DST Geospatial Hackathon

CONTACTLESS ENFORCEMENT OF CYCLE/PRIORITY LANES TO PROMOTE ACTIVE MOBILITY IN INDIA AS SAFE AS POSSIBLE.

Real-time vehicle tracking system based on surveillance videos

Hackathon Stage-Idea Submission:

1. A detailed description of the scope of the solution

In the case of CCTV, as its location is fixed and its hardware performance is superb, it is highly effective for the monitoring of car movements in a predefined area.

A number of video surveillance devices used for traffic condition analysis, people identification , and event detection. As they are dealing with a single video source, the analysis results were limited, and combining the results from separate video sources would be both time-consuming and labor-intensive. Vehicle tracking based on surveillance videos suffers from the same problem.

To solve this, I propose a **Kafka-based real-time vehicle tracking system** that can collect data from different video sources, extract relevant features from vehicles for monitoring, and integrate them in a consistent manner.

The effectiveness of vehicle tracking relies on diverse information, such as
- plate number
- time
- place
- direction

collected from numerous different places.

Fortunately, modern CCTV cameras provide diverse metadata including global positioning system (GPS) and time-stamping.

In addition, plate number and moving direction can be easily detected from the captured images using popular image processing or machine learning techniques.

————-xx————-xx————
Real-time vehicle tracking system based on surveillance videos from diverse devices including CCTV, dashboard cameras, and drones.

For scalability and fault tolerance, our system is to be built on a distributed processing framework and comprises a Frame Distributor, a Feature Extractor, and an Information Manager.

- The **Frame Distributor** is responsible for distributing the video frames from various devices to the processing nodes.

- The **Feature Extractor** extracts principal vehicle features such as plate number, location, and time from each frame.

- The **Information Manager** stores all the features into a database and handles user requests by collecting relevant information from the feature database.

To illustrate the effectiveness of our proposed system, we implemented a prototype system and performed a number of experiments. I report some of the results mentioned below.

———-xx———-xx———
An integrated vehicle tracking system, IVATS, based on Kafka and HBase was designed and developed.

Our system could assign a significant number of frames from diverse video sources, such as CCTVs cameras, to processing nodes using Apache Kafka.

Primary vehicle features such as plate number, time, and location data were extracted accurately from the frames using image and metadata processing.

The feature data were stored in HBase clusters and retrieved for query processing.

For effective query processing, an indexing structure based on R-Tree was proposed.

In the experiments, I demonstrate that our system can handle diverse user queries, including vehicle tracking and traffic congestion, efficiently.

Based on the data distribution, storage structure, Rowkey design, and indexing structure, our system can effectively handle real-time requirements of vehicle tracking applications.
———-xx———-xx———

2. Improvements in functional PoC over Phase 1 submission

Drawbacks:
It is not easy to implement for a number of reasons.

Firstly, the system should have sufficient storage and processing capacity to handle the big data involved.
- The volume of data generated from the video devices in real time is significant.
- Therefore, the system should be sufficiently fast to avoid any data accumulation inside the node, otherwise, all the nodes in the system could experience a memory shortage and, in the worst case, the entire system might stop.
- To overcome this problem, a distributed processing platform can be used.

Secondly, the system should have a fault tolerance ability that is essential for the system to provide accurate and complete car tracking information.
- This means that when a node fault or transmission fault occurs, the system should be able to recover from the fault.

Thirdly, precise and fast image processing methods should be supported to efficiently extract all the critical information about the cars in the frames.

Finally, to answer user queries promptly, the system should have methods for managing a significant amount of data efficiently, including an index structure for query processing.

We investigated methods for recognising or tracking automobiles from video frames.
As numerous diverse vehicle feature extraction methods have been developed, complete systems for vehicle tracking have also been proposed.
- A system that collected the frames from surveillance videos, recognised the license plate, and provided the results to the user that consequently enabled remote monitoring.
- One proposed a video surveillance system in a cloud environment. Because of the automatic license plate recognition engine and the cloud environment, their system was able to cover wide areas and visualised the detection results using Google Maps.

We investigated frameworks for real-time distributed processing.
- Hadoop processes data in batches, HIPI (Hadoop image processing interface) is not appropriate for real-time processing. In addition, the Hadoop distributed file system uses a random-access approach to disks, which induces an amount of delay in accessing the data in a file system.

- Spark is one of the fastest frameworks for suitable for distributed processing. However, it has a critical weakness with insufficient memory. When it encounters insufficient memory, the processing speed of the system decreases rapidly and could even result in the data in the memory being lost.
- The abovementioned disadvantages of the two popular frameworks could be *significant stumbling blocks for real-time vehicle tracking*.

Therefore, ***I focused on Kafka***, which is a platform developed for real-time message transmission. Kafka comprises three parts: Producer, Consumer, and Broker.

1. Producer generates data and sends them to the Broker.
2. In Broker, the data are classified according to their topics and replicated for in- creased reliability.
3. Consumer, a processing part, obtains the data from Broker each time it finishes tasks.

Kafka has the following properties:

1. It stores temporary data in its own file system
2. Each Consumer schedules its own task
3. Saving data in the storage nodes enables Kafka to recover the data without data loss when an error occurs.

Although memory-based structures are typically faster than disk-based structures, the speed of data access in Kafka is comparable to that of memory-based structures because of efficient disk usage.

4. The second property indicates that a Kafka node need not wait for a job schedule from the cluster master. Therefore, bottleneck problems caused by scheduling can be avoided and the communication between nodes can be decreased, reducing the network load.

Because of these properties, *Kafka can be a suitable framework in a real-time environment, and it was validated.*

We investigated distributed databases, and index structures for HBase.

Due to popularity of distributed processing frameworks, distributed databases like Cassandra, MongoDB, and HBase are attracting increasing attention for managing large volumes of data.

- MongoDB is another open-source cross-platform NoSQL database program & is a categorized as document-based database.
- While Cassandra and HBase are column-based databases. Compared to other database management systems, it is easy to use and can process a number of query conditions.

HBase is an open-source, non-relational database based on Hadoop and Google Bigtable.

1. It ensures data consistency and provides fault tolerance.
2. HBase is based on Hadoop, it is easy to use MapReduce when implementing the various query processing methods.
3. For this reason, we use HBase for data management.

In HBase, a data tuple is called a row and data are managed in tables that are divided into small row sets known as region.

