

AUTOMOTIVE EMBEDDED

1. ADAS (Advanced Driver Assistance System) : It refers to a category of tech and systems designed to assist drivers in the operation of the vehicle and enhance the safety by providing features like adaptive cruise control, lane keeping assist, automotive parking and collision avoidance.

ADAS systems uses sensors such as cameras, radar, lidar, ultrasonic sensors to monitor the vehicle's surroundings and provide real time assistance to the driver.

ADAS systems includes adaptive cruise control, lane departure warning, blindspot monitoring and automatic emergency braking.

2. AUTOSAR(Automotive Open system Architecture): It is an open and standardized software architecture framework primarily focused on the development of software for Electronic control units(ECUs) within vehicles.

AUTOSAR defines a common platform, standards and specifications for automotive software development, aiming to make automotive software more modular, scalable and reusable.

AUTOSAR focuses on the overall software architecture communication, communication protocols, interfaces and development methodologies for building software systems in vehicles.

AUTOSAR is used to standardise and manage the software components responsible for functions like engine control, infotainment systems and more.

3. CAN(Controller Area Network) Networking:

It was developed to facilitate the communication between electronic control units(ECUs) in vehicles enabling them to exchange data without a central computer.

CAN typically uses a bus topology where multiple devices are connected to a single communication line. CAN high and CAN low are two bus lines kept at 3.75 V and 1.25 V.

Message frame from CAN are structured into frames, consisting of an identifier(ID), data length code(DLC), CRC(Cycle redundancy check) and Acknowledgment bits.

CAN and CAN FD protocol

CAN data frame

Start of Frame

Remote Transmission Request

Delimiter Bits

Bus Idle

SOF

Message Identifier

RTR

Control Field

Data Field

CRC

ACK

EOF

IFS

Bus Idle

Arbitration

Start of Frame – A dominant bit begins the frame and initiates arbitration

Message Identifier – 11-bit identifier used for arbitration priority

Remote Transmission Request – Indicates whether this is a data or request frame

Control Field – Specifies the length of the data to be transmitted

Data Field – Up to eight bytes of data

CRC Sequence – 15-bit cyclic redundancy check

ACK – Acknowledges the CRC status of receiving nodes

End of Frame – Marks the end of data and remote frames

1:58 / 18:36 • CAN data frame >

TEXAS INSTRUMENTS

2

7 bit IFS separates every CAN message. SOF is always dominant 0 and is used to synchronize the bus after the idle state.

CAN uses differential signaling where the state of a bit is determined by the voltage difference between two wires(CAN-high) and (CAN-low). This provides more noise immunity

CAN uses a priority based arbitration mechanism. When two devices attempt to transmit simultaneously, the one with the lower ID wins the bus and transmits the message.

Dominant bit is logical 0 and recessive bit is logical 1. Logical 0 overrides Logical 1 in AND operation.

CAN has robust error detection and error handling mechanisms including cyclic redundancy checks(CRC) and acknowledgment mechanisms.

Data Rate : CAN supports various data rates such as 125 kbps, 250 kbps, 500 kbps and 1 Mbps, allowing flexibility in communication speed.

Extended CAN : It introduces 29 bit identifiers(instead of standard 11 bit) increasing the number of unique message IDs and enabling more complex network.

CAN FD: It is CAN with Flexible Data rate which extends the original CAN protocol by allowing variable data lengths and faster data rates. This is especially useful for high bandwidth applications.

Bit Stuffing : A bit of opposite polarity is inserted after five consecutive identical bits.

OSI(Open System Interconnection) model

CAN has a seven layer model, the CAN lower layers cover some functions of the transport layer(eg. automatic retransmission of faulty frames)

Layer 1 - Physical layer 2 - Data link layer, Layer 3 - Network layer, 4 - Transport layer, 5 - Session , Layer 6 - Presentation Layer 7 - Appllication

4. CAN Physical Layer

The CAN physical layer is a critical component of the CAN protocol, responsible for transmitting data between devices on the network. It defines how bits are physically transmitted over the communication medium, ensuring reliability and robustness.

- Differential Signaling : It uses a differential signaling scheme which means it transmits data by measuring the voltage difference between two wires.

CAN High(CAN- H) - This wire carries a voltage that is higher than the reference voltage during a dominant bit(logical 0)

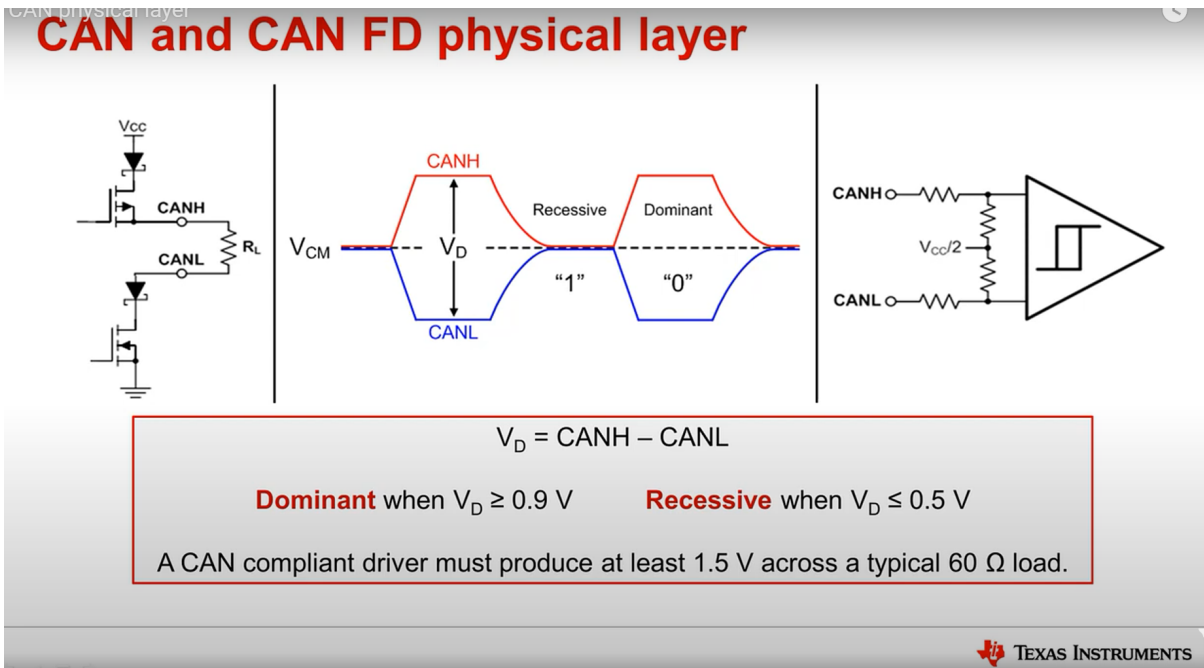
CAN Low(CAN L) : This wire carries a voltage that is lower than the reference voltage during a dominant bit(logical 0)

- Voltage levels

In differential mode, a dominant bit(0) is represented when CAN-H is greater than CAN-L whereas a recessive bit(1) is represented when CAN-H and CAN-L are approximately at same voltage levels.

