Mathworks (Sensor Fusion for Autonomous Systems)

Members

- Hagop Arabian
- Daniel Gallegos
- Roberto Garcia
- Gerardo Ibarra
- David Nielsen
- Patrick Emmanuel Sangalang
- Jonathan Santos
- Deepanker Seth
- Angel Tinajero
- Xiao Hang Wang

Agenda

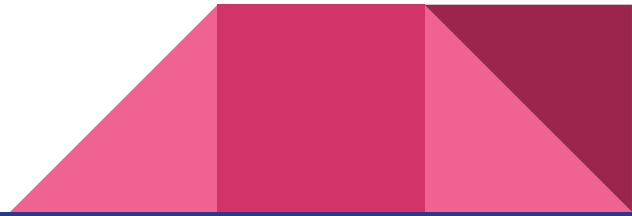
- Topic Overview
- Significance
- Goals/Purpose
- Progress (Research and Challenges)
- Future Goals

Topic Overview

- Autonomous System
- Sensor Fusion
- “Where am I?”
- Machine Learning and Deep Algorithm
- Improving localization performance and have been able to achieve those outcomes.
- Develop a sensor fusion algorithm for vehicle pose estimation using AI techniques.

Sensor Fusion

- Process of merging data from multiple sensors
  - Helps increase accuracy in autonomous systems
  - Three types of sensor configuration:
    - Complementary
    - Competitive (also referred to as Redundant Configuration)
    - Cooperative

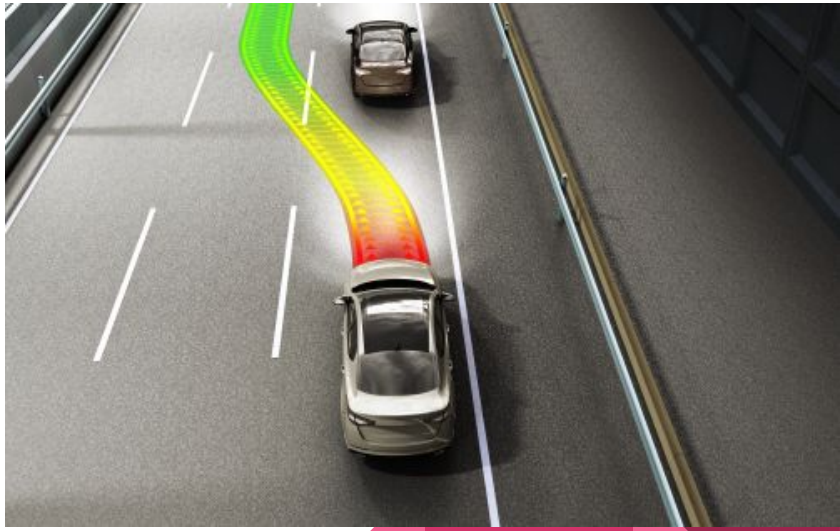


Project Description

- This project focuses on fusing sensors on a ground robot or quadcopter to determine position and orientation.
- We will use different tools to help simulate a vehicle trajectory and many commonly used sensors.
- We could use a variety of sensors such as GPS, LIDAR and visual odometry.
- In this project we will design a fusion algorithm for a group of these sensors to localize a ground robot or quadcopter.

