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

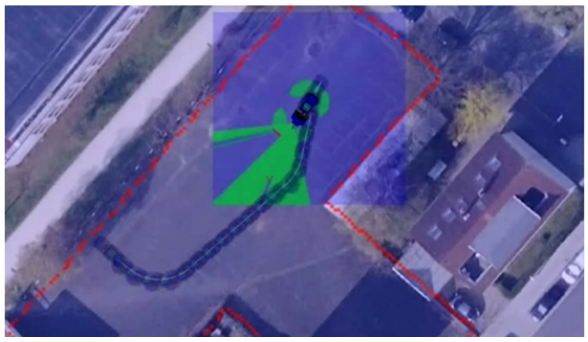
Xin Binjian

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; isomorphism; Fourier transform.

1 项目假设

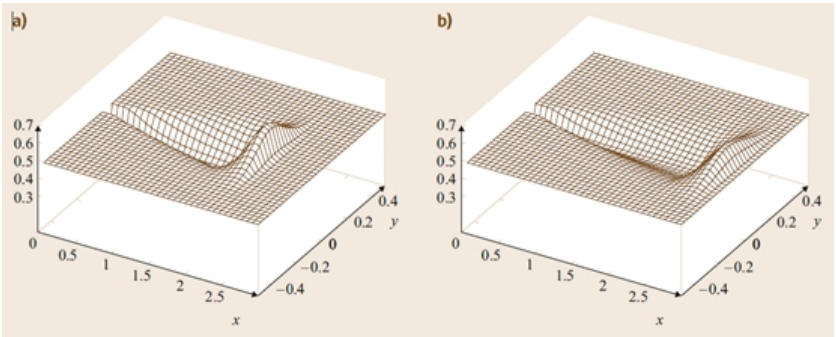
RTK，高精度地图，4 个 SRR，1 个 ESR；若干地标物（landmark, beacon, 交通标示）道路模型（局部地图）需要处理未知，占据和非占据信息。接口静态的占据格栅图地图 OGM+ 动态的目标列表。根据静态地标占据格栅图用来更新本车姿态，叠加动态的障碍物，局部地图：障碍物列表，车道，静态地标，RTK 车辆定位-->路径规划。



