泛亚汽车技术中心

阶段性报告

地图模块

忻斌健

协助
丁稼毅 鞠一鸣 钱士才 李亚光

2017 年 01 月 09 日

地图模块

忻斌健

2017 年 01 月 09 日

摘要：本文讨论了地图模块的接口设计和主要的功能模块以及车辆和目标在地理坐标系下的定位···.

关键词：地图；RTK；数据融合；路径规划；微波雷达；SRR, ESR.

HD Map implementation by RTK deployment for Path Planning with Vehicle and Object localization

Xin Binjian

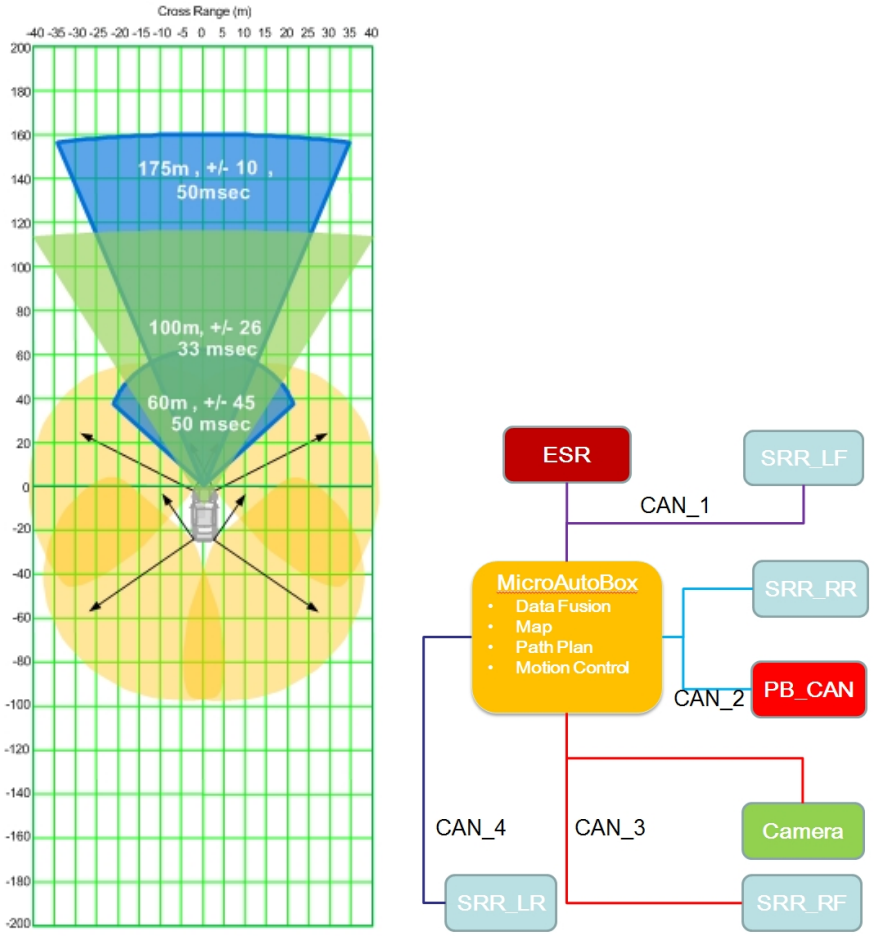
Abstract: This document has discussed the implementation of HD Map and the vehicle and objects localization in geographic coordinate system with RTK.

Key words: HD Map; RTK; Sensor Fusion; Path Planning; RADAR; SRR; ESR

1 传感器方案

本项目使用的传感器有如下：

- RTK
- 高精度地图（含车道线和其他地标物（landmark, beacon) 如交通标示牌））
- 4 个短距离毫米波雷达 SRR,
- 1 个长距离毫米波雷达 ESR；



(a) 传感器检测范围　　(b) 传感器 CAN 总线拓扑

图 1: 传感器方案图 [8]

