首先需要确定坐标系，在哪一种坐标系下面进行计算？

WGS84坐标系下还是要GCJ02坐标系

距离=速度X时间

distince=speed \* time

主要目的：

在处理卡口数据的过程中，遇到了一个问题：对于每个交叉口只知道其中心点的经纬度，而不知道每个进口道停车线的经纬度，对不同的进口道，难以将轨迹数据分开处理。

因此，采用了一种简化的方法，假设了每个交叉口中心点到每个停车线的距离为m米，根据相邻的两个交叉口中心经纬度，计算了路段方位角，进而根据交叉口中心点到每个停车线的距离、路段走向方位角、路段中心店经纬度这三个信息，计算出了每个进口道停车线的经纬度。其中用到的代码如下：

1.根据起始点经纬度、目标点经纬度计算它们之间的方位角

def calc\_azimuth(lat1, lon1, lat2, lon2):

lat1\_rad = lat1 \* math.pi / 180

lon1\_rad = lon1 \* math.pi / 180

lat2\_rad = lat2 \* math.pi / 180

lon2\_rad = lon2 \* math.pi / 180

y = math.sin(lon2\_rad - lon1\_rad) \* math.cos(lat2\_rad)

x = math.cos(lat1\_rad) \* math.sin(lat2\_rad) - \

math.sin(lat1\_rad) \* math.cos(lat2\_rad) \* math.cos(lon2\_rad - lon1\_rad)

brng = math.atan2(y, x) \* 180 / math.pi

return float((brng + 360.0) % 360.0)

2.根据起始点经纬度、距离、方位角计算目标点经纬度

参考了这篇博客的计算方法:https://blog.csdn.net/sinat\_32857543/article/details/107207553

def calc\_situation(lon1, lat1, deg, dis):

arc = 6371.393\*1000

lon2 = lon1 + dis\*math.sin(deg)/(arc\*math.cos(lat1)\*2\*math.pi/360)

lat2 = lat1 + dis\*math.cos(deg)/(arc\*2\*math.pi/360)

return lon2,lat2

注意：距离dis的单位是米，deg是方位角，需要化为弧度。lon1，lat1是起始点经纬度信息。

情况1：无转向计算

情况2：有转向时算(仅水平转向计算，仅垂直转向计算，水平垂直转向同时计算)