Termination : Proper termination is essential in the CAN physical layer to minimize signal reflections and ensure signal integrity. Termination resistors(120 ohm) are placed at both ends of the resistors to match the characteristics impedance of the transmission line.

Bit Timing - CAN uses a bit time synchronization mechanism to ensure that all nodes on the network are in sync. It defines the length of time for each bit and the timing of the sampling points for bit reception. It is an asynchronous model.



The wiring of CAN are twisted together to reduce electromagnetic interference and form a differential signal. The wires are connected to each device on the bus through a transceiver that converts the logic level signals to the bus level signals

Connectors used in CAN networks are typically standardized such as Deutsch DT series or the AMP Superseal series.

Baud rate refers to the speed at which data is transmitted over the network, and is typically measured in bits per second(bps)

5. Introduction to Vector Tools

CANoe and **CANalyzer** are vector tools used for CAN network simulation, analysis and debugging.

- CANoe is a comprehensive tool that allows users to create and simulate a CAN network scenario. It provides features such as creating ECU nodes, CAPL scripting and advanced analysis capabilities. CANoe is used for simulating complex scenarios and is suitable for intermediate to advanced users.

- CANalyzer : It is another vector tool that focuses on analyzing and debugging CAN traffic. It offers advanced logging techniques, data analysis capabilities and features like filters and triggers for in-depth analysis. It is suitable for users who require advanced logging and analysis capabilities.

6. ECUs(Electronic Control Units)

What are the different ECUs/applications available?

| Power Train | Chassis | Body |
|--|--|---|
| <ul style="list-style-type: none"> ✓ Engine Control ✓ Transmission Control ✓ Hybrid Control | <ul style="list-style-type: none"> ✓ Steering, Braking, ABS ✓ Suspension ✓ Electronic Stability Control ✓ Four wheel drive (4WD) | <ul style="list-style-type: none"> ✓ Doors & Windows ✓ Lighting ✓ Key Fob |
| Multimedia, Telematics & HMI | Active/Passive Safety | ADAS |
| <ul style="list-style-type: none"> ✓ Infotainment ✓ Instrument Cluster ✓ Telematics Unit | <ul style="list-style-type: none"> ✓ Adaptive Cruise Control ✓ Air Bag / Seat-Belt ✓ Lane Keeping Assist ✓ Collision Warning | <ul style="list-style-type: none"> ✓ 360° Surround-View ✓ Traffic Sign Indication ✓ Parking Assistance |

Automotive Embedded Systems Domain Career Paths

| Requirements Management | Software Development | Software Testing | Hardware Development |
|---|--|---|---|
| <ul style="list-style-type: none"> ✓ Vehicle Features Understanding ✓ Documentation - DOORS | <ul style="list-style-type: none"> ✓ Application S/w ✓ Device Drivers ✓ Boot loader ✓ Board Support Package ✓ Board bring-up ✓ Flashing Protocol ✓ Flashing Tool ✓ IoT ✓ C / C++ programming ✓ AUTOSAR Development/Integration ✓ Functional Safety Engineer ✓ Embedded Linux / Kernel development ✓ Model-based development ✓ Computer Vision / Image Processing ✓ Algorithm development / hardware accelerator optimization ✓ Build/Release Integration ✓ RTOS / Scheduler | <ul style="list-style-type: none"> ✓ Software-In-Loop ✓ Model-In-Loop ✓ Hardware-In-Loop ✓ White-box Testing ✓ Regression Testing ✓ Static Analysis ✓ Memory Profiling | <ul style="list-style-type: none"> ✓ Component Selection ✓ Schematic Design ✓ PCB Design ✓ Circuit Simulation ✓ Hardware Benchmarking ✓ Wiring Harness Designer |
| HMI Design | | | Hardware Testing |
| <ul style="list-style-type: none"> ✓ UI/UX Design ✓ Graphics Design <ul style="list-style-type: none"> – Photoshop & Illustrator ✓ HMI/Animation Design <ul style="list-style-type: none"> – Altia, QT, Kanzi – EB Guide, GL Studio | | | <ul style="list-style-type: none"> ✓ Prototype Testing ✓ System Integration Testing ✓ Board bring-up |
| Network Design | | Test Automation | Vehicle Testing |
| <ul style="list-style-type: none"> ✓ Vehicle Features Understanding ✓ Protocol Understanding ✓ Design Messages, Signals, data packaging | | <ul style="list-style-type: none"> ✓ CAPL Scripting ✓ Python ✓ C# (DLLs) ✓ Excel Automation ✓ XML Automation ✓ Test Bench/ Framework | <ul style="list-style-type: none"> ✓ Integration Tester ✓ Vehicle Tester (Production milestones) |

7 CAN dump and CAN send

CAN dump also known as CAN logging or CAN recording, refers to the process of capturing and string the data transmitted over a CAN bus. This data can include messages from various ECUs, such as sensor readings, control commands and other information.

CAN dump is mostly used for troubleshooting, debugging and analyzing the behaviour of a CAN bus network. It allows engineers and technicians to monitor the CAN bus's data traffic

to identify issues, diagnose faults and analyze the performance of the network. It's used in automotive applications.

CAN send refers to the process of transmitting messages or data onto a CAN bus. This involves sending CAN frames with specific IDs and data payloads to other devices or ECUs on network.

CAN send is used for a range of purposes, such as controlling or configuring the ECUs, sending commands to actuators and communicating with other devices on the CAN bus. It allows for real time interaction with the CAN network, enabling features like remote control updates and reconfiguration of connected devices.

8. Common Jargons used while writing a DBC file

```
VERSION ""
NS_ :
NS_DESC_
CM_ :
BA_DEF_
BA_
BA_DEF_DEF_
BA_DEF_SGTYPE_
BA_DEF_SIGNAL_
BA_DEF_ENVVAR_
BA_DEF_VAL_
BA_DEF_ERROR_
BA_DEF_FAULT_
BA_DEF_SGTYPE_
BA_
BA_SGTYPE_
BA_SIGNAL_
BA_RELATION_
BA_RELATION_SG_
BA_RELATION_EV_
BA_RELATION_DA_
BA_RELATION_PD_
BA_CONSUMER_
BA_PRODUCER_
BA_SGTYPE_TABLE_
BA_DEF_SGTYPE_REL_
BA_DEF_SGTYPE_EV_
BA_DEF_SGTYPE_ENVVAR_
BA_DEF_SGTYPE_DATA_
BA_DEF_SGTYPE_DECODE_
BA_DEF_SGTYPE_CHECK_
BU_ : Main Control
BO_ 1 MainControlStatus: 8 MainControl
    SG_ MainControlState : 0|1@1+ (1,0)
[0|1] "" MainControl

BU_ : Door
BO_ 2 DoorStatus: 8 Door
```

The provided text is a snippet of a DBC (Diagnostics and Calibration Data) file, which is commonly used for defining the communication messages and signals in a CAN network.

