Self Driving Car

An autonomous car is a vehicle capable of sensing its environment and operating without human involvement. A human passenger is not required to take control of the vehicle at any time, nor is a human passenger required to be present in the vehicle at all. An autonomous car can go anywhere a traditional car goes and do everything that an experienced human driver does.

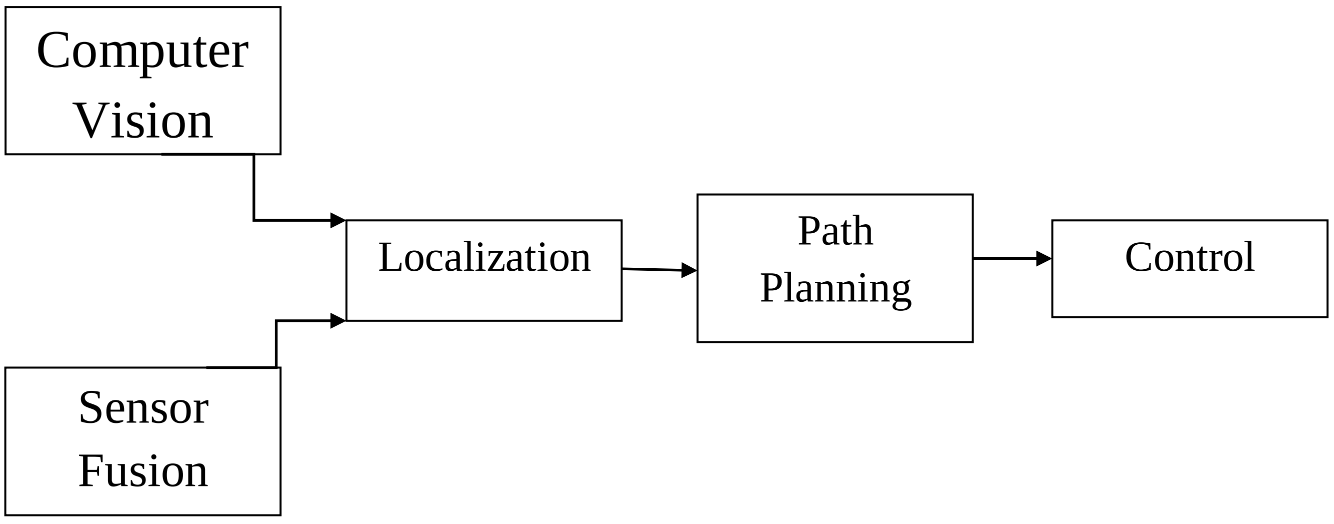
Task

1. Perceiving the Environment
2. Planning how to reach from point A to point B
3. Controlling the vehicle

Operations

1. Lateral control (steering)
2. Longitudinal control (braking,accelerating)
3. Object and Event Detection and Response(automatic emergency response)
4. planning

Block Diagram



Perception

Identification and understanding motion

AI Helps:

- use sensor data to **paint the scene**
- Identify **signs** & **road rules**
- **Learn** continuously to improve **safety and performance**

Goals

- **Road & lane markings**
- **Curbs(off-road,slide)**
- **Traffic light**
- **Road signs**
- **Construction sign & obstruction**
- **Vehicle and Pedestrain detection**

Challenges

- **Robust detection & segmentation**
- **Sensor uncertainity (While raining, fog)**
- **Occlusion & Reflection(While sunlight)**
- **Illuminance,lensflare**
- **Weather**
- **precipitation**

HardWare

Sensors:

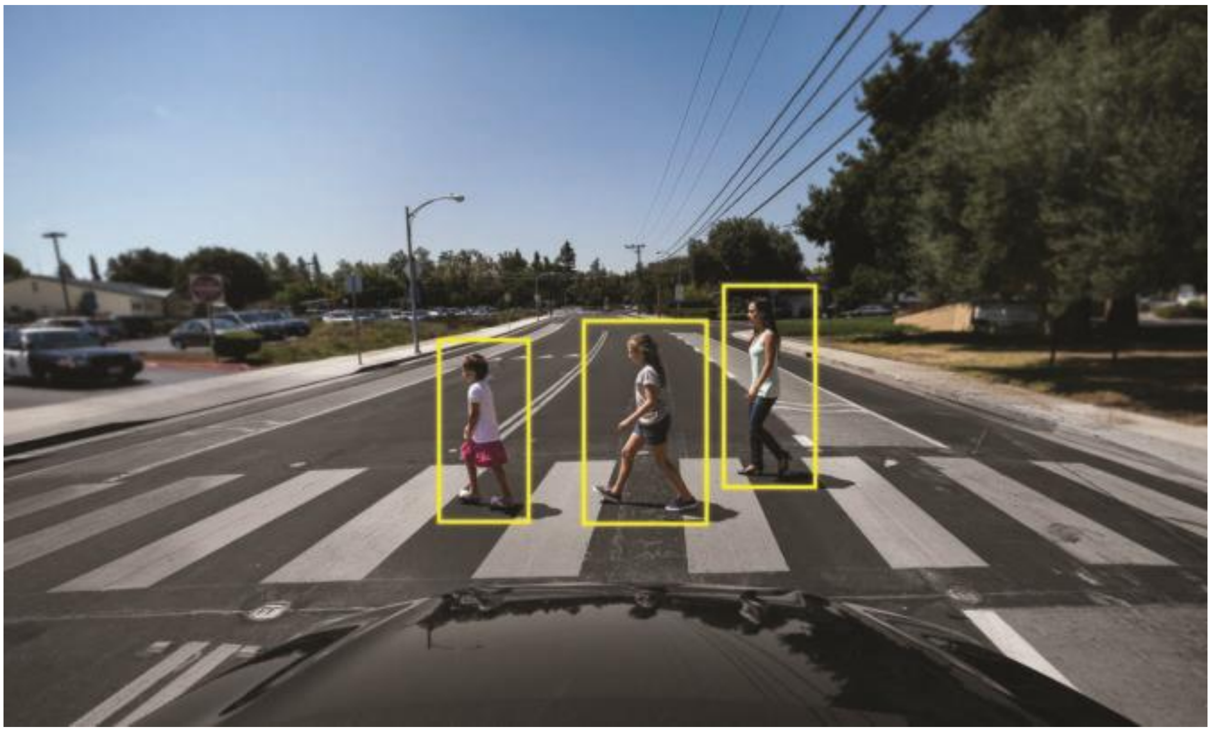
1. LiDAR(3D Scene)
2. RADAR(Works in poor visibility like fog and precipitation)