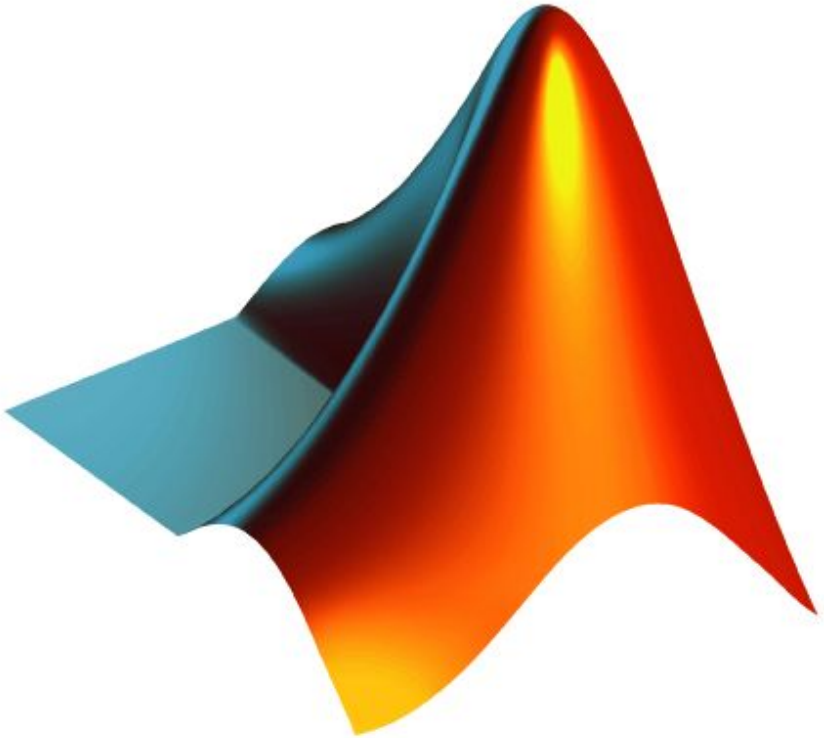
Problem Statement

- How do we get LIDAR and Camera data into our system ?
- How do we track data when it is in motion and motionless ?
- How do we use this data to create a machine learning algorithm that focuses on “lane changing?”



Sponsor Information

- Mathworks
- Specializes in mathematics computing software
- MATLAB
- Simulink

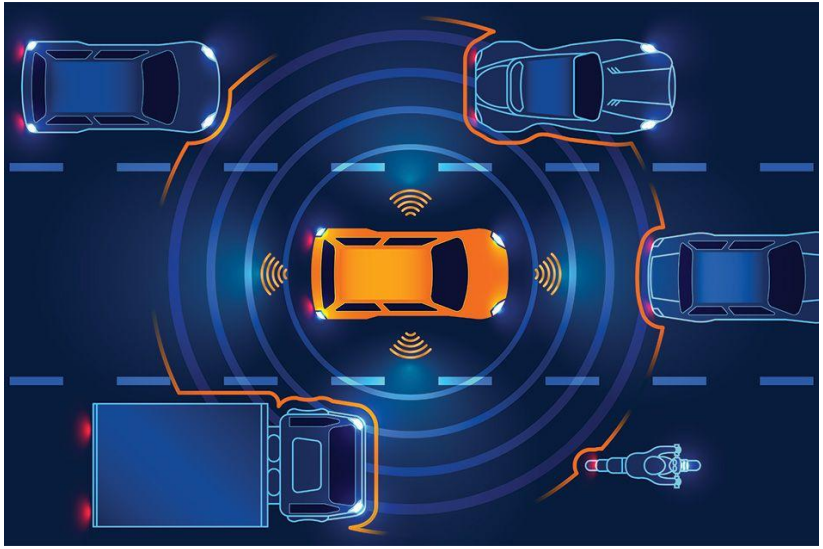


Significance

- 9-percent of all accidents in the US are caused by unsafe lane changes.
- Almost 35,000 people are injured per year.
- Almost 6,000 people die per year.

Significance

- Autonomous Vehicles
- “An extra set of eyes”
- Blind-spot monitoring system that monitors the area next to and behind the car
- If car is present, blinking light will illuminate.
- Not perfect



Goals and Purpose

- Goals
  - Develop a machine-learning algorithm with the focus on “lane changing”
    - Provide a driving simulator that improves lane changing data
- Purpose
  - Improve already given “lane changing” algorithm for autonomous cars.