2 基于静态离线地图的目标定位和融合

本方案是基于静态采集的地图结合实时 RTK 信号来实现车辆和目标定位。基本算法流程如图2

在仿真及实时 Autobox 环境中的接口如图3

地图模块的 Simulink 模型及接口如图4

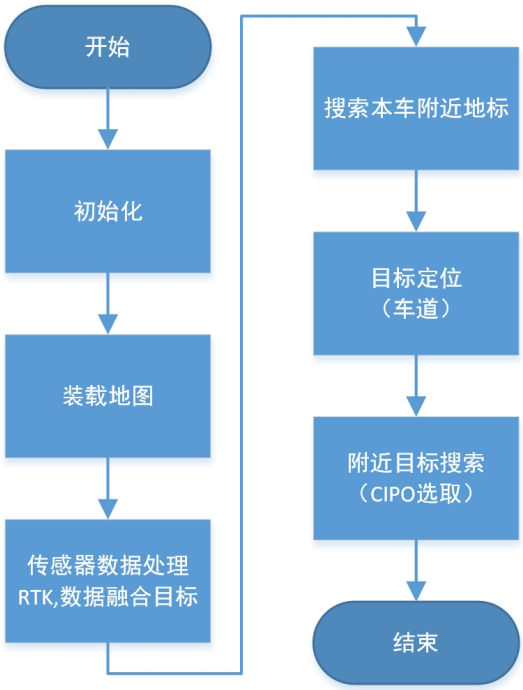


图 2: 基本算法流程图

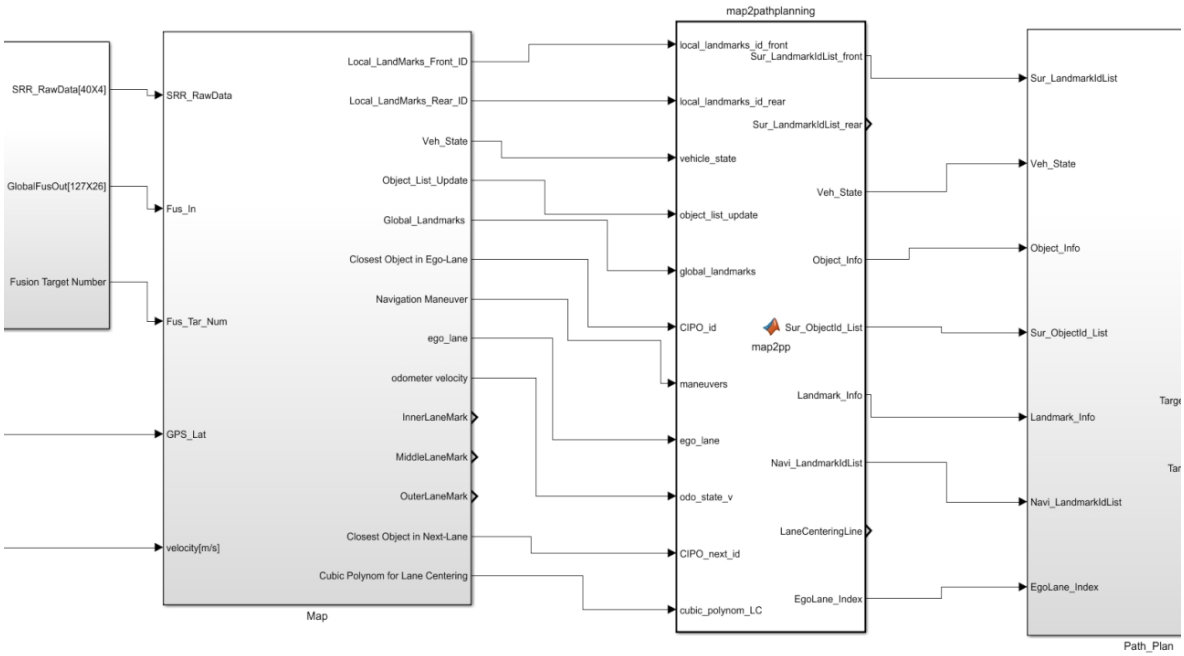


图 3: 仿真环境 Simulink 模型

2.1 静态地图采集和存储

静态地图是车道线和地标的集合。此处地标主要包括限速标志，停止线，人行横道线等。本项目园区道路是由道路分隔线，路边沿组成的；地标有如下种类。

```
enum LandMarks{
// 十字路口停止线
    XCrossing_Stop = 11,
```