1. **`VERSION `**: This line typically specifies the version of the DBC file format. In this case, it's left empty, which is common practice when you don't have a specific version to specify.
2. **`NS_ `**: This line indicates the start of a new namespace block, which is used for organizing objects within the DBC file. The colon (:) indicates that this is an open namespace block, but it doesn't specify a namespace name.
3. **`NS_DESC_ `**: This line is typically used to provide a description for the current namespace, but it's empty in this case, meaning there's no description provided.
4. **`CM_ `**: Similar to line 2, this line indicates another namespace block, but it's left empty without specifying a namespace name.
5. The following lines starting with **`BA_DEF_ `**, **`BA_ `**, **`BA_DEF_DEF_ `**, etc., are used to define attributes and attribute values for objects within the DBC file. These attributes can be used to provide additional information and metadata about messages, signals, nodes, etc., in the CAN network. For example, **`BA_DEF_SGTYPE_ `** defines a signal type attribute, **`BA_DEF_SIGNAL_ `** defines a signal attribute, and so on.
6. **`BU_ : Main Control`**: This line defines a new ECU (Electronic Control Unit) node named "Main Control." ECU nodes represent the electronic control units in your network, and this line establishes one of them.
7. **`BO_ 1 MainControlStatus: 8 MainControl`**: This line defines a message (or frame) with ID 1, named "MainControlStatus," belonging to the "Main Control" ECU. The message has a data length of 8 bytes.
8. **`SG_ MainControlState : 0|1@1+ (1,0) [0|1] "" MainControl`**: This line defines a signal named "MainControlState" within the message "MainControlStatus." The signal has two states: 0 (Off) and 1 (On). The signal is associated with the "Main Control" ECU.
9. **`BU_ : Door`**: Similar to line 6, this line defines another ECU node named "Door."
10. **`BO_ 2 DoorStatus: 8 Door`**: This line defines a message with ID 2, named "DoorStatus," belonging to the "Door" ECU. The message also has a data length of 8 bytes.
11. **`SG_ DoorState : 0|1@1+ (1,0) [0|1] "" Door`**: This line defines a signal named "DoorState" within the message "DoorStatus." Like the previous signal, it has two states: 0 (Close) and 1 (Open). The signal is associated with the "Door" ECU.

In summary, this DBC snippet defines two ECU nodes ("Main Control" and "Door") and messages with associated signals for each ECU. It specifies signal names, signal states, message IDs, message names, and the ECU to which each signal or message belongs.

within a CAN network. This information is crucial for simulating and communicating between electronic control units in a CAN network.

Alternatively, We can define a database via Vector CANdb++ and define messages inside it.

1. Define Network node which is ECU
2. Define Messages
3. Define Signal
4. Define Map signal database.
5. Edit signal with map signal database.
6. Map signal to message
7. Map network node to message.

9. Creating GUI in CANoe

1. Go to home > Panel > create Panel.
2. Add buttons from standard control panel on the right side similar to Netbeans JDK.

10. CAPL Scripting Primer for Beginners

CAPL(Communication Access Programming Language) is a scripting language commonly used in the field of automotive testing and simulation. It allows you to create and control communication and simulation scenarios in various automotive software tools.

>> HELLO, CAPL !

A CAPL script consists typically of two parts : variables and functions



```
variables
{
    message MyMessage;
}

on start
{
    // This code runs when the script
    starts
    output("Hello, CAPL!\n");
}

on message MyMessage
{
    // This code runs when the message
    MyMessage is received
    output("Message received!\n");
}
```

2. Variables and Data Types

CAPL supports various data types like 'int' and 'float' and custom types like 'message'. Here's how to declare and initialise variables



```
variables
```

```
{
```

```
    int myInteger = 42;
```

```
    float myFloat = 3.14;
```


```
    message myMessage;
```

```
    byte myArray[8];
```

```
}
```

3. Sending Messages

CAPL is often used to simulate messages on the CAN bus. To send a bus following snippet maybe used.



```
on key 's'
{
    myMessage = { 0x123, 8, "Hello",
"CAPL" };
    output("Sending message: %.*H\n",
myMessage.dlc, myMessage.byte(0));
    output("Data: %s %s\n",
myMessage.byte(0), myMessage.byte(1));
    output("ID: 0x%X\n", myMessage.id);
    output("DLC: %d\n", myMessage.dlc);
}
```

4. Receiving Messages


To react to incoming messages, following snippet is being used



```
on message myMessage
{
    output("Received message: %.*H\n",
this.dlc, this.byte(0));
    output("Data: %s %s\n", this.byte(0),
this.byte(1));
    output("ID: 0x%X\n", this.id);
    output("DLC: %d\n", this.dlc);
}
```

5. Flow Control

CAPL supports common programming constructs like loops and conditionals. For example



```

on key 'p'
{
    for (int i = 0; i < 5; i++)
    {
        output("Count: %d\n", i);
    }

    if (myInteger == 42)
    {
        output("Flow control working!\n");
    }
}

```

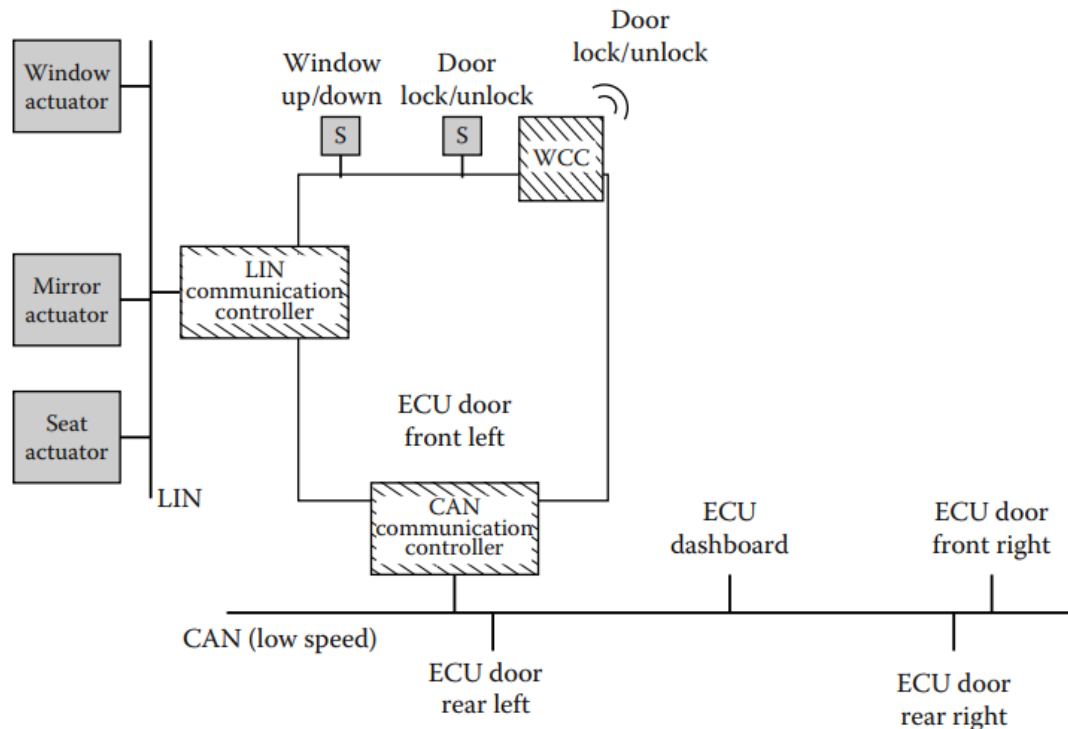
Testing the script includes plugging the CAPL script to an ECU node and compiling

11. DID YOU KNOW ?

- Today, upto 2500 signals are exchanged through up to 70 electronic control units(ECUs) on five different types of networks.
- Besides AUTOSAR, There is AEE*(Automotive Electronics Environment), EAST(Enterprise Architect for Automotive Systems), OSEK/VDEX(Open system and the corresponding interfaces for automotive electronics) are several other standards and frameworks.
- "Power train"(Control of engine and transmission) + "Chasis"(Control of suspension, braking and steering) domains are ECUs concerned with real time control and safety.
- - Principal players in the automotive industry can be divided into - Vehicle manufacturers, Automotive third part suppliers and Tool-embedded software suppliers.
- Controlling the volume of radio as per vehicle speed, When an accident causes an airbag to inflate, its microcontroller emits a signal to the embedded global positioning system receiver that then communicates with the cell phone, making it possible to give the vehicle's position to the rescue service.

