supplementary file

The core source program of the positioning model:

Software: Matlab

```
% Algorithm 1: Core Source Program of the Positioning Model

% Dynamic demonstration of path planning for automated parking.
function AutomatedParkingDemo()

    % Initialize MATLAB environment
    clc;
    clear;
    close all;

    % Parking area limits
    UpLimit = 8.1; % Upper boundary vertical coordinate
    RightLimit = 15; % Right border of horizontal coordinate

    % Parking information point C
    FrontP = [5.1, 2.6];

    % Parking spots
    O3 = [0.5, 1];

    % Initial Point
    InitePlot = [6, 4];

    % Car specifications
    CarLong = 4.9; % Car Captain
    CarWidth = 1.8; % Car width
    RMin = 4.99; % Minimum turning radius
    PlotVec = 0.01; % Animation speed
    FitModel = 0; % Fitting form, 1-bit higher polynomial, other numbers are Fourier series fitting

    % Plot the limited parking area
    figure(1)
    hold on

    plot([0, RightLimit], [UpLimit, UpLimit], 'LineWidth', 2, 'Color', [0 0 0]);
    plot([0, FrontP(1)], [0, 0], 'LineWidth', 2, 'Color', [0 0 0]);
    plot([FrontP(1), FrontP(1)], [0, FrontP(2)], 'LineWidth', 2, 'Color', [0 0 0]);
    plot([FrontP(1), RightLimit], [FrontP(2), FrontP(2)], 'LineWidth', 2, 'Color', [0 0 0]);

    axis([0 RightLimit 0 UpLimit], 'equal');
```

```
    % Parameters
    x = InitePlot(1);
    y = InitePlot(2);
    CO = [x, y]; % Vehicle rear axle center point coordinates

    % Rest of the code...

end


% Algorithm 2: Car Parking Information and Route Planning

% Car parking information
CarLongine(1,:) = [x,           x,           y - CarWidth/2, y + CarWidth/2];
CarLongine(2,:) = [x,           x + CarLong, y + CarWidth/2, y + CarWidth/2];
CarLongine(3,:) = [x + CarLong, x + CarLong, y + CarWidth/2, y - CarWidth/2];
CarLongine(4,:) = [x + CarLong, x,           y - CarWidth/2, y - CarWidth/2];

% Initialization
figure(1);
for i = 1:4
    plot([CarLongine(i,1), CarLongine(i,2)], [CarLongine(i,3), CarLongine(i,4)], 'Color', [0 0
0]);
end

% Description of the starting point
% plot(x, y, 'ro');

%% Route Planning

% Your existing code for route planning here...

% Collision judgment
DCo1 = sqrt(CarLong^2 + (CarWidth/2 + DMiddle)^2);
ThetaCo1 = atan2((CarWidth/2 + DMiddle), CarLong);
if ThetaCo1 < Alpha
    % Collision judgment logic...
end

DCo2 = sqrt(CarLong^2 + (CarWidth/2 + RMin)^2);
ThetaCo2 = atan2((CarWidth/2 + RMin), CarLong);
if ThetaCo2 + Alpha >= pi/2
    % Collision judgment logic...
```

```
end

ErrX = 0;
ErrY = 0;

%% Discrete data construction

% Arc Information
Cir1R = DMiddle;
Cir1O = [x, YMiddle];
Cir2R = RMin;
Cir2O = [O3(1), O3(2) + RMin];

% Constructing sequences
AlphaArr = linspace(0, Alpha, 50);
AlphaArr = AlphaArr';
Cir1Data = [x - DMiddle.*sin(AlphaArr), YMiddle + DMiddle.*cos(AlphaArr), AlphaArr];
AlphaArr = linspace(Alpha, 0, 50);
AlphaArr = AlphaArr';
Cir2Data = [Cir2O(1) + RMin.*sin(AlphaArr), Cir2O(2) - RMin.*cos(AlphaArr), AlphaArr];
CirData = [Cir1Data; Cir2Data];

%% Graphical interface

for item = 1:length(CirData)
    CO = [CirData(item, 1) + ErrX, CirData(item, 2) + ErrY];
    CarLongine = CarP(CarLong, CarWidth, CO);
    for i = 1:4
        bCarLongine = RotTheta(CarLongine(i,:), CirData(item, 3), CO(1), CO(2));
        plot([bCarLongine(1), bCarLongine(2)], [bCarLongine(3), bCarLongine(4)], 'Color',
[0 0 0]);
        % Description of the current point
        % plot(x, y, 'r.');
    end
    pause(0.02)
end

% End

% Algorithm 3: Automatic Parking Assistance System
clc
close all
clearvars
%% Declaring global variables
```

```
global wl;
global ww;
global D;
global L;
global WSeta;
    wl=7;
    ww=2;
    A=15;
    L=20;
    D=5;
    F=8;
    B=L+D+F;
    %% Rectangular coordinates of the parked vehicle 1
    ParkedCar1x1=0;