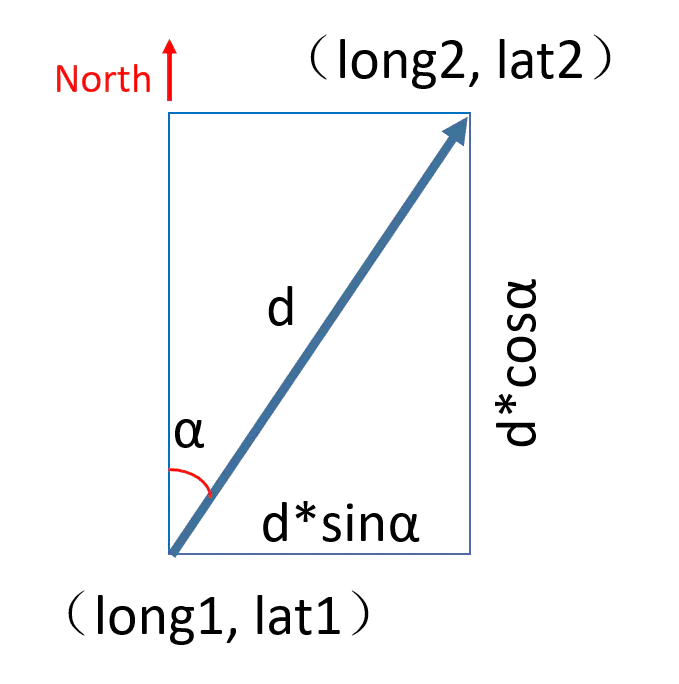
已知一点经纬度和距离，计算另一点的经纬度

因为有需求，在知道一点经纬度和距离的情况下，需要知道另一点的经纬度。

之前在网上也有查找，也有不少的案例，但是多数都是语焉不详，看不太明白。后来自己整理和重新再思考了下，这里给出一个自认为说得还算比较简明扼要的版本。

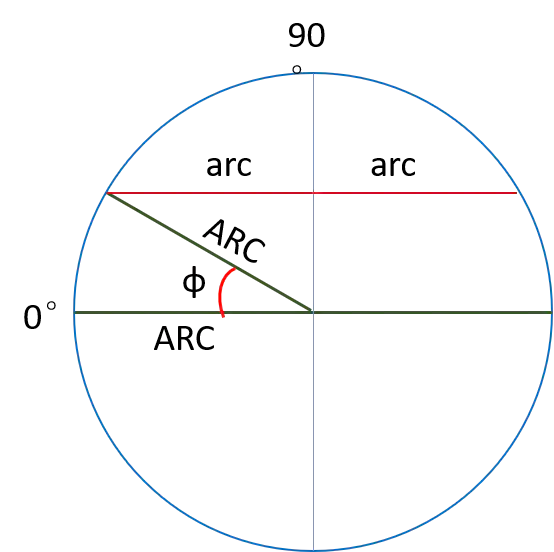
话不多说，具体如下：

假设方位角是α， 那从点1到点2的平移距离分别如下所示d\*sinα， d\*cosα。 这里正北为0度。基中点1经纬度（long1, lat1）和距离d是已知的。 求点2的经纬度（long2,lat2）



还有一个隐藏的信息，就是点1所在的纬度，其实也是一个有用的角度，通过它可以知道当前纬度的那个切面的半径长度，下图红线所示用arc表示。这里假设地球是近似球体，赤道圆的半径是ARC，侧从下图中可以得出：

就是知道φ是当前点1的纬度。则当前的纬度的切面半径 arc = ARC\*cos(φ) ，其中φ其实就是当前的方位的纬度值，即arc = ARC\*cos(lat1)



这里还要再讨论下地球半径，其实地球是一个椭球体。

极半径 从地心到北极或南极的距离，大约3950英里(6356.9088千米)（两极的差极小，可以忽略）。

赤道半径 是从地心到赤道的距离，大约3963英里(6377.830千米)。

如果只是做近似计算的，我们这里取平均距离，平均半径 大约3959英里(6371.393千米) 。这个数字是地心到地球表面所有各点距离的平均值。

这里取平均半径那么ARC=6371.393\*1000（米）

通过上面的知识铺垫后， 计算就简单化了，

【计算思路】

1. 计算第二点的经度，就是 水平平移的距度（d\*sinα）除以 当前纬度切面周长（2π\*arc），再每乘以360度) ，就知道了水平横向平移了多少度，再加上long1，就是long2的值了。

2. 计算第二点的纬度，比较简单，就是, 垂直平移的距离d（d\*cosα）除以 地球纵向周长，再乘上360度，就知道纵向平移了多少度，再加上lat1，就知道lat2的值了。

long2 = long1 + d\*sinα/[ARC\*cos(lat1)\*2π/360]

lat2 = lat1 +d\*cosα/ (ARC \*2π/360)

BTW, 最后再啰嗦一句，如果你是要通过编程实现的，一定要注意所用编程语言中，三角函数的值用的是度数还是弧度，不然也是得不到正确的结果的。

C#代码实现：

/// <summary>

/// 计算移动后的经纬度

/// </summary>

/// <param name="lon">经度</param>

/// <param name="lat">纬度</param>

/// <param name="a">方位角（弧度）</param>

/// <param name="dst">移动距离</param>

/// <returns></returns>

public double[] LongLatOffset(double lon, double lat, double a, double dst)

{

double arc = 6371.393 \* 1000;

lon += dst \* Math.Sin(a) / (arc \* Math.Cos(lat) \* 2 \* Math.PI / 360);

lat += dst \* Math.Cos(a) / (arc \* 2 \* Math.PI / 360);

return new[] { lon, lat };

}

数据有两类

1已经有GPS坐标数据，不需要计算。直接使用经纬度定位即可。

2无GPS坐标数据，需要先查找最后一次GPS坐标数据，Last Position(lon,lat)

通过将已经经过的点进行计算，得到当前位置。

最后一次GPS点数据通过先查缓存得到。

缓存中没有则从数据库中获取。数据库获取后放到缓存中。

计算时同步执行即可。