- Various automotive embedded networks include LIN(Local interconnected network), CAN, TTP/C, FlexRay, MOST & IDB-1394.

- Illustrated is the examples of door/window/seat/mirror control



12. Operating Systems

- OSEK/VDX is a multitask operating system that is becoming a standard in the European automotive industry. This standard is divided into four parts - OSEK/VDX OS is the specification of the kernel itself; OSEK/VDX COM concerns the communication between tasks(internal or external ECU); OSEK/VDX NM addresses the network management and finally OSEK/VDX OIL is the language that supports the description of all the components of an application.

- OSEK/VDX OS provides services on objects like tasks("basic tasks" without blocking points and "extended tasks" that can include blocking points), events, resources and alarms. It proposes a fixed priority(FP) scheduling policy that is applied to tasks that can be preemptive or non-preemptive and combined with a reduced version of the priority ceiling protocol. Intertask synchronization is achieved through private events and alarms.

13. Middleware

It refers to software components or layers that facilitate communication and interaction between various hardware and software components within a vehicle's electronic control system or ECU.

14. Architectural Description Language(ADL)

ADL for automotive embedded systems is a specialized modeling language and framework used in the design and development of embedded software and electronic control units(ECUs) for automotive applications.

One example of ADL in the automotive industry is the AUTOSAR standard. It is a standardized software architecture and methodology for automotive software development.

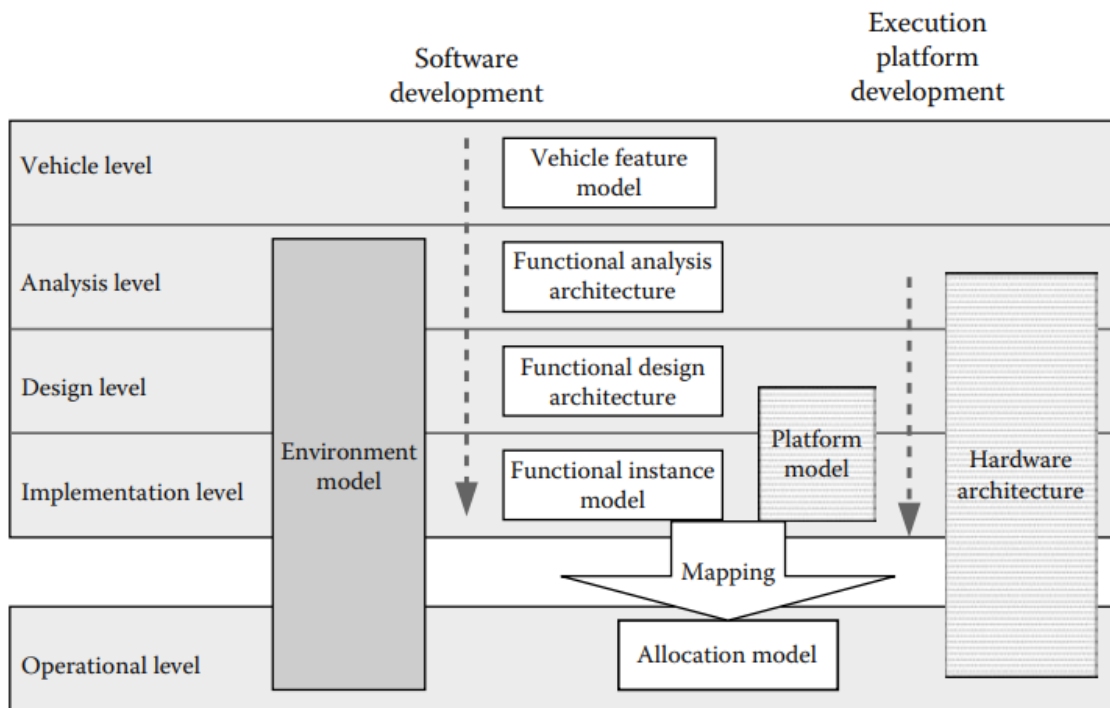


FIGURE 1.3 The abstraction levels and the system views in EAST-ADL.

15. Matlab, Simulink, Stateflow, Simscape & ASpice

Matlab:

- MATLAB is like a digital math toolbox.
- It's a software that helps people do math and data analysis.
- You can use it to perform calculations, plot graphs, and solve equations.
- It's like a calculator, but way smarter and more powerful.
- Scientists, engineers, and researchers often use MATLAB.

Simulink:

- Simulink is like a digital model playground.
- It's part of MATLAB and used for making models of systems or processes.
- You can create models by connecting blocks that represent different parts of a system.
- It's great for simulating and testing how things work before building them in the real world.

- Engineers and scientists use Simulink for designing and testing systems like cars and airplanes.

Stateflow:

- Stateflow is like a digital decision maker.
- Also part of MATLAB, it helps create decision-based systems.
- You design these systems using flowcharts that show how something behaves depending on different conditions.
- It's used for making decisions in complex systems, like in aviation and automotive control.
- Stateflow helps make sure that things work the way they should in different situations.

Simspace:

- It is like a virtual playground for engineers. It's a special software that helps us design and test stuff in a computer before making it in real life.
- We use it to create and check how things like machines, vehicles or systems will work.

ASpice

- ASpice(Automotive Software Process Improvement and Capability Determination) is like a quality certificate for car software.
- It's a set of rules and guidelines that carmakers follow to make sure the software in their vehicles is top notch.
- Car companies use ASpice to ensure their software is safe, reliable and works well.

16. ADAS Functionalities

16.1 - Lane Keeping Assist vs. Lane Departure Warning vs. Lane Centring

LKA:

- Think of Lane Keeping Assist as your "gentle steering buddy" on the road.
- LKA helps keep your car centered within your lane by making subtle steering adjustments.
- It's like having a friend gently nudging the steering wheel to prevent you from accidentally drifting out of your lane.
- LKA uses cameras and sensors to monitor the lane markings and gives you a little extra help in maintaining a straight path.
- Two common methods used for implementing LKA are the Pure Pursuit method and the Stanley method.

