

地图定位误差

地图定位误差

前言

常用坐标系介绍

北斗定位系统坐标

坐标系的相互转换

总结

前言

腾讯，高德地图使用的坐标与上位机的坐标不一样，我们的上位机调用了天地图的API（WGS-84标准），所以转换出来的数据会有差异。在获取到GPS/北斗定位信息后，要把GPS/北斗的定位信息转化成天地图（WGS-84标准）的坐标之后再进行转化。

常用坐标系介绍

WGS-84(GPS)

国际标准，一般从国际标准的GPS设备获取的坐标都是WGS-84，以及国际地图提供商使用的坐标系。

GCJ-02

中国标准，国测局02年发布的坐标系。又称“火星坐标”。在中国，必须至少使用“GCJ-02”对地理位置进行首次加密。比如谷歌中国、高德、腾讯都在用这个坐标系。

BD-09

百度标准，在“GCJ-02”的基础上进行二次加密。

北斗定位系统坐标

如果有对接过北斗定位系统的小伙伴可能会发现一个问题，北斗坐标经纬度好像都被乘了100？

```
$GNRMC,083735.000,A,2429.53531,N,11810.78036,E,0.54,171.11,190621,A*7E
```

这个，就是我们北斗定位系统接收到的坐标值，可以发现，北纬是2429.53531，东经是11810.78036，倘若我们直接将这个经纬度除100，北纬24.2953531，东经118.1078036，在百度或者高德地图中坐标偏移的离谱。

高德地图定位：

○ 按关键字搜索 ● 按坐标搜索

118.1078036,24.2953531 搜索 118.204621,24.216474

坐标获取结果:

118.21286,24.240894

百度地图定位：



可以发现，两者的距离都偏移了几十公里，这个原因就在于，国内，无论是高德还是百度，它们所使用的经纬度值，都是经过加密之后的值，而不是直接用GPS的实际经纬度来定位的，因此，我们如果想要将北斗定位系统的坐标系，用于商业地图上，则需要将它经过对应的加密转换处理才能够正确的使用。

坐标系的相互转换

首先，要强调一点，其实我们北斗定位的坐标系值并不是简单的100倍关系，而是需要做一次度分秒的转换的。则我们获取的北斗坐标值，北纬2429.53531，东经11810.78036，需要做如下计算： $24 + (29.53531/60) \approx 24.49225517$ $118 + (10.78036/60) \approx 118.17967267$

这两个值才是我们应该带入转换公式的坐标值，如果直接用100倍关系的值去转换，定位一样会偏差，所以需要先转化为gps84标准的坐标值。

```
def str_To_Gps84(self, in_data1, in_data2):  
    len_data1 = len(in_data1)  
    str_data2 = "%05d" % int(in_data2)  
    temp_data = int(in_data1)  
    symbol = 1  
    if temp_data < 0:  
        symbol = -1  
    degree = int(temp_data / 100.0)
```

```

        str_decimal = str(in_data1[len_data1-2]) + str(in_data1[len_data1-1]) + '.' +
str(str_data2)
        f_degree = float(str_decimal)/60.0
        # print("f_degree:", f_degree)
        if symbol > 0:
            result = degree + f_degree
        else:
            result = degree - f_degree
        return result

```

WGS-84转化成GCJ-02:

```

def gps84_to_gcj02(self, lat, lon):
    if (self.outofChina(lat, lon)):
        return [0, 0]
    dLat = self.transformLat(lon - 105.0,
lat - 35.0)
    dLon = self.transformLon(lon - 105.0,
lat - 35.0)
    radLat = lat / 180.0 * self.PI
    magic = math.sin(radLat)
    magic = 1 - self.EE * magic * magic
    sqrtMagic = math.sqrt(magic)
    dLat = (dLat * 180.0)/((self.A * (1 -
self.EE))/(magic * sqrtMagic) * self.PI)
    dLon = (dLon * 180.0) / (self.A /
sqrtMagic * math.cos(radLat) * self.PI)
    mgLat = lat + dLat
    mgLon = lon + dLon
    return [mgLat, mgLon]

```

GCJ-02转化成BD-09:

```
def gcj02_to_bd09(self, gg_lat, gg_lon):
    x = gg_lon
    y = gg_lat
    z = math.sqrt(x * x + y * y) + 0.00002
    * math.sin(y * self.PI)
    theta = math.atan2(y, x) + 0.000003 *
    math.cos(x * self.PI)
    bd_lon = z * math.cos(theta) + 0.0065
    bd_lat = z * math.sin(theta) + 0.006
    return [bd_lat, bd_lon]
```

其他中间需要的函数和变量:

```
self.PI = 3.1415926535897932384626
self.A = 6378245.0
self.EE = 0.00669342162296594323

def outofChina(self, lat, lon):
    if (lon < 72.004 or lon > 137.8347):
        return True
    if (lat < 0.8293 or lat > 55.8271):
        return True
    return False

def transform(self, lat, lon):
    if (self.outofChina(lat, lon)):
        return [lat, lon]
```

```

        dLat = self.transformLat(lon - 105.0,
lat - 35.0)
        dLon = self.transformLon(lon - 105.0,
lat - 35.0)
        radLat = lat / 180.0 * self.PI
        magic = math.sin(radLat)
        magic = 1 - self.EE * magic * magic
        sqrtMagic = math.sqrt(magic)
        dLat = (dLat * 180.0) / ((self.A * (1 -
self.EE)) / (magic * sqrtMagic) * self.PI)
        dLon = (dLon * 180.0) / (self.A /
sqrtMagic * math.cos(radLat) * self.PI)
        mgLat = lat + dLat
        mgLon = lon + dLon
        return [mgLat, mgLon]

```

```

def transformLat(self, x, y):
    ret = -100.0 + 2.0 * x + 3.0 * y + 0.2
* y * y + 0.1 * x * y + 0.2 *
math.sqrt(abs(x))
    ret += (20.0 * math.sin(6.0 * x *
self.PI) + 20.0 * math.sin(2.0 * x *
self.PI)) * 2.0 / 3.0
    ret += (20.0 * math.sin(y * self.PI) +
40.0 * math.sin(y / 3.0 * self.PI)) * 2.0 /
3.0
    ret += (160.0 * math.sin(y / 12.0 *
self.PI) + 320 * math.sin(y * self.PI /
30.0)) * 2.0 / 3.0
    return ret

```

```
def transformLon(self, x, y):  
    ret = 300.0 + x + 2.0 * y + 0.1 * x * x  
    + 0.1 * x * y + 0.1 * math.sqrt(abs(x))  
    ret += (20.0 * math.sin(6.0 * x *  
self.PI) + 20.0 * math.sin(2.0 * x *  
self.PI)) * 2.0 / 3.0  
    ret += (20.0 * math.sin(x * self.PI) +  
40.0 * math.sin(x / 3.0 * self.PI)) * 2.0 /  
3.0  
    ret += (150.0 * math.sin(x / 12.0 *  
self.PI) + 300.0 * math.sin(x / 30.0 *  
self.PI)) * 2.0 / 3.0  
    return ret
```

高德地图请用的GCJ-02坐标系转换结果，百度坐标系请用BD-09坐标系转换结果。

总结

谷歌地图那些，用的是WGS-84坐标系，北斗定位系统给到的坐标值要做一次度分秒的转换，转换后才是84坐标，像高德地图那些用的是GCJ-02火星坐标系，等于要先做一次GCJ-02加密，坐标系才能在高德地图用，百度地图则需要再做一次BD-09的加密。