Software Design Specification

for

Autonomous Vehicle System: IntelliDrive

Prepared by Joshua Concepcion, Logan Scully, Xiangming Xie

27 October 2023

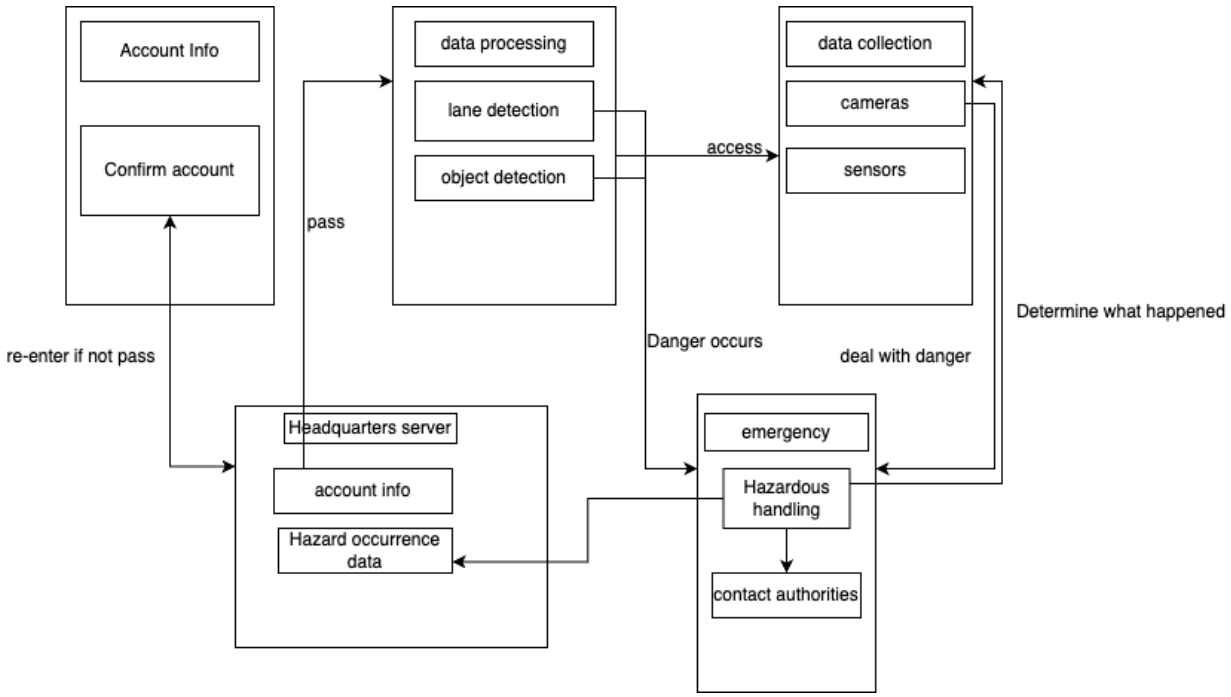
1. System Description

IntelliDrive is a software system designed to keep the driver, passengers, and others on the road safe. Our system will utilize lane and object detection, road stability, and real-time data gathering through the use of a variety of cameras, sensors and other similar components. In the event of an emergency, the system will send signals via satellite to the nearest rescue center. This feature also detects obstacles in the car's path and sends that information to the vehicle's computer to adjust road stability.