3. Ultrasonic Sensor
4. Camera
5. Wheel Odometry(Tracks wheel velocities and orientation ,speed accuracy , position drift)
6. GNSS(position, velocity)
7. IMU(angular rotation rate ,acceleration)
8. GPS
9. ACC(Adaptive Cruise Control)(controls longitudinal speed)

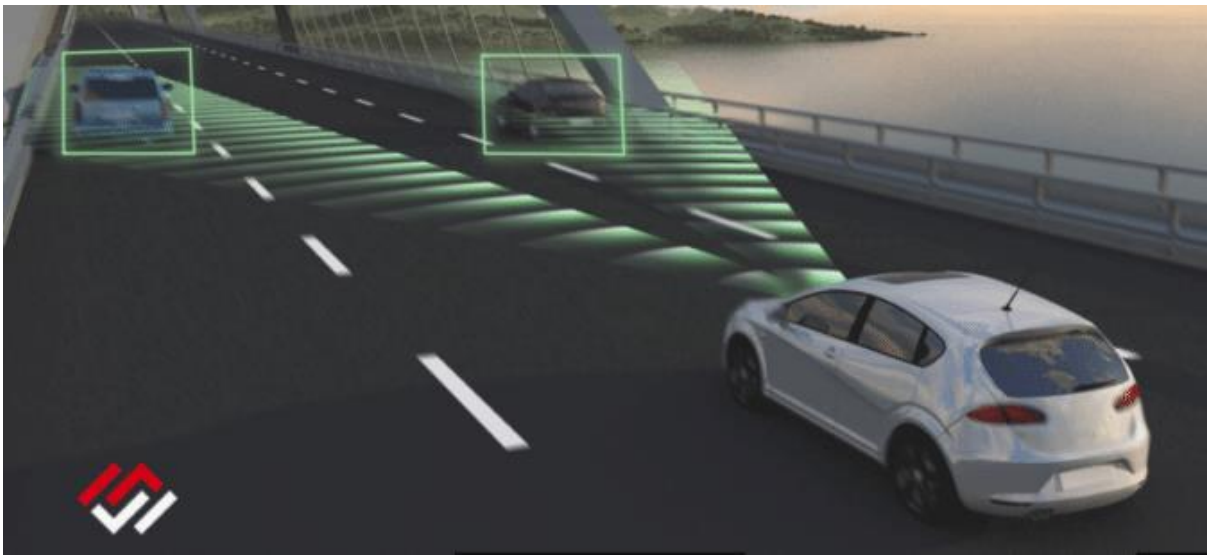
Computing Hardware:

1. Image processing, Object detection, Mapping
    - GPUs -Graphic Processing Unit
    - FPGAs -Field Programmable Gate Array
    - ASICs -Application Specific Integrated Chip
2. Drive PX/AGX
3. Intel
4. Mobileye EyeQ etc...

Camera:



RADAR:



LiDAR:

Softwares Related items:

1. C++
2. Python
3. Linux
4. ROS: Robots Middleware
5. Opencv
6. Matlab
7. Git
8. Vehicle Design
    - CAE**(**Computer Aided Engineering**)**
    - CAD**(**2D and 3D modeling**)**
    - PLM(Product Lifecyce Management)
    - FEA**(**Finite Element Analysis**)(**control strees and strain of body of the software)
    - CATIA, Solidworks, Pro/E, Autodesk 360, Enovia, STAR-CCM+, ANSA, Altair HyperWorks and OptiStruct, ANSYS, MSC NASTRAN, Abaqus, Polarion

Multibody Vehicle Dynamics and Vehicle Model Simulation

1. LS-DYNA**, MSC ADAMS**, CarSim, CarMaker, Dymola, OptimumG, SusProg3D, Oktal SKANeR

Analog and Digital Hardware Development

1. LTSpice
2. Altium

Vehicle Software Development

1. **Docker, CMake**, Shell, Bash, Perl**, JavaScript**, Node.js, React, Go, Rust, Java, Redux, Scala, R, Ruby, **Rest API**, gRPC, protobuf, Julia, HTML5, PHP

Programming ROS

1. Rviz
2. MoveIt
3. Gazebo
4. PCL
5. C, MISRA C, Embedded C**, RTOS**
6. **Unity 3D, Unreal Engine**

Testing

1. Mocha, SysML, Jasmine, Jest

Machine Learning, AI, Deep Learning

1. **TensorFlow**,
2. Keras**,**
3. **Torch/PyTorch**,
4. CAFFE,
5. Apache MXNet,
6. Theano,CNTK

Software for hardware access

1. Velodyne Development Kit,
2. ZED Stereo Camera SDK,
3. Scanse LIDAR SDK (shutdown)**,**
4. **SLAM**

Programming GPUs

1. **CUDA,**
2. **OpenCL**,
3. OpenGL,
4. DirectX,
5. DirectCompute,
6. Vulkan
7. **Velilog**

Game and Physics Engines for Simulation

1. **Unity 3D,**
2. **Unreal Engine**,
3. CryEngine,
4. Lumberyard,
5. Bullet,
6. Havok,
7. PhysX

Vehicle Communication Protocols

1. **CAN**,
2. LIN,
3. FlexRay,
4. Ethernet (Automotive Ethernet),
5. SPI/I2C,
6. TSN (Time-Sensitive Networking),

7. TCP/IP,
8. WLAN (Wifi),
9. Bluetooth**,**
10. Cryptography Primitives and Cryptoschemes

Data Storage

1. **Redundant Array of Independent Disks (RAID)**,
2. Network Attached Storage (NAS)

Databases

1. HBase**,**
2. **NoSQL,**
3. **MongoDB**,
4. PostgreSQL,
5. **SQL**,
6. **MySQL**,
7. DynamoDB,
8. HDP,
9. Cloudera,
10. EMR,
11. Cassandra,
12. Vertica

