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

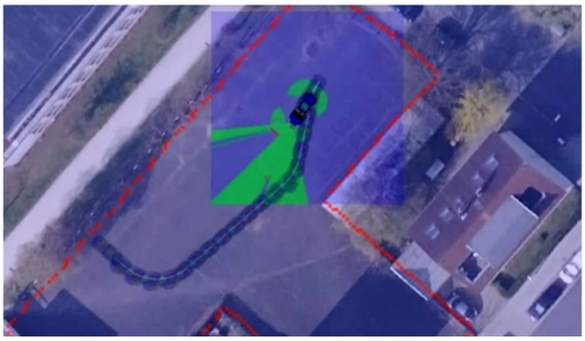
Xin Binjian

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

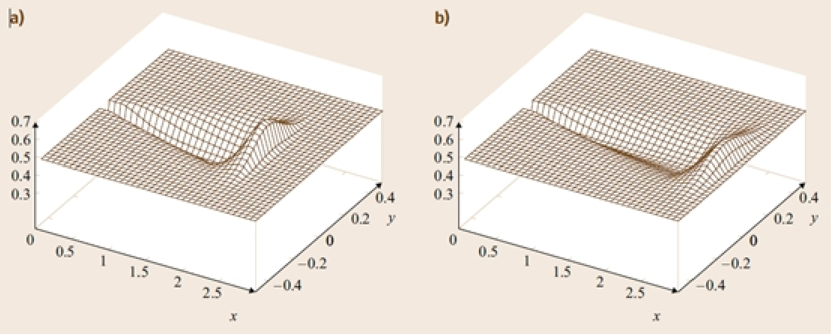
RTK，高精度地图，4个SRR，1个ESR；若干地标物（landmark, beacon, 交通标示）道路模型（局部地图）需要处理未知，占据和非占据信息。接口静态的占据格栅图地图OGM+动态的目标列表。根据静态地标占据格栅图用来更新本车姿态，叠加动态的障碍物，局部地图：障碍物列表，车道，静态地标，RTK车辆定位--> 路径规划。



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

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

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

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

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

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

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

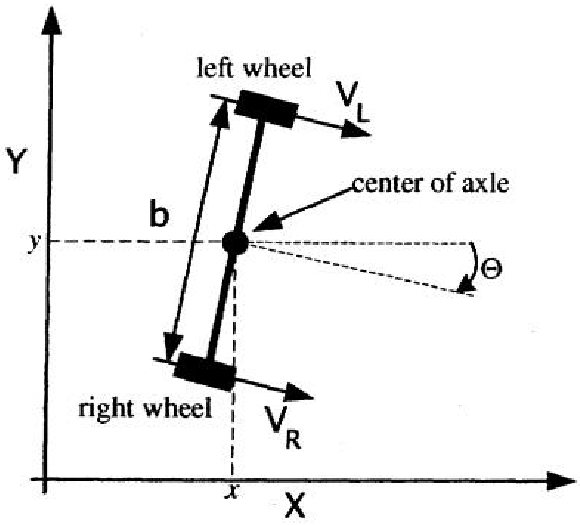
$$\omega_k = \frac{v_{R,k} + v_{L,k}}{b} = \frac{\omega_{R,k}R - \omega_{L,k}R}{b} \tag{8}$$

标定

$$\begin{aligned} v_{t,k} &= \frac{k_1 \cdot v_{L,k} + k_2 \cdot v_{R,k}}{2} && (9) \\ \omega_k &= \frac{k_2 \cdot v_{R,k} - k_1 \cdot v_{L,k}}{k_3 \cdot b} && (10) \end{aligned}$$

$u_k = (v_k, \omega_k)^T$
预测

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, v_k) && (11) \\ f(x_k, u_k, v_k) &= \begin{cases} x_k + (D_k + v_{1,k} \cdot \cos(\theta_k + \Delta\theta_k + v_{2,k}) \\ y_k + (D_k + v_{1,k} \cdot \sin(\theta_k + \Delta\theta_k + v_{2,k}) \\ \theta_k + \Delta\theta_k + v_{2,k} \end{cases} && (12) \end{aligned}$$