- Pure Pursuit Method

- It is a path tracking algorithm used in LKA systems. It calculates the steering angle to follow a reference path, typically the centerline of the lane.
- The vehicle's current position is determined through sensors, such as cameras or LiDAR and the algorithm calculates the error or the difference between the desired path and the vehicle's actual position.
- Using the error, the Pure Pursuit method computes the steering angle needed to minimize the error and keep the vehicle on the desired path.
- It is called "Pure Pursuit" because it pursues a point on the desired path ahead of the vehicle, taking into account the vehicle's dynamics.

- Stanley Method

- The Stanley method is similar to Pure pursuit method, It also calculates the vehicle's position relative to a reference path.
- It uses a different approach to determine the steering angle. It computes the "cross-track error" which is the lateral distance between the vehicle and the reference path. It also takes into account the "heading error" which is the difference between the vehicle's current orientation and the desired orientation on the reference path.
- By considering both the cross track error and the heading error, the Stanley method determines the appropriate steering angle to keep the vehicle on the desired path.

LDW:

- Imagine Lane Departure warning as your "Lane boundary alarm"
- LDW watches the road and warns you if it senses you're unintentionally leaving your lane.
- It's like having a helpful friend who taps you on the shoulder and says "Hey, stay in your lane !"
- LDW relies on the same cameras and sensors as LKA to detect lane markings but doesn't actually steer your car.

LC:

- Lane centering is like having a "self guided car" that stays perfectly in the middle of the lane.
- This advanced system not only warns you about drifting but actively keeps your vehicle centered in the lane.
- It's as if your car has a super smart co-pilot that takes care of steering and helps maintain your position in the middle of the lane.
- Lane centering typically combines the features of LKA and LKW offering a more hands-free driving experience.

16. CARLA simulator

- CARLA(Car learning to Act) is a cutting edge, opensource driving simulator designed to mimic real world automotive environments and scenarios.
- Developed by the Computer Vision Center(CVC) and Barcelona Super computing, It serves as a valuable tool for testing and training autonomous driving systems.
- It offers a highly realistic and dynamic simulation platform that can generate complex urban landscapes and traffic conditions.
- It has advanced features like sensor models and a wide range of configurable params.

17. Self Driving Cars specialisation by University of Toronto

Course 1

Glossary of Important Terms

ACC: Adaptive Cruise Control

A cruise control system for vehicles which controls longitudinal speed. ACC can maintain a desired reference speed or adjust its speed accordingly to maintain safe driving distances to other vehicles.

Ego

A term to express the notion of self, which is used to refer to the vehicle being controlled autonomously, as opposed to other vehicles or objects in the scene. It is most often used in the form ego-vehicle, meaning the self-vehicle.

FMEA: Failure Mode and Effects Analysis

A bottom up approach of failure analysis which examines individual causes and determines their effects on the higher level system.

GNSS: Global Navigation Satellite System

A generic term for all satellite systems which provide position estimation. The Global Positioning System (GPS) made by the United States is a type of GNSS. Another example is the Russian made GLONASS (Globalnaya Navigazionnaya Sputnikovaya Sistema).

HAZOP: Hazard and Operability Study

A variation of FMEA (Failure Mode and Effects Analysis) which uses guide words to brainstorm over sets of possible failures that can arise.

IMU: Inertial Measurement Unit

A sensor device consisting of an accelerometer and a gyroscope. The IMU is used to measure vehicle acceleration and angular velocity, and its data can be fused with other sensors for state estimation.

LIDAR: Light Detection and Ranging

A type of sensor which detects range by transmitting light and measuring return time and shifts of the reflected signal.

LTI: Linear Time Invariant

A linear system whose dynamics do not change with time. For example, a car using the unicycle model is a LTI system. If the model includes the tires degrading over time (and changing the vehicle dynamics), then the system would no longer be LTI.

LQR: Linear Quadratic Regulation

A method of control utilizing full state feedback. The method seeks to optimize a quadratic cost function dependent on the state and control input.

MPC: Model Predictive Control

A method of control whose control input optimizes a user defined cost function over a finite time horizon. A common form of MPC is finite horizon LQR (linear quadratic regulation).

NHTSA: National Highway Traffic Safety Administration

An agency of the Executive Branch of the U.S. government who has developed a 12-part framework to structure safety assessment for autonomous driving. The framework can be found here.

https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf

ODD: Operational Design Domain

The set of conditions under which a given system is designed to function. For example, a self driving car can have a control system designed for driving in urban environments, and another for driving on the highway.

OEDR: Object and Event Detection and Response

The ability to detect objects and events that immediately affect the driving task, and to react to them appropriately.

PID: Proportional Integral Derivative Control

A common method of control defined by 3 gains.

- 1) A proportional gain which scales the control output based on the amount of the error
- 2) An integral gain which scales the control output based on the amount of accumulated error
- 3) A derivative gain which scales the control output based on the error rate of change

RADAR: Radio Detection And Ranging

A type of sensor which detects range and movement by transmitting radio waves and measuring return time and shifts of the reflected signal.

SONAR: Sound Navigation And Ranging

A type of sensor which detects range and movement by transmitting sound waves and measuring return time and shifts of the reflected signal.

What makes up a driving task ?

Lateral control - Steering

Longitudinal control - braking and acceleration

Object and event detection and response - detection and reaction

Planning - Long term and short term

Can the above be automated to make an automotive system ?

Level 1 - Driving Assistance

1. **Adaptive Cruise Control** - can control speed but the driver has to steer.
2. **Lane Keeping Assistance** - can help you stay in the lane, if you drift

Level 2 - Partial Driving Automation

1. GM Super Cruise - Longitudinal and Lateral control enabled
2. Nissan ProPilot
3. Tesla FSD

Level 3 : Conditional Driving Automation

Longitudinal Control + Lateral control + OEDR

- Audi A8 Sedan

Level 4 - High Driving Automation

Longitudinal Control + Lateral Control + OEDR + Fallback

Waymo

Level 5 - High Driving Automation

It can operate in any condition. No examples, yet. Ideal for society deployment. Automated taxies.

.

Goals for perception

1. **Detection of static objects** - Road and lane markings(on road), curbs(off road), traffic lights(off-road), road signs(off road), construction signs, obstructions and more(on - road)
2. **Detection of dynamic objects** - Motion needs to be predicted of vehicles(4 wheelers, 2 wheelers), pedestrian
3. **Ego Localization** : Position, velocity, acceleration, orientaion, angular motion via GPS, GNSS and IMU sensors