    ParkedCar1y1=40;
    ParkedCar1x2=ParkedCar1x1;
    ParkedCar1y2=75;
    ParkedCar1x3=15;
    ParkedCar1y3=ParkedCar1y2;
    ParkedCar1x4=ParkedCar1x3;
    ParkedCar1y4=ParkedCar1y1;
    %% Rectangular coordinates of parked vehicle 2
    ParkedCar2x1=40;
    ParkedCar2y1=40;
    ParkedCar2x2=ParkedCar2x1;
    ParkedCar2y2=75;
    ParkedCar2x3=55;
    ParkedCar2y3=ParkedCar2y2;
    ParkedCar2x4=ParkedCar2x3;
    ParkedCar2y4=ParkedCar2y1;
    %% Rectangular coordinates of parked vehicle 3
    ParkedCar3x1=-40;
    ParkedCar3y1=40;
    ParkedCar3x2=ParkedCar3x1;
    ParkedCar3y2=75;
    ParkedCar3x3=-25;
    ParkedCar3y3=ParkedCar3y2;
    ParkedCar3x4=ParkedCar3x3;
    ParkedCar3y4=ParkedCar3y1;
%%   Adjustment parameters
    WSeta=50;
    WSeta=(WSeta/180)*pi; % Curvature
    xi=ParkedCar1x2-3;
    yi=ParkedCar1y2+25;
```

```
    f1e=40;
   %% Simulation device
    axis([-60 100 -20 100],'equal');
    t=0:0.01:pi;
    seta=2*t;
    Xc=xi-(L/tan(WSeta));
    Yc=yi+D;
    Rbl=sqrt(D^2+(L/tan(WSeta))^2);
    phi=atan(D/(L/tan(WSeta)));
%% Initialization of the scene begins
            i=1;
            x=Xc+Rbl*cos(seta(i)+phi);
            y=Yc-Rbl*sin(seta(i)+phi);
            patch([ParkedCar2x1 ParkedCar2x2 ParkedCar2x3
ParkedCar2x4],[ParkedCar2y1 ParkedCar2y2 ParkedCar2y3 ParkedCar2y4],[1 0 0]);
            patch([ParkedCar1x1 ParkedCar1x2 ParkedCar1x3
ParkedCar1x4],[ParkedCar1y1 ParkedCar1y2 ParkedCar1y3 ParkedCar1y4],[0 0 1]);
            patch([ParkedCar3x1 ParkedCar3x2 ParkedCar3x3
ParkedCar3x4],[ParkedCar3y1 ParkedCar3y2 ParkedCar3y3 ParkedCar3y4],[1 0 1]);
            title("Automatic parking Assistance System");
            text(ParkedCar1x2,ParkedCar1y2+5,"Car 1");
            text(ParkedCar2x2,ParkedCar2y2+5,"Car 2");
            text(ParkedCar3x2,ParkedCar3y2+5,"Car 3");
            text(ParkedCar2x2,ParkedCar2y2+40,"Parking Lot",'Color','red','FontSize',16);
            Turn1(x,y,A,B,seta(i));
            pause(4);
%% Parking simulation scenario
    for i=2:f1e
            x=Xc+Rbl*cos(seta(i)+phi);
            y=Yc-Rbl*sin(seta(i)+phi);
            cla;
            patch([ParkedCar2x1 ParkedCar2x2 ParkedCar2x3
ParkedCar2x4],[ParkedCar2y1 ParkedCar2y2 ParkedCar2y3 ParkedCar2y4],[1 0 0]);
            patch([ParkedCar1x1 ParkedCar1x2 ParkedCar1x3
ParkedCar1x4],[ParkedCar1y1 ParkedCar1y2 ParkedCar1y3 ParkedCar1y4],[0 0 1]);
            patch([ParkedCar3x1 ParkedCar3x2 ParkedCar3x3
ParkedCar3x4],[ParkedCar3y1 ParkedCar3y2 ParkedCar3y3 ParkedCar3y4],[1 0 1]);
            text(ParkedCar1x2,ParkedCar1y2+5,"Car 1");
            text(ParkedCar2x2,ParkedCar2y2+5,"Car 2");
            text(ParkedCar3x2,ParkedCar3y2+5,"Car 3");
           text(ParkedCar2x2,ParkedCar2y2+40,"Parking Lot",'Color','red','FontSize',16);
            Turn1(x,y,A,B,seta(i));
            pause(.05);
    end
```

```
      xi=x+A*cos(seta(f1e));
      yi=y-A*sin(seta(f1e));
    Xc=xi+Rbl*cos(seta(f1e));
    Yc=yi-Rbl*sin(seta(f1e));
    pause(1);
    for i=f1e:-1:1
            x=Xc-Rbl*cos(seta(i));
            y=Yc+Rbl*sin(seta(i));
            cla;
            patch([ParkedCar2x1 ParkedCar2x2 ParkedCar2x3
ParkedCar2x4],[ParkedCar2y1 ParkedCar2y2 ParkedCar2y3 ParkedCar2y4],[1 0 0]);
            patch([ParkedCar1x1 ParkedCar1x2 ParkedCar1x3
ParkedCar1x4],[ParkedCar1y1 ParkedCar1y2 ParkedCar1y3 ParkedCar1y4],[0 0 1]);
            patch([ParkedCar3x1 ParkedCar3x2 ParkedCar3x3
ParkedCar3x4],[ParkedCar3y1 ParkedCar3y2 ParkedCar3y3 ParkedCar3y4],[1 0 1]);
            text(ParkedCar1x2,ParkedCar1y2+5,"Car 1");
            text(ParkedCar2x2,ParkedCar2y2+5,"Car 2");
            text(ParkedCar3x2,ParkedCar3y2+5,"Car 3");
           text(ParkedCar2x2,ParkedCar2y2+40,"Parking Lot",'Color','red','FontSize',16);
