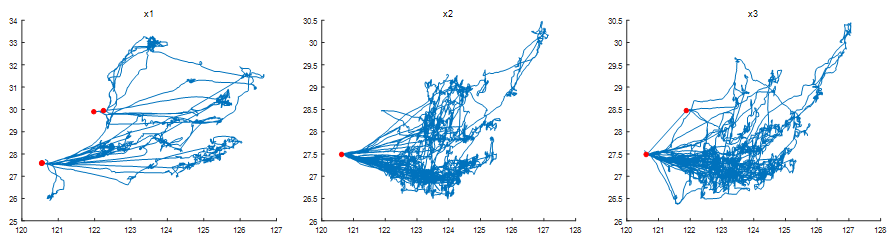
e.根据轨迹的周期性变化判断。其中前三种方法（a/b/c）都是在数据点上做文章，方法d是分类问题，

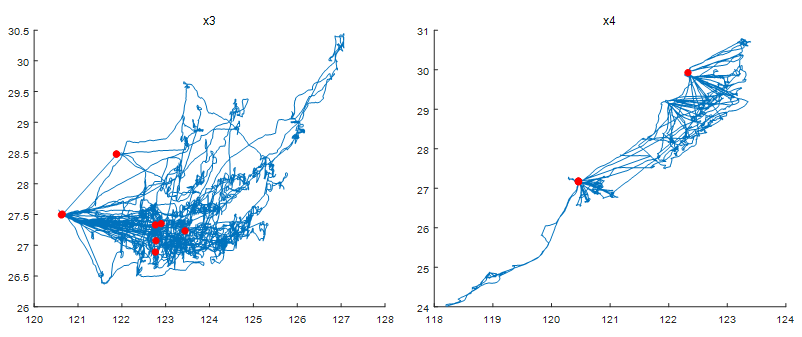
方法e是根据轨迹形状进行判断。

2.1累计停留时间

累计停留时间的计算，是根据两两数据点连线在每个网格的长度对时间进行加权。计算量极大：一方面，计算量随着网格密度减小成指数级增长，这句话的逻辑不对啊；另一方面，无效计算大。遂放弃这一方法。

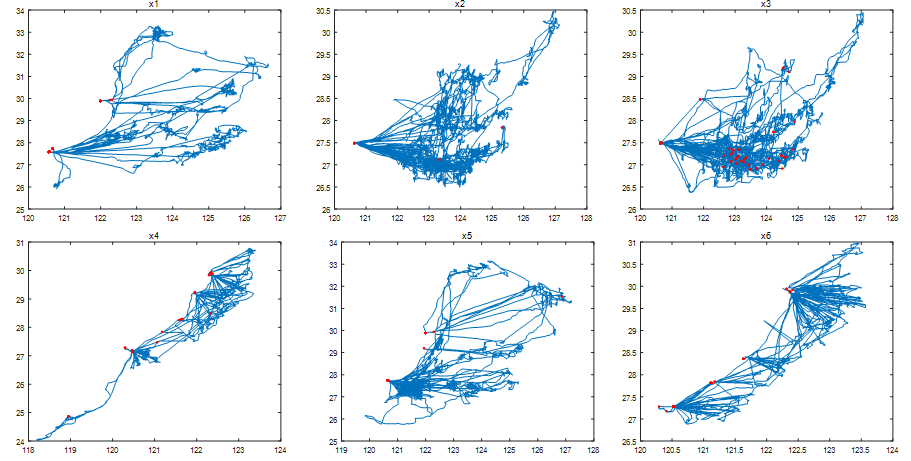
 2.2坐标重复。

 对于x2（第二艘渔船）数据进行分析，共存在48次坐标重复情况，其中港口38次（右图）。同样通过网格法，设置阈值判断港口。这里，网格大小影响着港口判断的经度，定为1’ ×1’，相当于1平方海里的误差。对于不同渔船数据的验证见下图。对于港口点较少的x1/x2/x3而言，判断准确；但是对于多港口的x4，只能判断出个别主要港口。为了识别更多港口，放宽阈值限制会导致下图中x3/x4的情况——难以平衡阈值与港口识别准确的关系。这是因为一方面在海上同样存在停船现象，另一方面，渔船在部分港口停船时间短，访问次数少。