Challenges to perception

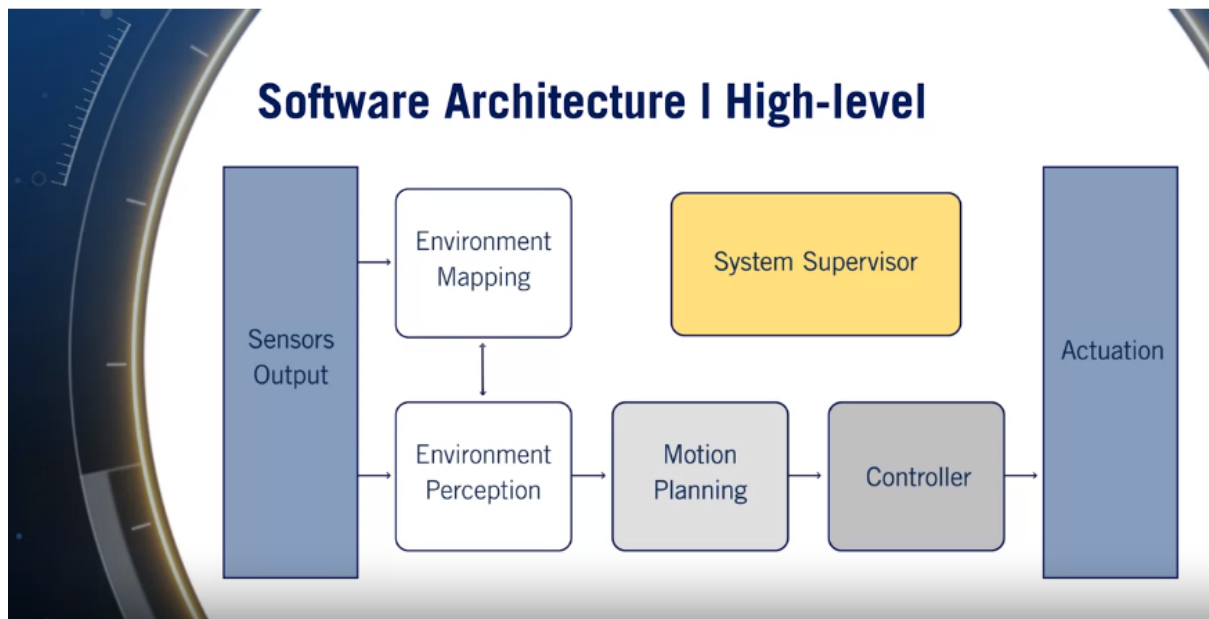
Robust detecton and segmentation, sensor uncertainty, occlusion, reflection, illumination, lens fare, weather, precipitation

Sensors

1. **Exteroceptive** : surroundings
2. **Proprioceptive** : internal

Software Architecture of a Self driving software system

1. **Environment Perception**
2. **Environment Mapping**
3. **Motion Planning**
4. **Vehicle Controller**
5. **System Supervisors**



The raw sensor measurements gathered from variety of sensors equipped around a self driving car are passed into two sets of modules dedicated to understanding the environment around the car.

The environment perception modules have two key respon. i.e first identifying the current location of the autonomous vehicle in space and second classifying and locating important elements of the environment for the driving task.

Examples of these elements includes other cars, bikes, pedestrians, the road, road markings and road signs

The environment mapping module creates a set of maps which locate objects in the environment around the autonomous vehicle for a range of diff uses from collision avoidance to ego motion tracking and motion planning.

The motion planning module i.e the third module makes all the decisions about what actions to take and where to drive based on all of the information provided by the perception and mapping modules. The main output of this module is a safe, efficient and comfortable planned path that moves the vehicle towards its goal.

The planned path is then executed by the fourth module i,e controller. The controller module takes the path and decides on the best steering angle, throttle position, brake pedal position and gear settings to precisely follow the planned path.

The fifth and final module is the system supervisor. It monitors all parts of the software stack as well as the hardware output to make sure all the systems are working as intended.

Environmental Mapping

The three types of environmental maps used by self driving cars are as follows

- Localization point cloud or feature map.
- Occupancy grid map
- Detailed roadmap

The first map i.e feature maps is created using a continuous set of lidar maps or camera image features as the car moves through the environment.

This map is then used in combination with GPS, IMU and wheel odometry by the localization module to accurately estimate the precise position of the vehicle at all times.

The second map is the occupancy grid map. The occupancy grid map also uses a continuous set of LIDAR points to build a map of the environment which indicates the location of all static, or stationary obstacles. This map is used to plan a safe, collision free path for all the autonomous vehicle.

The third and final map is the detailed roadmap. It contains detailed positions for all regulatory elements, regulatory attributes and lane markings. This map is used to plan a path from the current position to the final destination.

Safety Assurance for Autonomous vehicles

The realization about how crucial it is to make self driving cars supersafe.

In lesson 2, standard and regulations + safety case development was discussed and in lesson 3, testing methods, safety frameworks like SOTIF, ASIL were discussed

Kinematic Modeling in 2D

Kinematic Vs Dynamic Modeling

At low speeds, it is often sufficient to look only at kinematic models of vehicles.

Dynamic modeling is more involved, but captures vehicle behaviour more precisely over a wide operating range.

Coordinate frames

Right handed by convention, Inertial frame is fixed usually relative to earth, body frame is attached to the vehicle, origin at vehicle center of gravity or center of rotation whereas sensor frame is attached to sensor, convenient for expressing sensor measurements.

PID Control in Automotive

<https://www.linkedin.com/pulse/mastering-road-pid-control-self-driving-cars-suraj--mmbpf%3FtrackingId=ud7dmbinRPmRvXKDUokihg%253D%253D/?trackingId=ud7dmbinRPmRvXKDUokihg%3D%3D>

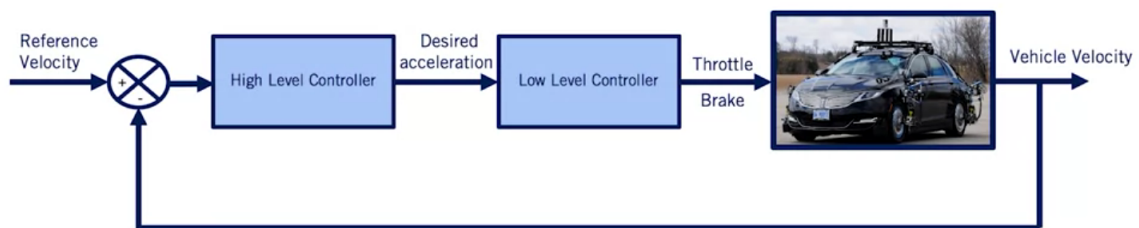
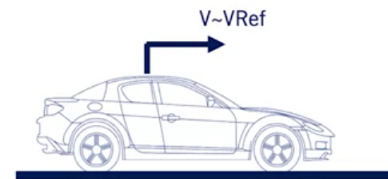
Longitudinal speed control via PID

Cruise Control

Longitudinal Speed Control

Cruise control:

- Speed of the vehicle is controlled (by throttling and braking) to be kept at the reference speed



Vehicle Dynamic Blockset by Matlab for simulation of driving systems in 3D controlled environments

<https://in.mathworks.com/products/vehicle-dynamics.html?requestedDomain=>

https://in.mathworks.com/videos/getting-started-with-the-vehicle-dynamics-blockset-v-dbs--1555080044769.html?s_eid=PSM_15028

Lateral vehicle control

Lateral vehicle control is about keeping your vehicle on the right path, especially during turns or curves. Two crucial methods, Pure Pursuit and Stanley, use clever geometric principles to achieve precise control.

In Pure Pursuit, it's like following a point on your path. Choose a spot ahead, the "target point," and calculate the angle between your car's direction and a line to that point. This angle becomes your guide for steering. The formula for the steering angle (δ) is:

"Steering angle (δ) equals the arctangent of (2 times the car's length (L) times the sine of the angle (α) divided by the lookahead distance (L_f))."

