V1.0.0 2012.2.21

[bin.zhang@autonavi.com](mailto:bin.zhang@autonavi.com)

1. 删除CallBack\_PointFixed的返回值错误定义

润波：

通过上次来上海的交流，我觉得你们已经了解dvr项目对道路轨迹匹配的基本需求，在这里我把需求细化一下到具体的实现功能接口，你看看，如有出入请及时与我们讨论并予以更正。

2011/12/17 [bin.zhang@autonavi.com](mailto:bin.zhang@autonavi.com)

# 1.道路轨迹修正

1.1描述:

1. Dvr设备在采集gps数据和视频数据时，由于设备并不具有惯导辅助，所以采集的轨迹当处在闹市、高架等区域出现gps丢失和漂移的情况，这对我们后期的影像和轨迹处理带来了困难。
2. 提供一种方法，输入gps轨迹，抛弃飞点，修正漂移坐标并输出给调用者。



1. 对于gps短时间丢失情况，希望能提供一种gps坐标点补偿的能力。



1. A图中P1到P2点轨迹之间出现gps丢失情况，如果是短时间的丢失，希望能提供一种坐标点补偿的能力，处理结果返回连续的修正之后的轨迹，如B图。
2. 如果轨迹无法匹配到路网（新路），则处理之后轨迹尽量显得光滑

1.2数据结构

1. GpsPoint\_t

|  |  |  |
| --- | --- | --- |
| 描述一个gps轨迹点信息 | | |
| 字段 | 类型 | 说明 |
| lon | float | 精度 |
| lat | float | 纬度 |
| timetick | Uint32 | Unix timestamp 1970 ~ |
| speed | float | 速度 |
| angle | float | 角度 |
| Delta | Void\* | 用户层私有数据作为上下文传递  (Fileoffset Gps时间对应影像文件偏移量) |
| Speed,angle也许并不能获取 | | |

2. CallBack\_PointFixed

typedef void (CallBack\_PointFixed\*)(GpsPoint\_t\* pts, size\_t size);

函数: void (\*)(GpsPoint\_t\* pts, size\_t size)

功能: 地图轨迹匹配完成回调返回给用户的接口

参数: pts – 轨迹点地址; size – 轨迹点数量

返回: void

1.3 接口

a.) gpsfix\_begin

函数: void gpsfix\_begin(CallBack\_PointFixed user);

功能: 开始gps校正处理

参数: user – 完成gps修复之后传递给用户的接收接口

返回: void

b.) gpsfix\_end

函数 :　void gpsfix\_end();

功能: 结束一次校正处理,匹配接口接收到用户调用gpsfix\_end之后，将内部残留的gps点即刻全部通过CallBack\_PointFixed返回给用户，并且再追加调用一次CallBack\_PointFixed函数，并设置pts为 NULL,或者 size 为0 ，以便用户层可以获知已经调用gpsfix\_end()。

参数: n/a

返回: void

1. gpsfix\_data

函数: void gpsfix\_data(GpsPoint\_t\* pt,size\_t ptsize)

功能: 用户输入gps坐标点。此函数被用户连续调用，gps数据流式的被要求提交进处理模块。

参数: pt – 坐标点地址; ptsize – 坐标点数量

返回: void

1.4 处理流程:

void cb\_gpsdata(GpsPoint\_t\* pts, size\_t size){

这里接收修正的gps数据

If ( pts == NULL ){

已经到达Gpsfix\_end

}

}

Gpsfix\_begin(cb\_gpsdata);

GpsPoint\_t \* pt;

While( pt = readgps() ){

Gpsfix\_data(pt,1);

}

Gpsfix\_end();

开始时，用户初始化地图模块，调用gpsfix\_begin()函数，

设置好数据返回的入口(CallBack\_PointFixed),连续读入dvr的gps数据，调用gpsfix\_data()将gps轨迹提交给处理模块； 处理模块内部实现缓冲、匹配等处理之后通过CallBack\_PointFixed接口将修正的数据返回给用户层； 用户结束输入调用gpsfix\_end()通知处理模块，处理模块完成最后的匹配之后将残存的所有gps轨迹提交给用户(callback\_pointfixed)，并清除自己内部的缓冲，重新初始化自己的状态。

