**Better Approach (my opinion):**

**Using magnetometer:**

In our mobile phones, we can see Compass/Navigator, or in google maps, we can see an arrow symbol that shows where we’re pointing at.



Based on this input we can tag the images.

Which\_Turn():

If the deflection is around +900 :

“Right Turn”

If the deflection is around -900 :

“Left Turn”

If the deflection is around +1800 :

“Right U-Turn”

If the deflection is around -1800 :

“Left U-Turn”

Based on this sensor input we’ll tag the images:

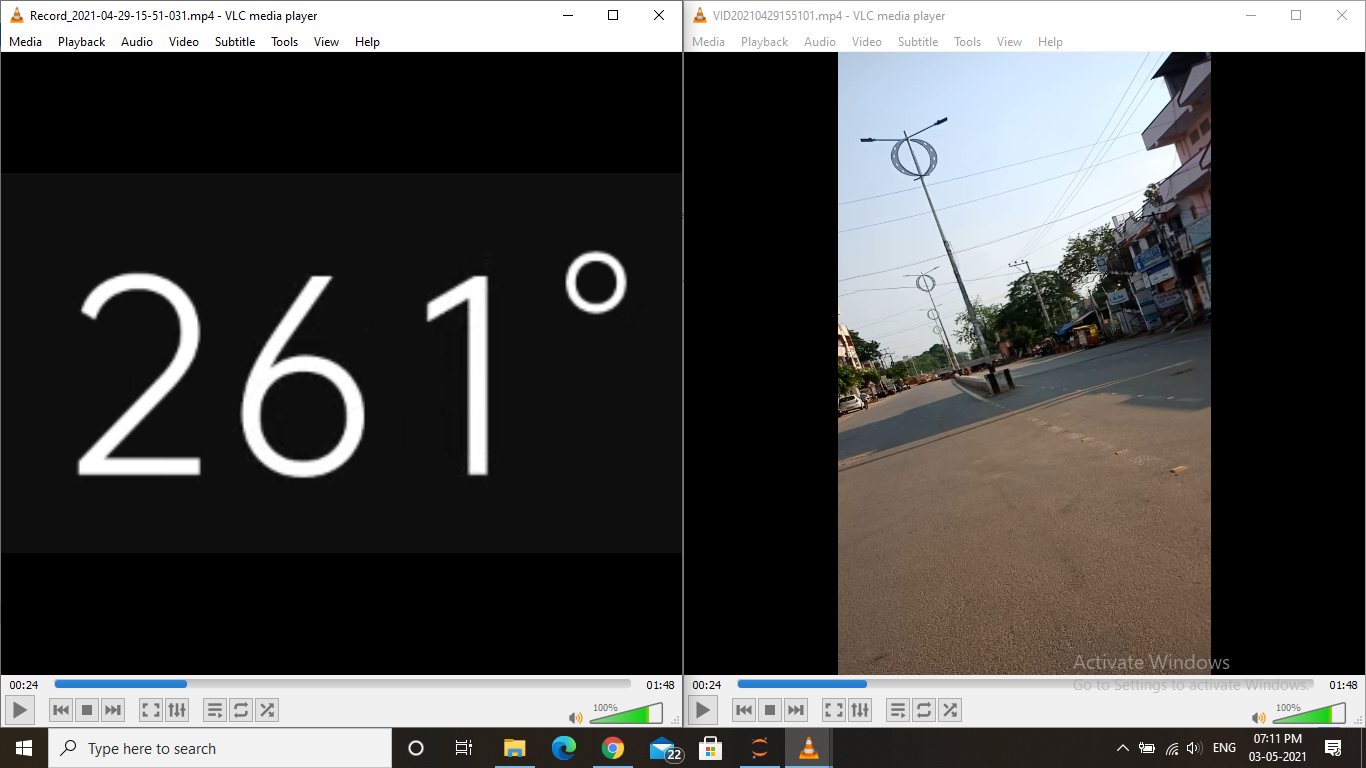
If the deflection starts we’ll start appending images, until it gets stabilized means the car is now traveling in a straight path,

Based on the value returned by the Which\_Turn() we’ll tag those images.