4.2 观测模型

基于距离变换 DT

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

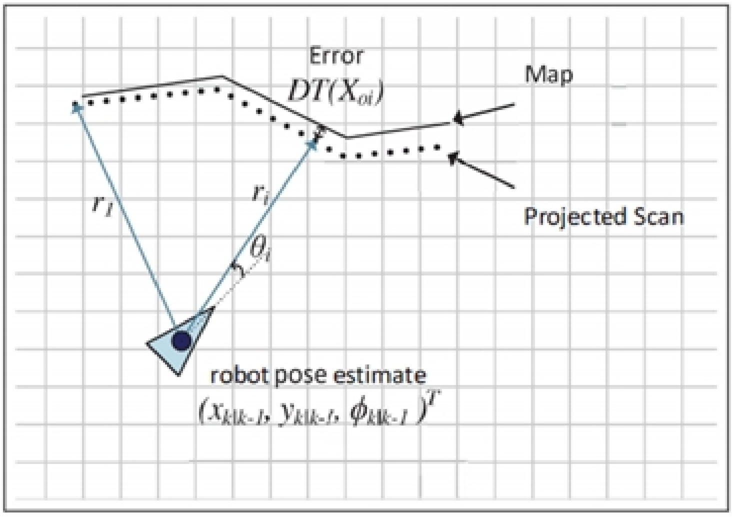
Chamfer distance CD

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

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

隐式观测模型

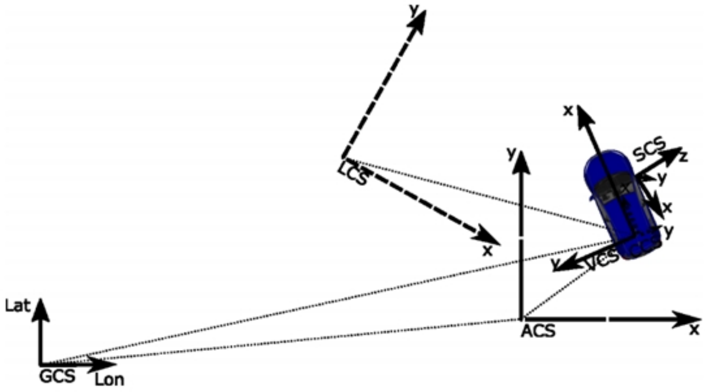
$$h(\mathrm{X}, \mathrm{z})$$



4.3 更新

$$
\begin{aligned}
K &= P_{k|k-1}\nabla h_{\mathrm{X}}^T(\nabla h_{\mathrm{X}}P_{k|k-1}\nabla h_{\mathrm{X}}^T + \nabla h_{\mathrm{z}}^T\Sigma_{\mathrm{z}}\nabla h_{\mathrm{z}})^{-1} \quad (13)\\
X_{k|k} &= X_{k|k-1} + K(-h((X)_{k|k-1}, \mathrm{z})) \quad (14)\\
P_{k|k} &= (I - K\nabla h_{\mathrm{X}})P_{k|k-1} \quad (15)
\end{aligned}
$$