Streaming technologies

1. **Apache Kafka**,
2. Storm,
3. Flink,
4. Spark Streaming

Batch technologies

1. **Apache Hadoop**,
2. MapReduce,

3. Apache Spark,
4. Hive,
5. Presto,
6. Impala

Serialization

1. Avro,
2. Parquet,
3. JSON

Vehicle Test — MIL, SIL, HIL, In-Vehicle Test

1. **NI LabVIEW**,
2. NI TestStand,
3. NI VeriStand**,**
4. **dSpace HIL Simulation Systems,**
5. **dSpace RTMaps**,
6. Proemion PETools CAN Tools,
7. Vector CANalyzer,
8. Vector CANape,
9. Vector Capl

Data Visualization and Analysis

1. **Microsoft Excel**,
2. NI DIAdem,
3. Splunk,
4. Datadog,
5. Logz.io,
6. ELK Stack,
7. Looker,
8. Tableau

Web Services

1. Azure and Azure ML,
2. Google Cloud and Google AI,
3. Amazon Web Services (AWS)

Issue Tracking Products

1. Jira

What We have to Achieve in Self Driving Car ?

1. Lane line Detection
2. Traffic sign classifier
3. Traffic Light classifier
4. Behavioural cloning
5. Vehicle detection
6. Extended kalman filter(simulated RADAR and LiDAR)
7. Unscented kalman filter
8. Kidnapped vehicle(if your vehicle somewhere use gps location to get back your car.build localization map with GNSS)
9. PID control
10. MPC control
11. Path planning
12. Road segmesntation

1. Lane line Detection

The goal was to **create a simple pipeline to detect road lines in a frame taken from a roof-mounted camera**.

- Opencv
- Python

  Step1: write a function to draw the line with color,depth
  Step2:Write a Lane Detection Function. That should have the following functions

    1. Read an image
    2. Convert it to grayscale image
    3. perform gaussian blur
    4. Use cany edge detection method for edge detection

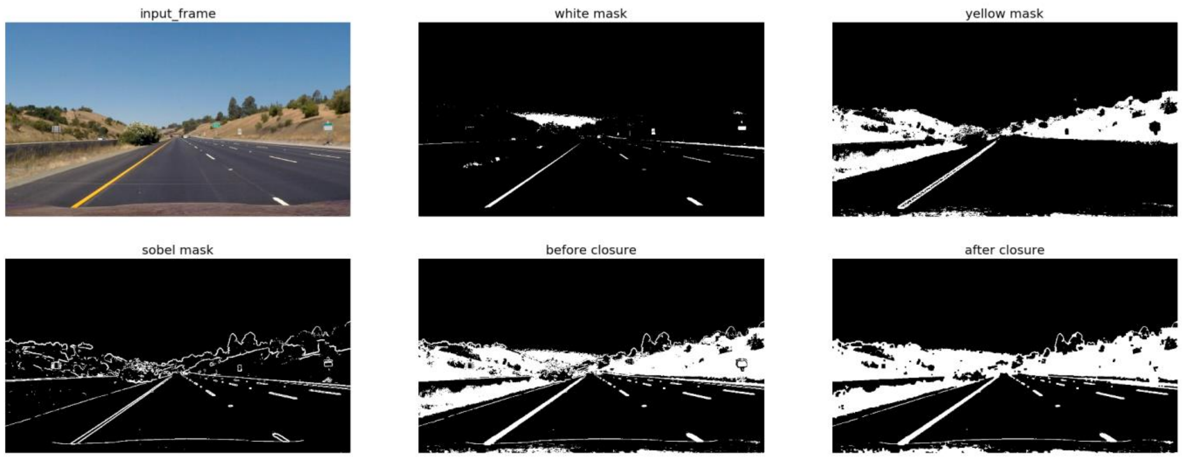
5. perform hough transform
6. detect line
7. write line on image
8. sharpen the image for better result

step3: wite a main function

1. load sensor data from camera(as video)
2. start the video stream
3. check video streaming or not
4. call the necessary function to achieve the goal

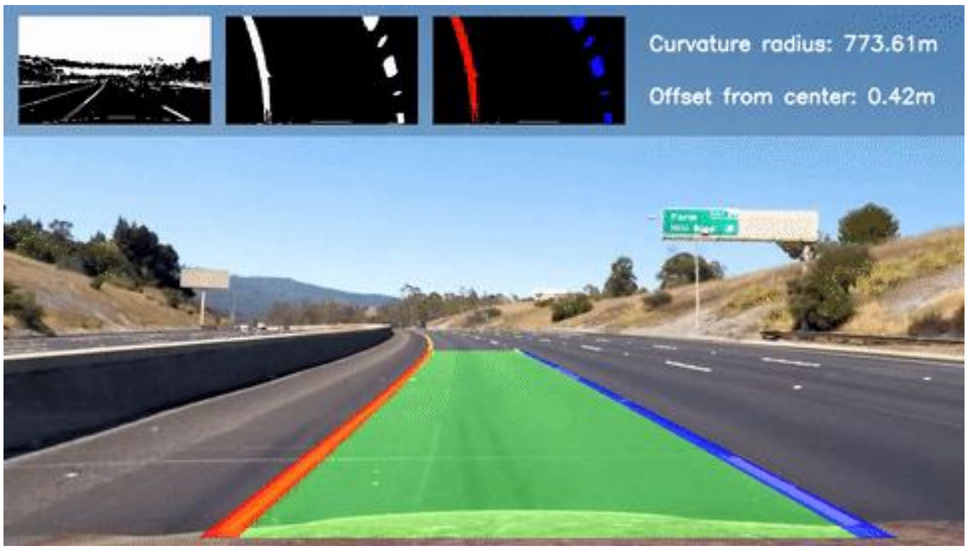
Advanced

- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image.
- Detect lane pixels and fit to find the lane boundary.
- Determine the curvature of the lane and vehicle position with respect to center.
- Warp the detected lane boundaries back onto the original image.
- Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.
- Camera Calibration OpenCV provide some really helpful built-in functions for the task on camera calibration.
- we can use the function `cv2.findChessboardCorners(image, pattern_size)`. Once we have stored the correspondeces between 3D world and 2D image points for a bunch of images, we can proceed to actually calibrate the camera through.`cv2.calibrateCamera()`.
- camera matrix
- distortion coefficients, which we can use to undistort the frames.



you calculated the radius of curvature of the lane and the position of the vehicle with respect to center.

we can approximate the car's deviation from the lane center as the distance between the center of the image and the midpoint at the bottom of the image of the two lane-lines detected. During the previous lane-line detection phase, a 2nd order polynomial is fitted to each lane-line using `np.polyfit()`. This function returns the 3 coefficients that describe the curve, namely the coefficients of both the 2nd and 1st order terms plus the bias.