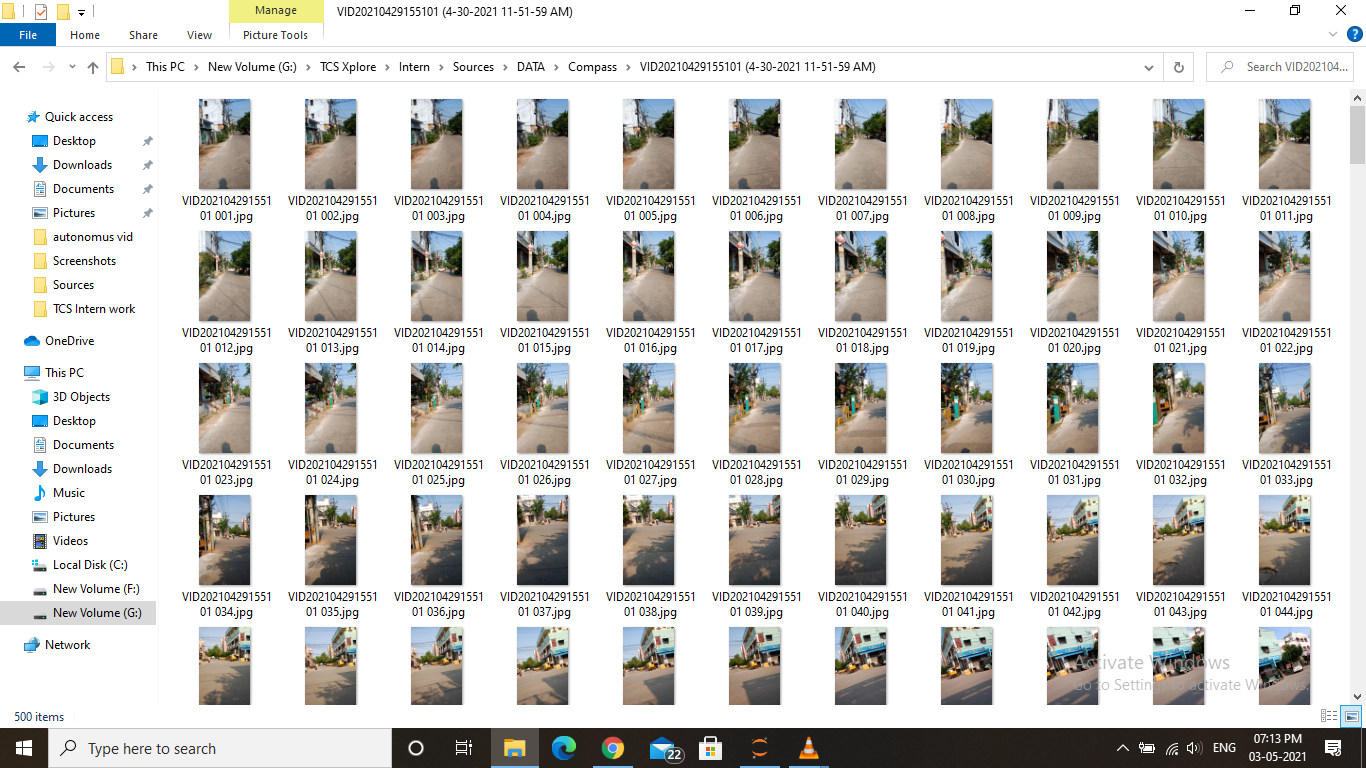
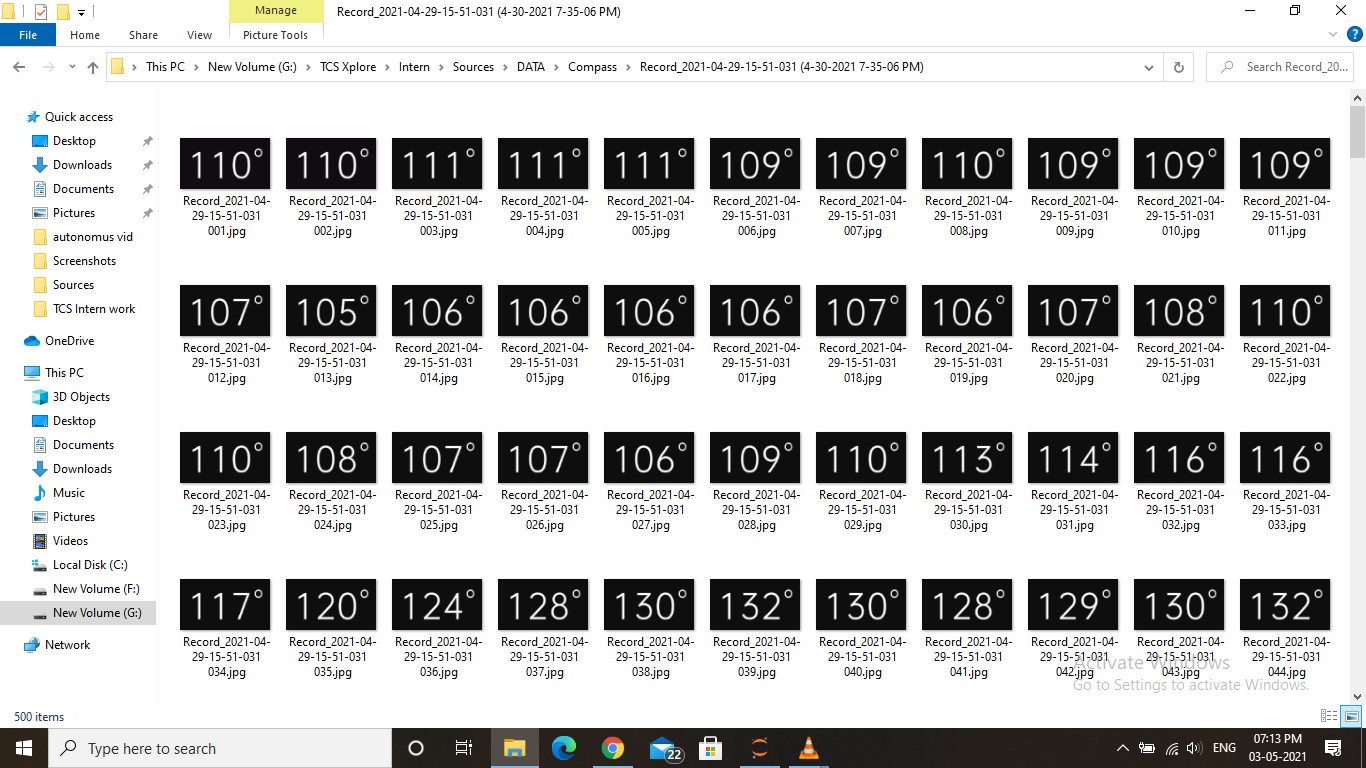
Now we can have the images of the surrounding (car’s view) data when a car takes a right turn/ left turn/ U-turn.