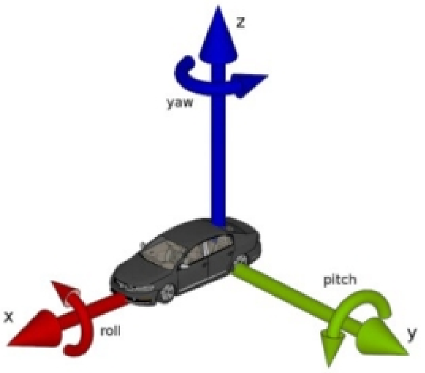
4.4 参考坐标系

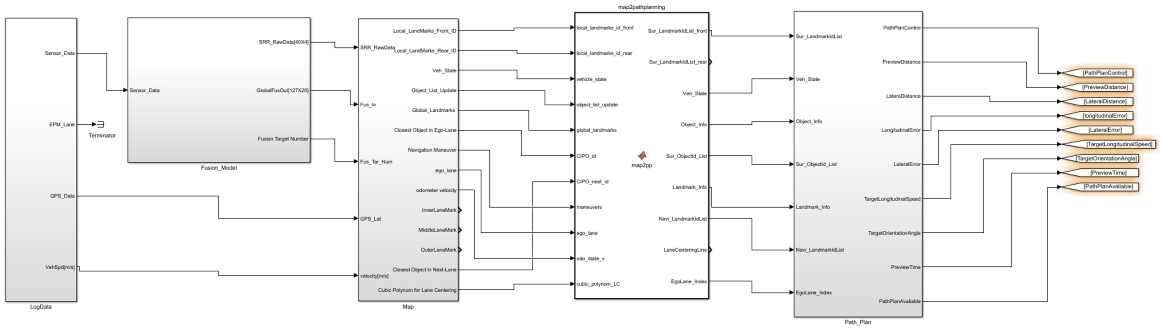


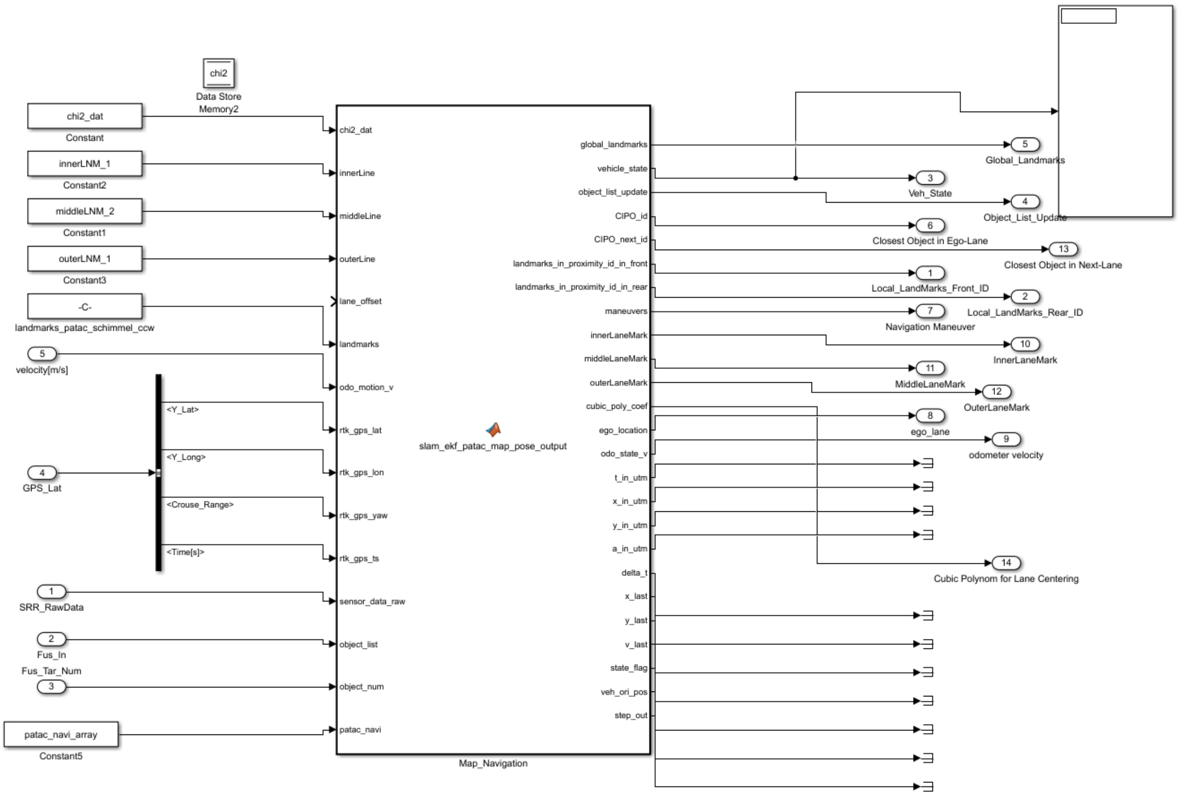
5 附录

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

    % ok, here we go

    %%observations = get_observations(ground, sensor, step);
    [x1,y1,utmzone,utmhemi] = wgs2utm(rtk_gps_lat,rtk_gps_lon,51,'N');
    veh_origin_pose.x = x1;
    veh_origin_pose.y = y1;
    veh_origin_pose.yaw = rtk_gps_yaw;
    veh_origin_pose.ts = rtk_gps_ts;
    veh_pose = veh_origin_pose;
    veh_vel = [0; 0]; % Start point with 0 velocity.
    observations = get_observations (ground, landmarks, veh_pose, ...
                                    sensor_data_raw, sensor, proximity);
    draw_observations (observations, configuration, step);

%      GT = zeros(1, size(sensor_data_raw,1));
%       H = zeros(1, size(sensor_data_raw,1));

    map = add_features(map, observations);
    % plot map
    draw_map (map, ground,configuration,step);

   % steps = length(ground.motion);
else

    step = step+1;
    disp('---------------------------------------------------------------------------');
%      disp(sprintf('Step: %d', step));