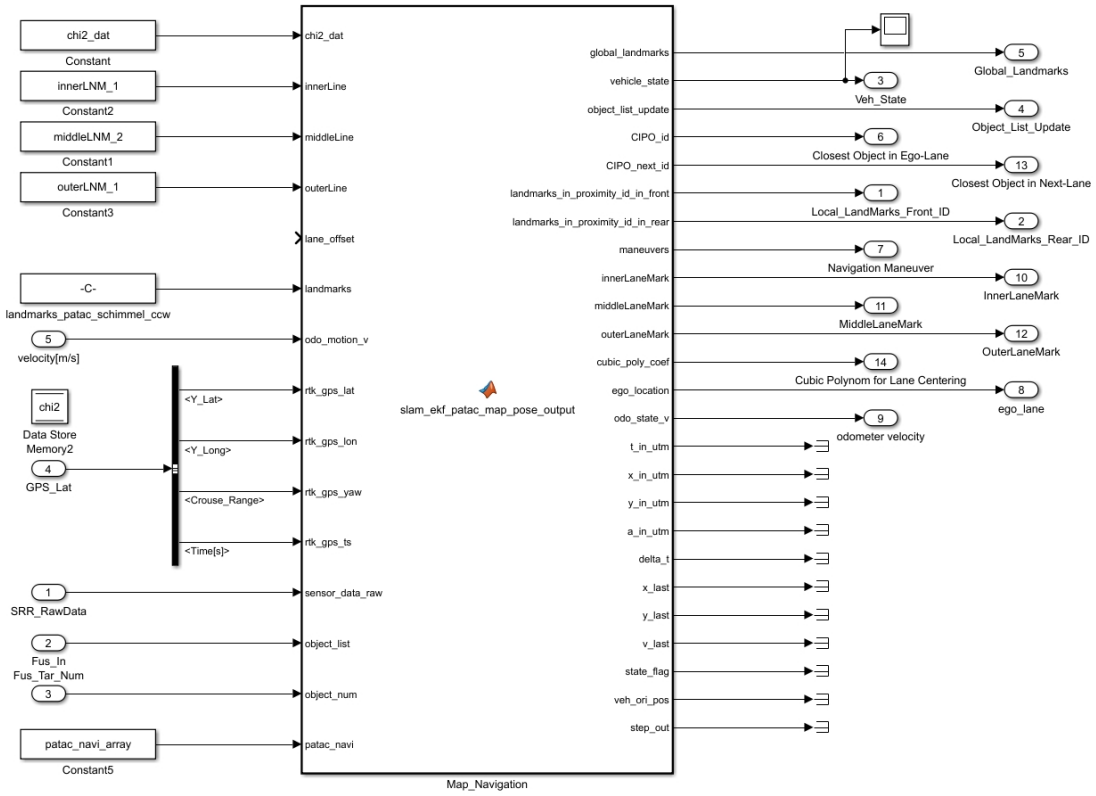


图 4: 地图模块 Simulink 模型

```
// 十字路口目标车道进入线
    XCrossing_Start = 12,
//T 字路口停止线
    TCrossing_Stop = 21,
//T 字路口目标车道进入线
    TCrossing_Start = 22,
//人行横道线
    ZCrossing = 30,
//边线 1（沿车道横向边线），摆渡车停车点。
//停车点可能没有实际的标记，请按估计的停车点选取位置，
//比如添加 2 号楼门口的停车点
    Stop_1 = 51,
//边线 2（沿车道横向边线）
    Stop_2 = 52,
// 左转 L 形路口停止线
    LeftTurn_Stop = 61,
// 左转 L 形路口进入线
    LeftTurn_Start = 62,
// 右转 L 形路口停止线
    RightTurn_Stop = 71,
// 右转 L 形路口进入线
    RightTurn_Start = 72,
// 路边停车位边线 1（沿车道横向边线）
    ParkSlotOnRoad_Stop = 81,
```

```
// 路边停车位边线 2（沿车道横向边线）
    ParkSlotOnRoad_Start = 82,
//  路外停车位，只需测量沿道路行驶方向的开始点和停止点
    ParkSlotOffRoad = 90,
//  路外停车位，只需测量沿道路行驶方向的开始点
    ParkSlotOffRoad_Start = 91,
//  路外停车位，只需测量沿道路行驶方向的停止点
    ParkSlotOffRoad_Stop  = 92,
// 车道宽度变化点，如两点间距离暗示车道宽度，
//则此时两点选取应保证，两点连线同测量处的车道垂直，
//或者增加车道宽度的域。
    RoadChange = 100,
//限速标志，点类型
    SpeedLimit = 110,
//地上箭头（参考 EPM3 的箭头位置定义），点类型
    GroundArrow = 120,
//交通灯位置，点类型
    TrafficLight = 130,
//其他
    UnClassified = 180,
}
```

地标分三类，点，线和面。所有地标统一主要由两对浮点数和类型值构成。如果两对浮点数相等的，表示点地标。线地标的两对浮点数表示线段的起点和终点。面坐标有 Bounding Box 的两个对角顶点描述。车道线用位于线上顺序的点序列描述。

车道线与路沿采集都由 RTK 设备离线采集。由于采集到的点是 GPS 的经纬度信号，需要转化为量纲为米的长度单位，这里的转换是从地理坐标系到局部坐标系，有 Mercator 变换完成。

图5(a)显示的是用 RTK 设备采集的车道线和地标。图5(b)是对应的园区卫星图 [2]。

2.1.1 参考坐标系定义和转换

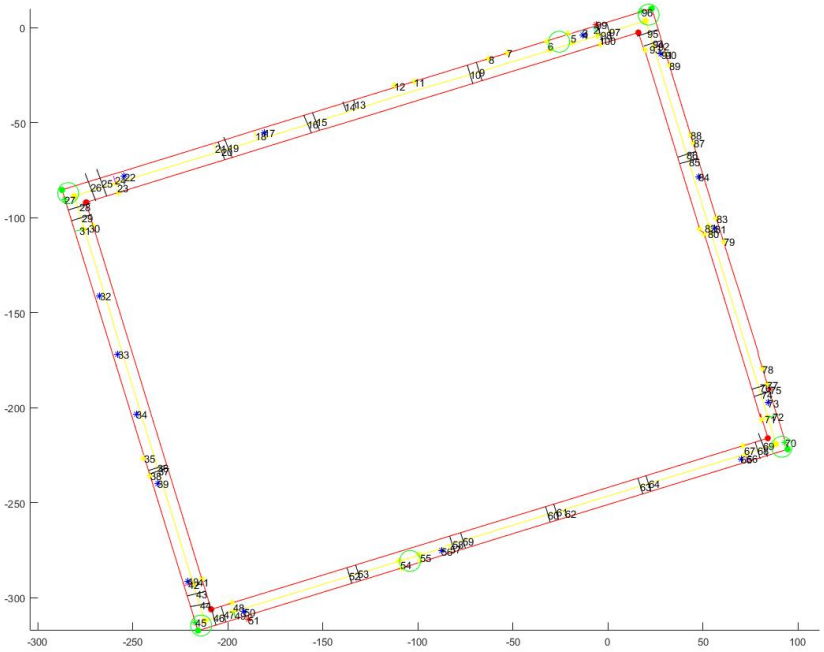
本项目用到的坐标系有（参见图6）：

- 地理坐标系 GCS (Geographic Coordinate System)，由经纬度表示。
- 局部坐标系 LCS (Local Coordinate System)，原点在地面上的选定的某个点（此处为车辆上电时的位置），正东方向为 X 轴，正北方向为 Y 轴，Z 轴向上，旋转角正向为沿 Z 轴逆时针方向。
- 车辆坐标系 VCS (Vehicle Coordinate System) 以车辆形心为原点，车辆行驶方向为 X 轴，垂直行驶方向向左为 Y 轴，Z 轴向上。
- 传感器坐标系 SCS (Sensor Coordinate System) 以车辆前保险杠中点为原点，坐标轴方向与车辆坐标系一致。

图7显示了车辆坐标系。

2.2 RTK 数据处理

本模块用到的 RTK 数据有四个，经度，纬度和航向角和 GPS 时间。本车定位由 RTK 信号给出。由于 RTK 信号更新周期大于与地图模块运行周期，以及由于上电导致启动