- Therefore, MapReduce performs data processing in the unit of region.
- Except for Rowkey, which is an identifier of a row, the attributes of a table are not indexed. This means that HBase must access all stored data to answer user queries.
- Therefore, data retrieval takes significantly longer than data insertion and the query processing time increases rapidly as the volume of stored data increases.

To overcome this problem, a number of studies have proposed the index structure, specifically for geometric information.

A popular index structure for geometric information is R-tree-based indexing scheme for trajectory data of cars in a distributed environment. However, we optimised the index structure by indexing GPS data *using Quad-tree-based indexing scheme and Hilbert space-filling curves* ***to obtain improved performance***.

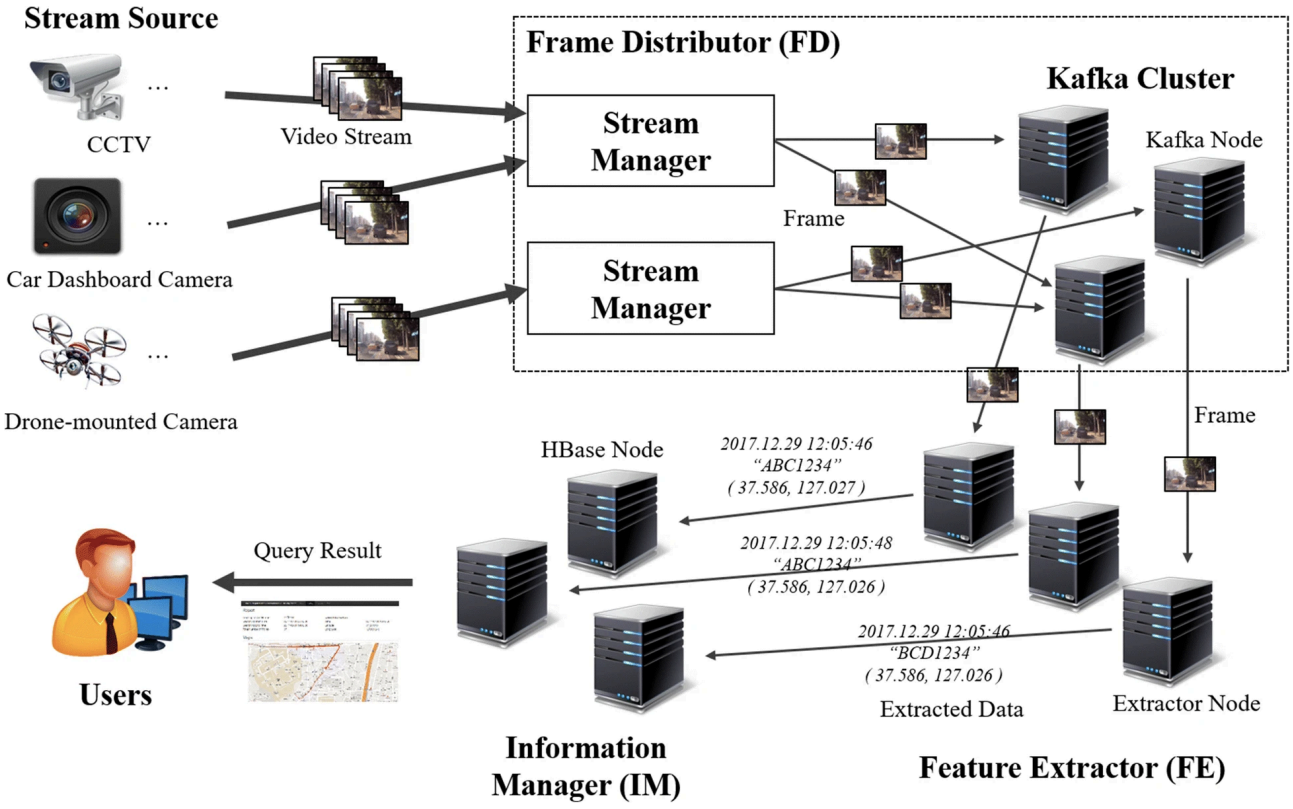
3. What are the technologies used in the functional PoC?

I propose a real-time vehicle tracking system IVATS (Integrated Video-based Automobile Tracking System) that can collect video big data, extract and store principal vehicle features, and process user queries in a real-time environment.

Our proposed system comprises three components: Frame Distributor (FD), Feature Extractor (FE), and Information Manager (IM).

- **FD** is responsible for reliable distribution of the video frames from a number of devices to the processing nodes. *(The role of the FD is to assign a significant amount of frames from numerous video sources to processing nodes using Apache Kafka)*
- **FE** extracts diverse vehicle features such as plate number, time, and location from each frame. *(Each node in the FE extracts principal vehicle features such as plate number, time, and location from the frame and transfers them to the IM)*
- **IM** stores all the extracted feature data, processes user queries by collecting relevant information from the feature database, and presents the query results to the user. (The IM that is built on HBase clusters is responsible for storing all the extracted features, constructing index structures for them, and retrieving all the relevant data to answer user queries)

Overall structure of IVATS:



Hackathon Stage-UI/Design/Code Submission:

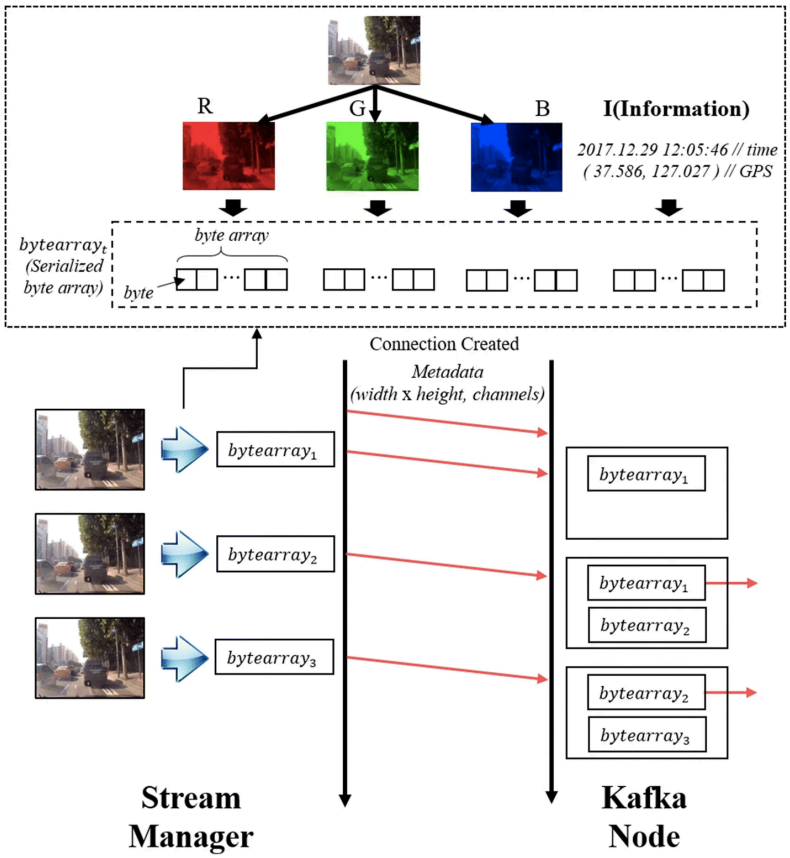
1. DB structure & explanation of the key features and algorithms:

Frame distributor:

i. The role of FD is transferring frames from diverse video sources to FE for feature extraction.
ii. One critical task of this module for accurate car tracking is reliable data transfer.
iii. As frames are generated from diverse video devices in real time, the data volume is significant and reliable data transfer is not trivial.

iv. In addition, the frames must be processed rapidly, or the entire system could stop because of buffer overflow.
v. To prevent this, we used a Kafka cluster for frame distribution and storage.
vi. Stream Manager (SM) in FD divides the data from the stream channels into frames and transmits them to the Kafka cluster.
vii. The stream channels can be directly connected to a video device or receive data from a remote video device using a real-time protocol, such as the real-time streaming protocol.
viii. SM is responsible for either one stream channel or multiple stream channels, depending on its capacity. Each frame has three RGB (red, green, and blue) color channels, and SM transforms the frame into a byte array by serialization.
ix. The serialized byte array will be restored to its original form in FE for image processing.
x. Therefore, SM must provide the metadata, including the width, height, and the number of color channels, to the FE node through the Kafka cluster.
xi. However, sending this information each time can result in a significant overhead.
xii. To reduce this over- head, the frame metadata for reconstructing frames is only sent once when the connection between SM and the Kafka cluster is created, and the Kafka cluster records this information for the FE.
xiii. Frame-related metadata such as GPS data and time are sent in a byte array. As we are unable to get the exact data of the captured automobile, the GPS data and time refer to the location and time of capture of the video devices, and not the automobile.
xiv. For instance, the GPS data in the CCTV video is identical to the vehicle data in the video. However, this is significant for vehicle tracking in an actual environment.

Transmission steps between Stream Manager and a node in a Kafka cluster:



As soon as a connection is made between SM and a node, SM sends the metadata needed for image reconstruction and starts to send frames by converting each frame into a serialized byte array, bytearray$_t$ in the figure, at time t. bytearray$_t$ consists of RGB and I, which are the byte arrays of three color channels and the metadata respectively. bytearray$_t$ at a Kafka node will be sent to a node in the next step when the node requires the data.

xv. While the FD is responsible for preparing frames to send and distributing them, the Kafka cluster actually connects the stream channels to the FE.
xvi. The Kafka cluster receives a byte array from SM and forwards it to the FE.
xvii. The frame then becomes located in a Kafka topic, which is a message queue in Kafka. The FE nodes take a frame from Kafka topics each time they finish a task.
xviii. Kafka is superior to other frame- works such as Hadoop and Spark in a number of aspects.

1. Firstly, the nodes in a Kafka cluster store the data in their local file system. Therefore, the data being transmitted in the Kafka cluster are always protected from data loss, even when a fault occurs.

- In addition, because of this property, Kafka can play a role as a data buffer and restore the original data without any loss when a node is at a standstill.
- The data can be deleted only when the data duration exceeds some predefined threshold.

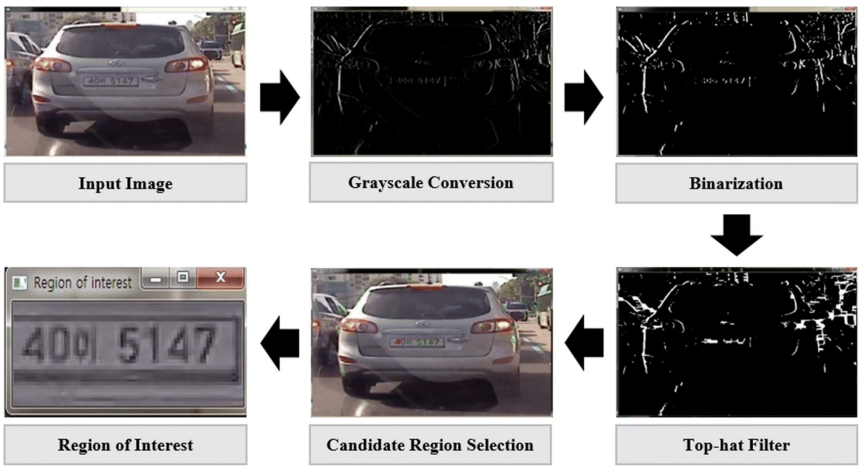
2. Secondly, similar to other frameworks, the Kafka Cluster also adopts replication for the situation when a number of nodes stop abruptly. These two properties give Kafka high availability.
3. Thirdly, Kafka nodes can schedule their own tasks. Because of this, Kafka does not experience the overhead problems that occur when the master node must schedule all the slave nodes.
4. Fourthly, Kafka nodes can be operated asynchronously. Synchronous operations are typically safe from errors caused by an incorrect processing order, although they are slower than asynchronous operations.

xix. However, Kafka supports asynchronous operations without errors and can accommodate data rapidly.
xx. Finally, an FE node takes the frames to process as soon as it finishes its current task. This policy frees Kafka from having to monitor the task of nodes for job scheduling.