Breaking it down:

- L is your car's length (wheelbase).
- L_f is your lookahead distance.
- α is the angle between your car's direction and the line to the target point.

Now, let's add a twist with Stanley. It's like Pure Pursuit but smarter. Measure how far you are from the ideal path (cross-track error or CTE), and adjust your steering angle based on both your off-course distance and the direction you're facing. This ensures you stay right on track. The formula for the steering angle (δ) in Stanley is:

"Steering angle (δ) equals your car's current direction (θ) plus the arctangent of (the tuning parameter (K) times the off-course distance (e) divided by the speed (v))."

Breaking it down:

- θ is your car's current direction.
- e is your off-course distance (cross-track error).
- v is your speed.
- K is a tuning parameter.

In summary, whether you're driving manually or letting your car take charge, these methods help you stay on course. Pure Pursuit and Stanley make sure your vehicle moves smoothly and safely, making every journey a breeze.

- Model Predictive Control (MPC):

- Steering Wizard: MPC is like a smart steering guide.
- Look Ahead: It predicts the future to decide how your car should steer.
- Car Blueprint: Understands your car's behavior, like how heavy it is and how tires grip.
- Best Steering Path: Solves a problem to find the best steering moves for a smooth and safe ride.
- Step-by-Step: Plans the first move and adjusts as your car moves along.
- Smart Math: Uses simple math to make decisions while considering the rules of the road.
- Adaptable: Handles surprises on the road, making your car drive smartly.

The Kalman filter is unbiased, consistent and Best Linear Unbiased Estimator(BLUE)

- Extended Kalman Filter(EKF)

- Extended Kalman Filter (EKF): An extension of the Kalman Filter designed for nonlinear systems.

- Linearization: Handles nonlinearities by linearizing the system dynamics and measurement equations around the current state estimate. Linearizing a system just means picking some operating point a , and finding a linear approximation to the nonlinear function in the

neighborhood of a . In two dimensions, this means finding the tangent line to the function f of x , when x equals a .

- **Jacobian Matrix:** Represents the partial derivatives used in linearization, allowing adaptation to changing conditions. Linearizing a system just means picking some operating point a , and finding a linear approximation to the nonlinear function in the neighborhood of a . In two dimensions, this means finding the tangent line to the function f of x , when x equals a . Intuitively, the Jacobian matrix tells you how fast each output of your function is changing along each input dimension.

- **Prediction and Correction Steps:** Similar to the Kalman Filter, with the EKF adapting the linearized equations for nonlinear systems.

- **Example:** If tracking a robot's position with non-linear motion, the EKF accommodates changes in direction or speed by iteratively linearizing and updating predictions based on sensor measurements.

- **Challenges:** Accuracy depends on the quality of linearization, and performance may degrade with highly nonlinear systems.

- **Iterative Refinement:** Continuously refines state estimates, balancing nonlinear complexities with the Kalman framework.

Improved EKF - The Error State Extended Kalman Filter

Previously, we introduced the Extended Kalman Filter, which uses local linearization as a way to allow us to apply the Kalman filter equations to non-linear systems.

In this, we're going to look at a variant of the EKF called the Error-State Extended Kalman Filter, or ES-EKF, which has a couple of nice properties that will come in handy later in the course.

Summary

- **Kalman Filter Basics:**

- Used for estimating the state of a dynamic system from noisy measurements.
- Traditional Kalman Filter assumes linear dynamics and Gaussian noise.

- **Extended Kalman Filter (EKF):**

- Extension of Kalman Filter for non-linear systems.
- Linearizes the system dynamics and measurement equations to make them manageable.

- **Challenges with EKF:**

- Linearization introduces errors, especially for highly non-linear systems.
- Cumulative errors can lead to inaccurate state estimates.

- **Error State Extended Kalman Filter (EKF):**

- Improves upon EKF by explicitly modeling and correcting errors.
- Instead of directly estimating the state, it estimates the errors in the state.

- Key Concepts:
 - Represents the system state as a nominal state and an error state.
 - Estimates the error state, which is then added to the nominal state to get an accurate estimate.
- Advantages:
 - Handles non-linearities more effectively by directly addressing the errors introduced by linearization.
 - Better accuracy over time by continuously updating and correcting error estimates.
- Applications:
 - Widely used in navigation systems, robotics, and other fields where accurate state estimation in non-linear systems is crucial.
- Overall:
 - The Error State Extended Kalman Filter is a refined version of the EKF, addressing its limitations in handling non-linear systems and providing more accurate state estimates by explicitly accounting for errors.

State Estimation and Localization for Self Driving Cars

Localization is the method by which we determine the position and orientation of a vehicle within the world.

If we want to drive autonomously, we certainly need to know where we are. To accomplish this, we can use state estimation. This is the process of computing a physical quantity like position from a set of measurements. Since any real-world measurements will be imprecise, we will develop methods that try to find the best or optimal value given some assumptions about our sensors and the external world. Related to state estimation is the idea of parameter estimation. Unlike a state, which we will define as a physical quantity that changes over time, a parameter is constant over time. Position and orientation are states of a moving vehicle, while the resistance of a particular resistor in the electrical sub-system of a vehicle would be a parameter.

Part 1 : Squared Error Criterion and the method of least squares

In this part, we will cover a common technique in state estimation, the method of least squares. By the end of this week, you'll know a little bit about the history of least squares and you'll learn about the method of ordinary least squares and its cousin, the method of weighted least squares. Then, we'll cover the method of recursive least squares and finally, discuss the link between least squares and the maximum likelihood estimation technique

As per Carl F. Gauss, The most probable value of the unknown quantities will be that in which the sum of the squares of the differences between the actually observed and the computed values multiplied by the numbers that measure the degree of precision is a minimum

Arithmetic mean minimizes method of least square

i.e $\text{sq}(x_1 - v) + \text{sq}(x_2 - v) = \min$
take derivative and push to zero
 $x_1 + x_2 / 2 = v$

Linear Kalman Filter

The Kalman filter remains an important tool to fuse measurements from several sensors to estimate in real-time the state of a robotic system such as a self-driving car.

The kalman filter works as a two stage filter - prediction and filter.

- Concept: The Kalman Filter is a mathematical tool used for estimating the state of a system by combining noisy measurements with a dynamic model.
- Linear System: It assumes the system's dynamics and measurements can be represented by linear equations.
- Prediction Step: Predict the next state based on the previous state and the system dynamics, incorporating uncertainties.
- Correction Step: Update the prediction using actual measurements, adjusting for measurement noise and improving state estimation.
- Covariance Matrix: Represents the uncertainty in both the state prediction and measurements.
- Kalman Gain: Balances the contributions of the prediction and measurement in the correction step.
- Example: Imagine tracking the position of a moving car. Predict its next position based on its current speed and direction, then adjust with GPS measurements, giving more weight to more accurate measurements.
- Iterative Process: Repeatedly predict and correct, refining the state estimate over time.

It's mostly used in Sensor fusion.

Unscented Kalman Filter

- Unscented Kalman Filter (UKF):
 - Developed to address limitations of EKF in handling non-linear systems.
 - Avoids explicit linearization by using a set of carefully chosen sample points.
- Key Features:
 - Uses a set of sigma points to capture the distribution of the state.

