Lane-change assistance using on-vehicle sensor data fusion

Team Members: Hagop Arabian, Daniel Gallegos, Roberto Garcia, Gerardo Ibarra, David Neilsen, Patrick Emmanuel Sangalang, Jonathan Santos, Deepanker Seth, Angel Tinajero, Xiao Hang Wang
Faculty Advisor: Dr. Manveen Kaur
MathWorks Liaison: Sumit Tandon
Department of Computer Science
College of Engineering, Computer Science, and Technology
California State University, Los Angeles

ML SENSOR FUSION TEAM

[Background]

- Sensor fusion is the merging of data, from a combination of sensors, which provides a better understanding of the surrounding area.
- Types of sensors may include a camera, LiDAR, and Sonar.
- Compared to classic filtering techniques, previously used, machine learning allows the system to learn from patterns to make predictions based on training data.
- Vehicular technology can integrate sensor fusion to assist in creating safer driving conditions by improving situational awareness.

[Algorithm]

Algorithm YoMo (You Only Merge Once) Algorithm

Initialize variable *video_frame*
Initialize data structure variable *edge* to store modified frame
video_frames gets the individual video frame from the dataset

for *frame* in *video_frames*:
    Apply 'Canny edge detection' algorithm to the *frame*
    Append the resulting frame into *edge*
end for
for frame in *edge*:
    Store third quadrant of image in *third_quad* variable
    Store fourth quadrant of image in fourth_quad variable
    Initialize *num_of_cars* variable
    Use "Cascade classifier" to count the number of cars detected in *third_quad* and add count in *num_of_cars* variable
    Use "Cascade classifier" to count the number of cars detected in *fourth_quad and add it to the count in num_of_cars* variables
end for
for vec in lidar_vectors:
    vec_mag gets the magnitude (Math.sqrt(vec.x**2 + vec.y**2 + vec.z**2) of *vec*
    Append values of *vec_mag* to the *magnitude* data structure
end for

result = Algorithm MLPClassifier is applied with parameters: [edge, num_ of_ cars, magnitude]

[Methods]

- The YoMo (You Only Merge Once) algorithm uses video frames along with LiDAR data as inputs from the KITTI Vision Benchmark Suite.
- "Canny edge detection" is used to identify edges in the frames.
- Each video frame is divided into four quadrants and the algorithm focuses on the third and fourth quadrants.
- Cascade classifier is used to identify the number of cars present in each of the selected quadrants.
- Lidar sensor data is processed to compare the distances to all objects and calculate their proximity to the vehicle.
- The resulting data are fused by the Multi-layer Perceptron Classifier, a machine learning algorithm, to determine if it is safe for the vehicle to change lanes.

[Objectives]

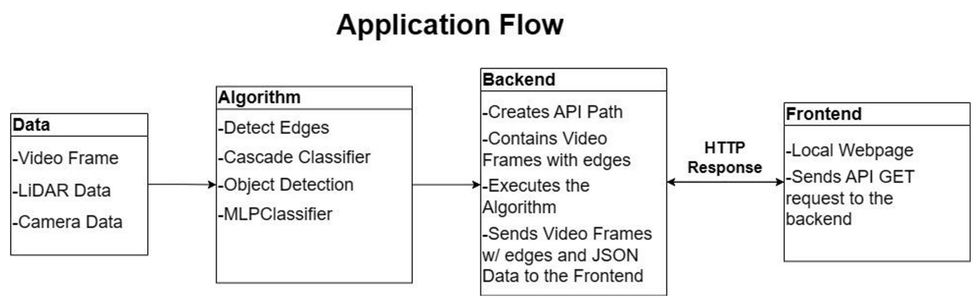
A vehicle operator must be aware of its position and surroundings, which may include path markings, pedestrians, and other vehicles.

The purpose of this project is to develop a sensor fusion algorithm to assist in determining whether it is safe to change lanes. The objective is to create an algorithm which fuses data captured from sensors to guide the future actions of the vehicle and increasing lane change safety.

[Datasets and Tools]

- **These are the sources of Data that were used:**
  - The KITTI Vision Benchmark Suite
    - Frame images from a mounted camera
      - png images
    - Velodyne point cloud from Velodyne HDL-64E lidar.
      - txt text files
  - Matlab
    - Data simulation
- **The Following Algorithms were used to preprocess data:**
  - ***Canny edge detection:*** Edge detection
  - ***Cascade classifier:*** Car detection
  - ***MPL classifier:*** Algorithm fusion
- **Tools and Libraries**
  - ***Flask:*** Back-end web framework
  - ***React:*** Front-end framework
  - ***Sklearn:*** Machine learning algorithm training and testing
  - ***Numpy:*** Data normalization and high-level mathematical functions
  - ***Pandas:*** Data normalization
  - ***Opencv:*** Object detection
  - **Python:** Programing Language

Sample of the conversion from video frame to an edge detection frame.



Application flowchart