I think a CNN could be employed to successfully make this step more robust.

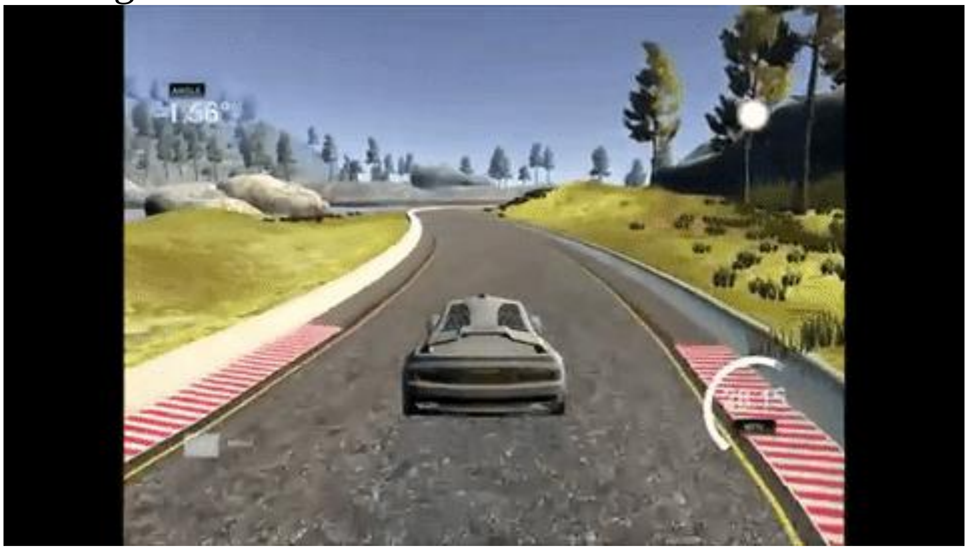
2. Traffic sign classifier

- dataset exploration and visualization
- data augmentation
- deep network design and training
- visualization of the certainty/uncertainty of the model's predictions
- CNN
- Tensorflow or Pytorch
- Keras

3. Behavioral Cloning

The goal was to train a Deep Network to replicate the human steering behavior while driving.
To this purpose, the network takes as input the frame of the frontal camera (say, a roof-mounted camera) and predicts the steering direction at each instant.

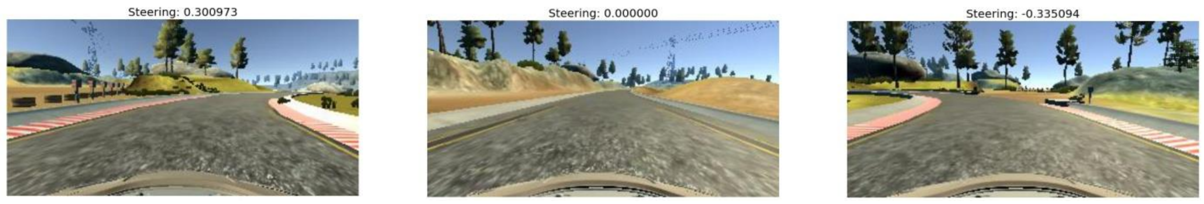
The dataset
Data for this task can be gathered with simulator(gazebo with ros) itself. Indeed, when the simulator is set to *training mode*, the car is controlled by the human though the keyboard, and frames and steering directions are stored to disk.



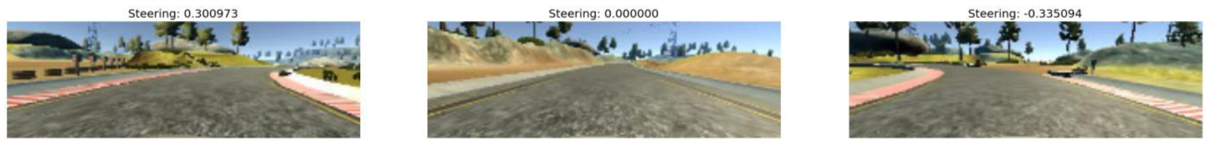
Information gathered from the simulator three frames from the frontal, left and right camera respectively and the corresponding steering direction

Visualizing training data

Here's how a typical sample looks like. We have three frames from different cameras as well as the associated steering direction.



every frame is preprocessed by cropping the upper and lower part of the frame: in this way we discard information that is probably useless for the task of predicting the steering direction. Now our input frames look like these:



As we see, each frame is associated to a certain steering angle. Unfortunately, there's a huge skew in the ground truth da

Data Augmentation

The "test" track is completely different from the "training" one form a visually point of view. Thus, a network naively trained on the first track would **hardly** generalize to the second one. Various forms of data augmentation can help us to deal with these problems.

Exploting left and right cameras

For each steering angle we have available three frames

- captured from the frontal,
- left
- right camera.

Frames from the side cameras can thus be employed to augment the training set, by appropriately correcting the ground truth steering angle.

Brightness changes

Before being fed to the network, each frame is converted to HSV and the Value channel is multiplied element-wise by a random value in a certain range.

The wider the range, the more different will be on average the augmented frames from the original ones.

Normal noise on the steering value

Given a certain frame, we have its associated steering direction.

However, being steering direction a continuous value, we could argue that the one in the ground truth is not *necessarily* the only working steering direction.

Given the same input frame, a slightly different steering value would probably work anyway. For this reason, during the training a light normal distributed noise is added to the ground truth value. In this way we create a little more variety in the data without completely twisting the original value.

Shifting the bias

Bias belonging to range [0, 1] in the **data generator** to be able to mitigate the bias towards zero of the ground truth.