---------------------------------------------------------------------------------------------------------------------

1. I have taken the videos of the car journey (car’s view) parallelly I have started recording the compass readings in my phone (screen recording).

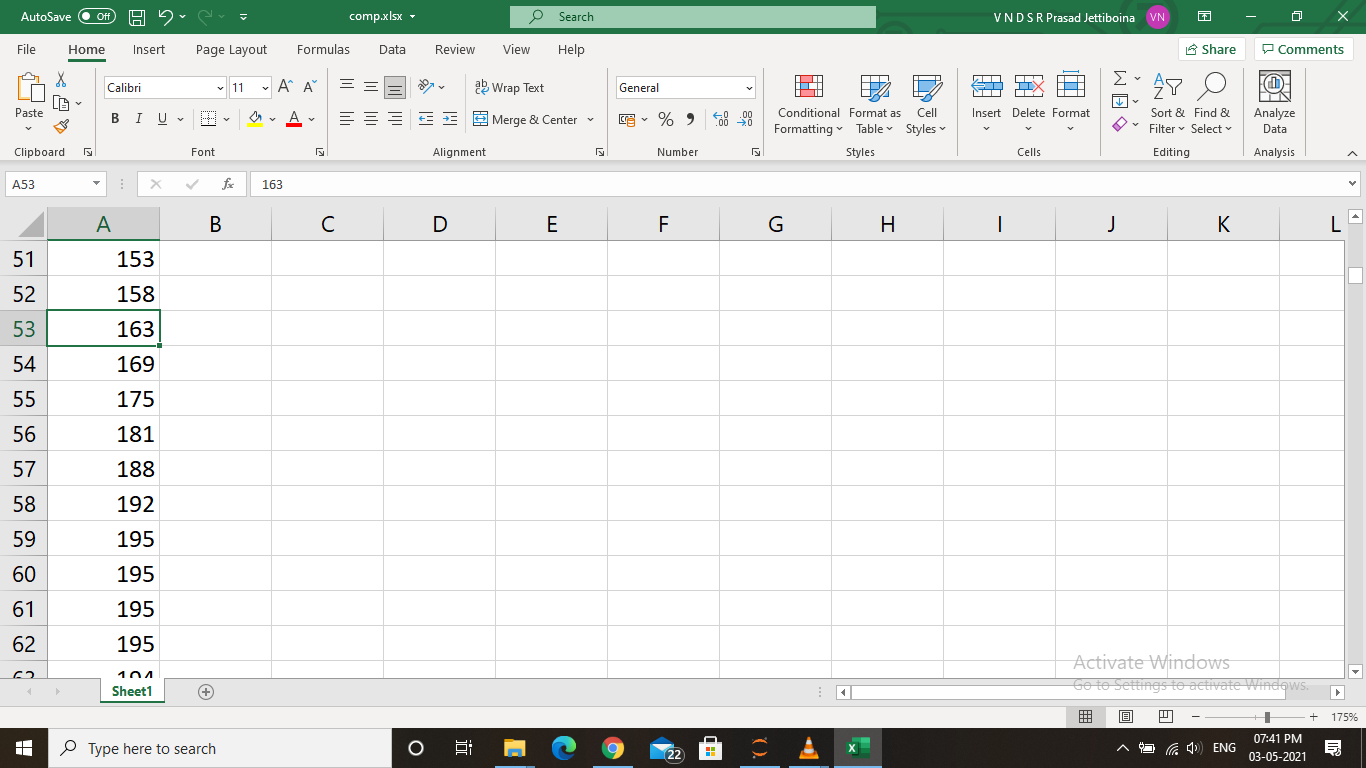


1. Both the videos have captured at the same time and I convert those videos into sets of frames (images)

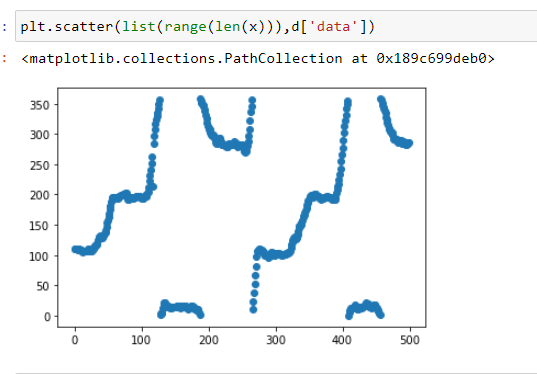


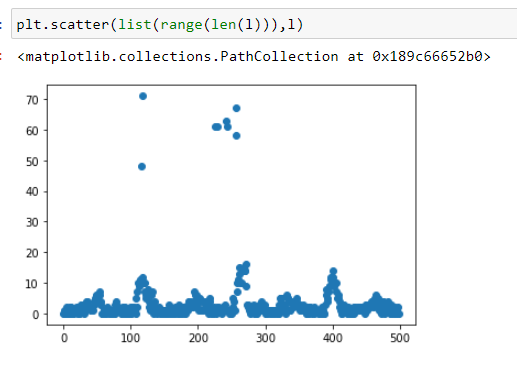
1. I have extracted the compass numerical data by converting compass readings in images into text/numbers using OCR and then organize compass data in excel/CSV format,



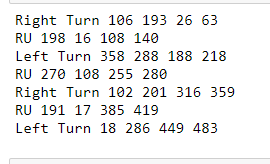


1. Now I did the analysis on that compass data and build logic to determine whether a vehicle takes Right turn / Left Turn / U-Turn





1. Now we can have start points and endpoints where the vehicles take Turns, with these points we can tag images.



**Turn detection by using steering angle:**

**And detect whether it is a right/left or U turn using image comparison:**

When a vehicle is about to take a turn the steering angle changes.

Right turn or Left turn can be detected just by looking into the steering angle.

But what about the U-turn?

1. How much angle and how much time it steered

To take U-turn we need to steer somewhat more angle compared to right/left

And we’re in that turning angle for more time to take U-turn compared to right/left

1. By comparing images:

Attach 3 cameras to the car,

Cam1 to Right side view

Cam 2 to Left side view

Cam3 to Backside view



When a car is about to take right side turn :

Start appending pictures (that we’ll use for the comparison task) of the left side view camera and backside view camera

Similarly When a car is about to take right side turn :

Start appending pictures (that we’ll use for the comparison task) of the right side view camera and backside view camera

Take pictures until the car got stabilized and moving in a straightway.