    % EKF prediction step
    odometry.x = [odo_motion_x, odo_motion_y,odo_motion_yaw]'; ;%odo_motion.x;
    odometry.P = diag([0.25 0.1 5*pi/180].^2);


    map = EKF_prediction (map, odometry);

    % sense
    [x1,y1,utmzone,utmhemi] = wgs2utm(rtk_gps_lat,rtk_gps_lon,51,'N');
    veh_pose.x = x1;
    veh_pose.y = y1;
    veh_pose.yaw = rtk_gps_yaw;
    veh_pose.ts = rtk_gps_ts;

    [x2,y2,utmzone,utmhemi] = wgs2utm(rtk_gps_lat_last,rtk_gps_lon_last,51,'N');
    veh_vel = [x1-x2; y1-y2]/(rtk_gps_ts - rtk_gps_ts_last)/1000;%ms-->s

    rtk_gps_lat_last =rtk_gps_lat;
    rtk_gps_lon_last = rtk_gps_lon;
    rtk_gps_ts_last = rtk_gps_ts;


    motion.x = [x1 y1 rtk_gps_yaw]';
    motion.P = diag([0.02 0.02 2*pi/180].^2); % expectation of std of RTK
    ground = move_vehicle (ground, motion, step);

    observations = get_observations(ground, landmarks, veh_pose, ...
                                    sensor_data_raw, sensor, proximity);

    % individual compatibility
    prediction = predict_observations (map, ground);
    compatibility = compute_compatibility (prediction, observations);

    disp(compatibility.HS);
    disp(compatibility.AL);

    disp(' ');

    % ground truth
    % your algorithm here!
    % 1. Try NN
    % 2. Complete SINGLES and try it
    % 3. Include people and try SINGLES5
    % 4. Try JCBB

    H = NN (prediction, observations, compatibility,configuration);

    draw_map (map, ground,configuration, step);
    draw_observations (observations, configuration, step);
```

```
    draw_compatibility (prediction, observations, compatibility,configuration);

    disp(' ');

    draw_hypothesis (prediction, observations, H, 'NN:', 'b-',configuration);

    % update EKF step
    map = EKF_update (map, prediction, observations, H, step);


    % only new features with no neighbours
    new = find((H == 0) & (compatibility.AL == 0));