Progress

- Used Technology
  - MATLAB
  - Python
  - Unreal Engine

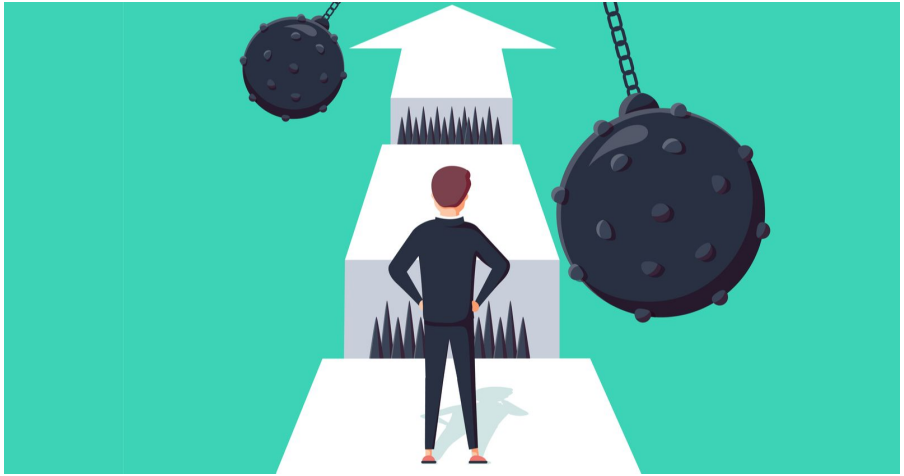
Progress

Research Conducted

- Matlab Machine Learning and Driving Simulation tools
- Machine learning frameworks scikit-learn and Tensorflow
- Random-forest technique to leverage multiple ML algorithms at once

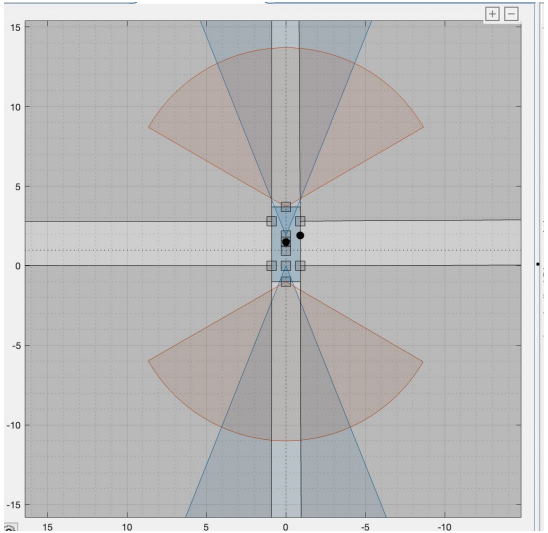
Progress

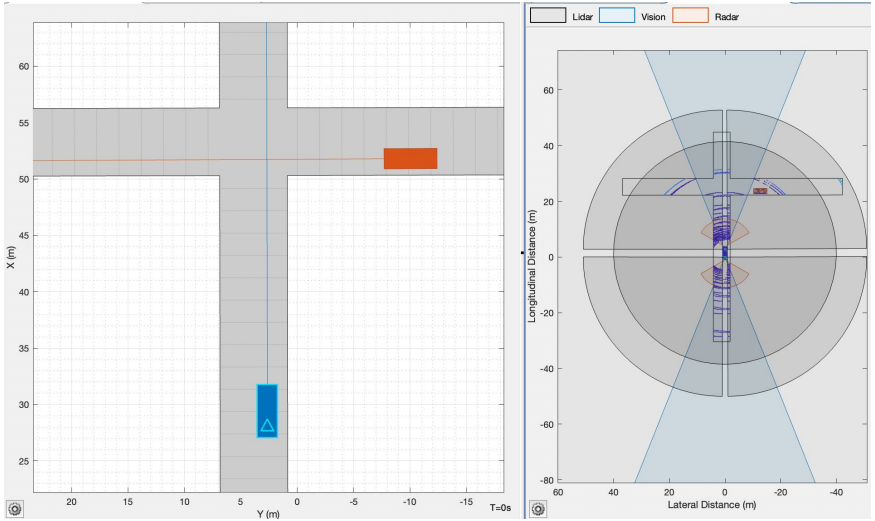
- Challenges
  - Time conflicts between team members
  - Organizing meetings
  - Planning
  - Lack of communication between team members



Progress

We have designed several sensor layouts for our vehicle actor in the Matlab Driving Scenario Designer app.