Let we have captured ‘K’ images i.e from the starting point (when it starts taking turn) till the endpoint (when the car got stabilized and moves in a straight path)

But there is a delay between the left/right side view cam and backside view cam.

The left/right side view cam is placed front and it captures pictures and that same pictures will be captured by the back camera after some time.

Right/Left camera images list [a1,a2,a3,......ak]

Back side camera images list [b1,b2,b3,......bk]

We use image comparison here: <https://www.geeksforgeeks.org/measure-similarity-between-images-using-python-opencv/>

It gives the similarity scores between 2 images

Step1:

Compare b list images with a1:

Whenever we got a very good similarity score we consider that as a starting point

Let say that is at bx

So, that x is the drift/delay

We’ll remove those x images from the backside cam list from the start.

Left/right side cam from the end.

Step2:

Now

left/right [a1,a2,a3,......,~~ax,a(x+1),a(x+2),.....ak~~]

Back [~~b1,b2,b3,......~~,bx,b(x+1),b(x+2),...bk]

Now we got ‘K-x’ images

left/right a[a1,a2,....,a(k-x)]

Back b[b1,b2,....,b(k-x)]

Now we’ll compare image from a with image from b lists (a[i] with b[i])

a[1] compare with b[1]

a[2] compare with b[2]

.

.

.

a[k-x] compare with b[k-x]

Step3:

if(number of images with good similarity scores > (k-x)//2):

print(“It’s U turn)

else:

print(“It’s Right/Left turn)

**Data sets searching:**

<https://github.com/kenshiro-o/CarND-Behavioral-Cloning-P3>

There is a pretrained model which maps steering wheel angle and images

Using this we can predict the steering angle using regular images.

Simulator for creating datasets

<https://github.com/udacity/self-driving-car-sim>

Udacity datasets:

<https://github.com/udacity/self-driving-car>

All are in torrent format (I extracted but have to browse on how to use them since they are in different format compared to the general pictures of jpeg or png format)

<https://medium.com/udacity/challenge-2-using-deep-learning-to-predict-steering-angles-f42004a36ff3>

#Challenge 2 description in detail

**Approaches:**

1. The combined input of Steering wheel angle and Cameras inputs

A)Based on the steering wheel angle we can assess how much angle a car turns.

B)Consider left and right cameras as a trigger and centre cam as a detector

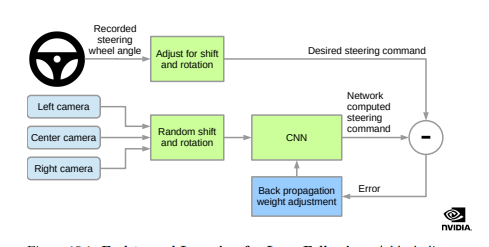
Detect the object which covers the major portion in the left camera and compare it with the center camera view.

If that object (which occupies a major portion and is captured in the left camera) is diminishing from the left camera and capturing in the center camera we can say that the car might take a left turn.

Similar to the right camera-> right turn.

Now A and B are the 2 inputs we have, based on these 2 we can tell whether a car turns right turn or left turn or no turn.

Input (A and B) -> right/left/no turn

****

1. Can detect using the angular velocity of the car

TACHOMETER: An instrument used for the measurement of angular velocity.

GYROSCOPE: is a device used for measuring or maintaining orientation and angular velocity.

<https://www.hindawi.com/journals/ddns/2014/950219/>

<https://www.mdpi.com/1424-8220/17/3/633/htm>

1. Can take a video of one of the front wheel

Using RNNS (sequence of Images data) we can predict the turn.

**Left and right turn detection:**

Since there are cameras, every frame at every instant is captured and we have to analyze a bunch of them for direction detection, say “k” frames as a set.

If the car is going straight then the objects, becomes larger in size(w.r.t the camera and analogous to the human eye,) or in other words the objects will be “zoomed in” in that particular direction of motion while moving and relatively in the opposite direction they will be zoomed out.

Using this, we can detect what direction it is moving based on the frame captured by the camera(exists outliers, that require further analysis).