Frame extractor:

i. Frame Extractor extracts diverse features for car tracking from the frames in the Kafka cluster nodes through image and metadata processing.
ii. FE nodes obtain a frame from the FD whenever they complete their current task.
iii. When an FE node is finished with a frame, the extracted features are stored in IM together with the GPS data and timestamp that were sent with the frame.
iv. Although for simplicity, the plate number is used as the only feature of the vehicle in this study; additional features such as color, vehicle type, and moving direction can be used for more versatile car tracking.
v. The FE node receives frames in the form of both byte arrays and their metadata for restoration such as width, height, and the number of color channels.
vi. Based on this metadata, the node transforms the byte arrays into images and then performs feature extraction.

Steps for identifying license plate:



1) The first step in feature extraction is the removal of noise in the frame by transforming the image from RGB scale to grayscale, followed by Gaussian blurring.

(i) We then choose the **region of interest** (ROI) that contains the plate number of a vehicle.
(ii) To do that, the grayscale image is converted into a binary image using Otsu's method that reduces a grayscale image into a binary image, considering that the intra-class variance of two classes, white and black, should be minimal.
(iii) When an optimal threshold is found, the frame is converted into a binary image using this threshold. For the binary image, we apply the top-hat filter that is one of the morphology operations.
(iv) After that, we calculate candidate ROI regions by using templates. Based on various license plate templates, we search which region in the image contains a license plate. The selected regions become ROI.
(v) Now, we are ready to identify characters in the license plate region. Based on the identification, we can confirm that the region is a license plate.
(vi) To identify characters, pixels in the region of a license plate should be split into a number of areas that could possibly indicate single characters.
(vii) However, the size of the ROI could be different depending on the distance and angle between the video device and the license plate. We use the affine transformation to solve this problem. This is a mapping function between two affine spaces with points, straight lines, and planes retained.
(viii) Because of its property that maintains ratios of distances between points lying on a straight line, it can make ROIs have similar sizes and reduce the distortion introduced during the transformation of an image.

2) The next step for the plate number identification is to divide ROIs into character regions by projecting the pixels in each ROI onto a horizontal axis and counting the foreground pixels in the axis.

- Based on this, we can construct a pixel histogram and use it to find character regions.
- If an interval has any pixels, this interval is considered as a character region; otherwise, the interval is deemed as a space between two adjacent characters.

After all, if a ROI has the required number of character regions, it really corresponds to a license plate. The character in each character region can be read using Tesseract-OCR: an open-source OCR library.

Information manager:
All the vehicle information including the license plate number, GPS data, and timestamp should be stored and processed efficiently to support diverse applications and user requests.

For this, we use HBase, an open-source, non-relational database based on Hadoop and Google Bigtable. An index structure is required for the spatial data as spatial data should be continually retrieved for vehicle tracking.

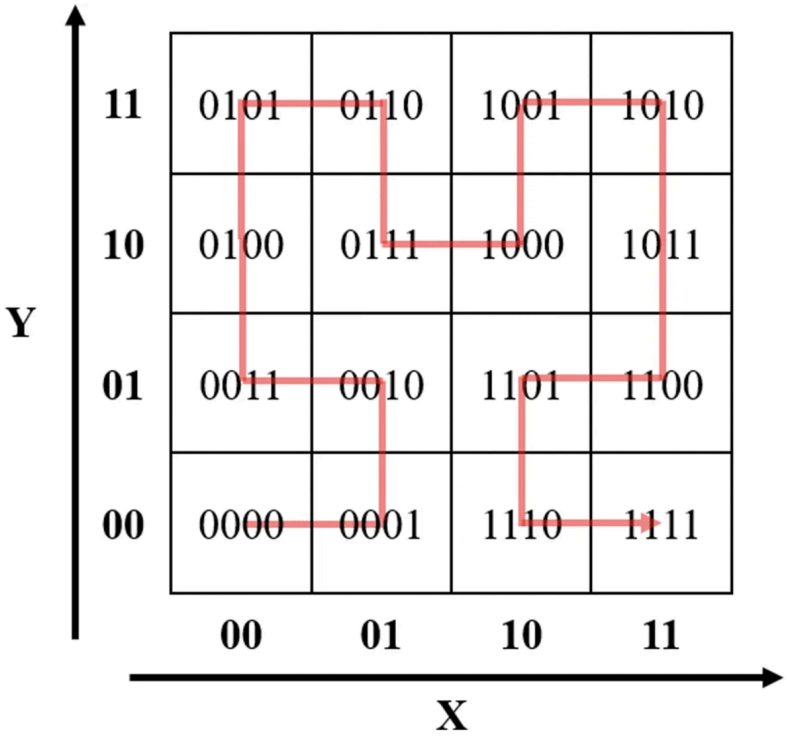
Sample HBase table schema for storing license plate number, time, and GPS data:

	ColumnFamily	
Rowkey	Latitude	Longitude
117ㅏ1111_1488960532177	37.58268	127.02621
117ㅏ1111_1488960532183	37.58274	127.02672
117ㅏ1111_1488960532199	37.58307	127.02813

