Software Requirements Specification (SRS)

PEDAC

Authors: Samuel Chung, Christopher Cummings, Wan Kim, Tyler Maklebust, Mark Velez

Customer: Mr. David Agnew, Director Advanced Engineering, Mobis NA

Instructor: Dr. Betty H.C. Cheng

1 Introduction

This SRS document provides an overall project description, specific requirements, models of our proposed solutions, and a description of our prototype.

1.1 Purpose

This SRS document is intended for Mr. David Agnew, Director Advanced Engineering of Mobis NA. Its purpose is to outline our understanding of the project specifications and to describe the expected behavior of our product to be in agreement with Mr. David Agnew’s requirements.

1.2 Scope

The software solution proposed here is the Automated Pedestrian Collision Avoidance System, or PEDAC for short. PEDAC is a submodule in fully autonomous driving systems that is responsible for recognizing and responding to pedestrian hazards. PEDAC seeks to maximize safety and efficiency in automated driving systems in which it is included. It will be implemented as embedded software in its respective automotive system. The PEDAC system will determine whether the brake-by-wire system should be activated or if the steady-state velocity can be maintained.

1.3 Definitions, acronyms, and abbreviations

PEDAC: Automated Pedestrian Collision Avoidance System
Steady State Velocity: A velocity to be maintained by the autonomous driving system in cruise control
Brake-By-Wire: The mechanism by which the software can apply the brakes on the vehicle

1.4 Organization

The rest of this document contains detailed descriptions of the product including its functions, behaviors, dependencies, and requirements in figures, diagrams, and prose.
2.0 Overall Description

This section covers the overall product description. This includes the product perspective, product functions, user characteristics, constraints, assumptions and dependencies, and the agreed requirements.

2.1 Product Perspective

PEDAC is an embedded system in autonomous vehicles that recognizes and avoids collision with pedestrian hazards. PEDAC manages messages to the brake-by-wire system and the cruise control system. PEDAC’s relationships to other systems is depicted in Section 4: Modeling Requirements.

The system consists of the following interfaces:

System Interfaces: Brake-By-Wire System, Cruise Control System
Hardware Interfaces: Stereo Cameras
Software Interfaces: Brake-By-Wire System, Cruise Control System
Communication Interfaces: Stereo Cameras I/O, Brake-By-Wire messages, Cruise Control System messages

Our system has the following operational constraints: Only brake or resume steady state speed
2.2 Product Functions

PEDAC will comprise of the following main functions:

1. Monitor path in front of vehicle while driving, looking for pedestrians and identifying potential collisions with them.
2. Determine potential collisions by analyzing collision path between vehicle and pedestrian.
3. Take action to avoid pedestrians by executing velocity reduction commands (automatic braking) which override the current steady state velocity of the vehicle.
4. An automatic brake by wire system in the vehicle to reduce velocity as requested by the system.
5. Vehicle velocity will automatically return to steady state velocity when the hazard no longer exists.

In the case that the PEDAC cannot avoid a collision with the pedestrian, it will apply the brakes maximally to mitigate the impact force of the collision.

2.3 User Characteristics

As a fully autonomous system, we have no external users. The system is self-contained and manages itself.

2.4 Constraints

The PEDAC system is constrained by vehicle acceleration and deceleration limits. Acceleration of the vehicle is limited to 0.25g and maximum deceleration from braking is 0.7g. Along with change in speed, maximum speed is limited to 50kph. The system is also constrained by the output rate of the camera sensors. Information can only be retrieved from the sensors every 100 milliseconds, which limits how quickly the PEDAC system can react to changing conditions.

2.5 Assumptions and Dependencies

One assumption about the PEDAC system that must hold true for it to be functional is that the maximum deceleration for the vehicle is accurate under all conditions. If the vehicle cannot slow down at this rate, the core requirement of the system, not hitting a pedestrian, might not hold true. Another assumption is that the pedestrians can never move at a speed greater than the speed defined, 6kph. The system algorithm will be based on the assumption that pedestrians cannot exceed this speed.

In order for the PEDAC system to function properly, other systems which it depends on must be functional as well. The camera system, along with pedestrian identification, is one of these dependencies. Pedestrians must be accurately identified in order for the system to respond.
accompanying. Another critical component is the braking system. This is the only available means of avoiding a collision and must be functional.

2.6 Apportioning of Requirements

There are some requirements that fall outside the scope of this project. The implementation of pedestrian identification is to be handled by the camera sensors, but this would likely be part of the PEDAC system in a production solution. The effect of lighting or any other conditions on pedestrian detection will not be taken into account for this project, either. Factoring in the effect of weather on braking efficiency is also outside the scope of this project, deceleration will be assumed to be effective regardless of any conditions.