- These sigma points are selected to represent the mean and covariance of the current state estimate.
- Propagates these sigma points through the non-linear process model, avoiding the need for linearization.
- Advantages:
 - More accurate than EKF in non-linear scenarios, as it avoids potential errors from linearization.
 - Works well even with highly non-linear systems.
- Sigma Points and Transformations:
 - Sigma points are chosen to capture the statistical properties of the state distribution.
 - The process and measurement models are applied to these sigma points, and their means and covariances are calculated.
- Applications:
 - Widely used in scenarios where EKF might struggle, such as in sensor fusion for autonomous vehicles or robotics.
- Overall:
 - The Unscented Kalman Filter is an enhancement over the Extended Kalman Filter, designed to handle non-linear systems more effectively by using carefully chosen sigma points to represent the state distribution without the need for explicit linearization.

Summary | Nonlinear Kalman Filtering

| | EKF | ES-EKF | UKF |
|----------------------------|----------------------------|-----------------------------|---------------------|
| Operating Principle | Linearization (Full State) | Linearization (Error State) | Unscented Transform |
| Accuracy | Good | Better | Best |
| Jacobians | Required | Required | Not required |
| Speed | Slightly faster | Slightly faster | Slightly slower |

References

https://www.youtube.com/watch?v=mzTEZUNJFKM&ab_channel=SimplyDone

https://cdn.vector.com/cms/content/products/VectorCAST/Events/TechNights/CANalyzer_QuickStart_Apr19_Local.pdf

https://www.youtube.com/watch?v=tbhK4uA4REE&t=78s&ab_channel=AutomotiveEngineering

<https://www.youtube.com/watch?v=fiMw3o4U-n0>

<https://youtube.com/playlist?list=PLvRkgRDxp5cqQE50te4IAMqggg-AtKbkf&si=fyl6-UB89R4AWY28>

<https://www.youtube.com/watch?v=8QM7oin8P4E>

https://www.youtube.com/watch?v=T6xrJv9SRLI&ab_channel=EmmersiveInfotech

https://www.youtube.com/watch?v=fPjOWekzeGI&ab_channel=CARinfo3d%28En%29

https://www.youtube.com/watch?v=_Ewdksjh7zU&ab_channel=SareaHA

<https://www.mathworks.com/videos/introduction-to-simulink-quadcopter-simulation-and-control-100476.html>

https://www.youtube.com/watch?v=hGcGPUqB67Q&list=PLPNM6NzYyzYqMYNc5e4_xip-yEu1jiVrr

<https://in.mathworks.com/solutions/automotive/electric-vehicle.html>

<https://in.mathworks.com/help/autoblks/ug/explore-the-electric-vehicle-reference-application.html>

<https://caradas.com/understanding-adas-lane-keep-assist/>

https://carla.readthedocs.io/en/latest/python_api/

https://www.sae.org/standards/content/j3016_201806/

<https://www.cvlibs.net/datasets/kitti/>

http://wavelab.uwaterloo.ca/sharedata/ME597/ME597_Lecture_Slides/ME597-4-Measurement.pdf

<https://repository.tudelft.nl/islandora/object/uuid:2ae44ea2-e5e9-455c-8481-8284f8494e4e/datastream/OBJ/download>

Lanelets: <https://ieeexplore.ieee.org/abstract/document/6856487>

<https://www.iso.org/standard/68383.html>

<https://www.iso.org/standard/70939.html>

<https://www.damtp.cam.ac.uk/user/tong/dynamics/clas.pdf>

<https://publications.lib.chalmers.se/records/fulltext/244369/244369.pdf>

https://link.springer.com/chapter/10.1007/978-1-4614-1433-9_4

https://d3c33hcgiewev3.cloudfront.net/vjVDuyDXEemj-RKX93anOA_be4b8ab020d711e9a1134155eeec67a0_Lateral-Vehicle-Dynamics.pdf?Expires=1700611200&Signature=fUJVKyfHk0LEyDYC1VzAPXgSX71zBqVcwo2rhXUpbafCYol4OS2-UCQB9r8YpHG1HCFReXCUNvnRY3ORoyAAD3gmtSobD2L0JFyb5HcprTHWY2KD8qYqJ0HUa~7jGcck~E2~LzbJYcy39qivBYDZEm0s2U3pTj7UlrzzAT~i59o_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A

https://d3c33hcgiewev3.cloudfront.net/fdEeaiDvEem3Cw5hhdQCGg_7e168b3020ef11e99a25a9f602841cc7_Feedforward-speed-control.pdf?Expires=1700784000&Signature=H1zoDwGc2W~ZXWSLNiejRUhguLn7GBHPSurphZXUaHkQVwoT3dqYlbHdOJyy31wDf7sss8tem5doGgFEdqvFDeOCtoTmx6uy4JEC58YdwHDTbIDWGdq6jequs7fHpA7AI9gArXcEGonbIOQfLKTWirLdYUVTdzzXpLAW4NOpbw_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A

<https://hal.science/hal-01690792/document>

https://d3c33hcgiewev3.cloudfront.net/Tjbr4CDxEem9_wqPhE1wLA_4e4bfa0020f111e990ff73ab14e458cc_Automatic_Steering_Methods_for_Autonomous_Automobile_Path_Tracking.pdf?Expires=1700784000&Signature=FaNm~DJ9ij1Gg0rXxjsSKe70lpqt72s~2ZvedP8kzpzJNprR0~tma6cEXGSecj0frWcsfyzM-P7EABdQtri6wqvYuW9qMF7C-omDwfIOvFMOTzhRpD95v8AxTp9YPdvXJBAXlZ581KAhwZ2W4lhU1yJiEDMe3uSRke0y5lx88_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A

https://d3c33hcgiewev3.cloudfront.net/wM8kyDxEem9_wqPhE1wLA_ff19d14020f111e99167b944be537fd0_Automatic_Steering_Methods_for_Autonomous_Automobile_Path_Tracking.pdf?Expires=1700870400&Signature=F5k~N1kY-9L8XKU83AQzaoydANXHHGBpHWRuSLRfobd27FHCtH~Nf19eQx4ln48ABegKfSF2Zayl4~RjbvXPIHKuAeEe-3DASLKztb~K73oQ7BxajfEOCQHyc41-Wku3zQLd0sl37TA7zOKhSzQPgT5rRLNemtoRb93D6xkxpg8_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A

https://d3c33hcgiewev3.cloudfront.net/2tyroyDyEem9HA6xGGaRfg_daf207a020f211e990ff73ab14e458cc_CARLA--An-Open-Urban-Driving-Simulator.pdf?Expires=1700870400&Signature=c9c3UrMe-6mPx1ZJVpaLrNEsLSqBTZ3LbRfcR37v-LIJQ~3JUBxSAm7qJs4CvVvapKmt2fpWBIO1xEFXdnTZBXfrrBE9nl6L7S76iqppLavNYZ7GmzwWPHXudXa6gLDWK99dj8GhN79EZA-OWMqaKNX9PZGsGiag5-T3H2YRG4E_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A

Certificate 1 -

<https://www.coursera.org/account/accomplishments/certificate/M7UZXWGEHPA4>