HAN AUTOMOTIVE RESEARCH

HAN_UNIVERSITY OF APPLIED SCIENCES

Automated docking maneuvering of an articulated vehicle in the presence of obstacles

Software Guide

Supervisors: Dr. Joop Pauwelussen
Dr. Karel Kural

Author: Manojpriyadharson Kannan

Date: 01.02.2021

VERSION HISTORY

Version #	Submitted By	Revision Date	Submitted to	Approval Date	Remarks
0.1	Manojpriyadharson Kannan	February 2021	*Joop Pauwelussen (Company Supervisor) and Karel Kural (HAN Supervisor)*		

1. Introduction

1.1 Purpose

As a part of Master thesis work, a bi-directional path planner based on lattice based motion planning technique in combination with A* search algorithm has been developed and intended to be used for the purpose of auto-docking in a distribution center at Oirschot. This document will serve as guide to run the path planner along with a path tracking controller in loop in a MATLAB/SIMULINK environment.

1.2 Prerequisites

The reader is requested to refer the master thesis document (Kannan, 2020) titled "Major project report-638628.pdf" for the details about a lattice based motion planning technique and working function of A* search algorithm.

A bi-directional path planner is responsible to generate a reference path between an initial and final configuration of Single Articulated Vehicle (SAV) assuming constant forward and reversing speed. Tractor semi-trailer (TST) combination is considered in this work and vehicle configuration $q = (x_1, y_1, \theta_1, \gamma)$ is described by vehicle states, namely position of the semi-trailer axle (x_1, y_1), orientation angle of the semi-trailer θ_1 and vehicle articulation angle γ i.e. difference between orientation angle of tractor θ_0 and semi-trailer θ_1 respectively. The related files of path planner can be found in HAN-AR GitHub repository i.e. *HAN-AR/CATALYST-obstacle-avoidance*. The generated reference paths is given as input to a path tracking controller to validate whether the vehicle can follow the reference path. An already available path tracking controller at HAN-AR is used in this work. The related files of path tracking controller can be found in HAN-AR GitHub repository i.e. *HAN-AR/INTRALOG-vehicle-models-controller*.

Since the path planner is developed in MATLAB/SIMULINK (version 2020b), the reader is required to have a same version of MATLAB. The following toolbox must be installed.

- Simulink Compiler (version 1.1)
- Simulink Coder (version 9.4)
- Simulink (version 10.2)
- Simulink Check (version 5.0)
- Simulink Code Inspector (version 3.7)
- Mapping Toolbox (version 5.0)
- MATLAB (version 9.9)

2. MATLAB/SIMULINK files

It is assumed that the reader has read the master thesis document (Kannan, 2020) titled "Major project report-638628.pdf". The developed path planner rely on motion primitives, a set of precomputed paths from one vehicle state to another state where state is defined by orientation angle of the semi-trailer θ_1 and vehicle articulation angle γ respectively. To generate such paths, optimal control problem is defined. To solve it, ACADO toolkit is employed in this work. As a first step, download the ACADO toolkit based on your Operating System. Kindly refer (Github, n.d.), where a detailed explanation on installing it on MATLAB and its working details are given.

After installing it on MATLAB, run the script named "Motionprimitives_acado.m". It can be found in "HAN-AR/CATALYST-obstacle-avoidance/MATLAB FILES-PATHPLANNER". The constraints and parameters mentioned in the script are related to this work (Kannan, 2020). But the reader is welcomed to implement his own logic in generating the motion primitives. And, also constrains

value can be altered as per user-defined cases. After running this script, a set of motion primitives respecting the mentioned constraints will be generated. Note that, the motion primitives with initial orientation angle of $\theta_{1i} = 0°$ are generated first. Then, for other discretized initial semi-trailer orientation angles a rotational matrix is used and is mentioned in the below equation 2.1.

$$\begin{bmatrix} x_{1\Delta\theta} \\ y_{1\Delta\theta} \end{bmatrix} = \begin{bmatrix} \cos(-(\theta_{1i} + \Delta\theta)) & -\sin(-(\theta_{1i} + \Delta\theta)) \\ \sin(-(\theta_{1i} + \Delta\theta)) & \cos(-(\theta_{1i} + \Delta\theta)) \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \tag{2.1}$$