2 2D 世界模型

2D 占据格栅图

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



3 追踪列表（动态）

```
int main() {
printf("hello, world");
return 0;
}
```

```
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 SLAM（同时定位与生成地图）

本车姿态（车辆位置，航向）和被观测目标的位置相关（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. (fastSLAM/DP-SLAM 适用于其他场合)

4.1 运动模型

$$x_{k+1} = x_k + D_k \cdot \cos\theta_{k+1} \tag{2}$$

$$y_{k+1} = y_k + D_k \cdot \sin\theta_{k+1} \tag{3}$$

$$\theta_{k+1} = \theta_k + \Delta\theta_k \tag{4}$$

$$D_k = v_{t,k} \cdot T \tag{5}$$

$$\Delta\theta_k = \omega_k \cdot T \tag{6}$$

$$v_{t,k} = \frac{v_{L,k} + v_{R,k}}{2} = \frac{\omega_{L,k}R + \omega_{R,k}R}{2} \tag{7}$$

$$\omega_k = \frac{v_{R,k} + v_{L,k}}{b} = \frac{\omega_{R,k}R - \omega_{L,k}R}{b} \tag{8}$$

标定

$$v_{t,k} = \frac{k_1 \cdot v_{L,k} + k_2 \cdot v_{R,k}}{2} \tag{9}$$

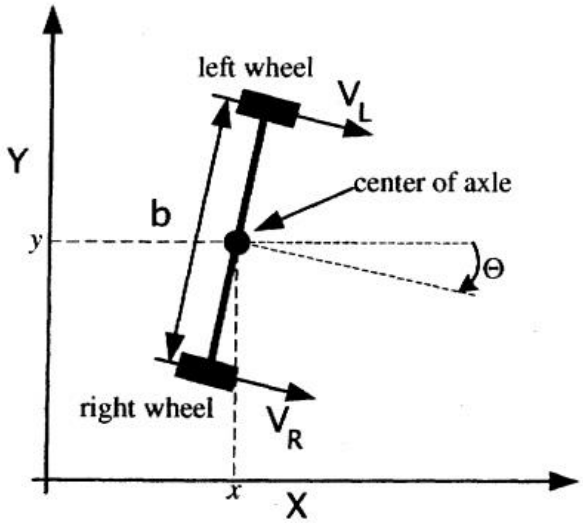
$$\omega_k = \frac{k_2 \cdot v_{R,k} - k_1 \cdot v_{L,k}}{k_3 \cdot b} \tag{10}$$

$u_k = (v_k, \omega_k)^T$

预测

$$x_{k+1} = f(x_k, u_k, v_k) \tag{11}$$

$$f(x_k, u_k, v_k) = \begin{cases} x_k + (D_k + v_{1,k} \cdot \cos(\theta_k + \Delta\theta_k + v_{2,k}) \\ y_k + (D_k + v_{1,k} \cdot \sin(\theta_k + \Delta\theta_k + v_{2,k}) \\ \theta_k + \Delta\theta_k + v_{2,k} \end{cases} \tag{12}$$



4.2 观测模型

基于距离变换 DT

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

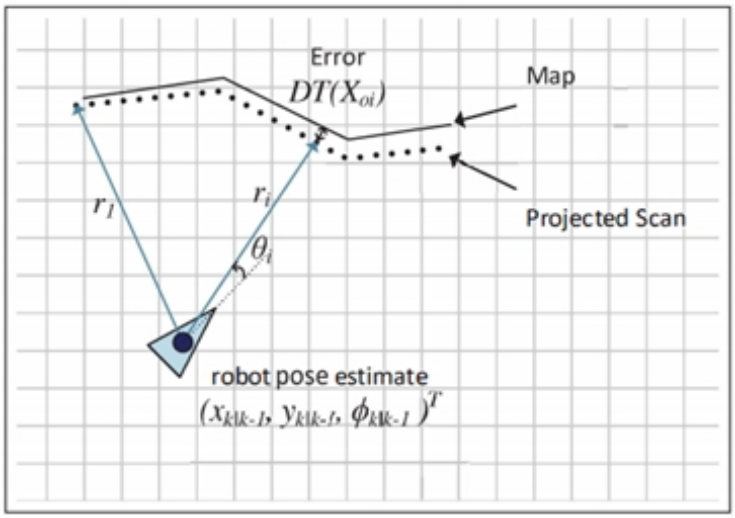
Chamfer distance CD

$$h(\mathrm{X}, \mathrm{z}) = \frac{1}{n} \sum_{i=0}^{n-1} DT(X_{O_i}) = CD$$

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

隐式观测模型

$$h(\mathrm{X}, \mathrm{z})$$



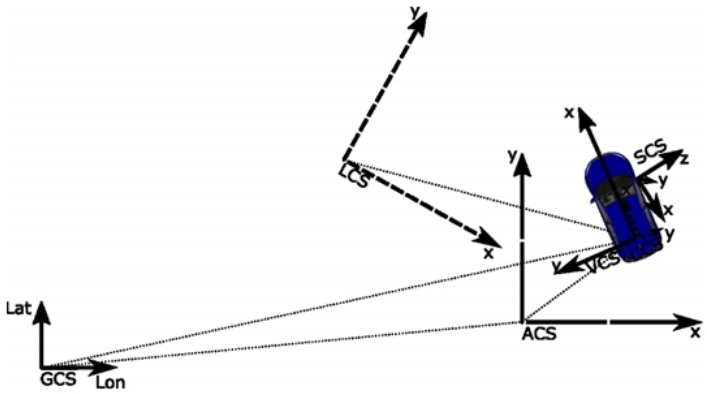
4.3 更新

$$K = P_{k|k-1}\nabla h_{\mathrm{X}}^T(\nabla h_{\mathrm{X}} P_{k|k-1}\nabla h_{\mathrm{X}}^T + \nabla h_{\mathrm{z}}^T \Sigma_{\mathrm{z}} \nabla h_{\mathrm{z}})^{-1} \quad (13)$$

$$X_{k|k} = X_{k|k-1} + K(-h((X)_{k|k-1}, \mathrm{z})) \quad (14)$$

$$P_{k|k} = (I - K\nabla h_{\mathrm{X}})P_{k|k-1} \quad (15)$$

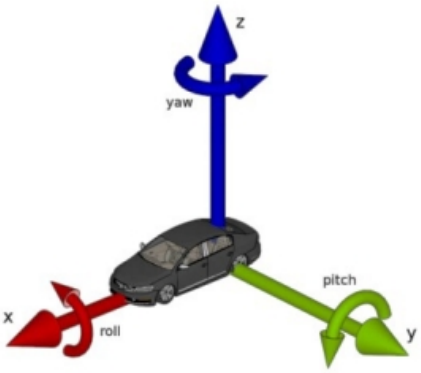
4.4 参考坐标系

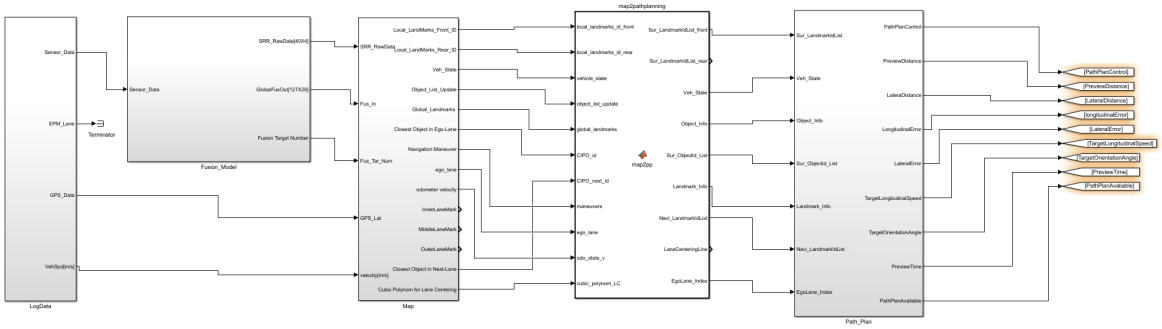


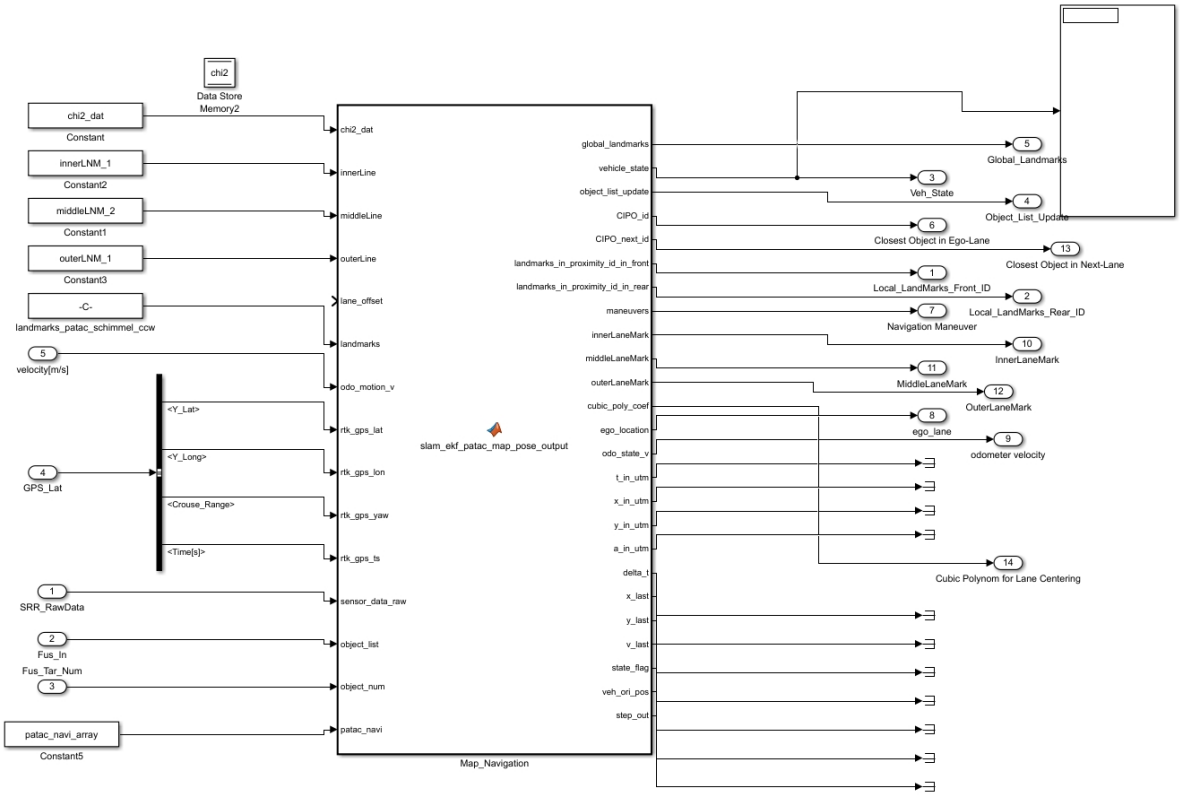
5 附录

5.1 主控模型 m 函数