Every time an example is loaded from the training set, a *soft* random theshold `r = np.random.rand()` is computed.

Then the example `i` is discarded from the batch if `steering(i) + bias < r`.

In this way, we can tweak the ground truth distribution of the data batches loaded. The effect of the bias parameter on the distribution of the ground truth in a batch of 1024 frames.

Network architecture

Input normalization is implemented through a `Lambda` layer, which constitutes the first layer of the model. In this way input is standardized such that lie in the range [-1, 1]: of course this works as long as the frame fed to the network is in range [0, 255].

Convolutional layers are followed by 3 fully-connected layers.

finally, a last single neuron tries to regress the correct steering value from the features it receives from the previous layers.

Training Details

Model was compiled using Adam optimizer with default parameters and mean squared error loss w.r.t. the ground truth steering angle.

Training took a couple of hours on the **GPU**. During the training **`bias`** parameter was set to 0.8, frame brightness was augmented in the range [0.2, 1.5] with respect to the original one.

Normal distributed noise added to the steering angle had parameters **mean=0** , **std=0.2**. Frame flipping was random with **probabililty 0.5**.

Testing the model

After the training, the network can successfully drive on both tracks.

it drives better and smoother on the test track with respect to the training track (at least from a qualitative point of view).

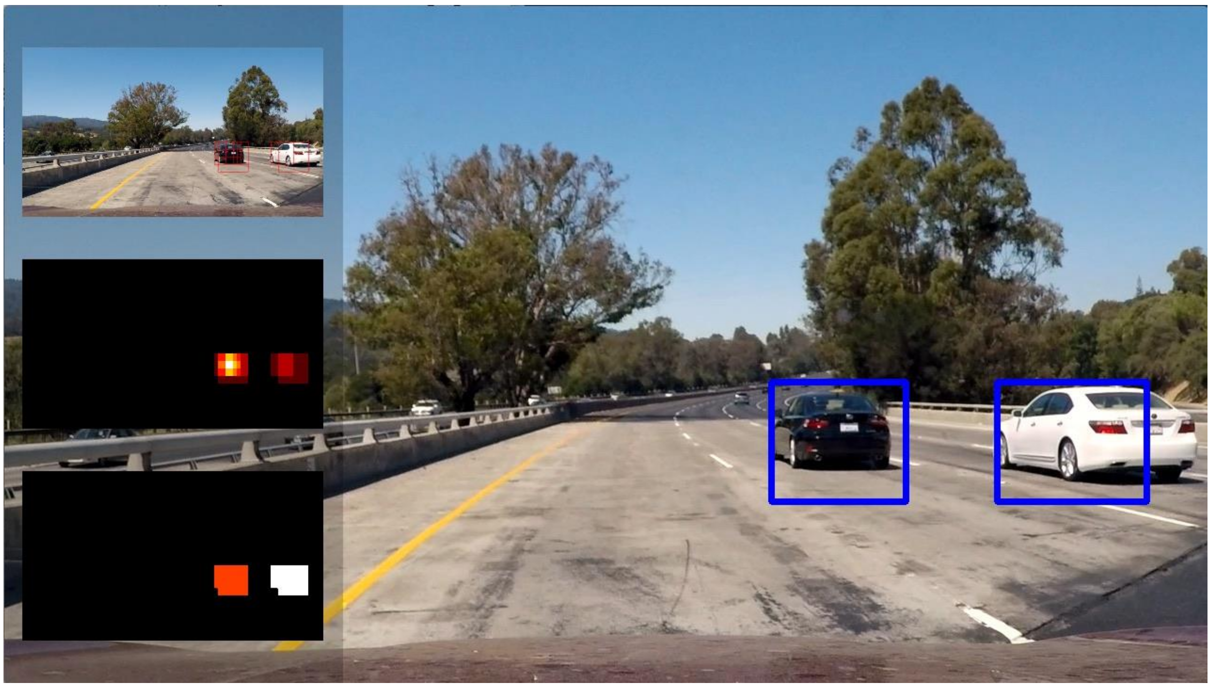
4. Vehicle Detection & Pedestrain Detection

The goal was to develop a pipeline to reliably detect cars given a video from a roof-mounted camera.

HOG features + linear SVM to detect cars, temporal smoothing to discard false positive

SSD deep network for detection, thresholds on detection confidence and label to discard false positive

Histogram of Oriented Gradients (HOG)



- **Feature Extraction**
  In the field of computer vision, a features is a compact representation that encodes information that is relevant for a given task.

  In our case, features must be informative enough to distinguish between car and non-car image patches as accurately as possible.

  Actual feature extraction is performed by the function `image_to_features`, which takes as input an image and the dictionary of parameters, and returns the features computed for that image.

  In order to perform batch feature extraction on the whole dataset (for training), `extract_features_from_file_list` takes as input a list of images and return a list of feature vectors, one for each input image.
- **Choosing HOG parameters.**
  The main parameters are

  the size of the cell in which the gradients are accumulated, as well as the number of orientations used to discretize the histogram of gradients.

  Furthermore, one must specify the number of cells that compose a block, on which later a feature normalization will be performed.

  Finally, being the HOG computed on a single-channel image, arises the need of deciding which channel to use, eventually computing the feature on all channels then concatenating the result.
- **parameters used in the phase of feature extraction**

```
feat_extraction_params = {'resize_h': 64,             # resize image height
before feat extraction
                          'resize_w': 64,             # resize image height before
feat extraction
                          'color_space': 'YCrCb',     # Can be RGB, HSV, LUV,
HLS, YUV, YCrCb
                          'orient': 9,                # HOG orientations
                          'pix_per_cell': 8,          # HOG pixels per cell
                          'cell_per_block': 2,        # HOG cells per block
                          'hog_channel': "ALL",       # Can be 0, 1, 2, or "ALL"
                          'spatial_size': (32, 32),   # Spatial binning dimensions
                          'hist_bins': 16,            # Number of histogram bins
                          'spatial_feat': True,       # Spatial features on or off
                          'hist_feat': True,          # Histogram features on or off
                          'hog_feat': True}           # HOG features on or off
```