In this work, both the path planner and path tracking controller are available at SIMULINK platform itself. First, the working of that will be explained. Under the "HAN-AR/CATALYST-obstacle-avoidance/SIMULINK-PATHPLANNER", two matlab files named "**Path_planning_run.m**" and "**checkpathplanning.m**" can be found where former is for static obstacle avoidance alone and latter for moving obstacle avoidance in addition with static obstacle. To avoid confusion, two separate files are being created.

First, "**Path_planning_run.m**" contains the script for running the Simulink. In the file, initial and final configuration along with controller parameters like steer sensitivity and look ahead distance are provided. In addition, the main-related matlab files will be listed below.

1. Add_MP_toworkspace.m – used for loading the motion primitives in the workspace
2. DPDscenario.m – denotes the static map generation
3. closed_checkCopy.m - matlab file for closed list generation
4. g_costCopy.m – matlab file for calculating the length of path (in this work, it is pre-computed during motion primitives generation itself).
5. h_costnewslx.m – matlab file for calculating heuristic cost i.e. estimated cost to travel from n node to final node
6. loop_checkfinalslx.m – matlab file for checking whether the vehicle configuration reached near the given final configuration
7. open_checkCopy.m – matlab file for open list generation
8. staticobs_checkCopy.m – matlab file for static collision check module

Now, inside the SIMULINK environment there are six subsystems which will be listed below.

1. Path planner subsystem – main matlab function which contains A* search algorithm used to find an optimal path between given initial and final configuration. This subsystem takes input from MATLAB script and performs its job with the help of matlab files listed above. Once the path is generated, it is send to next subsystem "Path extraction" where post processing work before moving on to controller part is done.
2. Path extraction subsystem – where post processing work like interpolation is done to be compatible with the controller. The output from this subsystem are a.) path_input b.) checkgoal_x c.) checkgoal_y d.) v_0. It will be sent to controller subsystem and stop simulation matlab function As mentioned, Controller working can be found in *HAN-AR/INTRALOG-vehicle-models-controller*.
3. Stop simulation matlab function – used to check whether the vehicle configuration reached the final configuration or not.

To check whether the uploaded file is working or not. The reader is requested to run the script but keep in mind the add the entire folder "CATALYST-obstacle avoidance" to get rid of error saying as shown in Figure 1. The reader should get the output as shown in Figure 2 finally. In addition, the controller depends on tuning parameters so those parameters have to be tuned to obtain smooth path.

```
Error using run (line 66)
Add_MP_toworkspace not found.

Error in Path_planning_run (line 5)
run Add_MP_toworkspace;
```

Figure 1 Common error

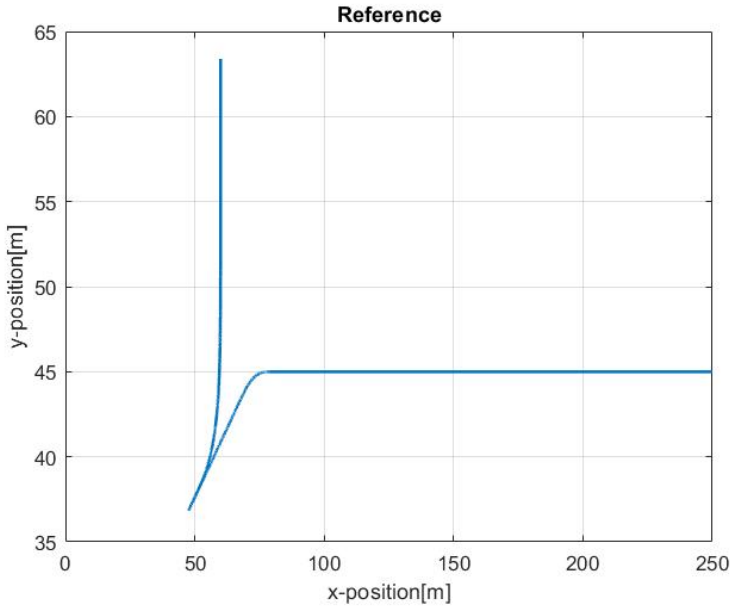

Figure 2 Reference output from static obstacle avoidance Simulink

For the moving obstacle avoidance, run the script "checkpathplanning.m". The above mentioned points are applicable to this module as well. In addition, the moving obstacle starting configuration i.e. position_x, position_y, orientation angle is mentioned manually. In this work, shunt vehicle operating at distribution center is considered.