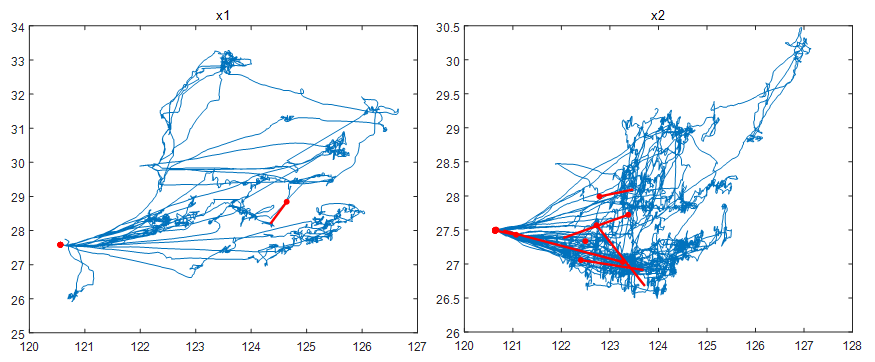
 这一方法目前不能使用。但是，当数据量累计到足够大，该方法可能是目前能想到的效率最高的港口识别方法。

2.3、超低速

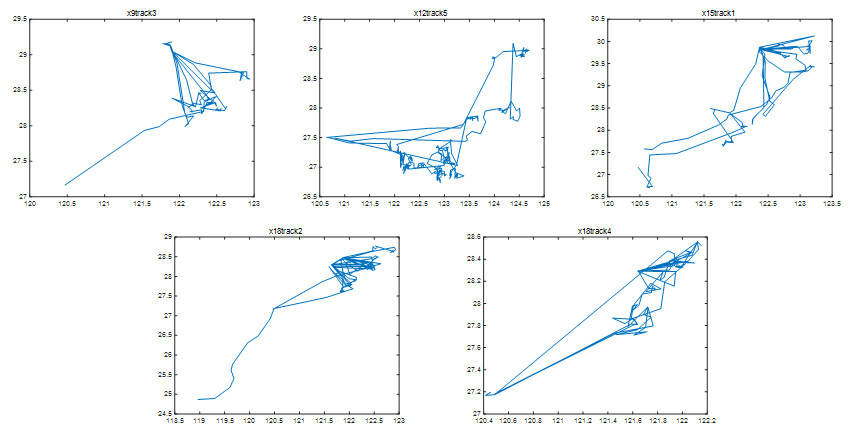
在对数据x2的研究中，有两种方法确定超低速的阈值。动态方法：10倍最低速；静态方法：用常数作为所有渔船的超低速阈值。经过分析发现，最低速基本发生在7到9月，对应着禁渔期（当然也存在在禁渔期内的数据）。时间间隔短则半个月，长则接近两个月，不稳定。所以，最后确定用静态常量作为阈值。

下图中红点是用常数1.5\*1e4节作为阈值保留下的数据。

在寻找超低速的阈值时遇到的问题是数据缺失，下图中的红色线段是以相邻数据间隔3天为标准筛选得到的，而正常的数据间隔为3分钟。这还没有考虑更大幅度的数据缺失问题。观察左侧x1的数据可以发现，在两个明显的港口中，上方的港口不存在相隔三天的数据点，有两种可能：船舶始终开机；开关机间隔不超过3天。我认为前一种可能性比较小，而后一种则意味着船舶靠港时间不超过三天，也为“超低速”这种方法带来了困难。

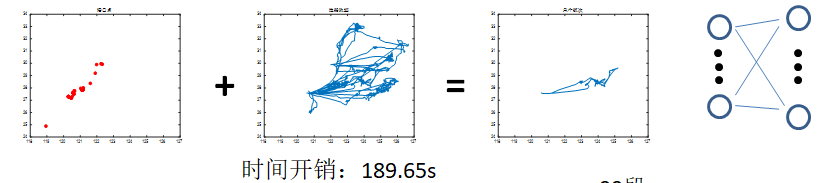


为了避免这些干扰，将阈值设置偏小，这样一来部分港口就难以识别。于是不再单独用一艘渔船识别其港口，而是用全部渔船识别出的所有港口视为公共港口。

 右图是通过不同阈值判断港口确定了最终阈值设为2\*1e5节，共获得43个港口点。这里用了网格法去重，存在小于0.13nm的误差。相对来说效果不错，但仍然存在下图中因港口未识别或识别不准确造成的多个航次未被划分成功的情况。