```
function [veh_pose, veh_vel] = slam_ekf_patac (innerLine, middleLine, outerLine,...
                landmarks, ...
                odo_motion_x, odo_motion_y, odo_motion_yaw, ...
                rtk_gps_lat, rtk_gps_lon, rtk_gps_yaw, rtk_gps_ts,...
```








```
                    sensor_data_raw,...
                    proximity)
coder.extrinsic('EKF_prediction');
coder.extrinsic('draw_ellipse');
```

```
rng;%randn('state', 0);

% determines execution and display modes
coder.inline('never');
%global configuration sensor;

persistent sensor configuration step veh_origin_pose map ground; %step = 0;
%chi2 = chi2inv(configuration.alpha,1:1000);
persistent rtk_gps_lat_last rtk_gps_lon_last rtk_gps_ts_last;

if isempty(step)
    %adapt to applied sensors (SRR)!
    step = 1;

configuration = struct('ellipses',true,'tags',false,'odometry',true, ...
                       'noise',true,'alpha',0.99,'step_by_step',false,...
                       'people',false,'ground',1,'map',2,'observations',3,...
                       'compatibility',4,'ground_hypothesis',5,'hypothesis',6,...
                       'tables',7);

    sensor.range = 5;
    sensor.minangle = -pi/2;
    sensor.maxangle = pi/2;
    sensor.srho = 0.01;
    sensor.stita = 0.125*pi/180;

    rtk_gps_lat_last =rtk_gps_lat;
    rtk_gps_lon_last = rtk_gps_lon;
    rtk_gps_yaw_last = rtk_gps_yaw;
    rtk_gps_ts_last = rtk_gps_ts;
    % generate the ground data from hdmap and RTK
    ground = generate_rtk_ground(innerLine, middleLine, outerLine);

    % start with a fresh map
    [map, ground] = new_map(ground);

    % plot ground
    draw_ground(ground,landmarks, configuration);
    %%pause
```

```
48
49      % ok, here we go
50
51      %%observations = get_observations(ground, sensor, step);
52      [x1,y1,utmzone,utmhemi] = wgs2utm(rtk_gps_lat,rtk_gps_lon,51,'N');
53      veh_origin_pose.x = x1;
54      veh_origin_pose.y = y1;
55      veh_origin_pose.yaw = rtk_gps_yaw;
56      veh_origin_pose.ts = rtk_gps_ts;
57      veh_pose = veh_origin_pose;
58      veh_vel = [0; 0]; % Start point with 0 velocity.
59      observations = get_observations (ground, landmarks, veh_pose, ...
60                                       sensor_data_raw, sensor, proximity);
61      draw_observations (observations, configuration, step);
62
63  %     GT = zeros(1, size(sensor_data_raw,1));
64  %      H = zeros(1, size(sensor_data_raw,1));
65
66      map = add_features(map, observations);
67      % plot map
68      draw_map (map, ground,configuration,step);
69
70    % steps = length(ground.motion);
71  else
72
73      step = step+1;
74      disp('------------------------------------------------------------');
75  %     disp(sprintf('Step: %d', step));
76
77      % EKF prediction step
78      odometry.x = [odo_motion_x, odo_motion_y,odo_motion_yaw]'; ;%odo_motion.x;
79      odometry.P = diag([0.25 0.1 5*pi/180].^2);
80
81
82      map = EKF_prediction (map, odometry);
83
84      % sense
85      [x1,y1,utmzone,utmhemi] = wgs2utm(rtk_gps_lat,rtk_gps_lon,51,'N');
86      veh_pose.x = x1;
```