    if nnz(new)
        map = add_features(map, observations, new);
    end

    draw_map (map, ground, configuration, step);
end
veh_pose = map.x(1:3);
```

5.2 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;
% distance            = zeros(size(object_list,1),3);
side_direction       = zeros(size(object_list,1),3);
% nearestIndex       = 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(p,outerLine_coordinate, outerLine_Vertex_index);
        side_direction(i,3) = point_inside_lane_offset(p,outerLine_coordinate, outerLine_Vertex_in
    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,middleLin
%     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
        if (nearestIndex_ldm(i) >= nearestIndex_Ego && ...%middleLine Index-->(2);in front of t
                (nearestIndex_ldm(i) < ldm_closest_in_path_ID_Index...
                 || ldm_closest_in_path_ID <0 )...
            )%nearer than the closest so far
            ldm_closest_in_path_ID = i;
            ldm_closest_in_path_ID_Index = nearestIndex_ldm(i);
        end
    else %clockwise
        if (nearestIndex_ldm(i) <= nearestIndex_Ego && ...%middleLine Index-->(2);in front of t
                (nearestIndex_ldm(i) > ldm_closest_in_path_ID_Index...
                 || ldm_closest_in_path_ID <0)...
            )%nearer than the closest so far, clockwise is greater Index --> ”>”
            ldm_closest_in_path_ID = i;
            ldm_closest_in_path_ID_Index = nearestIndex_ldm(i);
        end
    end
end

landmarks_in_proximity_id_in_front = zeros(size(landmarks,1),1);
landmarks_in_proximity_id_in_rear  = zeros(size(landmarks,1),1);
if(ldm_closest_in_path_ID<0)%no front landmarks all are behind the ego vehicle.
    if(option.clockWise == 0) %counter-clockwise
        landmarks_in_proximity_id_in_rear(1:size(visible,1)) = flipud(visible);
    else%clockwise
        landmarks_in_proximity_id_in_rear(1:size(visible,1)) = visible;
    end
elseif(abs(ldm_closest_in_path_ID-1)<1e-3)
    if(option.clockWise == 0) % counter-clockwise
        if(middleLine_Vertex_index(1,1)<nearestIndex_Ego && nearestIndex_Ego<middleLine_V
            ldm_ID = visible(ldm_closest_in_path_ID);
            if(nearestIndex_Ego>2)
                front_cyclic_1 = find(visible<=79);
                front_cyclic_2 = find(visible>=ldm_ID);
                if(isempty(front_cyclic_1))
                    front_cyclic_1=zeros(0,1);
                end
                if(isempty(front_cyclic_2))
                    front_cyclic_2=zeros(0,1);
                end
```

```
                    front_cyclic = intersect(front_cyclic_1,front_cyclic_2);
                    rear_cyclic_1 = find(visible>79);
                    rear_cyclic_2 = find(visible<ldm_ID);
                    rear_size_1 = size(rear_cyclic_1,1);
                    rear_size_2 = size(rear_cyclic_2,1);
%                      rear_cyclic = union(rear_cyclic_1,rear_cyclic_2);
                    front_size = size(front_cyclic,1);
%                      rear_size  =  size(rear_cyclic,1);
                    if(front_size~=0)
                        landmarks_in_proximity_id_in_front(1:front_size) = visible(front_cyclic);
                    end
                    if(rear_size_2~=0)
                        landmarks_in_proximity_id_in_rear(1:rear_size_2)= flipud(visible(rear_cyclic_
                    end
                    if(rear_size_1~=0)
                        landmarks_in_proximity_id_in_rear(rear_size_2+1:rear_size_2+rear_size_1)=
                    end
                else %if(nearestIndex_Ego==2
                    front_cyclic_1 = find(visible<=79);
                    front_cyclic_2_1 = find(nearestIndex_ldm==2);
                    front_cyclic_2_2 = find(visible>79);
                    if(isempty(front_cyclic_2_1))
                        front_cyclic_2_1=zeros(0,1);
                    end
                    if(isempty(front_cyclic_2_2))
                        front_cyclic_2_2=zeros(0,1);
                    end
                    front_cyclic_2 = intersect(front_cyclic_2_1,front_cyclic_2_2);
                    front_size_1 = size(front_cyclic_1,1);
                    front_size_2 = size(front_cyclic_2,1);
%                      front_cyclic = union(front_cyclic_1,front_cyclic_2);
%                  front_size = size(front_cyclic,1);
                    rear_cyclic_1 = find(visible>79);
                    rear_cyclic_2 = find(nearestIndex_ldm~=2);
                    if(isempty(rear_cyclic_1))
                        rear_cyclic_1=zeros(0,1);
                    end
                    if(isempty(rear_cyclic_2))
                        rear_cyclic_2=zeros(0,1);
                    end
                    rear_cyclic = intersect(rear_cyclic_1,rear_cyclic_2);
                    rear_size  =  size(rear_cyclic,1);
%                   front_cyclic = find(visible<=79 || visible>=ldm_ID);
%                    rear_cyclic = find( visible>79 && visible<ldm_ID);