            [xa1, xb1, xb2, xa2, ya1, yb1, yb2, ya2]=Turn2(x,y,A,B,seta(i));
            pause(.05);
    end
 pause(1);
 for i=1:50
 ya1=ya1-.5;
 yb1=yb1-.5;
 ya2=ya2-.5;
 yb2=yb2-.5;
 cla;
 patch([ParkedCar2x1 ParkedCar2x2 ParkedCar2x3 ParkedCar2x4],[ParkedCar2y1
ParkedCar2y2 ParkedCar2y3 ParkedCar2y4],[1 0 0]);
 patch([ParkedCar1x1 ParkedCar1x2 ParkedCar1x3 ParkedCar1x4],[ParkedCar1y1
ParkedCar1y2 ParkedCar1y3 ParkedCar1y4],[0 0 1]);
 patch([ParkedCar3x1 ParkedCar3x2 ParkedCar3x3 ParkedCar3x4],[ParkedCar3y1
ParkedCar3y2 ParkedCar3y3 ParkedCar3y4],[1 0 1]);
 patch([xa1 xb1 xb2 xa2],[ya1 yb1 yb2 ya2],[0 1 0]);
 text(ParkedCar1x2,ParkedCar1y2+5,"Car 1");
text(ParkedCar2x2,ParkedCar2y2+5,"Car 2");
text(ParkedCar3x2,ParkedCar3y2+5,"Car 3");
text(ParkedCar2x2,ParkedCar2y2+40,"Parking Lot",'Color','red','FontSize',16);
 ya11=ya1+(D-wl/2);
 yb11=ya11+wl;
 ya21=ya2+(D-wl/2);
 yb21=ya21+wl;
```

```
 xa11=xa2+(ww/2);
 xb11=xa2+(ww/2);
 xa21=xa2-(ww/2);
 xb21=xa2-(ww/2);
 patch([xa11 xb11 xb21 xa21],[ya11 yb11 yb21 ya21],[0 0 0]);
 xa22=xa1-(ww/2);
 xb22=xa1-(ww/2);
 xa12=xa1+(ww/2);
 xb12=xa1+(ww/2);
 patch([xa12 xb12 xb22 xa22],[ya11 yb11 yb21 ya21],[0 0 0]);
 ya13=ya11+L;
 yb13=yb11+L;
 patch([xa11 xb11 xb21 xa21],[ya13 yb13 yb13 ya13],[0 0 0]);
 patch([xa12 xb12 xb22 xa22],[ya13 yb13 yb13 ya13],[0 0 0]);
 pause(.02);
 end
 disp("This car is parked between the purple and blue cars because this is the shortest distance from the car's initial position to the available parking space")

% Algorithm 4: MATLAB GUI Initialization
function varargout = jieshu(varargin)
% JIESHU M-file for jieshu.fig
%      JIESHU, by itself, creates a new JIESHU or raises the existing
%      singleton*.
%      H = JIESHU returns the handle to a new JIESHU or the handle to
%      the existing singleton*.
%      JIESHU('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in JIESHU.M with the given input arguments.
%      JIESHU('Property','Value',...) creates a new JIESHU or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before jieshu_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to jieshu_OpeningFcn via varargin.
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help jieshu
% Last Modified by GUIDE v2.5 07-Dec-2010 22:40:03
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @jieshu_OpeningFcn, ...
                   'gui_OutputFcn',  @jieshu_OutputFcn, ...
```

```
                    'gui_LayoutFcn',    [] , ...
                    'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before jieshu is made visible.
function jieshu_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to jieshu (see VARARGIN)
% Choose default command line output for jieshu
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
global qian
global rate
set(handles.abc,'String',num2str(rate));
set(handles.bbb,'String',num2str(qian));
% UIWAIT makes jieshu wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = jieshu_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% Algorithm 5: CreateFcn for edit2 in the GUI
% --- Executes during object creation, after setting all properties. function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
End

% Algorithm 6: Callbacks for pushbutton1, pushbutton2, and CreateFcn for bbb
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
close all;
openfig('jiemian1.fig')
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
close all
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles       structure with handles and user data (see GUIDATA)
% --- Executes during object creation, after setting all properties.
function bbb_CreateFcn(hObject, eventdata, handles)
% hObject       handle to bbb (see GCBO)
% eventdata     reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called
plot3(a(:,1),a(:,2),a(:,3),'ro-');
% End of Algorithm 6
```