- **Training the classifier**

Once decided which features to used, we can train a classifier on these. In [`train.py`](train.py) I train a linear SVM for task of binary classification *car* vs *non-car*. First, training data are listed a feature vector is extracted for each image:

```
cars = get_file_list_recursively(root_data_vehicle)

notcars = get_file_list_recursively(root_data_non_vehicle)
```

car_features = extract_features_from_file_list(cars, feat_extraction_params)

notcar_features = extract_features_from_file_list(notcars, feat_extraction_params)

Then, the actual training set is composed as the set of all car and all non-car features (labels are given accordingly). Furthermore, feature vectors are standardize in order to have all the features in a similar range and ease training.

feature_scaler = StandardScaler().fit(X) # per-column scaler

scaled_X = feature_scaler.transform(X)

Now, training the LinearSVM classifier is as easy as:

svc = LinearSVC() # svc = SVC(kernel='rbf')

svc.fit(X_train, y_train)

In order to have an idea of the classifier performance, we can make a prediction on the test set with svc.score(X_test, y_test).

Sliding Window Search

In a first phase implement a naive sliding window approach in order to get windows at different scales for the purpose of classification.

This is shown in function `compute_windows_multiscale`.

This turned out to be very slow.

Implement a function to jointly search the region of interest and to classify each window as suggested by the course instructor.

The performance boost is due to the fact that HOG features are computed only once for the whole region of interest, then subsampled at different scales in order to have the same effect of a multiscale search, but in a more computationally efficient way.

Whole classification pipelin using CV approach is implemented.

Each test image undergoes through the `process_pipeline` function, which is responsbile for all phases: feature extraction, classification and showing the results.

In order to optimize the performance of the classifier, training with different configuration of the parameters, and kept the best one.

Performing detection at different scales also helped a lot, even if exceeding in this direction can lead to very long computational time for a single image.

At the end of this pipeline, the whole processing, from image reading to writing the ouput blend, took about 0.5 second per frame.

Computer Vision on Steroids, Deep Learning

1. SSD (Single Shot Multi-Box Detector) network

In order to solve the aforementioned problems, use a deep network to perform the detection, thus replacing the HOG+SVM pipeline. For this task employed the recently proposed for detection. This paved the way for several huge advantages:

- the network performs detection and classification in a single pass, and natively goes in GPU (is fast)

- there is no more need to tune and validate hundreds of parameters related to the phase of feature extraction (is robust)

- being the "car" class in very common, various pretrained models are available in different frameworks (Keras, Tensorflow etc.) that are already able to nicely distinguish this class of objects (no need to retrain)

- the network outputs a confidence level along with the coordinates of the bounding box, so we can decide the tradeoff precision and recall just by tuning the confidence level we want (less false positive)

The whole pipeline has been adapted to the make use of SSD network in file

Actually, while using SSD network for integrating detections over time was not only useless, but even detrimental for performance. Indeed, being detections very precide and false positive almost zero, there was no need anymore to carry on information from previous detections.

Other object detection methods:

1. **SSD**
2. **MobilNet**
3. **YOLO**

PID controls

Implement a PID controller for keeping the car on track by appropriately adjusting the steering angle.

What's a PID controller

A proportional integral derivative controller is one of the most common control loop feedback mechanisms.

A PID controller continuously calculates an error function (which in our case is the distance from the center of the lane) and applies a correction based on proportional (P), integral (I), and derivative (D) terms.

Choosing PID Parameters The behavior of a PID controller depends on three main parameters, namely the **proportional, integral and derivative gain**.

Each one of these three parameters controls the strenght of the respective controller's response.

1. **Proportional gain** regulates how large the change in the output will be for a given change in the error. If the proportional gain is too high, the system can become unstable (see *p controller* gif above).

2. **Integral gain** contributes in proportion to both the magnitude of the error and the duration of the error. In this way controller is able to eliminate the residual steady-state error that occurs with a pure proportional controller (*i.e.* a purely proportional controller

operates only when error is non-zero) and is able to deal with systematic biases.

3. **Derivative gain** decides how much the error's rate of change is taken into account when computing the response.

In other words, if the desired setpoint is getting closer (= error is decreasing) the response must be smoothed in order not to overshoot the target.

Derivative component benefits the system's stability and settling time.

In the current project, parameters have been manually tuned by qualitatively inspecting the driving behaviour in the simulator in response to parameter's changes.

Parameter's validation could also be easily performed automatically in a simulator in which headless mode was available.

MPC Control

Model predictive control (MPC) is an advanced method of process control which relies on dynamic models of the process.

MPC controller has the ability to anticipate future events and can take control actions accordingly. Indeed, future time steps are taking into account while optimizing current time slot.

The MPC controller framework consists in four main components:

- **Trajectory**

  Taken in consideration during optimization.

  This is parametrized by a number of time steps ***N*** spaced out by a time ***dt***.

  Clearly, the number of variables optimized is directly proportional to *N*, so this must be considered in case there are computational constraints.

- **Vehicle Model**

  which is the set of equations that describes system behavior and updates across time steps.

  used a simplified kinematic model (so called *bycicle model*) described by a state of six parameters:

- **x** car position (*x-axis*)
- **y** car position (*y-axis*)
- **psi** car's heading direction
- **v** car's velocity
- **cte** cross-track error
- **epsi** orientation error

- **Contraints**

  necessary to model contrants in actuators' respose.

  For instance, a vehicle will never be able to steer 90 deegrees in a single time step. In this project we set these constraints as follows:

steering: bounded in range [-25°, 25°]
acceleration: bounded in range [-1, 1] from full brake to full throttle

- **Cost Function**
  Usually cost function is made of the sum of different terms. Besides the main terms that depends on reference values (*e.g.* cross-track or heading error), other regularization terms are present to enforce the smoothness in the controller response (*e.g.* avoid abrupt steering).

Changing Reference System

Simulator provides coordinates in global reference system. In order to ease later computation, these are converted into car's own reference system

Dealing with Latency

To mimic real driving conditions where the car does actuate the commands instantly, a *100ms* latency delay has been introduced before sending the data message to the simulator.
In order to deal with latency, state is predicted one time step ahead before feeding it to the solver

For visualization this C++ [matplotlib wrapper] could be helpful.