Take a frame/image. Round the center pixels(a box that contains center pixels),

Say if(p,q) is the centroid then consider a box through the diagonal (p-” constant”,”q+constant”), to (p+”constant”,q-”constant”), which is the principal diagonal of the square. so extract the pixels in this square and use them as reference images, and for the next frames, use this box as a reference image.

Use this image as input and detect this in the next k-1 frames(you can observe the width of the principal diagonal will be increased, for each frame) as the model detects that respective reference image in these new k-1 frames.

So this indicates a direction/ state of being “zoomed in”.

For Turn detection:(note: there will be both right and left cams, so every turn has its own respectance!!)

Now, take an image, from a cam, and consider a 60% portion of an image, starting from “LEFT”, and store it, this 60% part will be the reference image(from that reference frame).

Now use it for the next “p” images and detect & recognise this in them. And also observe the portion of this reference image in those new frames, it gradually decreases hence indicating the turn.

The amount of this ref image in those new frames gradually decreases/ it diminishes in those new frames). Thereby clearly representing turn.

Envisage this, whenever there is turn since angular motion takes place the respective rigid body w.r.t which the motion is being captured will be decreased in size in the view of the moving body(relatively)

Haar cascade

Bounding boxes

YOLO

Doubt:

Sir, should we try this task only using Computer Vision (by analyzing images)

Why can’t we try using sensors data (like Gyroscope kind of sensors)

**RENOVO** - 1st scalable data management solution (10 years)

Auto OEM (Original Equipment manufacturer) cloud

**What it does:**

All formats of data (cam, sensor, vehicle) -> available to developers

All data ->indexing (classification) into sensor data...vehicle data….then

->Enrichment (using ML models) like diff objects...scenarios...conditions

-> Build correlations for particular extractions.

**Why it does:**

Autonomous vehicle data-> tbs/day

Processing and Managing-> Time consuming and error prone (difficult to collect properly eg: blurred images..errors in sensors etc., and handle)

**How it does:**

5 steps



**Ingest**: Car moves data gathered -> Comes to hubs -> Data collected from drives-> Uploaded to cloud

**Index**: Classifying the type of input collected into sensor data, vehicle data cam data….

**Enrich**: Using ML models specific classification

Objects: pedestrians, other vehicles etc.

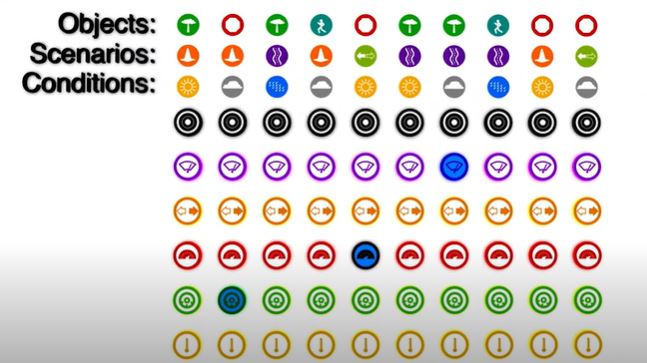
Scenarios: Right turn, Slow down, Sudden speed increment etc

Conditions: Day night, Climatic conditions etc

With these specific classifications we can do correlations between them

With these inputs we can get proper insights on how one feature effects other and the relation

With these we can extract data based on the the situation we want



**Search**: Extract data based on the query

Eg: Unprotected left turns in rains

It will narrow down the data

First it filters out the data based on condition : Weather condition :: Rainy

Then Scenario: Turn:: Left turn

Then It collected data based on the logs: if there is any error while driving it mean its not correct or else if there is any object at the instant when we turn left  
The data having these will be filtered out.

**Delivery**:

By this they can deliver the data needed by the users

Demand data will be sent first

Like priority based delivery

Rest data will be stored safely.