(a) 地标及车道线

(b) OpenStreetMap 园区卫星图

图 5: 静态全局地图

不同步或者潜在建筑物遮挡导致地图模块收到无效信号，需要进行 RTK 信号有效性判断和相应的插值。

经纬度和 Mercator 坐标转换由转换函数 [6] 完成。

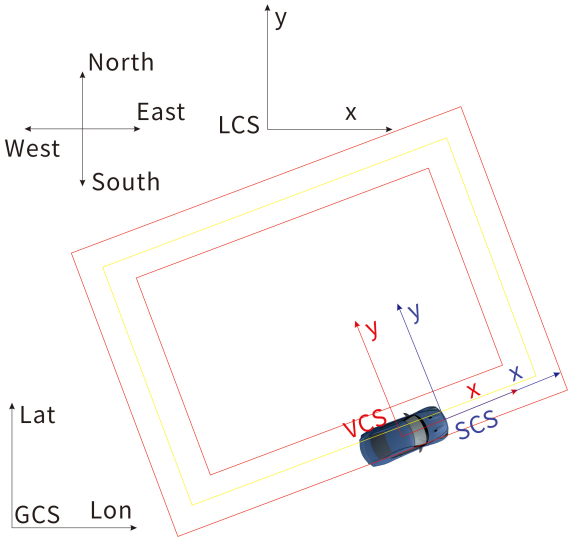


图 6: 用到的坐标系

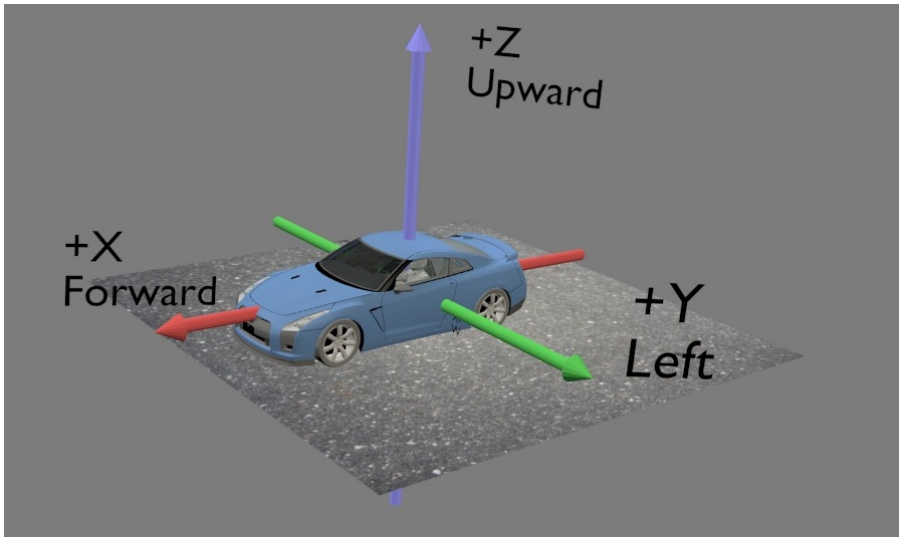


图 7: 车辆坐标系 VCS

2.3 附近地标搜索

车辆行驶过程中需要知道附近的地标信息以决定相应的行驶动作（Maneuver）。查询实时更新，由本车当前位置和本车查询范围（本车附近的搜索半径，比如 50 米）确定。搜索到的地标需要进行排序，区分本车前方路标和本车后方路标并按行驶方向顺序输出。

2.4 本车和目标的车道定位

本车的定位由 RTK 实时信号确定，由于 RTK 输入信号周期是地图模块任务更新周期的 5 倍。在 RTK 信号更新之间的 4 个周期用定速和定航向角模型进行位置的插值。再结合静态地图中车道线的坐标判断本车在哪个车道，以及在车道中的位置。

数据融合的目标追踪模块的参考坐标系是传感器坐标系 SCS，需要经过一个刚体变换转换到车辆坐标系，叠加 RTK 信号给出本车在局部坐标系中的位置，就可以得出每一个目标在车道的位置，确定所属的车道。图8给出了局部地图生成和定位算法的流程