```
87      veh_pose.y = y1;
88      veh_pose.yaw = rtk_gps_yaw;
89      veh_pose.ts = rtk_gps_ts;
90
91      [x2,y2,utmzone,utmhemi] = wgs2utm(rtk_gps_lat_last,rtk_gps_lon_last,51,'N');
92      veh_vel = [x1-x2; y1-y2]/(rtk_gps_ts - rtk_gps_ts_last)/1000;%ms-->s
93
94      rtk_gps_lat_last =rtk_gps_lat;
95      rtk_gps_lon_last = rtk_gps_lon;
96      rtk_gps_ts_last = rtk_gps_ts;
97
98
99      motion.x = [x1 y1 rtk_gps_yaw]';
100     motion.P = diag([0.02 0.02 2*pi/180].^2); % expectation of std of RTK
101     ground = move_vehicle (ground, motion, step);
102
103     observations = get_observations(ground, landmarks, veh_pose, ...
104                                    sensor_data_raw, sensor, proximity);
105
106     % individual compatibility
107     prediction = predict_observations (map, ground);
108     compatibility = compute_compatibility (prediction, observations);
109
110     disp(compatibility.HS);
111     disp(compatibility.AL);
112
113     disp(' ');
114
115     % ground truth
116     % your algorithm here!
117     % 1. Try NN
118     % 2. Complete SINGLES and try it
119     % 3. Include people and try SINGLES5
120     % 4. Try JCBB
121
122     H = NN (prediction, observations, compatibility,configuration);
123
124     draw_map (map, ground,configuration, step);
125     draw_observations (observations, configuration, step);
```

```
126
127
128     draw_compatibility (prediction, observations, compatibility,configuration);
129
130     disp(' ');
131
132     draw_hypothesis (prediction, observations, H, 'NN:', 'b-',configuration);
133
134     % update EKF step
135     map = EKF_update (map, prediction, observations, H, step);
136
137
138     % only new features with no neighbours
139     new = find((H == 0) & (compatibility.AL == 0));
140
141     if nnz(new)
142        map = add_features(map, observations, new);
143     end
144
145     draw_map (map, ground, configuration, step);
146 end
147 veh_pose = map.x(1:3);
```

5.2 目标定位模块 m 函数

```
1  function object_list_update = object_localization(object_list, object_num,...
2      innerLine_coordinate, innerLine_Vertex_index, ...
3      middleLine_coordinate, middleLine_Vertex_index,...
4      outerLine_coordinate, outerLine_Vertex_index,lane_offset,...
5       option)
6
7  object_list_update  = object_list;
8  % distance          = zeros(size(object_list,1),3);
9  side_direction      = zeros(size(object_list,1),3);
10 % nearestIndex      = zeros(size(object_list,1),3);
11
12 for  i = 1:object_num
13     if(object_list(i,1)~=0)
14         p = [object_list(i,2),object_list(i,4)];%2nd and 4th colomn are range x and range y
15         side_direction(i,1) = point_inside_lane(p,innerLine_coordinate, innerLine_Vertex_index);
```

```
        side_direction(i,2) = point_inside_lane(p,middleLine_coordinate,middleLine_Vertex_index);
%         side_direction(i,3) = point_inside_lane(p,outerLine_coordinate, outerLine_Vertex_index);
        side_direction(i,3) = point_inside_lane_offset(p,outerLine_coordinate, outerLine_Vertex_inc
    end
end

for i = 1:object_num
    if(object_list(i,1)~=0)
        if(option.clockWise == 0)%counter-clockwise
            switch(sum(side_direction(i,:)))
                case  3  %( 1  1  1) outside outer ring
                    object_list_update(i,9) = 3;
                case  1  %( 1  1 -1)
                    object_list_update(i,9) = 2; %outer lane
                case -1  %( 1 -1 -1)
                    object_list_update(i,9) = 1; %inner lane
                case -3  %(-1 -1 -1)
                    object_list_update(i,9) = 0; %inside inner ring
                otherwise
                    object_list_update(i,9) = 5;%warning('Unexpected location!')
            end
        else%clockwise
            switch(sum(side_direction(i,:)))%couterclockwise
                case -3 %( -1  -1  -1)
                    object_list_update(i,9) = 3;
                case -1 %( -1  -1   1)
                    object_list_update(i,9) = 2; %outer lane
                case  1   %( -1   1   1)
                    object_list_update(i,9) = 1; %inner lane
                case  3   %(  1   1   1)
                    object_list_update(i,9) = 0; %inside inner ring
                otherwise
                    object_list_update(i,9) = 5;%warning('Unexpected location!')
            end
        end
    end
end
```

5.3 地标搜索模块 m 函数

```
function [landmarks_in_proximity_id_in_front, landmarks_in_proximity_id_in_rear] = quest_m
    configuration,option)

% landmarks_in_proximity = zeros(size(landmarks));
%Landmarks
x1 = (double(landmarks(:,1))+double(landmarks(:,3)))/2 - ones(size(landmarks,1),1)*x_in_lcs;%lan
y1 = (double(landmarks(:,2))+double(landmarks(:,4)))/2 - ones(size(landmarks,1),1)*y_in_lcs;
% visible = find( (abs(x1) <= roi.x) & (abs(y1) <= roi.y) );
% distance2ego = (x1.^2 + y1.^2);
% [min_dist, ind] = min(distance2ego);
visible = find( (x1.^2 + y1.^2) <= configuration.proximity*configuration.proximity);