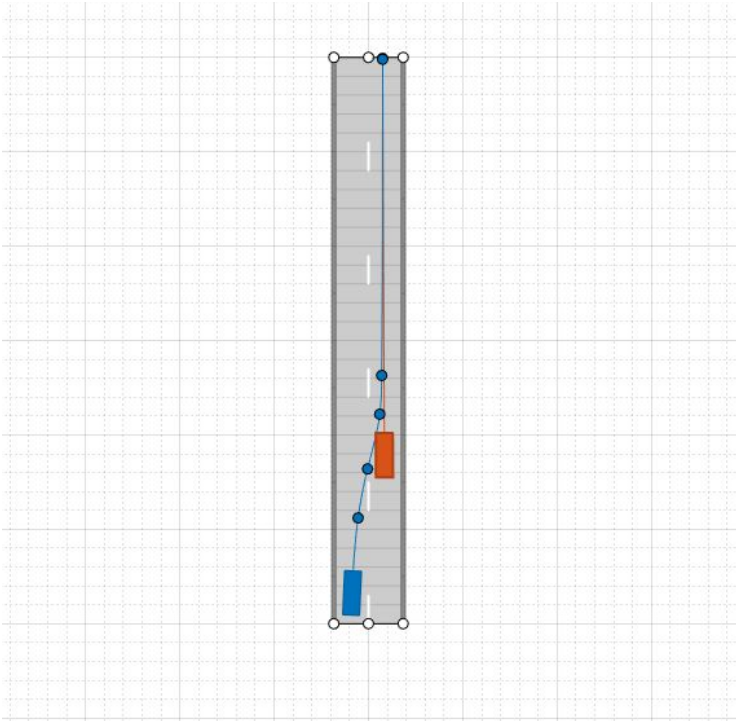


We then place these vehicle agents in various driving scenarios to collect the sensor data.

Progress

- Once we collect the sensor data from the driving scenario simulation our machine learning algorithms can begin training.
- We currently have various models training, including KNN, decision tree and neural network.
- These outputs are then fed into a random forest selector algorithm which takes a vote from each one to arrive at the final output.

One Simple Lane-changing Demo



Exporting Sensor Data from Matlab

```
fileID = fopen('dataset.json','w');

fprintf(fileID, jsonencode(SENSOR_DATA))

fclose(fileID);
```

- Benefits of exporting as JSON file
  - Cross-platform, easier importing to Python
  - Multidimensional structure
  - Internet transportability, possible remote services

Importing and Normalization

```
import json

with open('dataset.json') as json_file:

    data = json.load(json_file)
```

```
dataset = []

# For some reason the thrid and following datas are in a additional array,
# remove them first
for i in range(2, 6):
    data[i] = data[i][0]

import numpy as np
import pandas as pd
import math

tempSet = []

for i in range(0, 6):
    # Conver 3D array into an 1D array
    tempSet.append(np.array(data[i]['PointClouds'][0]['Location']).ravel())

for i in range(0, 6):
    safe = True
    if i >= 4:
        safe = False
    for j in range(0, len(tempSet[i])):
        cordinate = tempSet[i][j]
        # Eliminate NaN
        if cordinate is None:
            cordinate = 0
        dataset.append({"cordinate": cordinate, "safe": safe})

df = pd.DataFrame(dataset)
```