图。图9展示了本车及目标在局部地图上定位的一个结果。

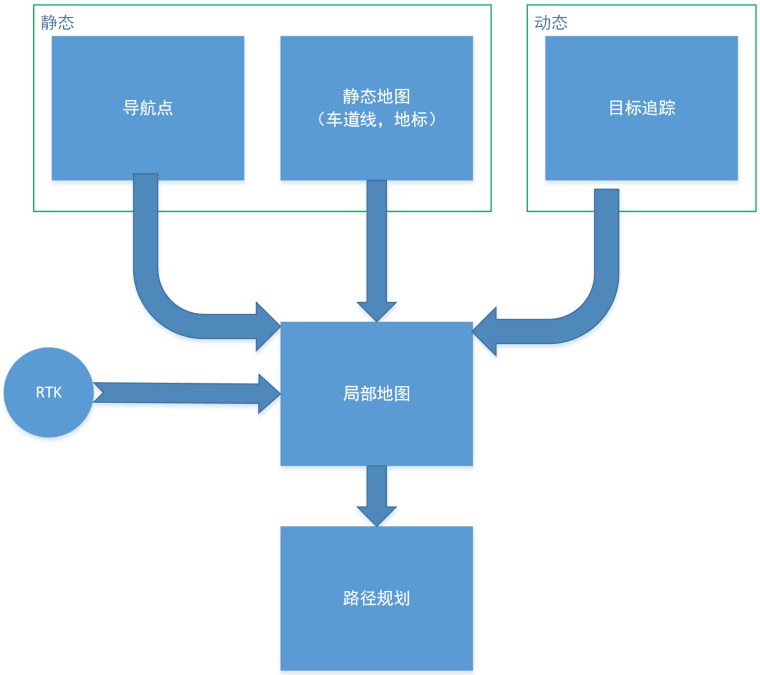


图 8: 局部地图生成

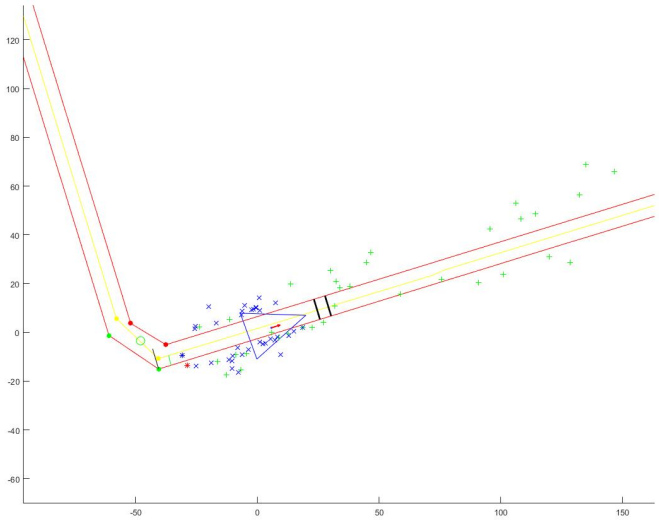


图 9: 本车及目标在局部地图上的定位

2.5 附近目标搜索及最近本车道目标 CIPO 选取

搜索本车附近的目标由两部分组成，先筛选车道上的目标，区分本车道和相邻车道的目标。然后按每个车道上目标沿车道行驶方向的纵向距离进行排序，并确定本车道最近目标 (CIPO)。排序是按车道坐标系（一维坐标系）。先将每个目标投影到车道线上，再按车道线投影的先后次序排序。

3 利用 SRR 数据生成地图和定位 SLAM

地图生成和定位 (SLAM) 是为了解决在没有 RTK 定位系统的情况下，如何生成局部地图并确定本车在局部地图中的定位，10展示了一个占据格栅图的实例。通常的方法是将本车定位和地图都作为未知量来估计，并利用二者之间的关联性降低参数估计的复杂度，简化和加速计算。局部地图用地标 (land mark) 结合占据格栅的方式构造 [7]。

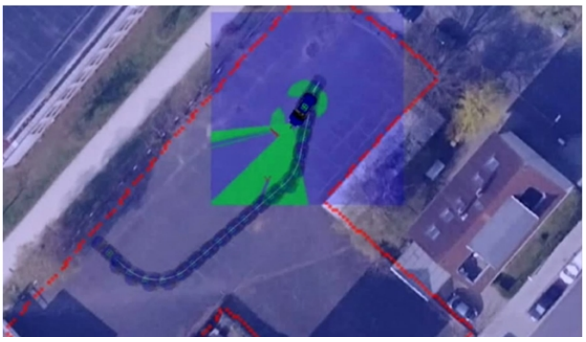


图 10: 占据格栅图实例

SLAM 需要用到传感器的原始数据 (object data), 在本方案中能得到 4 个 SRR 的原始数据，所以 SLAM 算法的输入就是 SRR 的点迹数据。另外为了降低算法复杂度，区分静态和动态的目标，只考虑静态地标构成的局部地图，SRR 检测出的动态目标和前置目标追踪模块给出的动态目标会叠加到局部地图上。根据静态地标占据格栅图用来更新本车姿态，目标追踪给出的动态目标通过坐标转换叠加到局部地图上。最后，静态地图和动态目标都输出给路径规划使用。

本车姿态（车辆位置，航向）和被观测目标的位置统计相关（地图, 此处只考虑静态的地标），应该同时求解。当特征（地标）较少，使用基于 EKF 的 SLAM[1, 5]。下面分别给出 EKF-SLAM 所需的世界模型，运动模型，和预测模型。