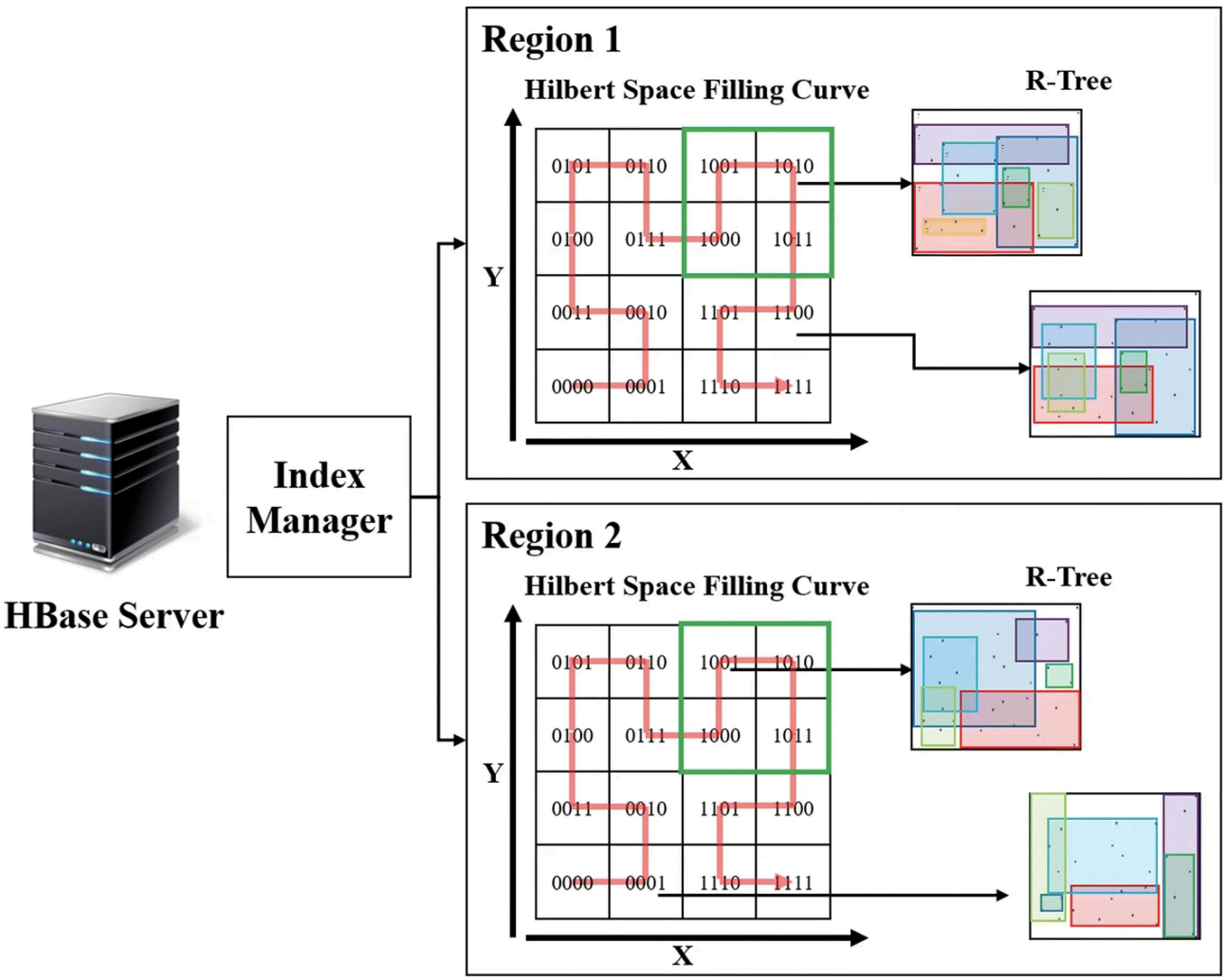
- We make the Rowkey of the table by combining the plate number and the time when the frame was captured and the other attributes are tied to one ColumnFamily that is a set of attributes in HBase.
- If additional attributes need to be stored, they will be contained in ColumnFamily in the current table, resulting in ColumnFamily extension.
- This enables HBase to simply append new rows instead of up- dating the previous row.
- The advantage of this is that there is no update overhead that is typically introduced in version management.
- In addition, searching for a specific vehicle data becomes considerably quicker as the data of an identical vehicle converges.

- In addition, considering that Rowkey is saved in each attribute, there is no requirement to create attribute columns for plate number and time and storage space could be decreased.
- However, HBase cannot process queries with attributes other than Rowkey. Fortunately, this restriction can be overcome by using an index structure.
- Specifically, as latitude and longitude are two primary attributes in our system for representing geographical information, we can construct an index structure for the two attributes.
- We revised the method to meet our requirements. The index structure comprises two parts: R-tree and Hilbert space-filling curve.
- R-tree is one of the most popular data structures for spatial data indexing and has a number of versions. The basis of R-tree is a rectangle binding a number of data points and minimizing its boundaries that is known as the minimum bounding rectangle.
- The significant attribute of R-tree is that it is a height-balanced structure and provides stable performance regardless of data location.
- The greater the volume of data stored, the greater the height of the R-tree, and the longer the query response time becomes.
- To solve this problem, we spread R-tree using a space-filling curve.
- A space-filling curve is a curve whose range contains the entire two-dimensional (2D) square. It is used to convert 2D data into one-dimensional (1D) data.
- The Hilbert space-filling curve, which is a fractal space-filling curve, exhibits the best performance in preserving locality. Because of this property, we use Hilbert space-filling curves in this study to map the spatial data (X, Y) to a 1D point.
- The total length of the Hilbert space-filling curve varies according to its order.

<u>The second order Hilbert space-filling curve:</u>



- The range of X and Y is from 0 to 3, respectively, in this case.
- The value in each rectangle is the value of the Hilbert curve corresponding to X and Y and the red line indicates the Hilbert curve line.
- In this curve, (3, 2) is converted to (1011) and other points, (0, 0), (3, 3), and (3, 1) are converted to (0000), (1010), and (1100), respectively.



Overall Index structure for GPS Data and Hilbert space-filling curve of order 2:

- Each node in HBase has an IndexManager that manages the index of all regions in the node.
- The IndexManager uses the two methods; Hilbert space-filling curve and R-tree. A Hilbert space-filling curve divides all possible spaces.

However, in the real application, **Hilbert space-filling curve of order 7** is used to map actual GPS data.

- As the Hilbert space-filling curve does not use floating-point values, GPS data (latitude, longitude) should be converted into integer values.
- We divide all possible regions expressed by (latitude, longitude) to fit the Hilbert space-filling curve and map the regions into the areas in the curve.
- Thus, GPS data are allocated to some integer values of the curve and we use these values.

- Each independent area then has one R-tree whose nodes have latitude, longitude, and the name of the address where the data are stored.
- In order to decrease the overhead of index updates because of the data insertion, we perform index updates only when HBase flushes all the data in the memory to the disk.
- Such flushing occurs just before the amount of data exceeds the memory capacity.

- This lazy update does not affect the performance significantly as the inserted data exists in the memory and data searching remains rapid, even if they are not indexed.

We now describe the detailed steps for processing spatial queries using a simple example.
- When a spatial query covering from (2, 2) to (3, 3) is given, its spatial condition is transformed into the range of a Hilbert space-filling curve from (1000) to (1011).
- This range is indicated as a green rectangle.
- Then, we investigate each region whether an R-tree exists in the area contained in the range. For instance, the IndexManager in the figure has the index of two regions.

i) Region 1 has R-trees in areas (1010) and (1100)
ii) Region 2 has R-trees in areas (0001) and (1001)

- Considering the query range, we can find that region 1 has an R-tree in (1010) and region 2 has an R-tree in (1001).
- If the area has an R-tree, the index structure retrieves the data address that is appropriate for the spatial condition.
- Accordingly, region 1 starts to search the R-tree in (1010) and collects the data addresses, and region 2 does the same for (1001).
- After all data addresses are collected through the R-tree searches, HBase records them as a result for users.