%                   front_size = size(front_cyclic,1);
%                   rear_size  =  size(rear_cyclic,1);
                    if(front_size_2~=0)
                        landmarks_in_proximity_id_in_front(1:front_size_2) = visible(front_cyclic_2);
                    end
                    if(front_size_1~=0)
                        landmarks_in_proximity_id_in_front(front_size_2+1:front_size_2+front_size_
                    end
                    if(rear_size~=0)
                        landmarks_in_proximity_id_in_rear(1:rear_size)= flipud(visible(rear_cyclic));
                    end
               end
            elseif(middleLine_Vertex_index(4,1)<nearestIndex_Ego && nearestIndex_Ego<middleLin
                ldm_ID = visible(ldm_closest_in_path_ID);
                front_cyclic_1 = find(visible <= 21);
                front_cyclic_2 = find(visible>=ldm_ID);
%               front_cyclic = union(front_cyclic_1,front_cyclic_2);
                rear_cyclic_1 = find(visible > 21);
                rear_cyclic_2 = find(visible < ldm_ID);
                if(isempty(rear_cyclic_1))
                    rear_cyclic_1=zeros(0,1);
                end
                if(isempty(rear_cyclic_2))
                    rear_cyclic_2=zeros(0,1);
                end
                rear_cyclic = intersect(rear_cyclic_1,rear_cyclic_2);
%               front_size = size(front_cyclic,1);
                rear_size  =  size(rear_cyclic,1);

                front_size_1 = size(front_cyclic_1,1);
                front_size_2 = size(front_cyclic_2,1);
%               rear_size_1  =  size(rear_cyclic_1,1);
%               rear_size_2  =  size(rear_cyclic_2,1);
                if(front_size_2~=0)
```

```
                    landmarks_in_proximity_id_in_front(1:front_size_2) = visible(front_cyclic_2);
                end
                if(front_size_1~=0)
                    landmarks_in_proximity_id_in_front(front_size_2+1:(front_size_1+front_size_2)
                end
                if(rear_size~=0)
                    landmarks_in_proximity_id_in_rear(1:rear_size)= flipud(visible(rear_cyclic));
                end
            else
                landmarks_in_proximity_id_in_front(1:size(visible,1)) = visible;
            end
        end
    else% (ldm_closest_in_path_ID>1)
        if(option.clockWise == 0) % counter-clockwise
            if(middleLine_Vertex_index(1,1)<nearestIndex_Ego && nearestIndex_Ego<middleLine_
                ldm_ID = visible(ldm_closest_in_path_ID);
                if(nearestIndex_Ego>2)
                    front_cyclic_1 = find(visible<=79);
                    front_cyclic_2 = find(visible>=ldm_ID);
                    if(isempty(front_cyclic_1))
                        front_cyclic_1=zeros(0,1);
                    end
                    if(isempty(front_cyclic_2))
                        front_cyclic_2=zeros(0,1);
                    end
                    front_cyclic = intersect(front_cyclic_1,front_cyclic_2);
                    rear_cyclic_1 = find(visible>79);
                    rear_cyclic_2 = find(visible<ldm_ID);
                    rear_size_1 = size(rear_cyclic_1,1);
                    rear_size_2 = size(rear_cyclic_2,1);
%                       rear_cyclic = union(rear_cyclic_1,rear_cyclic_2);
                    front_size = size(front_cyclic,1);
%                       rear_size  =  size(rear_cyclic,1);
                    if(front_size~=0)
                        landmarks_in_proximity_id_in_front(1:front_size) = visible(front_cyclic);
                    end
                    if(rear_size_2~=0)
                        landmarks_in_proximity_id_in_rear(1:rear_size_2)= flipud(visible(rear_cyclic_
                    end
                    if(rear_size_1~=0)
                        landmarks_in_proximity_id_in_rear(rear_size_2+1:rear_size_2+rear_size_1)=
                    end
                else %if(nearestIndex_Ego
                    front_cyclic_1 = find(visible<=79);
                    front_cyclic_2_1 = find(nearestIndex_ldm==2);
                    front_cyclic_2_2 = find(visible>79);
                    if(isempty(front_cyclic_2_1))
                        front_cyclic_2_1=zeros(0,1);
                    end
                    if(isempty(front_cyclic_2_2))
                        front_cyclic_2_2=zeros(0,1);
                    end
                    front_cyclic_2 = intersect(front_cyclic_2_1,front_cyclic_2_2);
                    front_size_1 = size(front_cyclic_1,1);
                    front_size_2 = size(front_cyclic_2,1);
                    rear_cyclic_1 = find(visible>79);
                    rear_cyclic_2 = find(nearestIndex_ldm~=2);
                    if(isempty(rear_cyclic_1))
                        rear_cyclic_1=zeros(0,1);
                    end
                    if(isempty(rear_cyclic_2))
                        rear_cyclic_2=zeros(0,1);
                    end