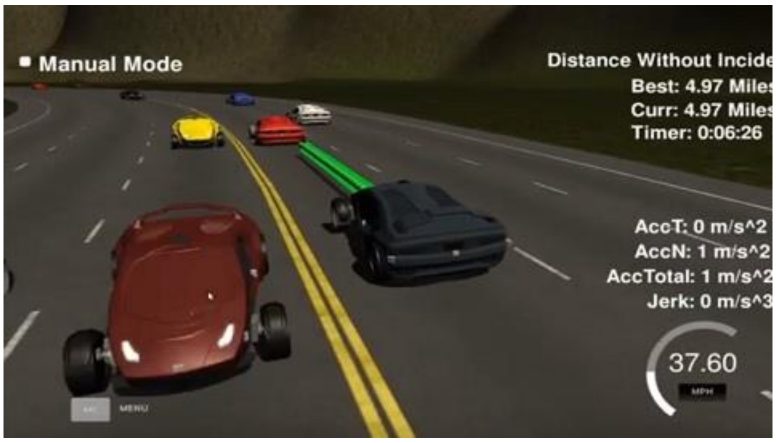
PATH Planning

- Rviz
- Gazebo
- ROS
- **MoveIt**

The goal is to build a path planner that is able to create smooth, safe trajectories for the car to follow. The highway track has other vehicles, all going different speeds, but approximately obeying the 50 MPH speed limit.

The car transmits its location, along with its sensor fusion data, which estimates the location of all the vehicles on the same side of the road.

A trajectory is constituted by as set of (x, y) points. Every 20 ms the car moves to the next point on the list.

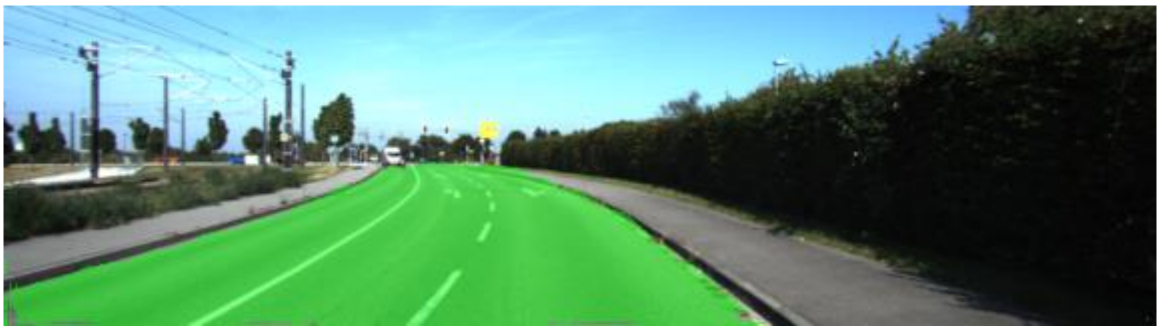


In this framework, the car moves from point to point perfectly.

Since the car always moves to the next waypoint after 20ms, the velocity of the vehicle is given by how much future waypoints are spaced.

The car has now entered the state `prepare_for_lane_change`. However, in order to safely change lane, the car has to check if at least another lane is sufficiently clear of traffic for allowing a safe lane change.

Road Segmentation



- **Tensorflow**
- **Keras**
- **Pytorch**
- **Python**
- **Numpy**
- **Scipy**

- label the pixels of a road in images using a Fully Convolutional Network (FCN).

IMPORTANT CONCEPTS Mostly Used:

- Linear Algebra
- Calculus
- Maths
- Probablity and Statistics
- Input source(collecting data)
- Camera
- LiDAR
- RADAR
- GPS
- IMU
- Fields
- AI,ML,DL,CV,RL
- Control theory
- Signal processing
- Motion planning
- SLAM
- Image processing
- Sensor Calibration
- Sensor Fusion
- Feature extraction

- Object detection, classification,tracking
- Algorithms
    - Search
    - Graph,complex data structure
    - Kalman filter
    - EKF
    - Bayesian network

- Languages
- Libraries
- Softwares
    - Linux
    - ROS middleware 7& tools
    - GBD / DDD (debugger)
    - Git
    - Jetkins ,travis
    - Docker
    - Hadoop,spark,storm,kafka,akka,zookeeper
    - GIS software(cloud)
- Physics
    - Force
    - Moments
    - Inertia
    - Newtons law etc..

CAR HARDWARES

ACTIVE SUSPENSION

it’s an electronically controlled, hydraulically powered system used as a means of maximising downforce by keeping the height of the car constant to the ground.

ACTUATORS

Typically, electronically controlled (but hydraulically powered) pistons that change length in accordancewith command signals from the on- board computer. In the active suspension era, these were used as part of the suspension system. Now they are also used for gear selection and DRS flap movement.

AIRBOX

Normally located above and behind the driver’s head inside the roll hoop, the airbox ducts air from the roll-hoop intake to the engine-intake trumpets and contains an air filter alongits length.

CAMBER

The angle of the tyre relative to the ground when viewed from the front. Typically, racing cars run around four degrees of front camber. This can be seen by the spectator as the tyres ‘leaning in’ towards the centre of the car.

CASTOR

The angle of the steering axis when viewed from the side. Castor is used to create a change in camber with steering lock and also stability; shopping trolleys, for instance, will often run a lot of castor to keep the wheels straight.

CHARGE AIR COOLER

On a turbo-charged engine, the action of the compressor is to raise its pressure, but, as a consequence, the temperature of the air is increased.

The engine loses power as a result of the raised air temperature, so the job of the ‘charge air cooler’ – effectively a radiator – is to cool the charge down again before it enters the engine.

COMPOSITE STRUCTURE

Strictly speaking, a composite structure is any structure made of more than one material. In motor racing it is commonly used to refer to any large component made out of carbon fibre, often containing inserts of aluminium or titanium.

DAMPER

An undamped spring will continue to oscillate after load input. If you twang the end of a ruler, for example, it will continue to oscillate for some time after the initial finger push. A damper is typically an oil-filled piston within a cylinder, whose job is to 'damp' this oscillation. The damper settings must be tunedto suit the spring rate and response to inputs, such as steering, that the race engineer prescribes.