ROS- Robot Operating System

Open source middleware

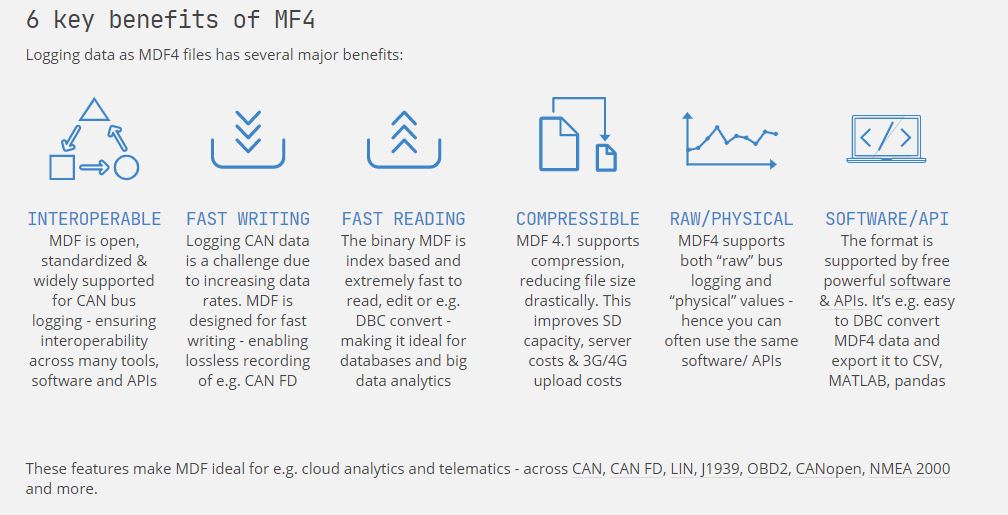
All formats of data (collecting datasets, testing algorithms etc,) combined into single file and it is called as bag files.

The rosbag package provides a command line tool for working with bags as well as code APIs for reading/ writing bags in C++ and Python

The rosbag command line tool and code APIs are stable.

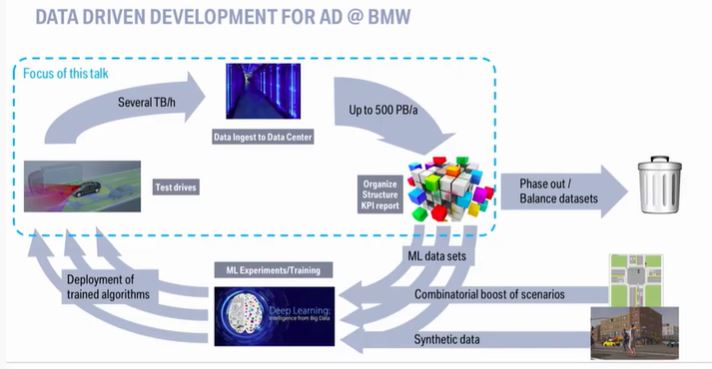
MDF4- Measurement Data format version 4

The ASAM MDF (Measurement Data Format) is a binary file format for recording e.g. [CAN](https://www.csselectronics.com/screen/page/simple-intro-to-can-bus), [CAN FD](https://www.csselectronics.com/screen/page/can-fd-flexible-data-rate-intro) and [LIN bus](https://www.csselectronics.com/screen/page/lin-bus-protocol-intro-basics) data. Today, the MDF format is the industry standard - ensuring interoperability across many CAN tools.



<https://www.asam.net/standards/detail/mdf/wiki/> More data for reference (in future).

* BMW goal - collect 500 petabytes per year
* Used **Hadoop**: software framework for distributed storage and processing of big data using the MapReduce programming model. **MDF4** file format.



* Why MDF4: supports fast reading/writing of data because it stores data in a binary format and the system also expects data in that format only. But, we can't use split and transfer so they set a limit of 2GB.
* There is a logger attached to the car that will read the data from all the available sensors (23 sensors they mentioned) and there is an SSD attached in the logger and the data stored in that SSD.
* The SSD consists of 2GB data in an unsorted format: Header file-> many small mdf4 files and they have links with the header and every link refers to the data block which stores 99.9% data. Upto this -- Data Gathering
* Now data collection. Transmitting data to the device is done through UDP.
* Bigger data will be splitted into multiple smaller parts - fragmentation.

Then that Fragment data after transmission will be defragmented using Reduce by keys or grouping but with this process they face network shuffling issues.(some data will miss)

So, they build their own fragmentation pipeline.(Read-> Defragment-> Decode-> Store)

* Then the data will be sent to users based on their requests, But, there are many types of Users: ML engineer, Software Engineer, Robotics Engineer. Etc,. different users requires different data so they need to extract the required data in a given interval of time.
* So, they will transfer the knowledge not the data (like insights from the query given)