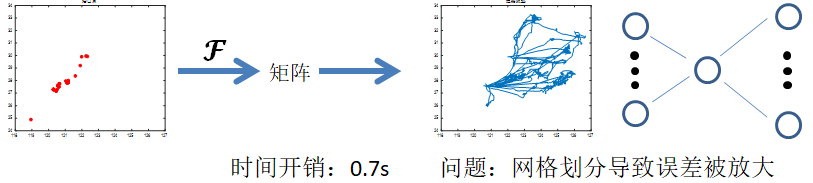
3、航次划分

由于识别出的港口是所有渔船共有的港口，直接拿一艘船的n条数据与m个港口比较，共要进行n\*m次计算。31条船划分航次的运算时间是189s。

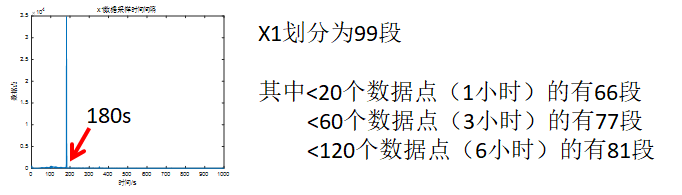


考虑这个时间开销本身的不足，以及随着数据增加，运算时间会不断增大。提出新的方法。用一个矩阵表示地图，将港口和数据映射到上面，时间开销0.7s，相当于用空间换时间。

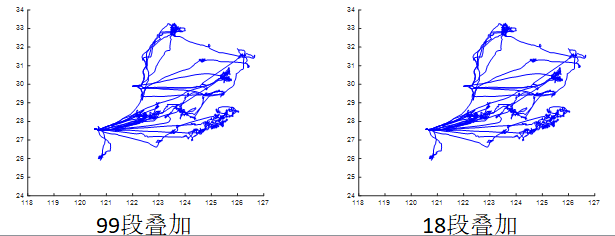
空间换时间写到系统实现中



两种方法（时间开销189s和时间开销0.7s），对于数据x1（第一艘渔船的数据），前者识别29个航次，而后者识别了99个航次，这是由于网格法扩大了港口范围，尤其是将港口附近的轨迹切分成多段。已知正常数据间隔为3分钟，统计如下：