DIFFUSER

A device that expands and slows air down; a hair dryer, for instance, will often have a diffuser attachment to take the concentrated hot blast of air and diffuse it into a slower-moving but wider flow. In motor racing, a diffuser is fitted to the back of the floor. If a low-pressure region of air is generated at the back of the diffuser, for instance by a rear wing, this createsthe slow-moving broad area analogous to that created by the hair- dryer attachment. Because of the contractionahead of it, the air flowing under the car is forced to travel muchfaster, thereby creating a low- pressure area underneath the car.

DOWNFORCE

it is the opposite of an aircraft’s lift – a means of pushing the car into the ground.

DRAG

The aerodynamic force that arises from the movement of an object through air. It’s what you feel when you try to stand upright on a windy day. In motor racing, it absorbs power from the engine and is ultimately what limits the speed of the car. ECU Or electronic control unit. Effectively an on-board computer that is used to control items such as the engine and the gearbox in response to the driver’s demands. All modern cars have them.

ENDPLATE

The vertical bit on the end of the front and rear wings, used to improve the efficiency of the wing.

EXCLUSION BOX

Areas on the car where the regulations state you cannot have any bodywork.

FREESTREAM

Undisturbed air, which in the real world would be stationary air, that the car passes through. In the wind tunnel, where the model is held still, it is the speed of the air passing through the wind tunnel.

GEARBOX DYNO

A factory-based piece of equipment, it typically contains three powerful and very fast-response electric motors used to replicate the action of the engine and rear wheels around a lap, allowing the actual car gearbox to be developed without having to run the car.

GEAR-DOGS

You can picture a gear-dog if you interlock your fingers and then try to slide your left hand past your right hand. That’s exactly what a gear-dog does. It allows the torque from the shafts within the gearbox to be transmitted through the gears.

HYPOID DRIVE

A bevel drive consists of two cone-shaped gears sitting at right angles to turn the longitudinal gearbox shafts through 90 degrees and off down the drive shafts. A hypoid drive is almost identical, except the shaft centres are offset vertically from each other by a small amount.

KART RACING TWO-STROKE

Normally 100cc for fixed-wheel karts and between 200cc and 250cc for gearbox karts (such as mine). Engines in my youth were always stolen out of motorcycles and re-tuned to suit karting. KERS Or Kinetic Energy Recovery System. On a Formula One car, an electric motor is mounted on the end of the crankshaft. Every time the car brakes, some of the braking force is provided by the electric motor performing a charging action, in a similar manner to the little dynamos that people have on their bicycles to operate lights. This electrical energy is stored in the battery and then used on button-request by the driver to augment acceleration down the next straight.

MONOCOQUE

The main structure, the chassis. It contains the driver and the fuel tank, and provides mountings for the front suspension, the engine, and the nose and side-impact structures.

NACA DUCT

Developed by the National Advisory Committee for Aeronautics in the US, this is an air inlet design to duct air to whatever component you wish to supply.

PARC FERMÉ

The parking area where the cars have to be placed at various times through the weekend under the surveillance of the FIA. During this time, teams are only allowed to perform routine checks on the car, with no alteration or maintenance that has not had prior FIA approval.

POWER-TO-WEIGHT RATIO

The power of the engine in horsepower divided by the weight of the car, including driver, in kilograms. Formula One cars have typically been in thearea of 1.2hp per kilogram over recent years.

PULLROD SUSPENSION

A way of transmitting the motion of the wheel to spring damper units mounted inside the chassis at the front and at the side of the gearbox at therear. A pull-rod runs diagonally from the upper wishbone at the wheel end, downto the bottom of the spring damper unit at the inboard end.

PUSHROD SUSPENSION

Essentially the same as pull-rod suspension, except that the rod runs from the lower wishbone at the wheel end, up to the top of the spring damper unit at the inboard end.

RIDE-HEIGHT

The proximity of the car to the ground. Downforce and the balance of downforce between the front and rear axles change as the ride height changes, and therefore ride-height is a key set-up consideration when adjusting a car to suit a particular circuit.

ROLL BAR STIFFNESS

Responsible for controlling the car's roll. A stiff roll bar will mean the car will roll very little, which is good for response and aerodynamic platform control but poor for the ride over bumps and kerbs. The balance of stiffness between the front and rear is known as mechanical balance; a very stiff front bar and soft rear, for instance, will lead to a very stable car that has a lot of understeer.

SPACER

This is often used by chassis designers as a slang term for the engine; i.e. the bit that joins the back of the chassis to the front of the gearbox. It is also a term used in long-distance racing for inserts fitted into the seat of the chassis to reduce its size for smaller teammates.

SPRING RATES

This refers to the stiffness of the suspension system, which seeks to find the optimum compromise between being very stiff foraerodynamic platform control and much softer for load fluctuation and the ride over bumps and kerbs. It is another key set-up variable between individual circuits.

TIP VORTEX

The vortex formed when the high pressure on one side of the wing tries to leak around the tip of the wing to the low pressure on its opposite side.

TOE-IN/TOE-OUT

The angle of the wheels when the steering wheel is straight ahead. Viewed from above, the wheels are angled slightly inwards when they are toe-in, and slightly outwards for toe-out.

TRACK ROD

The suspension member that controls the steering on the front wheels and prevents steering of the rear wheels.

TRACTION CONTROL

Considered a driver aid after the regulation changes of 1993, this system is designed to modulate the power of the engine to prevent the rear wheels from spinning up at a corner exit. It needs no input from the driver and is instead controlled by the ECU.

UNDERWING

A term originally used in the late 1970s to describe the huge wing shapes that were placed underneath the sidepods of the cars, with sliding skirts attached to the tips of the wing to prevent leakage and the consequent tip vortices.

WING

The most important part of an F1 car is the front wing . The front wing is responsible not only for generating most of the front downforce of the car but also for controlling the wake off the front wheels behind it.

YAW

When a car brakes, it ‘pitches forward’; when it accelerates, it ‘pitches rearwards’; and when it corners, it ‘rolls’. But also it rotates in ‘yaw’, which describes the rotation of the entire car as it follows the steering of the front wheels.