For the complete query result, the data in the memory should be checked because of the lazy update policy of our index structure. Therefore, spatial query results comprise data addresses from both the index and memory searches. For effective browsing of query results, they can be connected by means of a number of visualisation tools.

2. UI screen layout & Submit the code (pdf of file or URL of video)

- Now, I demonstrate the performance of my proposed system, I performed a number of experiments.
- I also show how a number of typical vehicle tracking queries are performed together with the query results.
- In the experiment, the FD was not considered as it is already reported that Kafka can be used as a satisfactory distributed-processing framework.

We consider the following three experiments:
(1) extracting vehicle features
(2) visualizing query results
(3) indexing spatial data

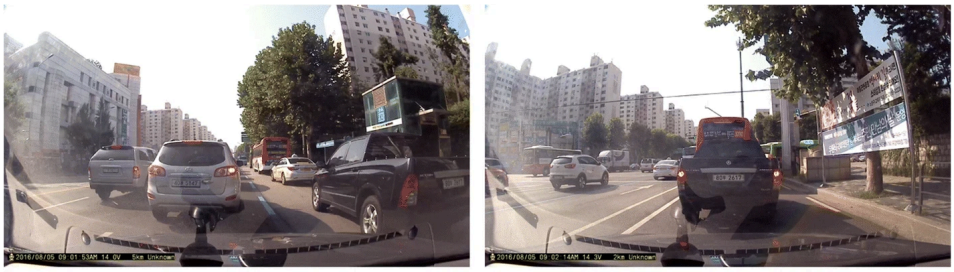
- The experiments were performed on an Intel® CoreTM i7-7700 with a 3.6 GHz processor and 32GB RAM, using virtual machines running Ubuntu 16.04.
- In addition, we used Hadoop version 2.7.3 and HBase version 1.2.4.
- The number of nodes in the HBase cluster was five, of which one was the master and four were slave nodes.
- The data used in the experiments were virtually generated except for the data for the first experiment.

Virtual data was used instead of actual data because the actual data available was insufficient for the needs of the tests.
The virtual data generated for the experiments were 16 million tuples in the HBase table.

Vehicle feature extraction:
Vehicle features of interest are extracted from frames by the FE node.

Feature Extraction: Consists two video frames from Camera & its extracted features:

1. In the features, the plate number is the outcome of the FE node and time, latitude, and longitude are transferred from the FD node with the frame.



(a)

{ “plate” : “40머5147”,
“time” : “2016/08/05 09:01:44”,
“latitude” : “37.637308”,
“longitude” : “127.067590”}

(b)

{ “plate” : “80우2617”,
“time”: “2016/08/05 09:01:54”,
“latitude” : “37.636116”,
“longitude” : “127.068125” }

2. Even though I used the license plate number as a visual feature of a vehicle for simplicity in this work, it is easily extended to cover other visual features such as color, type of car, and direction.

3. The plate number in the car, is the candidate that has the greatest confidence among the extracted plate numbers. In a typical situation, the plate number of the vehicle immediately in front of the camera has the highest confidence.

4. When the plate numbers of other vehicles in the same frame are detected depending on the angle and distance, they can also be used for more comprehensive car tracking.

5. Lastly, all the collected features through FD and FE nodes are sent to IM for storing into the database.

Visualization:

- In the vehicle tracking application, visualization could be an effective way for users to easily understand the query results.

- For this reason, we incorporated simple visualization functions in our system.

- Using these functions, the trajectory of a specific vehicle can be seen by using the data in IVATS, and the other vehicles can be seen sequentially.

- The trajectory of a vehicle is displayed on the map using Google Maps API.

- For example, given a plate number and possibly temporal or spatial condition, the system visualizes all the records that satisfy the query condition in both the spatial and temporal order.

Result of real-time tracking a particular vehicle:

The result consists of two parts.

1. The top part of the result shows the summary of the requested query and the query result including the first and last locations of the vehicle captured by IVATS and the number of times the vehicle was seen.
2. The bottom part shows the trajectory of the vehicle on the map. Black markers on the map indicate the locations where the vehicle was captured.

- In particular, the first and last locations of the tracking are marked in blue and red, respectively. The red line connects all the locations where the vehicle moved along.
- The line represents the trajectory of the vehicle.
- The pop-up window shows the location and time information of the selected marker so that users can browse the trace of the vehicle.
- Our proposed system can handle his kind of user query easily considering our policy for constructing and storing Rowkey for table tuples.

IVATS Home **Tracking** Congestion Grid

Query

Tracking Target Number	11가1111
Searching Start Time	2017-12-01 07:00:00
Searching End Time	2017-12-01 19:00:00

Report

Total Number of Points	37
Start	
Time	2017-12-01 08:01:20
(Latitude, Longitude)	(37.584207, 127.031024)
End	
Time	2017-12-01 18:02:56
(Latitude, Longitude)	(37.584518, 127.031240)

I. Even when the user query has a particular time range or a specific area of a rectangle expressed by two points, our system can easily give an answer to the query.
II. This type of query is highly effective for measuring the real-time location tracking for each particular vehicle or traffic congestion in a specific area.