%%%%%%%%%%%%%%%%%%%

if(size(visible,1)~=0)
    p = [x_in_lcs,y_in_lcs];
    [~,~,nearestIndex_Ego] = calDistance(p,middleLine_coordinate,middleLine_Vertex_index,option

    landmarks_in_proximity = landmarks(visible,:);
    lat_distance_ldm       = zeros(size(visible,1),1);
    side_direction_ldm     = zeros(size(visible,1),1);
    nearestIndex_ldm       = zeros(size(visible,1),1);

    %Initialize search
%     p = (landmarks_in_proximity(1,1:2)+ landmarks_in_proximity(1,3:4))/2; %
%     [lat_distance_ldm(1),side_direction_ldm(1),nearestIndex_ldm(1)] = calDistance(p,middleLine
%     ldm_closest_in_path_ID_Index = nearestIndex_ldm(1);
    ldm_closest_in_path_ID = -1;
    ldm_closest_in_path_ID_Index = nearestIndex_Ego;

    %Find the closest landmark in front
    for  i = 1:size(visible,1)
        p = (landmarks_in_proximity(i,1:2)+ landmarks_in_proximity(i,3:4))/2; %
        [lat_distance_ldm(i),side_direction_ldm(i),nearestIndex_ldm(i)]=calDistance(p,middleLine_
        if(option.clockWise == 0) %counter-clockwise
```

```
38            if (nearestIndex_ldm(i) >= nearestIndex_Ego && ...%middleLine Index-->(2);in front of t
39                (nearestIndex_ldm(i) < ldm_closest_in_path_ID_Index...
40                 || ldm_closest_in_path_ID <0 )...
41                )%nearer than the closest so far
42                ldm_closest_in_path_ID = i;
43                ldm_closest_in_path_ID_Index = nearestIndex_ldm(i);
44            end
45        else %clockwise
46            if (nearestIndex_ldm(i) <= nearestIndex_Ego && ...%middleLine Index-->(2);in front of t
47                (nearestIndex_ldm(i) > ldm_closest_in_path_ID_Index...
48                 || ldm_closest_in_path_ID <0)...
49                )%nearer than the closest so far, clockwise is greater Index --> ">"
50                ldm_closest_in_path_ID = i;
51                ldm_closest_in_path_ID_Index = nearestIndex_ldm(i);
52            end
53        end
54    end
55
56    landmarks_in_proximity_id_in_front = zeros(size(landmarks,1),1);
57    landmarks_in_proximity_id_in_rear  = zeros(size(landmarks,1),1);
58    if(ldm_closest_in_path_ID<0)%no front landmarks all are behind the ego vehicle.
59        if(option.clockWise == 0) %counter-clockwise
60            landmarks_in_proximity_id_in_rear(1:size(visible,1)) = flipud(visible);
61        else%clockwise
62            landmarks_in_proximity_id_in_rear(1:size(visible,1)) = visible;
63        end
64    elseif(abs(ldm_closest_in_path_ID-1)<1e-3)
65        if(option.clockWise == 0) % counter-clockwise
66            if(middleLine_Vertex_index(1,1)<nearestIndex_Ego && nearestIndex_Ego<middleLine_V
67                ldm_ID = visible(ldm_closest_in_path_ID);
68                if(nearestIndex_Ego>2)
69                    front_cyclic_1 = find(visible<=79);
70                    front_cyclic_2 = find(visible>=ldm_ID);
71                    if(isempty(front_cyclic_1))
72                        front_cyclic_1=zeros(0,1);
73                    end
74                    if(isempty(front_cyclic_2))
75                        front_cyclic_2=zeros(0,1);
76                    end
```

```
77                  front_cyclic = intersect(front_cyclic_1,front_cyclic_2);
78                  rear_cyclic_1 = find(visible>79);
79                  rear_cyclic_2 = find(visible<ldm_ID);
80                  rear_size_1 = size(rear_cyclic_1,1);
81                  rear_size_2 = size(rear_cyclic_2,1);
82  %                   rear_cyclic = union(rear_cyclic_1,rear_cyclic_2);
83                  front_size = size(front_cyclic,1);
84  %                   rear_size  =  size(rear_cyclic,1);
85                  if(front_size~=0)
86                      landmarks_in_proximity_id_in_front(1:front_size) = visible(front_cyclic);
87                  end
88                  if(rear_size_2~=0)
89                      landmarks_in_proximity_id_in_rear(1:rear_size_2)= flipud(visible(rear_cyclic_
90                  end
91                  if(rear_size_1~=0)
92                      landmarks_in_proximity_id_in_rear(rear_size_2+1:rear_size_2+rear_size_1)=
93                  end
94              else %if(nearestIndex_Ego==2
95                  front_cyclic_1 = find(visible<=79);
96                  front_cyclic_2_1 = find(nearestIndex_ldm==2);
97                  front_cyclic_2_2 = find(visible>79);
98                  if(isempty(front_cyclic_2_1))
99                      front_cyclic_2_1=zeros(0,1);
100                 end
101                 if(isempty(front_cyclic_2_2))
102                     front_cyclic_2_2=zeros(0,1);
103                 end
104                 front_cyclic_2 = intersect(front_cyclic_2_1,front_cyclic_2_2);
105                 front_size_1 = size(front_cyclic_1,1);
106                 front_size_2 = size(front_cyclic_2,1);
107 %                   front_cyclic = union(front_cyclic_1,front_cyclic_2);
108 %                 front_size = size(front_cyclic,1);
109                 rear_cyclic_1 = find(visible>79);
110                 rear_cyclic_2 = find(nearestIndex_ldm~=2);
111                 if(isempty(rear_cyclic_1))
112                     rear_cyclic_1=zeros(0,1);
113                 end
114                 if(isempty(rear_cyclic_2))
115                     rear_cyclic_2=zeros(0,1);
```