3.1 2D 世界模型

由于传感器误差，车道线和目标追踪的结果是含有误差的，所以局部地图需要处理未知的信息以及不完全准确的定位信息。因此需要用概率的语言描述已知信息，如方程1所示：

$$p(m|x_{1:t}, z_{1:t}) = \prod_{l=1}^{L} p(m_l|x_{1:t}, z_{1:t}) \tag{1}$$

世界模型是以占据格栅图的形式来描述，占据和非占据信息的描述方式如图11所示 [4]。

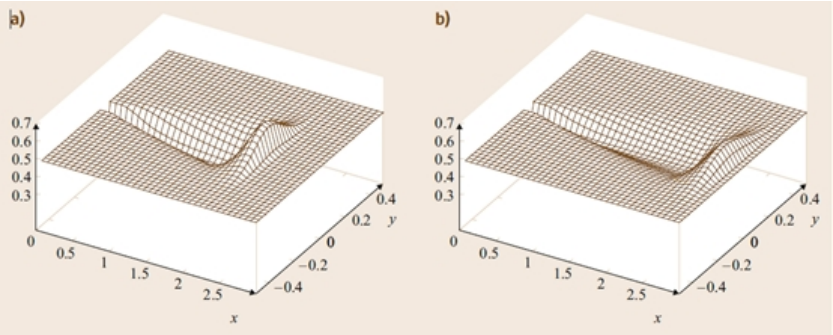


图 11: 占据格栅图概率模型

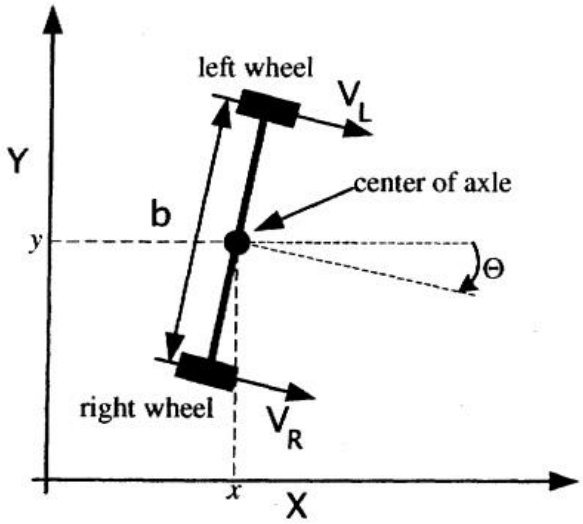


图 12: 运动模型

3.2 运动模型

本项目使用的运动学模型如图12所示。运动学关系见方程组2：

$$
\begin{aligned}
x_{k+1} &= x_k + D_k \cdot \cos\theta_{k+1} \\
y_{k+1} &= y_k + D_k \cdot \sin\theta_{k+1} \\
\theta_{k+1} &= \theta_k + \Delta\theta_k \\
D_k &= v_{t,k} \cdot T \\
\Delta\theta_k &= \omega_k \cdot T \\
v_{t,k} &= \frac{v_{L,k} + v_{R,k}}{2} = \frac{\omega_{L,k}R + \omega_{R,k}R}{2} \\
\omega_k &= \frac{v_{R,k} + v_{L,k}}{b} = \frac{\omega_{R,k}R - \omega_{L,k}R}{b}
\end{aligned}
\tag{2}
$$

输入量为：$u_k = (v_k, \omega_k)^T$ 为补偿驱动轮半径不相等带来的系统误差，引入 k_1,k_2 和 k_3 三个参数如下 [5]

$$
\begin{aligned}
v_{t,k} &= \frac{k_1 \cdot v_{L,k} + k_2 \cdot v_{R,k}}{2} \\
\omega_k &= \frac{k_2 \cdot v_{R,k} - k_1 \cdot v_{L,k}}{k_3 \cdot b}
\end{aligned}
\tag{3}
$$

方程3取代方程组2中的最后两个方程。这三个参数可以由标定来确定，不是待估计的参数。

预测方程如下：

$$x_{k+1} = f(x_k, u_k, v_k) \tag{4}$$

$$f(x_k, u_k, v_k) = \begin{cases} x_k + (D_k + v_{1,k} \cdot \cos(\theta_k + \Delta\theta_k + v_{2,k}) \\ y_k + (D_k + v_{1,k} \cdot \sin(\theta_k + \Delta\theta_k + v_{2,k}) \\ \theta_k + \Delta\theta_k + v_{2,k} \end{cases} \tag{5}$$

3.3 观测模型

观测模型如图13所示 [3]

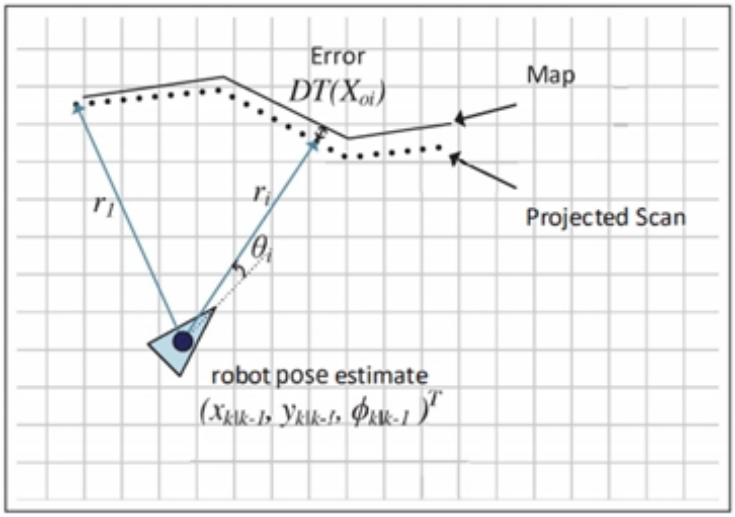


图 13: 观测模型

此预测模型采用基于距离变换 DT 方程6的 Chamfer distance CD 方程7

$$DT(\mathbf{x}) = \min_{v_j \in V} |\mathbf{x} - \mathbf{v}_j| \tag{6}$$

Chamfer distance CD 定义如下：

$$h(\mathbf{X}, \mathbf{z}) = \frac{1}{n} \sum_{i=0}^{n-1} DT(\mathbf{X}_{O_i}) = CD \tag{7}$$

其中 $\mathbf{X}_{O_i}$ 如下

$$\mathbf{X}_{O_i} = \begin{cases} x_{O_i} \\ y_{O_i} \end{cases} = \begin{cases} x_{k|k-1} + r_i \cos(\theta_i + \phi_{k|k-1}) \\ y_{k|k-1} + r_i \sin(\theta_i + \phi_{k|k-1}) \end{cases} \tag{8}$$

最后 EKF 更新周期需要的隐式观测模型如下:

$$h(\mathbf{X}, \mathbf{z}) \tag{9}$$

3.4 更新

基于隐式观测模型的更新周期计算如下：

$$K = P_{k|k-1}\nabla h_{\mathbf{x}}^T(\nabla h_{\mathbf{x}} P_{k|k-1}\nabla h_{\mathbf{x}}^T + \nabla h_{\mathbf{z}}^T \Sigma_{\mathbf{z}} \nabla h_{\mathbf{z}})^{-1} \quad (10)$$