<u>Traffic evaluation in a specific region:</u>

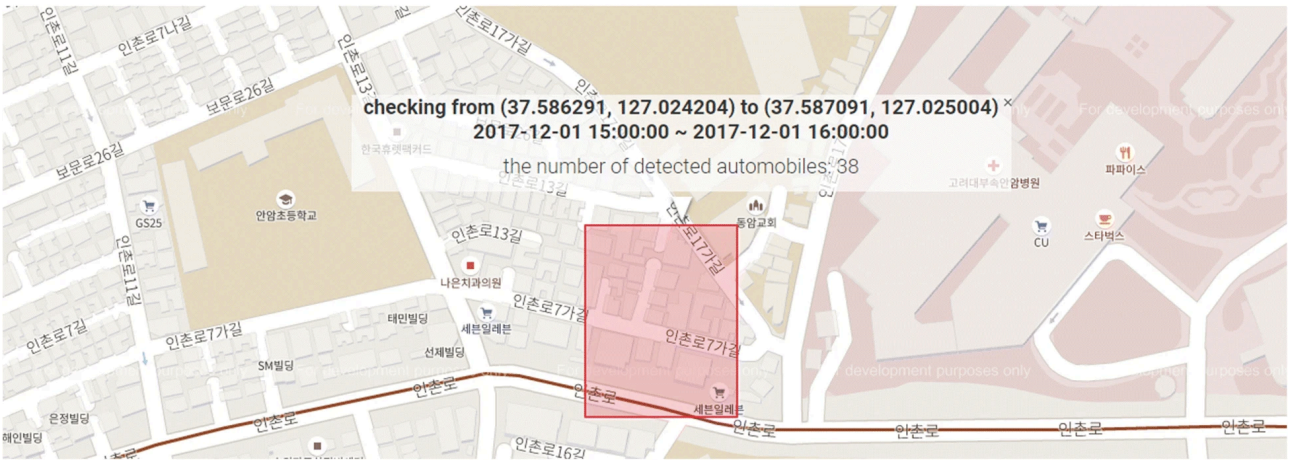
IVATS Home Tracking **Congestion** Grid

Query

Period	
Start Time	2017-12-01 15:00:00
End Time	2017-12-01 16:00:00
Space	
Start Point (Latitude, Longitude)	(37.586291, 127.024204)
End Point (Latitude, Longitude)	(37.587091, 127.025004)

Report

The number of Detected Car	**38**



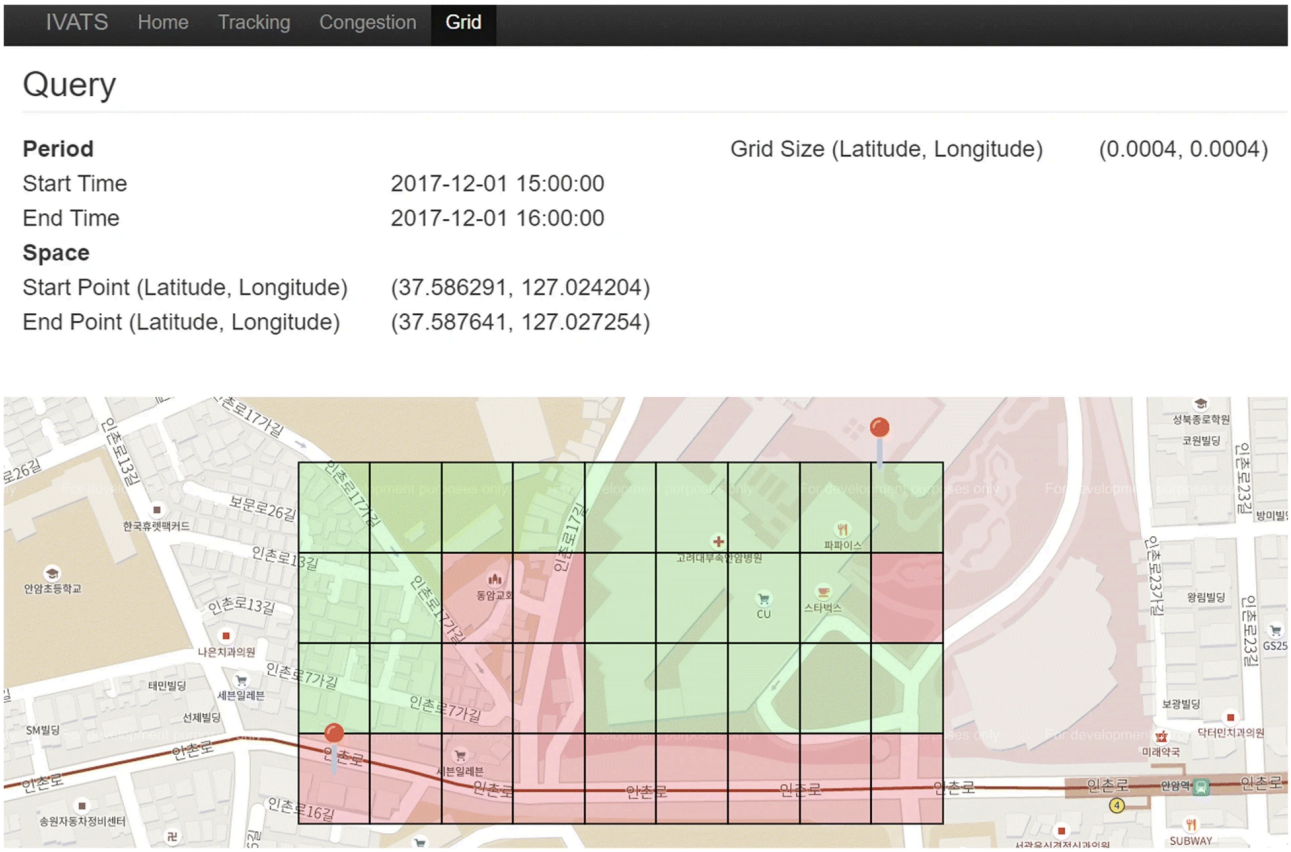
- The top part shows the user query, which contains the time range, area of interest, and the number of vehicles detected.

- The bottom part shows a map where the given query range is represented by a red box and the query result is shown on the pop-up window.

- To process such queries accurately, matched tuples should be sorted by the plate numbers as a vehicle could appear in numerous frames from different video sources.
- For instance, if a vehicle A was captured by vehicles B and C in rapid succession, there would be duplicate data in the query result.

- This type of query can be used to calculate the traffic congestion in a more spacious region at a specific time.

<u>Congestion checking in a spacious region:</u>



Our system divides the given query area into grids and calculates the traffic congestion for each grid.

1. The top part of the figure shows the query condition.
2. The bottom part indicates how much the region in the grid is congested at the given time by using different colors.