```
116                 end
117                 rear_cyclic = intersect(rear_cyclic_1,rear_cyclic_2);
118                 rear_size  =  size(rear_cyclic,1);
119 %                   front_cyclic = find(visible<=79 || visible>=ldm_ID);
120 %                    rear_cyclic = find( visible>79 && visible<ldm_ID);
121 %                   front_size = size(front_cyclic,1);
122 %                   rear_size  =  size(rear_cyclic,1);
123                 if(front_size_2~=0)
124                     landmarks_in_proximity_id_in_front(1:front_size_2) = visible(front_cyclic_2);
125                 end
126                 if(front_size_1~=0)
127                     landmarks_in_proximity_id_in_front(front_size_2+1:front_size_2+front_size_
128                 end
129                 if(rear_size~=0)
130                     landmarks_in_proximity_id_in_rear(1:rear_size)= flipud(visible(rear_cyclic));
131                 end
132             end
133         elseif(middleLine_Vertex_index(4,1)<nearestIndex_Ego && nearestIndex_Ego<middleLin
134             ldm_ID = visible(ldm_closest_in_path_ID);
135             front_cyclic_1 = find(visible <= 21);
136             front_cyclic_2 = find(visible>=ldm_ID);
137 %               front_cyclic = union(front_cyclic_1,front_cyclic_2);
138             rear_cyclic_1 = find(visible > 21);
139             rear_cyclic_2 = find(visible < ldm_ID);
140             if(isempty(rear_cyclic_1))
141                 rear_cyclic_1=zeros(0,1);
142             end
143             if(isempty(rear_cyclic_2))
144                 rear_cyclic_2=zeros(0,1);
145             end
146             rear_cyclic = intersect(rear_cyclic_1,rear_cyclic_2);
147 %               front_size = size(front_cyclic,1);
148             rear_size  =  size(rear_cyclic,1);
149
150             front_size_1 = size(front_cyclic_1,1);
151             front_size_2 = size(front_cyclic_2,1);
152 %               rear_size_1  =  size(rear_cyclic_1,1);
153 %               rear_size_2  =  size(rear_cyclic_2,1);
154             if(front_size_2~=0)
```

```
155                 landmarks_in_proximity_id_in_front(1:front_size_2) = visible(front_cyclic_2);
156             end
157             if(front_size_1~=0)
158                 landmarks_in_proximity_id_in_front(front_size_2+1:(front_size_1+front_size_2)
159             end
160             if(rear_size~=0)
161                 landmarks_in_proximity_id_in_rear(1:rear_size)= flipud(visible(rear_cyclic));
162             end
163         else
164             landmarks_in_proximity_id_in_front(1:size(visible,1)) = visible;
165         end
166     end
167 else% (ldm_closest_in_path_ID>1)
168     if(option.clockWise == 0) % counter-clockwise
169         if(middleLine_Vertex_index(1,1)<nearestIndex_Ego && nearestIndex_Ego<middleLine_
170             ldm_ID = visible(ldm_closest_in_path_ID);
171             if(nearestIndex_Ego>2)
172                 front_cyclic_1 = find(visible<=79);
173                 front_cyclic_2 = find(visible>=ldm_ID);
174                 if(isempty(front_cyclic_1))
175                     front_cyclic_1=zeros(0,1);
176                 end
177                 if(isempty(front_cyclic_2))
178                     front_cyclic_2=zeros(0,1);
179                 end
180                 front_cyclic = intersect(front_cyclic_1,front_cyclic_2);
181                 rear_cyclic_1 = find(visible>79);
182                 rear_cyclic_2 = find(visible<ldm_ID);
183                 rear_size_1 = size(rear_cyclic_1,1);
184                 rear_size_2 = size(rear_cyclic_2,1);
185 %                   rear_cyclic = union(rear_cyclic_1,rear_cyclic_2);
186                 front_size = size(front_cyclic,1);
187 %                   rear_size  =  size(rear_cyclic,1);
188                 if(front_size~=0)
189                     landmarks_in_proximity_id_in_front(1:front_size) = visible(front_cyclic);
190                 end
191                 if(rear_size_2~=0)
192                     landmarks_in_proximity_id_in_rear(1:rear_size_2)= flipud(visible(rear_cyclic_
193                 end
```

```
194                 if(rear_size_1~=0)
195                     landmarks_in_proximity_id_in_rear(rear_size_2+1:rear_size_2+rear_size_1)=
196                 end
197             else %if(nearestIndex_Ego
198                 front_cyclic_1 = find(visible<=79);
199                 front_cyclic_2_1 = find(nearestIndex_ldm==2);
200                 front_cyclic_2_2 = find(visible>79);
201                 if(isempty(front_cyclic_2_1))
202                     front_cyclic_2_1=zeros(0,1);
203                 end
204                 if(isempty(front_cyclic_2_2))
205                     front_cyclic_2_2=zeros(0,1);
206                 end
207                 front_cyclic_2 = intersect(front_cyclic_2_1,front_cyclic_2_2);
208                 front_size_1 = size(front_cyclic_1,1);
209                 front_size_2 = size(front_cyclic_2,1);
210                 rear_cyclic_1 = find(visible>79);
211                 rear_cyclic_2 = find(nearestIndex_ldm~=2);
212                 if(isempty(rear_cyclic_1))
213                     rear_cyclic_1=zeros(0,1);
214                 end
215                 if(isempty(rear_cyclic_2))
216                     rear_cyclic_2=zeros(0,1);
217                 end
218                 rear_cyclic = intersect(rear_cyclic_1,rear_cyclic_2);
219                 rear_size  =  size(rear_cyclic,1);
220
221                 if(front_size_2~=0)
222                     landmarks_in_proximity_id_in_front(1:front_size_2) = visible(front_cyclic_2);
223                 end
224                 if(front_size_1~=0)
225                     landmarks_in_proximity_id_in_front(front_size_2+1:front_size_2+front_size_
226                 end
227                 if(rear_size~=0)
228                     landmarks_in_proximity_id_in_rear(1:rear_size)= flipud(visible(rear_cyclic));
229                 end
230             end
231         elseif(middleLine_Vertex_index(4,1)<nearestIndex_Ego && nearestIndex_Ego<middleLin
232             ldm_ID = visible(ldm_closest_in_path_ID);
```