In SIMULINK environment, additionally two subsystems are included.

1. Detection module – The input for this matlab function are semi-trailer information from vehicle model and reference path from the path planner subsystem. In this work, only one moving obstacle is considered for which ID is mentioned. This ID helps to avoid repetitive detection of same obstacle in an environment. If the obstacle is found, clash points and new start points are calculated and sent to other subsystem "Obstacle check module" where obstacle avoidance maneuver is generated.
2. Obstacle check module – In addition to above mentioned inputs, parallel primitives library is also added which is extensively used for generating obstacle avoidance maneuver in this work. And, the generated path will be given to controller and same procedure mentioned for static obstacle avoidance SIMULINK repeats.

The MATLAB files associated with moving obstacle avoidance SIMULINK can be found in "HAN-AR/CATALYST-obstacle-avoidance/SIMULINK-PATHPLANNER/movingobsfiles" and are as follows:

1. Checkareafirststep.m – This is the first step in collision detection module where relaxed bounding box along the semi-trailer and moving obstacle trajectories are created and checked whether the collision will occur or not. Before that, virtual path of moving obstacle is computed with the help of moving obstacle information defined on script "checkpathplanning.m"
2. Checktimesecondstep.m – This is the second step where pessimistic circles are created along its trajectories and checked whether collision will occur or not. As a result of this test, approximate time at when collision will occur is found.
3. Clashpointfinder.m – Final step where exact collision time and clash points of both articulated vehicle and moving obstacle are computed. Along with that, start point of articulated vehicle is also calculated.

To check whether the uploaded file is working or not. The reader is requested to run the script and arrive at the output as mentioned in figure 3.

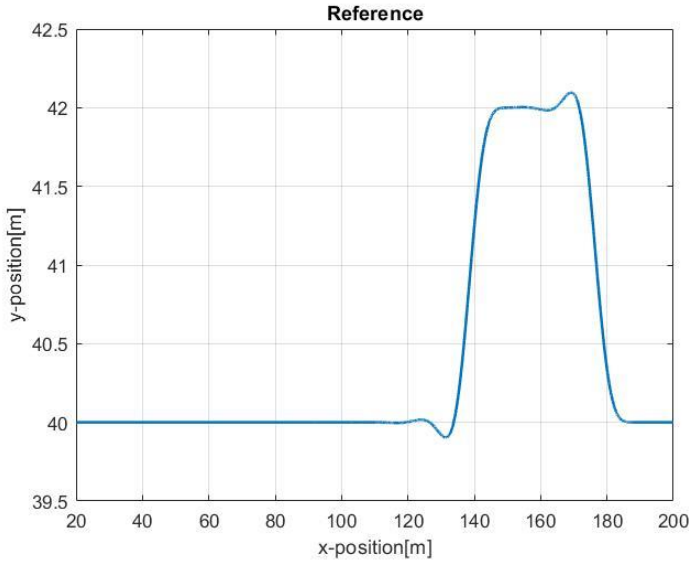

Figure 3 Reference - moving obstacle

PS: In this work, certain penalty techniques are also employed which can be found in the THESIS report. For moving obstacle avoidance, rectilinear moving obstacle alone is considered but for complex environments the files won't work. But the detection module and hierarchical check mentioned can be utilized which is generic for other work as well.

Bibliography

Github. (n.d.). From ACADO from MATLAB: https://acado.github.io/matlab_overview.html

Kannan, M. (2020). *Automated docking maneuvering of an articulated vehicle in the presence of obstacles.* Arnhem: HAN University of applied sciences.