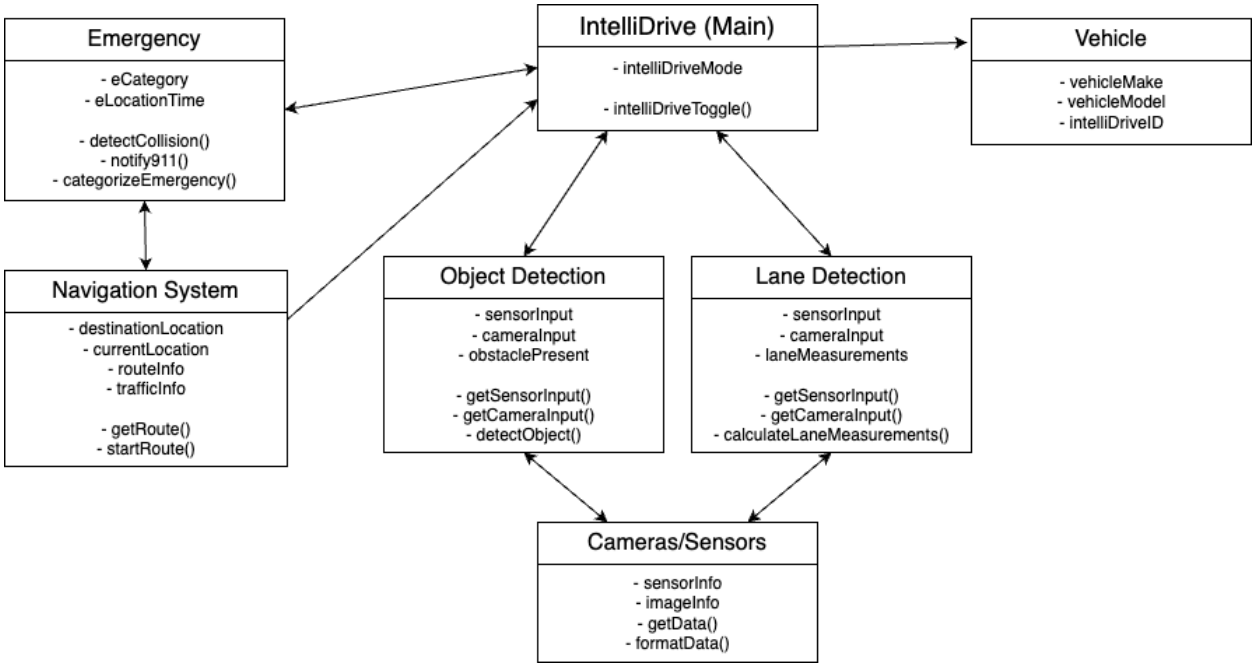
However, there are limits to our system that we do intend to expand upon in the future. For the time being, our AVS does not provide complete self-driving functionality, and users will be expected to handle a variety of aspects when operating the vehicle; the user is responsible for reacting to traffic signs, making turns, merging/exiting freeways, as well as accelerating or braking.

2. Software Architecture Overview

2.1 Architectural Diagram of all Major Components



2.2 UML Class Diagram



2.3 Description of Classes

IntelliDrive (Main)

- Contains method toggling of autonomous driving mode, calls classes such as object detection, lane detection, camera/sensors, and navigation system

Navigation System

- Contains methods for retrieving current location, determining efficient route

Vehicle

- Contains information regarding the vehicle's make, model, and registered IntelliDrive account

Lane detection

- Calls cameras and sensors
- Contains methods for processing information from the cameras and sensors, as well as methods for determining shape of lane

Object detection

- Calls cameras and sensors
- Calls Satellite class for information on set route

Camera/Sensors

- Provides information used for lane and object detection
- Contains methods for reading/processing information about scanned areas or images

Emergency

- Calls Navigation System to update route info and notify of accidents/obstacles
- Contains methods for determining accidents, system failures, and notifying authorities

2.4 Description of SWA diagram

- The system detects whether Account Info matches when booting Auto pilot. If the account does not match, you need to re-enter the account.

- When IntelliDrive Mode starts, Lane Detection and Object Detection will automatically start and run passively. Automatically detect whether Account Info matches when booting Auto pilot. Object Detection is responsible for using cameras/sensors to detect other vehicles/objects/pedestrians/and anything else on the road.

- Cameras and sensors are responsible for capturing information and sending it to Lane Detection and Object Detection

- If Objection Detection and Lane Detection detect danger, emergency will be automatically enabled. When emergency is used, cameras and sensors are automatically called to detect what happened and call the appropriate authorities.