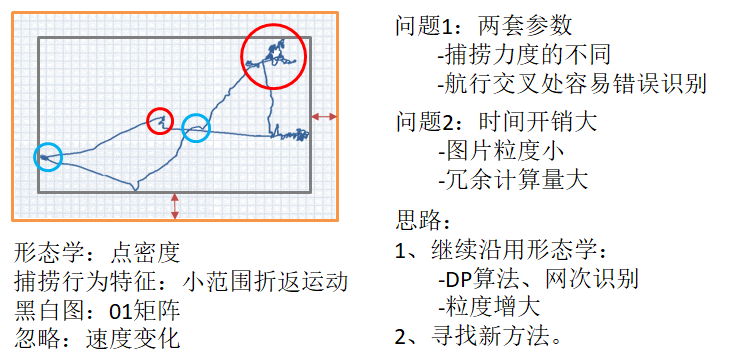


对小于120个数据点的轨迹进行抛除，比较筛选前后如下图，基本保留了全部的航次。



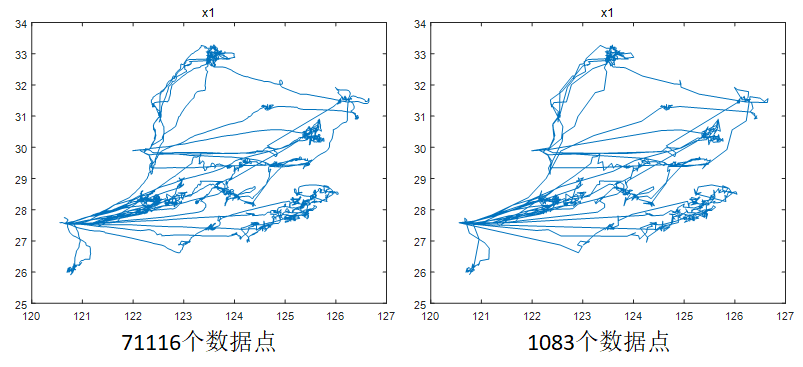
4、捕捞区识别

经过分析我们之前提出的形态学方法，发现的问题及解决方法如下：

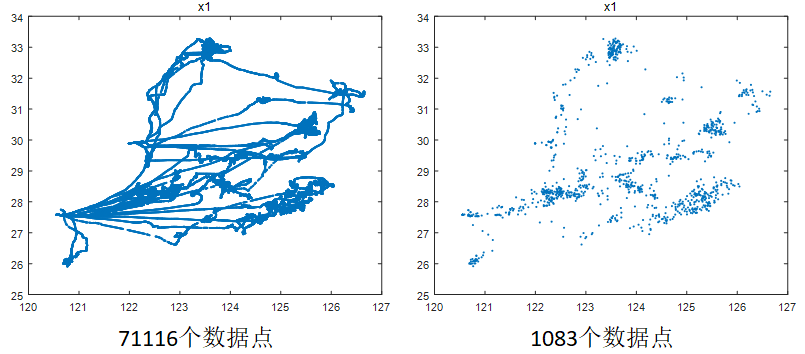


问题的重点在于解决时间开销大->图像计算量太大->减小矩阵，意味着每个像素代表的实际区域变大，在图像统计时会出现错误识别和遗漏的情况。于是考虑进行数据压缩。

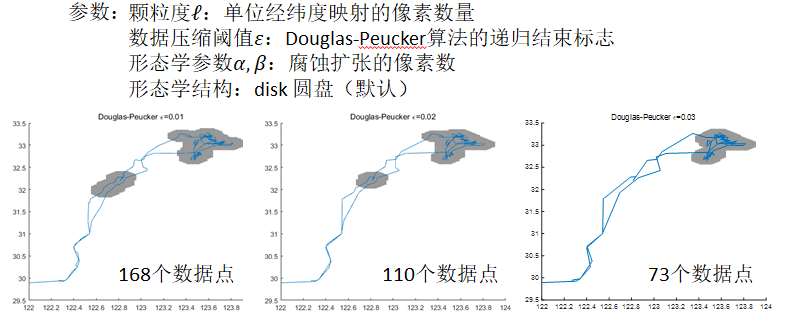
下图是在是用Douglas-Peucker算法压缩前后的轨迹图，可以看到保留了绝大部分的信息。



而观察上图对应数据点图（下图），虽然数据被大量的刨除，但直观上看，航行数据的比利明显高于捕捞状态，而我们的形态学方法也恰恰是为了忽略航行轨迹。另一个优点是，避免了航行状态交错引起局部点密度所导致的捕捞区错误识别。过去，就是为了降低航行轨迹被错误识别成捕捞区而不断的增大图像的像素，导致了高计算量。

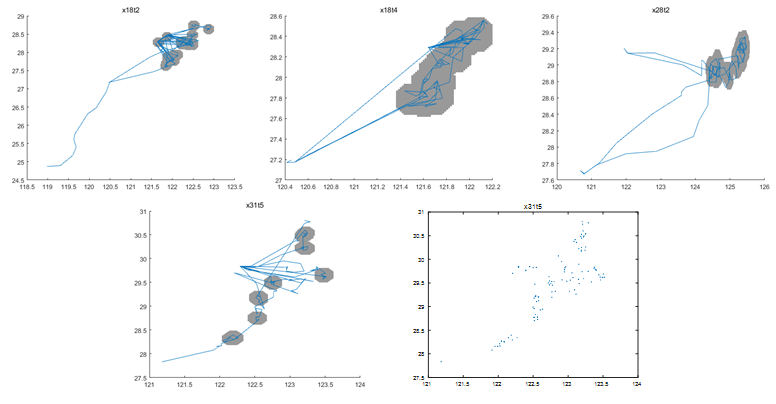


捕捞区识别算法中的参数如下



再引入数据压缩方法后，对于航行状态的捕捞区错误识别问题，可以通过增大压缩率来有效解决，而不必通过增加像素，这有效降低了捕捞区识别算法的时间开销，而Douglas-Peucker算法本身效率比较高，可以忽略不计。

但在检查各个航次的捕捞区识别过程中发现，之前的航次划分错误保留到了这个环节无法解决（下图）。



这个图看不明白错误是什么