                    rear_cyclic = intersect(rear_cyclic_1,rear_cyclic_2);
                    rear_size  =  size(rear_cyclic,1);

                    if(front_size_2~=0)
                        landmarks_in_proximity_id_in_front(1:front_size_2) = visible(front_cyclic_2);
                    end
                    if(front_size_1~=0)
                        landmarks_in_proximity_id_in_front(front_size_2+1:front_size_2+front_size_
                    end
                    if(rear_size~=0)
                        landmarks_in_proximity_id_in_rear(1:rear_size)= flipud(visible(rear_cyclic));
                    end
               end
            elseif(middleLine_Vertex_index(4,1)<nearestIndex_Ego && nearestIndex_Ego<middleLin
                ldm_ID = visible(ldm_closest_in_path_ID);
```

```
                front_cyclic_1 = find(visible <= 21);
                front_cyclic_2 = find(visible>=ldm_ID);
                rear_cyclic_1 = find(visible > 21);
                rear_cyclic_2 = find(visible < ldm_ID);
                if(isempty(rear_cyclic_1))
                    rear_cyclic_1=zeros(0,1);
                end
                if(isempty(rear_cyclic_2))
                    rear_cyclic_2=zeros(0,1);
                end
                rear_cyclic = intersect(rear_cyclic_1,rear_cyclic_2);
                rear_size  =  size(rear_cyclic,1);

                front_size_1 = size(front_cyclic_1,1);
                front_size_2 = size(front_cyclic_2,1);
                if(front_size_2~=0)
                    landmarks_in_proximity_id_in_front(1:front_size_2) = visible(front_cyclic_2);
                end
                if(front_size_1~=0)
                    landmarks_in_proximity_id_in_front(front_size_2+1:(front_size_1+front_size_2)
                end
                if(rear_size~=0)
                    landmarks_in_proximity_id_in_rear(1:rear_size)= flipud(visible(rear_cyclic));
                end
            else
                front_size = size(visible,1) - ldm_closest_in_path_ID+1;
                rear_size = ldm_closest_in_path_ID-1;
                landmarks_in_proximity_id_in_front(1:front_size) = visible(ldm_closest_in_path_ID
                landmarks_in_proximity_id_in_rear(1:rear_size)= flipud(visible(1:ldm_closest_in_pa
            end
        else%clockwise
            front_size = ldm_closest_in_path_ID;
            rear_size = size(visible,1) - ldm_closest_in_path_ID;
            landmarks_in_proximity_id_in_front(1:front_size) = flipud(visible(1:ldm_closest_in_pat
            landmarks_in_proximity_id_in_rear(1:rear_size)= visible(ldm_closest_in_path_ID+1:e
        end
    end
else
    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_in
    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_in
    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);
```

```
%find object in ego lane and next lane
object_list_lane_loc = object_list(1:object_num,9) ;

objects_in_ego_lane_id = find(object_list_lane_loc == ego_location);
objects_in_next_lane_id = find( object_list_lane_loc == next_lane);

% objects_in_ego_lane_id(1:size(objects_in_next_lane_id_temp,1))=objects_in_ego_lane_id_t
max_len = max(size(objects_in_ego_lane_id,1),size(objects_in_next_lane_id,1));
dist2StartCorner_obj      = zeros(max_len,2);

if(size(objects_in_ego_lane_id,1)~=0)
    objects_in_ego_lane  =  [object_list(objects_in_ego_lane_id,2),object_list(objects_in_ego_la
    obj_closest_in_path_ID_dist = dist2StartCorner_veh+500;
    obj_closest_in_path_ID = 0;
    %Find the closest in-path object (CIPO)
    for  i = 1:size(objects_in_ego_lane,1)
        p = objects_in_ego_lane(i,:);
        [~,dist2StartCorner_obj(i,1)]=point_projection2LNM(p, corner_1, corner_2); % projection o

        if (dist2StartCorner_obj(i,1) >= dist2StartCorner_veh && ...
                dist2StartCorner_obj(i,1) <= obj_closest_in_path_ID_dist)
            obj_closest_in_path_ID = objects_in_ego_lane_id(i);
            obj_closest_in_path_ID_dist = dist2StartCorner_obj(i,1);
        end
    end
else
    obj_closest_in_path_ID = 0;
end

% obj_closest_in_next_path_ID =0;
if(size(objects_in_next_lane_id,1)~=0)
    objects_in_next_lane =  [object_list(objects_in_next_lane_id,2),object_list(objects_in_next_
    obj_closest_in_path_ID_dist = dist2StartCorner_veh+500;
    obj_closest_in_next_path_ID = 0;

    %Find the closest Next Lane object
    for  i = 1:size(objects_in_next_lane,1)
```

```
        p = objects_in_next_lane(i,:);
        [~, dist2StartCorner_obj(i,2)] = point_projection2LNM(p, corner_1, corner_2);
        if (dist2StartCorner_obj(i,2) > dist2StartCorner_veh && ...
                dist2StartCorner_obj(i,2) < obj_closest_in_path_ID_dist)
            obj_closest_in_next_path_ID = objects_in_next_lane_id(i);
            obj_closest_in_path_ID_dist = dist2StartCorner_obj(i,2);
        end
     end
else
    obj_closest_in_next_path_ID = 0;
end
%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.