- Also when a vehicle is in danger, the server will automatically receive the information.

2.5 Description of Attributes

IntelliDrive (Main)

- intellidriveMode - a boolean attribute that would determine whether or not the autonomous driving mode is on

Navigation System

- destinationLocation - the address of desired destination

- currentLocation - the address/coordinates that indicate the vehicle's current location
- routeInfo - the route that the vehicle will take to reach the destinationLocation
- trafficInfo - provides information on current/expected traffic status

Vehicle

- vehicleMake and vehicleModel - provides basic information regarding the make and model of the vehicle
- intelliDriveID - the registered ID for the IntelliDrive user's vehicle

Lane Detection

- sensorInput - receives information from the sensors
- cameraInput - receives information from the camera
- laneMeasurements - information regarding size and shape of the lane

Object Detection

- sensorInput - receives information from the sensors
- cameraInput - receives information from the camera
- obstaclePresent - boolean that indicates whether there is an object in the road

Camera/Sensors

- sensorInfo - information regarding what the sensors have detected
- imageInfo - information regarding what the cameras have detected

Emergency

- eCategory - information regarding type of emergency (i.e. accident, system failure, etc)
- eLocationTime - information regarding the location of the accident/failure, as well as timestamp

2.6 Description of Operations

IntelliDrive (Main)

- intelliDriveToggle() - toggles on and off the autonomous driving mode

Navigation System

- getRoute() - determines route for vehicle to take
- startRoute() - begins navigation on screen
- updateRoute() - updates the route based on traffic and obstacle information

Lane Detection

- getSensorInput() - retrieves and processes input from sensor
- getCameraInput() - retrieves and processes input from cameras

- calculateLaneMeasurements() - uses input from sensor and camera to determine size and shape of lane
- notifyNoLaneMarkings() - based on information from sensors and cameras, if no lane markings are present, notifies the driver

Object Detection

- getSensorInput() - retrieves and processes input from sensor
- getCameraInput() - retrieves and processes input from cameras
- detectObject() - uses input from sensors and camera to determine if there is an obstacle on the path, and notifies the system of obstacle if present

Camera/Sensors

- getData() - enables scanning and imaging of sensors and cameras
- formatData() - formats the data for processing by lane and object detection systems

Emergency

- detectCollision() - calls cameras and sensors to detect collision to car, or collision nearby on the road
- notify911() - calls/notifies authorities in the area to collision or accident
- categorizeEmergency() - uses information from cameras and sensors to determine the type of accident

3. Development Plan and Timeline

3.1 Partitioning of tasks

- **Develop Software (9-12 months)**
    - IntelliDrive (Main)
        - intelliDriveMode
        - intelliDriveToggle
    - Navigation System
        - destinationLocation
        - currentLocation
        - routeInfo
        - trafficInfo
        - getRoute()
        - startRoute()
        - updateRoute()
    - Vehicle
        - vehicleMake
        - vehicleModel
        - intelliDriveID
    - Lane Detection
        - sensorInput
        - cameraInput
        - laneMeasurements
        - getSensorInput()
        - getCameraInput()
        - calculateLaneMeasurements()
        - notifyNoLaneMarkings()
    - Object Detection
        - sensorInput
        - cameraInput
        - obstaclePresent
        - getSensorInput()
        - getCameraInput()
        - detectObject()
    - Cameras/Sensors
        - sensorInfo
        - imageInfo

        - getData()
        - formatData()
    - Emergency
        - eCategory
        - eLocationTime
        - detectCollision()
        - notify911()
        - categorizeEmergency()
- **Debugging (6-9 months)**
    - Crosscheck colleagues code for errors
- **Implementation (4-6 months)**
    - Install hardware into select vehicles to run software
    - Install software into select vehicles for testing
- **Testing (2-3 years)**