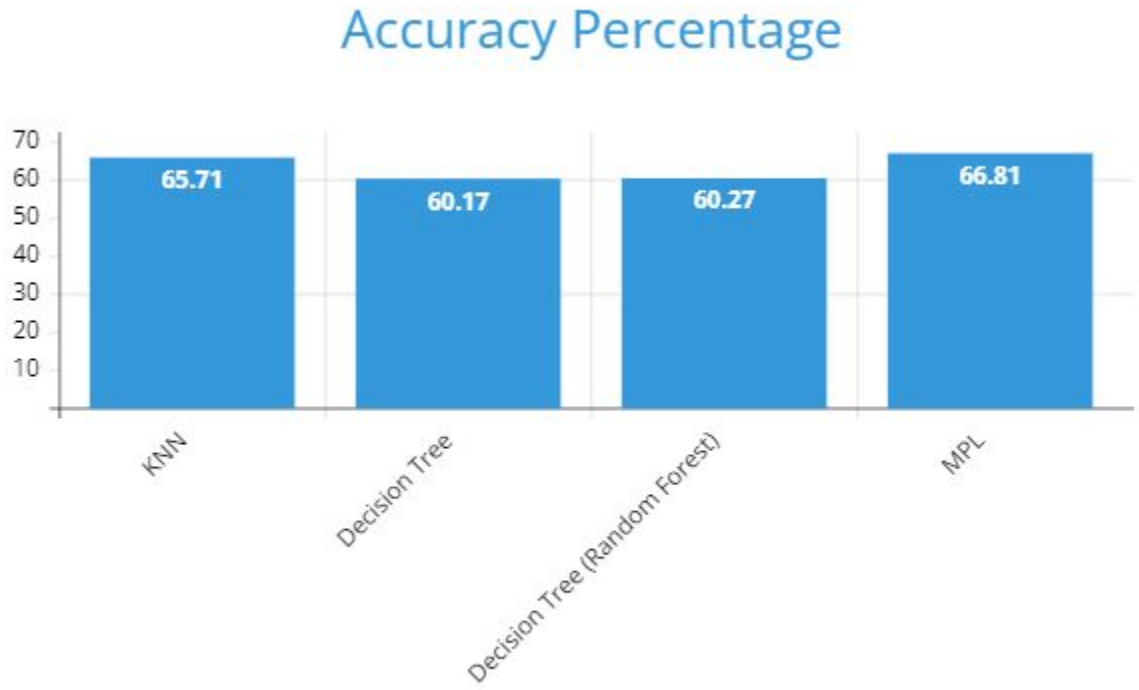
Training and Testing

- Splitting 70% of the dataset as training dataset
- The rest 30% of the dataset as testing dataset to test the accuracy

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

Accuracy


Accuracy Percentage
70
60
50
40
30
20
10
65.71
60.17
60.27
66.81
KNN
Decision Tree
Decision Tree (Random Forest)
MPL

3D Simulation

Birds Eye Plot

Tensorflow 2.0

We are considering using tensorflow 2.0

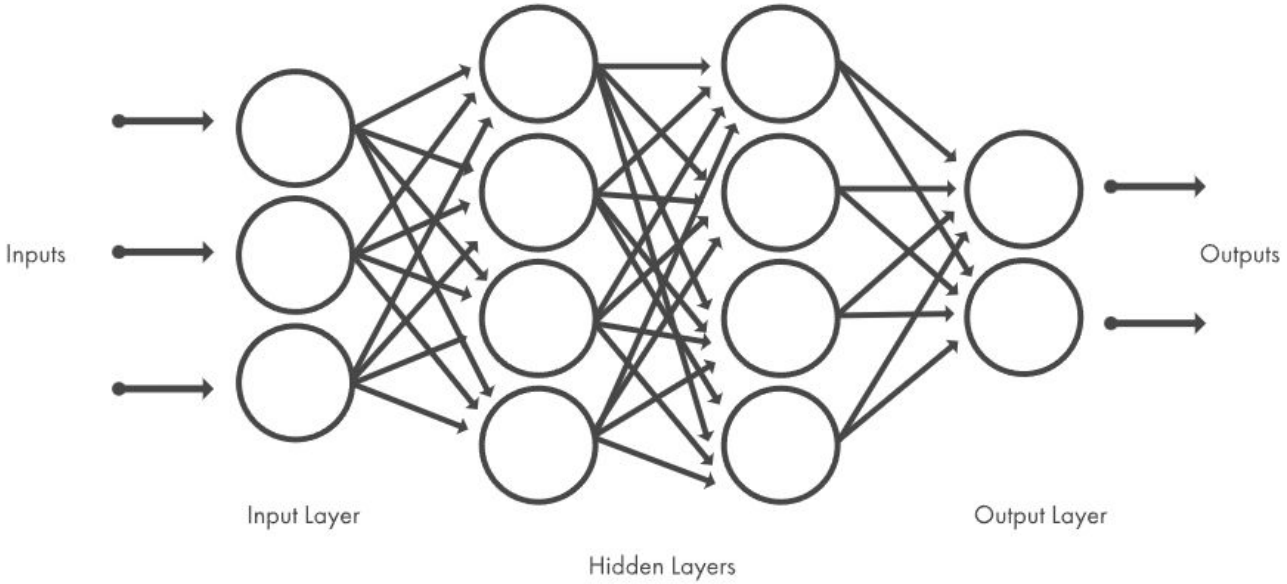
as a way to train, and make our

Machine Learning model portable

Why Tensorflow 2.0

- What is Tensorflow 2.0?
- A core platform and collection of applications that allows for the creation, development and deployment of machine learning models.
- Since we will be mainly using python as our programming language of choice, thus we will be using Keras which is a way for python to interface with Tensorflow 2.0

Our preliminary assessment of neural networks is to use it to aid in processing images to identify obstacles.