3 Specific Requirements

1. There should be zero collisions with pedestrians resulting from the test cases given:
   a. Correctly calculate current path of vehicle and all pedestrians
   b. Check all identified pedestrians for possible collisions
   c. Activate braking whenever a possible collision is detected and slow down as much as is needed to avoid collision
   d. Avoid collisions with pedestrians regardless of any action they may take by determining the worst case scenario of what they may do
2. Added time to trip caused by avoidance should be minimized for all non-collision scenarios
   a. Only react to pedestrians as much as is necessary to avoid collision, do not slow down more than needed
   b. Return to the steady state speed as soon as all collisions have been avoided
3. Under fail safe mode, system needs to fulfill all requirements above with an increased vehicle stopping time
4 Modeling Requirements

Use Cases:

Data Dictionaries:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedestrian Detection Sensor</td>
<td>PDS class that will relay the Pedestrians information</td>
</tr>
<tr>
<td>Attributes</td>
<td></td>
</tr>
<tr>
<td>PedestrianLocation:float</td>
<td>The location of the pedestrian relative to the car (+/- .5m)</td>
</tr>
<tr>
<td>PedestrianVelocity:float</td>
<td>The velocity of the pedestrian (+/- .2m/s)</td>
</tr>
<tr>
<td>PedestrianDirection:float</td>
<td>The direction of the pedestrian (+/- 5 deg)</td>
</tr>
<tr>
<td>CycleParam::vector</td>
<td>Vector that contains the above attributes to be transmitted to the Safety Controller class</td>
</tr>
<tr>
<td>Operations</td>
<td></td>
</tr>
<tr>
<td>Cycle(): vector&lt;float&gt;</td>
<td>Every 100ms, transmits the vector of information to the Safety Controller</td>
</tr>
</tbody>
</table>
The Pedestrian Detection Sensor gathers information about the pedestrian relative to the car and then relays that information to the safety controller.

### UML Extensions

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety Controller</td>
<td>The controller of the system that decides when to adjust the velocity of the car based on information that it receives</td>
</tr>
</tbody>
</table>

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VehicleSpeed:float</td>
<td>The desired constant velocity of the car (13.9m/s)</td>
</tr>
<tr>
<td>VehicleWidth:float</td>
<td>The width of the vehicle hence the collision zone (2m)</td>
</tr>
<tr>
<td>VehicleAcceleration:float</td>
<td>Acceleration to the steady state after auto brake (.25g)</td>
</tr>
</tbody>
</table>

**Operations**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA_Algorithm(VehicleSpeed:float, CycleParam:vector): bool</td>
<td>Operation will take the speed of the vehicle and parameters of the pedestrian and use this information in the algorithm to decide whether or not to brake. Returns Boolean value based on algorithm.</td>
</tr>
</tbody>
</table>

**Relationships**

| Relationships | Recieves information from Pedestrian Detection Sensors and issues brake signals to the class Vehicle Brake By Wire System. |

### UML Extensions

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle Brake By Wire System</td>
<td>Receives a signal from the signal controller when the brake actuator must be issued</td>
</tr>
</tbody>
</table>

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxDecel:float</td>
<td>Maximum deceleration rate (0.7g)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ReleaseTime: float</td>
<td>Release time (100ms)</td>
</tr>
<tr>
<td>DecelResponseTime: float</td>
<td>The response time needed to reach the requested deceleration (200ms)</td>
</tr>
<tr>
<td>DecelAccur: int</td>
<td>Accuracy of deceleration (+/- 2%)</td>
</tr>
</tbody>
</table>

**Operations**

- VehicleReq: void
  - Issues the request to brake the car

**Relationships**

- Receives signals from the vehicle to brake and then proceeds to brake using the deceleration settings

**UML Extensions**

- Class Diagram:
  - Pedestrian Detection Sensor
    - PedestrianLocation: float
    - PedestrianVelocity: float
    - PedestrianDirection: float
    - CycleParam: vector
  - Safety Controller
    - VehicleSpeed: float
    - VehicleVelocity: float
    - VehicleDeceleration: float
    - PCA_Algorithm(VehicleSpeed: float, CycleParam: vector)
  - Vehicle Brake by Wire System
    - MaxDecel: float
    - ReleaseTime: float
    - DecelResponseTime: float
    - DecelAccur: int
    - VehicleReq: void

- State Diagram:
  - Steady State
    - Threat Detected
      - Threat isn't a pedestrian or isn't in path of vehicle
      - Threat is a pedestrian and is in path of vehicle
  - Increase Speed
    - Steady state speed reached
      - Track Pedestrian
        - No threat of far away threat
        - Threat is no longer too close
        - Threat is too close
  - Breaking
    - Threat is too close
5 Prototype

Our prototype will allow the user to set the starting position and velocity of a car and pedestrian and run a simulation with our algorithm to see if the car can avoid the collision.

5.1 How to Run Prototype

To run the prototype a web browser that has HTML5 and Javascript enabled is required. The prototype is available at: http://www.cse.msu.edu/~cse435/Projects/F2016/Groups/PEDAC1/web/. The user will input their desired starting parameters and then press the “Start!” button. The simulation will run with realistic physics and is scaled properly. The simulation will run until the car and pedestrian collide (failure) or the car reaches the end of the simulation area (success). Currently the simulation area is scaled to 100 meters long by 20 meters high.
5.2 Sample Scenarios

Scenario 1:

This screenshot is at the very beginning of the simulation. The small dot in the image is that pedestrian and the line at the left is that front of the car.

After ~5.0 seconds of running the car collides with the pedestrian, giving an alert that the simulation has failed.
Scenario 2:

The beginning of scenario two is the same as scenario one except the pedestrian has a starting velocity of 4kph.

Neither the car nor the pedestrian are on screen at the end of the second scenario as they did not collide and therefore continued until the car reached the end of the screen. The alert lets the user know that the simulation was successful.

6 References

Project Description:
http://www.cse.msu.edu/~cse435/Projects/F2016/ProjectDescriptions/Mobis-Ped-Avoid-2016-Agnew.pdf

Website: http://www.cse.msu.edu/~cse435/Projects/F2016/Groups/PEDAC1/web/