$$X_{k|k} = X_{k|k-1} + K(-h((X)_{k|k-1}, \mathbf{z})) \quad (11)$$

$$P_{k|k} = (I - K\nabla h_{\mathbf{x}})P_{k|k-1} \quad (12)$$

至此，构造了完整的 EKF-SLAM 的计算流程。

4 附录

4.1 追踪列表（动态）

```
typedef struct{
int id;
// set to 1 in the very first cycle onl that an object is output, 0 in all other cycles.
bool newObj;
//one of GCS (WGS84/UTM), LCS,  CCS, VCS, SCS, ACS, ENUM tbd
int coordinate_system;
PatObjState objState;
PatObjSize objSize;
// Pedestrian/vehicle_car/vehicle_truck/unknown/... ---> ENUM tbd.
int objClass;
//1 if the object is moving;0 if it's still;
bool moving;
//tracking when the object is seen by which exteroceptive sensor;
PatTime lastSeenBySensor[NUM_SENSOR_EXTEROCEPTIVE];
//0 for invalid; 1 for valid;
float existenceProbability;
}PatObject;

//maximally 256 tracked objects by all sensors around the vehicle;
PatObject object_list[128];
```

4.2 主控模型 m 函数

```
%-------------------------------------------------------
% PATAC
% ADU
% SHD
%
% Authors:  Binjian Xin
% Date   :  11-2016
%-------------------------------------------------------
% slam, explore data association algorithms
% hd_map: offline HD map orginal ground
% odo_motion: vehicle odometer data: odo_motion.x; odo_motion.y; odo_motion.yaw;
% sensor_data_raw: raw sensor data 40x4 [ID, x, y, MoveInfo]
% sensor_data_raw.srr[4] srr(1)-srr(4) are four SRR Radar raw data.
```

```
% srr(i).Measure --> [rho tita;...]
% srr(i).Rn --> [rho tita;...]c
% sensor_data_fusion: sensor fuison data; sensor_data_fusion
% 127x26
% [ID,RangeX,VelX,RangeY,VelY,MoveInfo,Width,Type,Fus_State,Confidence,
% P11,...P44]; FuseState Byte 9 for Lane Position Judge
% hd_map offline hd map, supposed to be converted to RTM 51r : wgs2utm(Lat,Lon,51,'N');
% hd_map.lanemarking1/lanemarking2/lanemarking3:
%           three lanes, points in counterclockwise order; 1
% for left, 2 for middle, 3 for right;
% hd_map.landmark(:,5) (x,y,length, width, LandMarkType)
% rtk_gps: rtk_gps.[lat, lon, alt, yaw, pitch, roll, ts]
%


function [global_landmarks,...
vehicle_state, ...
object_list_update, ...
CIPO_id,...
CIPO_next_id,...
landmarks_in_proximity_id_in_front, ...
landmarks_in_proximity_id_in_rear,...
maneuvers, ...
innerLaneMark,...
middleLaneMark,...
outerLaneMark,...
cubic_poly_coef,...
ego_location, ...
odo_state_v, ...
t_in_utm,...
x_in_utm,...
y_in_utm,...
a_in_utm,...
delta_t,...
x_last,...
y_last,...
v_last,...
state_flag,...
veh_ori_pos,...
step_out] ...
= slam_ekf_patac_map_pose_output...
(chi2_dat, ...
innerLine, ...
middleLine, ...
outerLine, ...
lane_offset,...
landmarks,...
odo_motion_v,...
```

```
rtk_gps_lat, ...
rtk_gps_lon, ...
rtk_gps_yaw, ...
rtk_gps_ts, ...
sensor_data_raw, ...
object_list, ...
object_num, ...
patac_navi)
%Initialization
%...


%Convert rtk raw data to longitude, latitude, yaw in radian and ts in second
[wgs_lat,wgs_lon,wgs_yaw,wgs_ts_in_sec] = rtkraw2wgs (rtk_gps_lat,...
                                                      rtk_gps_lon,...
                                                      rtk_gps_yaw,...
                                                      rtk_gps_ts);
[x_in_utm,y_in_utm,~,~] = wgs2utm(wgs_lat,wgs_lon,51,'N');
wgs_yaw = 2*pi - wgs_yaw;


%rtk system --> VCS offset
x_rtk_offset = 0;

y_rtk_offset = 0;
a_rtk_offset = 90*pi/180;%rtk calibration info input!
x_in_utm = x_in_utm+x_rtk_offset;
y_in_utm = y_in_utm+y_rtk_offset;
a_in_utm = wgs_yaw + a_rtk_offset;
t_in_utm = wgs_ts_in_sec;

if a_in_utm>2*pi
a_in_utm = a_in_utm-2*pi;
end

chi2 = chi2_dat;
%Initialization
%Validity Check and initialization
%...


%Relocate object list in LCS
%SCS --> VCS offset
x_scs_offset = 5.24/2.0;%vehicle length 5.24m
y_scs_offset = 0;%vehicle width 1.8m

%VCS --> CCS (Control Coordinate System) offset
x_ccs_offset = 0;
```

```
y_ccs_offset = 0;
% x_offset = x_scs_offset+x_ccs_offset;
% y_offset = y_scs_offset+y_ccs_offset;
object_list_id = object_list(:,1);
valid = find(object_list_id);
object_num_meas = size(valid,1);

object_list_lcs = object_list;
object_num_lcs = object_num_meas;

object_loc = [object_list(1:object_num_meas,2),object_list(1:object_num_meas,4)]';
object_loc(1,:) = object_loc(1,:) + x_scs_offset;
object_loc(2,:) = object_loc(2,:) + y_scs_offset;
Tab = [ x_in_lcs; y_in_lcs; a_in_lcs];
Pa = tpcomp(Tab, object_loc);
%RangeX in LCS, displacement of VCS to SCS by 2.6m, half of vehicle length
object_list_lcs(1:object_num_meas,2)=Pa(1,:)';%object_list(:,2)+ x_scs_offset + x_in_lcs;
%RangeY in LCS