3.2 Team member responsibilities

- **Development**
    - Joshua - Main, Emergency
    - Tony - Navigation System, Vehicle
    - Logan - Lane Detection, Object Detection, Cameras/Sensors
- **Debugging**
    - Logan review Main Emergency
    - Joshua reviews Navigation System, Vehicle
    - Tony review Lane Detection, Object Detection, Cameras/Sensors
- **Implementation**
    - Outsource hardware installation
    - Tony is in charge of installing the software
- **Testing**
    - Logan and Joshua will come up with ways to test the software properly
    - Employee beta testers will receive versions of the software to test on the road

4. Verification Test Plan

4.1 Unit Tests

Set 1: Navigation System

Test Case: CalculateRoute

- Verify that the system is able to calculate (getRoute) the accurate and efficient route based on information stored in destinationLocation, currentLocation, trafficInfo, and routeInfo

Test Case: Reroute

- Verify that the system is able to update the route based on changes in trafficInfo and routeInfo, along with detection of objects by the ObjectDetection class

Set 2: Lane Detection

Test Case: DetectLaneMarkings

- Verify that the system is able to effectively and accurately determine the lane measurements and detect the lines (getLaneMeasurements()) using information from sensorInput and cameraInput, and correctly store the data in the laneMeasurements variable

Test Case: NoLaneMarkings

- Verify that the system is able to effectively and accurately determine the absence of lane markings/lines (getLaneMeasurements()) using information from sensorInput and cameraInput, and correctly notify the driver/user

4.2 Integration Tests

Test Set 1

Test Case: Sensor Integration and Perception

Test Objective:

- Evaluate the integration of sensors, perception, and collision avoidance.

Test Scenarios:

- Object Detection and Collision Avoidance

Test Objective:

- Verify the accuracy and reliability of object detection and collision avoidance features.

Test Steps:

1. Place stationary objects at varying distances around the vehicle.
2. Drive the vehicle at different speeds and monitor the real-time detection and tracking of these objects.
3. Verify that the system provides accurate information about object positions and velocities.
4. Ensure that the system triggers collision avoidance maneuvers if any object comes within a defined safety threshold.

Coverage:

- This test assesses the performance of LiDAR, cameras, radar, and the perception system in identifying and reacting to objects.

Test Case: Lane Keeping and Autonomous Lane Changes

Test Objective:

- Assess the integration between perception, control, and actuation systems for lane keeping and lane changes.

Test Steps:

1. Drive the vehicle on a multi-lane test track with traffic.
2. Monitor the vehicle's ability to maintain its lane position.
3. Initiate autonomous lane change requests and verify safe and smooth lane changes.
4. Ensure the system detects and responds to vehicles in adjacent lanes.

Coverage:

- This test evaluates the collaboration between perception, control, and actuation systems in maintaining safe lane changes and lane keeping.

Test Case: Sensor Redundancy and Failover

Test Objective:

- Evaluate the system's ability to handle sensor failures and ensure safety.

Test Steps:

1. Introduce simulated sensor failures (e.g., covering a camera lens).
2. Monitor the system's reaction to sensor failures.
3. Verify that the system seamlessly switches to redundant sensors.

Coverage:

- This test verifies the redundancy and failover mechanisms in place to ensure continued operation in case of sensor failures.

Test Set 2

Test Case: Communication and Control Integration

Test Objective:

- Assess the integration of communication systems, control interfaces, and emergency scenarios handling.

Test Scenarios:

- V2X (Vehicle-to-Everything) Communication

Test Objective:

- Evaluate V2X communication and the vehicle's response to external information.

Test Steps:

1. Simulate V2X communication with other test vehicles and traffic infrastructure.
2. Assess the ability of IntelliDrive to send and receive messages about traffic conditions, roadwork information, and emergency alerts.
3. Verify the vehicle's autonomous response to V2X information.

Coverage:

- This test assesses the V2X communication system's integration and its impact on vehicle behavior.

Test Case: Autonomous Parking and Remote Control

Test Objective:

- Assess the integration of remote control interfaces and autonomous parking features.