# 2. 路段匹配

2.1 描述

路段匹配是指输入连续的gps轨迹，要求地图接口计算返回轨迹所跨越的路段信息。路段信息包括: 图幅编号+路段节点编号 (mesh\_id+node\_id)

1. 路网路段节点匹配



如图，道路由A,B,C,D构成，GpsPath跨越了AB,BC,CD路段，要求输入GpsPath轨迹信息返回A,B,C,D四个道路节点,节点信息由mesh\_id和node\_id构成

Gps轨迹失锁的处理:



图中GpsPath1和GpsPath2之间产生失锁的现象，用户层将path1和path2的轨迹数据同时提交给地图匹配接口，匹配返回 [AB,BC]路段涉及的节点(A,B,C) 和 [CD]路段涉及的节点(C,D)。

1. 新道路处理：

？

2.2 数据结构

1. RoadNode\_t

|  |  |  |
| --- | --- | --- |
| 描述一个路段节点 | | |
| 字段 | 类型 | 说明 |
| Mesh\_id | Uint32 | 图幅号 |
| Node\_id | Uint32 | 节点编号 |
|  | | |

1. RoadPart\_t

|  |  |  |
| --- | --- | --- |
| 描述一个路段 | | |
| 字段 | 类型 | 说明 |
| first\_node | Roadnode\_t | 开始节点 |
| second\_node | Roadnode\_t | 结束节点 |
| First\_point | GpsPoint\_t | 出现在（first，second）路段上第一个gps坐标点 |
| Last\_point | GpsPoint\_t | 出现在（first，second）路段上最后一个gps坐标点 |
| First,second表示路段的两端节点，gps轨迹跨越路段具有方向性，first\_node表示开始，second\_node表示结束 | | |

3. CallBack\_NodeSplitted

typedef void (CallBack\_NodeSplitted\*)(RoadPart\_t\* parts, size\_t size);

函数: void (\*)(RoadPart\_t\* parts, size\_t size)

功能: 地图接口返回匹配的路段信息到用户的接口

参数: parts – 路段信息; size – 路段数量

返回: None

Parts数据由地图接口库完成内存分配和释放，用户层不能缓存parts数据，必须进行深拷贝

* 1. 接口

1. nodesplit\_begin

函数: void nodesplit\_begin(CallBack\_NodeSplitted user);

功能: 开始轨迹到路段匹配

参数: user – 用户接收接口

返回: void

1. nodesplit\_end

函数: void nodesplit\_begin();

功能: 结束gps轨迹到路段的匹配，地图SDK内部完成最后一次的处理工作，通过CallBack\_NodeSplitted接口返回所有的匹配路段。并且再追加调用一次CallBack\_NodeSplitted函数，并设置parts为 NULL,或者 size 为0 ，以便用户层可以获知已经调用nodesplit\_end()。

参数: none

返回: none

1. nodesplit\_data

函数: void nodesplit\_data(GpsPoint\_t\* pts, size\_t size);

功能: 用户提交dvr采集的gps轨迹数据到地图接口。 Gps轨迹是连续提交给地图接口，地图接口内部实现缓冲，完成处理之后通过callback\_nodesplitted反射回用户层。

参数: pts – gps轨迹点数组

Size - 轨迹点数量

返回: none

2.4 处理流程

void cb\_roadparts (RoadPart\_t\* pts, size\_t size){

这里接收路段数据

If ( pts == NULL ){

已经到达nodesplit\_end()

}

}

nodesplit\_begin(cb\_gpsdata);

GpsPoint\_t \* pt;

While( pt = readgps() ){

nodesplit\_data(pt,1);

}

nodesplit\_end();

程序开始调用nodesplit\_begin,设置路段匹配结果接收；

连续读取dvr轨迹数据并通过nodesplit\_data接口传递给地图SDK，地图sdk实现一定的缓存处理之后计算出轨迹跨越的路段通过 callback\_nodesplitted返回给调用层；

程序全部完成轨迹读取之后调用nodesplit\_end通知地图sdk结束道路轨迹匹配工作，地图sdk即刻处理内部所有的gps轨迹数据，通过callback\_nodesplitted返回给用户层, 并清除自己内部的缓冲，重新初始化自己的状态.