```
233                 front_cyclic_1 = find(visible <= 21);
234                 front_cyclic_2 = find(visible>=ldm_ID);
235                 rear_cyclic_1 = find(visible > 21);
236                 rear_cyclic_2 = find(visible < ldm_ID);
237                 if(isempty(rear_cyclic_1))
238                     rear_cyclic_1=zeros(0,1);
239                 end
240                 if(isempty(rear_cyclic_2))
241                     rear_cyclic_2=zeros(0,1);
242                 end
243                 rear_cyclic = intersect(rear_cyclic_1,rear_cyclic_2);
244                 rear_size  =  size(rear_cyclic,1);
245
246                 front_size_1 = size(front_cyclic_1,1);
247                 front_size_2 = size(front_cyclic_2,1);
248                 if(front_size_2~=0)
249                     landmarks_in_proximity_id_in_front(1:front_size_2) = visible(front_cyclic_2);
250                 end
251                 if(front_size_1~=0)
252                     landmarks_in_proximity_id_in_front(front_size_2+1:(front_size_1+front_size_2)
253                 end
254                 if(rear_size~=0)
255                     landmarks_in_proximity_id_in_rear(1:rear_size)= flipud(visible(rear_cyclic));
256                 end
257             else
258                 front_size = size(visible,1) - ldm_closest_in_path_ID+1;
259                 rear_size = ldm_closest_in_path_ID-1;
260                 landmarks_in_proximity_id_in_front(1:front_size) = visible(ldm_closest_in_path_ID
261                 landmarks_in_proximity_id_in_rear(1:rear_size)= flipud(visible(1:ldm_closest_in_pa
262             end
263         else%clockwise
264             front_size = ldm_closest_in_path_ID;
265             rear_size = size(visible,1) - ldm_closest_in_path_ID;
266             landmarks_in_proximity_id_in_front(1:front_size) = flipud(visible(1:ldm_closest_in_pat
267             landmarks_in_proximity_id_in_rear(1:rear_size)= visible(ldm_closest_in_path_ID+1:en
268         end
269     end
270 else
271     landmarks_in_proximity_id_in_front = zeros(size(landmarks,1),1);
```

```
    landmarks_in_proximity_id_in_rear  = zeros(size(landmarks,1),1);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.4 目标定位模块 m 函数

```
function object_list_update = object_localization(object_list, object_num,...
    innerLine_coordinate, innerLine_Vertex_index, ...
    middleLine_coordinate, middleLine_Vertex_index,...
    outerLine_coordinate, outerLine_Vertex_index,lane_offset,...
     option)

object_list_update  = object_list;
side_direction      = zeros(size(object_list,1),3);

for  i = 1:object_num
    if(object_list(i,1)~=0)
        p = [object_list(i,2),object_list(i,4)];%2nd and 4th colomn are range x and range y
        side_direction(i,1) = point_inside_lane(p,innerLine_coordinate, innerLine_Vertex_index);
        side_direction(i,2) = point_inside_lane(p,middleLine_coordinate,middleLine_Vertex_index);
        side_direction(i,3) = point_inside_lane_offset(p,outerLine_coordinate, outerLine_Vertex_inc
    end
end

for i = 1:object_num
    if(object_list(i,1)~=0)
        if(option.clockWise == 0)%counter-clockwise
            switch(sum(side_direction(i,:)))
                case  3  %( 1  1  1) outside outer ring
                    object_list_update(i,9) = 3;
                case  1  %( 1  1 -1)
                    object_list_update(i,9) = 2; %outer lane
                case -1  %( 1 -1 -1)
                    object_list_update(i,9) = 1; %inner lane
                case -3  %(-1 -1 -1)
                    object_list_update(i,9) = 0; %inside inner ring
                otherwise
                    object_list_update(i,9) = 5;%warning('Unexpected location!')
            end
        else%clockwise
```

```
            switch(sum(side_direction(i,:)))%couterclockwise
                case -3 %( -1  -1  -1)
                    object_list_update(i,9) = 3;
                case -1 %( -1  -1   1)
                    object_list_update(i,9) = 2; %outer lane
                case  1   %( -1   1   1)
                    object_list_update(i,9) = 1; %inner lane
                case  3   %(  1   1   1)
                    object_list_update(i,9) = 0; %inside inner ring
                otherwise
                    object_list_update(i,9) = 5;%warning('Unexpected location!')
            end
        end
    end
end
```