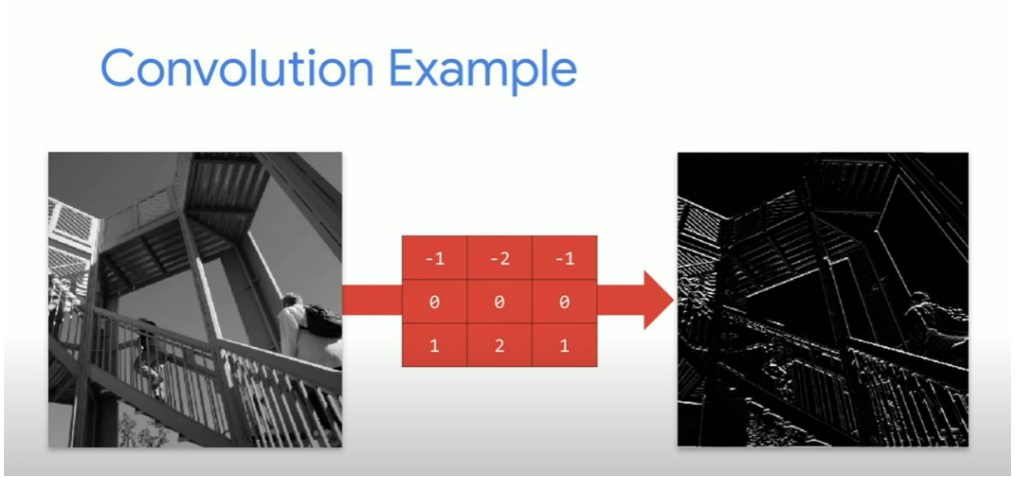


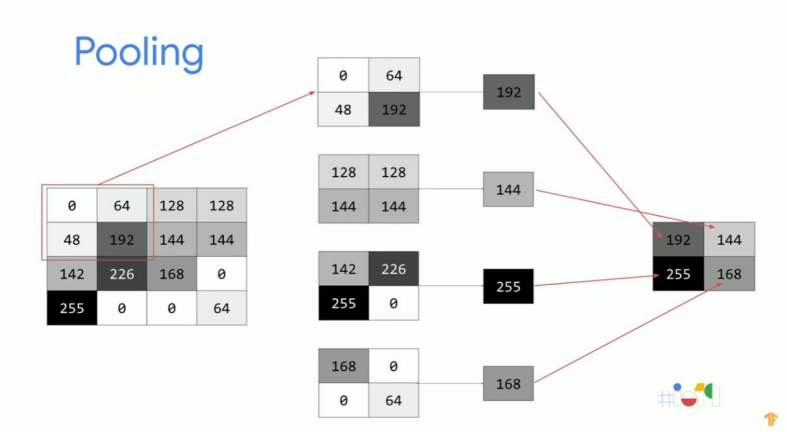
Typical neural network architecture.

By using image manipulation techniques such as:

- “Convolution” - applying filters to images
- “Pooling” - bunching up pixels and selecting the highest value from the bunch

With these techniques we can shrink images while accentuating key features for the neural network to find patterns while reducing noise.





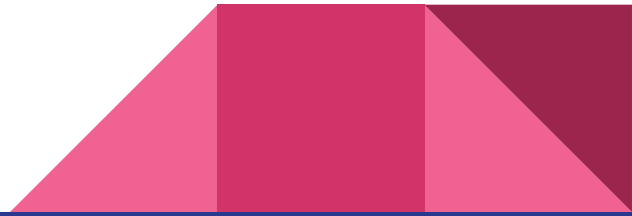
Deployment with TensorFlow 2.0

Making machine learning model more compact:

Shrinks the model using the TF lite converter to make a compact version of the model that can be then loaded directly on to either to an Arduino or raspberry pi or another embedded device.

Future Goals

- Working with the ME/EE team to test our data with their AI Device.
- Work on improving algorithm



Conclusion

- Lane changing accidents occur almost everyday and is 9 percent of car accidents that happen in the US.
- Improving lane changing ML algorithms in autonomous cars can help reduce that percentage.

Q & A

Any Questions ?