Test Steps:

- Provide a remote control interface for an operator to park the vehicle in a challenging scenario.
- Monitor the vehicle's execution of parking commands.
- Test remote control features like door control, engine start/stop, and horn operation.

Coverage:

- This test evaluates the integration of remote control features and autonomous parking mechanisms.

4.3 System Tests

IntelliDrive (Main)

Test: Toggling Autonomous Mode

- Input: intellidriveMode = false
- Action: Toggle intellidriveMode to true
- Expected Output: intellidriveMode is now true

Test: Setting Destination

- Input: destinationLocation = "123 Main St"
- Action: Set a new destinationLocation
- Expected Output: destinationLocation is updated

Test: Getting Traffic Info

- Input: No specific input
- Action: Request trafficInfo
- Expected Output: Traffic information is provided

Vehicle

Test: Checking Vehicle Information

- Input: intelliDriveID
- Action: Retrieve vehicleMake and vehicleModel using intelliDriveID
- Expected Output: Vehicle make and model are returned

Lane Detection

Test: Processing Sensor Data

- Input: sensorInput and cameraInput
- Action: Process sensor and camera data
- Expected Output: laneMeasurements are generated

Object Detection

Test: Detecting Obstacles

- Input: sensorInput and cameraInput
- Action: Check for obstacles
- Expected Output: obstaclePresent is updated

Camera/Sensors

Test: Reviewing Sensor Information

- Input: sensorInfo
- Action: Analyze sensor information
- Expected Output: Relevant information is extracted

Test: Reviewing Camera Information

- Input: imageInfo
- Action: Analyze camera information
- Expected Output: Relevant information is extracted

Emergency

Test: Reporting an Accident

- Input: eCategory = "Accident", eLocationTime
- Action: Report an accident
- Expected Output: Emergency system processes the accident report

Test: Reporting a System Failure

- Input: eCategory = "System Failure", eLocationTime
- Action: Report a system failure
- Expected Output: Emergency system processes the failure report

5. Software Design 2.0

5.1 Data Management Strategy

1. Database Choice:

Hybrid Approach: Utilizing a hybrid approach involving both SQL and NoSQL databases. SQL databases for structured data and NoSQL for unstructured or semi-structured data

Reasoning: SQL databases offer ACID compliance and are ideal for structured data like user profiles, vehicle information, and navigation routes. NoSQL databases are efficient for handling vast amounts of unstructured data from sensors, cameras, and real-time feeds.

2. Number of Databases and Data Split:

Two-Tier Architecture: Employing SQL databases (e.g., PostgreSQL) for structured data and NoSQL databases (e.g., MongoDB) for unstructured data.

Logical Split:

SQL Database: Holds user profiles, vehicle details, navigation routes, and critical structured information.

NoSQL Database: Stores unstructured data from cameras, sensors, real-time feeds, such as images, sensor data, and real-time analytics.

Potential Alternatives:

1. **All-in-One SQL Database:**

   Alternative: Using a single SQL database for all data storage.
   Trade Offs: While simpler to manage, it might lead to performance issues and scalability challenges when dealing with huge volumes of unstructured data.

2. **NoSQL Exclusivity:**

   Alternative: Solely relying on a NoSQL database for all data storage.
   Trade Offs: Could lead to complexity in managing relational data and potentially compromise the consistency and integrity of structured data.

Justifications:

SQL Database:

Ensures data consistency, integrity, and security for structured information such as user profiles, vehicle details, and routes.
ACID compliance provides reliability, critical for sensitive data.

NoSQL Database:

Allows for the scalability and flexibility required to handle unstructured data like images and sensor feeds.
Optimized for handling high-velocity data and unstructured information that doesn't fit neatly into tables.

Conclusion:

The chosen two-tier architecture provides a balance between structured and unstructured data management. It ensures data integrity, security, and efficiency while catering to the diverse and evolving data generated by IntelliDrive. The tradeoff involves the complexity of managing multiple databases but results in optimized storage and processing of varying data types.

5.2 Updated Software Architecture Diagram