5.5 地标搜索模块 m 函数

```
function [obj_closest_in_path_ID, obj_closest_in_next_path_ID, ego_location] = objects_of_int
    x_in_lcs, y_in_lcs, ...
    innerLine_coordinate, innerLine_Vertex_index, ...
    middleLine_coordinate, middleLine_Vertex_index,...
    outerLine_coordinate, outerLine_Vertex_index,...
     option)

inside      = zeros(1,3);
p = [x_in_lcs,y_in_lcs];
inside(1) = point_inside_lane(p,innerLine_coordinate, innerLine_Vertex_index);
inside(2) = point_inside_lane(p,middleLine_coordinate, middleLine_Vertex_index);
inside(3) = point_inside_lane(p,outerLine_coordinate, outerLine_Vertex_index);

if(option.clockWise == 0)%counter-clockwise
    switch(sum(inside))
        case  3  %( 1  1  1) outside outer ring
            ego_location = 3;
        case  1  %( 1  1 -1)
            ego_location = 2; %outer lane
        case -1  %( 1 -1 -1)
            ego_location = 1; %inner lane
```

```
        case -3  %(-1 -1 -1)
            ego_location = 0; %inside inner ring
        otherwise
            ego_location = 5;%warning('Unexpected location!')
    end
else%clockwise
    switch(sum(inside))%couterclockwise
        case -3 %( -1  -1  -1)
            ego_location = 3;
        case -1 %( -1  -1   1)
            ego_location = 2; %outer lane
        case  1 %( -1   1   1)
            ego_location = 1; %inner lane
        case  3 %(  1   1   1)
            ego_location = 0; %inside inner ring
        otherwise
            ego_location = 5;%warning('Unexpected location!')
    end
end

if( ~(ego_location==2 || ego_location==1) )
    obj_closest_in_path_ID = 0;
    obj_closest_in_next_path_ID = 0;
    return;
end
if(ego_location == 1)
    next_lane = 2;
else
    next_lane = 1;
end


[projection_dist,~]=point_dist2lane(p,outerLine_coordinate,outerLine_Vertex_index);
[~,sideIndex]=min(projection_dist);

%find the projected point of vehicle on the outer lane mark and its distance to the 1st corner.
corner_1=outerLine_coordinate(outerLine_Vertex_index(sideIndex,1),:);
corner_2=outerLine_coordinate(outerLine_Vertex_index(sideIndex,2),:);
[~, dist2StartCorner_veh] = point_projection2LNM(p, corner_1, corner_2);
```

```
61
62
63  %find object in ego lane and next lane
64  object_list_lane_loc = object_list(1:object_num,9) ;
65
66  objects_in_ego_lane_id = find(object_list_lane_loc == ego_location);
67  objects_in_next_lane_id = find( object_list_lane_loc == next_lane);
68
69  % objects_in_ego_lane_id(1:size(objects_in_next_lane_id_temp,1))=objects_in_ego_lane_id_te
70  max_len = max(size(objects_in_ego_lane_id,1),size(objects_in_next_lane_id,1));
71  dist2StartCorner_obj      = zeros(max_len,2);
72
73  if(size(objects_in_ego_lane_id,1)~=0)
74      objects_in_ego_lane  =  [object_list(objects_in_ego_lane_id,2),object_list(objects_in_ego_la
75      obj_closest_in_path_ID_dist = dist2StartCorner_veh+500;
76      obj_closest_in_path_ID = 0;
77      %Find the closest in-path object (CIPO)
78      for  i = 1:size(objects_in_ego_lane,1)
79          p = objects_in_ego_lane(i,:);
80          [~,dist2StartCorner_obj(i,1)]=point_projection2LNM(p, corner_1, corner_2); % projection on
81
82          if (dist2StartCorner_obj(i,1) >= dist2StartCorner_veh && ...
83                  dist2StartCorner_obj(i,1) <= obj_closest_in_path_ID_dist)
84              obj_closest_in_path_ID = objects_in_ego_lane_id(i);
85              obj_closest_in_path_ID_dist = dist2StartCorner_obj(i,1);
86          end
87      end
88  else
89      obj_closest_in_path_ID = 0;
90  end
91
92  % obj_closest_in_next_path_ID =0;
93  if(size(objects_in_next_lane_id,1)~=0)
94      objects_in_next_lane =  [object_list(objects_in_next_lane_id,2),object_list(objects_in_next_
95      obj_closest_in_path_ID_dist = dist2StartCorner_veh+500;
96      obj_closest_in_next_path_ID = 0;
97
98      %Find the closest Next Lane object
99      for  i = 1:size(objects_in_next_lane,1)
```

```
100         p = objects_in_next_lane(i,:);
101         [~, dist2StartCorner_obj(i,2)] = point_projection2LNM(p, corner_1, corner_2);
102         if (dist2StartCorner_obj(i,2) > dist2StartCorner_veh && ...
103                 dist2StartCorner_obj(i,2) < obj_closest_in_path_ID_dist)
104             obj_closest_in_next_path_ID = objects_in_next_lane_id(i);
105             obj_closest_in_path_ID_dist = dist2StartCorner_obj(i,2);
106         end
107     end
108 else
109     obj_closest_in_next_path_ID = 0;
110 end
111 %Sort objects in ego and next lane in driving direction and in the proximity sequence of ego vehicle.
```

References

[1] 那汤松. 实变函数论（第 5 版）. 徐瑞云译. 北京：高等教育出版社，2010.

[2] 郭大钧等. 实变函数与泛函分析（第二版）· 下册. 山东：山东大学出版社，2005.

[3] 夏道行等. 实变函数论与泛函分析（下册 · 第二版修订本）. 北京：高等教育出版社，2010.

[4] A.H. 柯尔莫戈洛夫，C.B. 佛明. 函数论与泛函分析初步（第 7 版）. 北京：高等教育出版社，2006.