object_list_lcs(1:object_num_meas,4)=Pa(2,:)';%object_list(:,4)+ y_scs_offset + y_in_lcs;

object_vec = [object_list(1:object_num_meas,3),object_list(1:object_num_meas,5)]';
Tab = [ veh_vel(1); veh_vel(2); a_in_lcs];
Pb = tpcomp(Tab, object_vec);
%VelX
object_list_lcs(1:object_num_meas,3)=Pb(1,:)';%object_list(:,3)+ veh_vel(1);
%VelY
object_list_lcs(1:object_num_meas,5)=Pb(2,:)';%object_list(:,5)+ veh_vel(2);

raw_loc = [sensor_data_raw(:,2),sensor_data_raw(:,3)]';
Tab = [ x_scs_offset + x_in_lcs; y_scs_offset + y_in_lcs; a_in_lcs];
Pa = tpcomp(Tab, raw_loc);
sensor_data_raw_lcs = sensor_data_raw;
%RangeX in LCS, displacement of VCS to SCS by 2.6m, half of vehicle length
sensor_data_raw_lcs(:,2)=Pa(1,:)';
%RangeY in LCS
sensor_data_raw_lcs(:,3)=Pa(2,:)';

%find local landmarks
option.firstTime=1;
option.clockWise=0;  %0 for counter-clockwise，1 for clockwise
[landmarks_in_proximity_id_in_front_lcs, landmarks_in_proximity_id_in_rear_lcs] = ...
            quest_map4landmark_new_lane (x_in_lcs, y_in_lcs, landmarks_lcs,...
                                         middleLine_lcs.coordinate(:,1:2),...
                                         middleLine_lcs.Vertex_index,...
                                         configuration,...
                                         option);
```

```
158
159 %locate objects in lanes
160 %decide object list location in lanes
161 option.firstTime=1;
162 option.clockWise=0;  %0 for counter-clockwise，1 for clockwise
163 object_list_update_lcs  = object_localization(object_list_lcs, object_num_lcs, ...
164     innerLine_lcs.coordinate,innerLine_lcs.Vertex_index,...
165     middleLine_lcs.coordinate(:,1:2),middleLine_lcs.Vertex_index,...
166     outerLine_lcs.coordinate,outerLine_lcs.Vertex_index,...
167     lane_offset, option);
168
169
170 %find the closest object in ego lane (probably CIPO)
171 option.firstTime=1;
172 option.clockWise=0;  %0 for counter-clockwise，1 for clockwise
173 [obj_closest_in_path_ID, obj_closest_in_next_path_ID,ego_location_lcs] = ...
174     objects_of_interest_fast(...
175         object_list_update_lcs,object_num_lcs,...
176         x_in_lcs, y_in_lcs, innerLine_lcs.coordinate, innerLine_lcs.Vertex_index,...
177         middleLine_lcs.coordinate(:,1:2), middleLine_lcs.Vertex_index,...
178         outerLine_lcs.coordinate, outerLine_lcs.Vertex_index, option);
179         obj_in_path_size = size(obj_closest_in_path_ID,1);
180 if(~isempty(obj_closest_in_path_ID))
181   CIPO_id_lcs = zeros(10,1);
182   if(obj_in_path_size<=10)
183     CIPO_id_lcs(1:obj_in_path_size,1) = obj_closest_in_path_ID(1:obj_in_path_size,1);
184   else
185     CIPO_id_lcs(1:10,1) = obj_closest_in_path_ID(1:10,1);
186   end
187 end
188 obj_in_path_size = size(obj_closest_in_next_path_ID,1);
189 if(~isempty(obj_closest_in_next_path_ID))
190   CIPO_id_next_lcs = zeros(10,1);
191   if(obj_in_path_size<=10)
192     CIPO_id_next_lcs(1:obj_in_path_size) = ...
193         obj_closest_in_next_path_ID(1:obj_in_path_size,1);
194   else
195     CIPO_id_next_lcs(1:10,1) = obj_closest_in_next_path_ID(1:10);
196   end
197 end
198 point_num = 30;
199 right_shift = 2; %counter-clockwise right shift 2m for outer lane; left shift for inner lane.
200 cubic_poly_coef_lcs = lanefit(vehicle_state, option.clockWise, middleLine_lcs, ...
201                               outerLine_lcs, right_shift, ego_location_lcs, point_num);
202 %Output
203 %...
```

5 参考文献

[1] José A Castellanos, José Neira and Juan D Tardós. "*Map Building and Slam Algorithms*". In: **2005**.

[2] Steve Coast. *OpenStreetMap*, 2016-10. http://www.openstreetmap.org/.

[3] L. Dantanarayana, G. Dissanayake, R. Ranasinghe *et al.* "*An extended Kalman filter for localisation in occupancy grid maps*". In: *2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS)*, 2015-12: 419–424.

[4] A. Elfes. "*Using Occupancy Grids for Mobile Robot Perception and Navigation*". *Computer*, 1989-06: 46–57. http://dx.doi.org/10.1109/2.30720.

[5] E. Ivanjko, M. Vasak and I. Petrovic. "*Kalman filter theory based mobile robot pose tracking using occupancy grid maps*". In: *2005 International Conference on Control and Automation*, 2005-06: 869–874 Vol. 2.

[6] Alexandre Schimel. *WGS2UTM*, https://cn.mathworks.com/matlabcentral/fileexchange/14804-wgs2utm--version-2-.

[7] Sebastian Thrun, Wolfram Burgard and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, **2005**.

[8] 王子涵. 园区自动驾驶数据融合阶段性报告 [内部报告]. 上海, 2017-01.