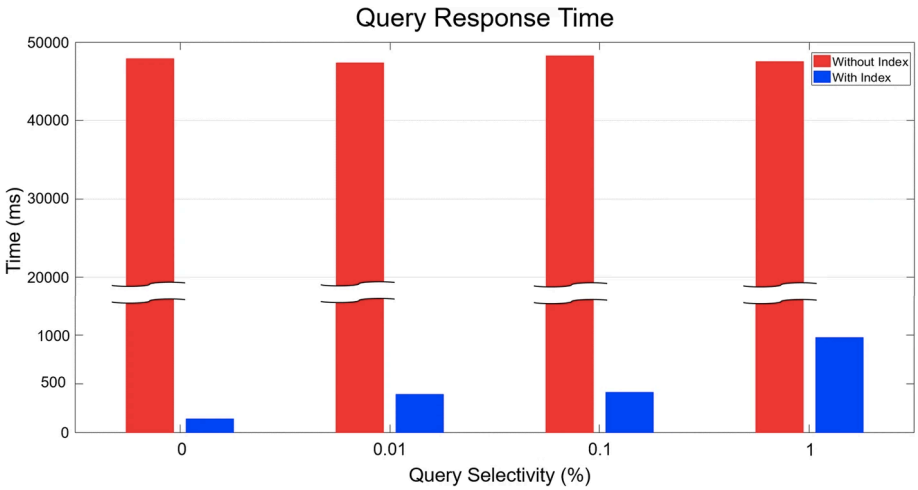
3. The green grid means that the number of the captured records in this region is under the average of the number of those in the query range.
4. The red grid means the opposite.

This function can be used to find out a faster way to a destination or plan the construction of a new road.

<u>Index efficiency:</u>
In the experiment, we evaluate the effect of our index structure by comparing the query response time when indexing is both used and not used for a user query whose range contains a portion of the entire data.

We repeated this ten times.

The query response times according to the query selectivity:



- The response times in the figure are the average of all response times for the queries.
- The query selectivity indicates the ratio of the query result to the total data.
- In the experiment, the query selectivity was set to 1%, 0.1%, 0.01%, and 0% (no relevant data).
- It can be seen in the figure that the response time when using an index structure differs depending on the query selectivity.
- On the contrary, the response times when not using an index structure are approximately identical regardless of the query selectivity.
- Overall, query processing can be achieved faster when using indexing.
- The difference in the response time between the two systems was the greatest when no data were in the query range, at which point the response time was approximately 300 times faster.

Multi-Camera Vehicle Tracking Using Edge-Computing and Low-Power Communication:

Hackathon Stage-Idea Submission:

1. A detailed description of the scope of the solution

1) Typical approaches to visual vehicle tracking across large area require several cameras and complex algorithms to detect, identify and track the vehicle route.

2) Due to memory requirements, computational complexity and hardware constrains, the video images are transmitted to a dedicated workstation equipped with powerful graphic processing units. However, this requires large volumes of data to be transmitted and may raise privacy issues.

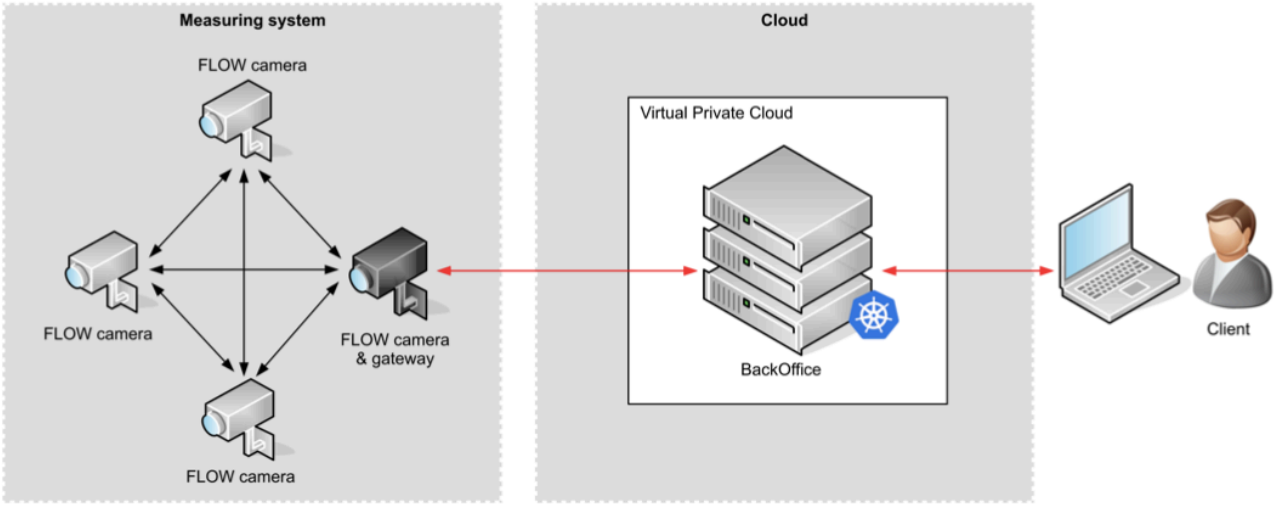
3) Therefore I present a dedicated deep learning detection and tracking algorithms that can be run directly on the camera's embedded system.

4) This method significantly reduces the stream of data from the cameras, reduces the required communication bandwidth and expands the range of communication technologies to use.

5) Consequently, it allows to use short-range radio communication to transmit vehicle-related information directly between the cameras, and implement the multi-camera tracking directly in the cameras.

6) The proposed solution includes detection and tracking algorithms, and a dedicated low-power short-range communication for multi-target multi-camera tracking systems that can be applied in parking and intersection scenarios. System components were evaluated in various scenarios including different environmental and weather conditions.

High-Level Architecture of Vehicle Tracking System: Black & Red arrows denote local and global communication



2. Improvements in functional PoC over Phase 1 submission

- Over the years a number of technologies and systems were proposed for vehicle traffic monitoring.
- For example, inductive loops or sonar/microwave-based detectors are popular at parking barriers, traffic lights and highways. However, their applications are limited, as they are less universal compared to video monitoring.

- The most **important advantages of video-based systems** are: relatively low installation cost, lack of need for complex construction work during installation, and ability to perform wide spectrum of tasks (e.g., detection, classification, tracking, traffic law violations monitoring) in various environments and weather conditions.
- Thanks to its easy installation, video based systems may be considered for both short and long-term deployments.

- In the latter case, **low-energy consumption is an important factor** since in many applications providing external power may be infeasible and the cameras need to operate on batteries. For such short-term application, where battery operation capability is a benefit.

- Interesting use-case for video based systems is traffic monitoring over a road stretch or an intersection. In these scenarios, **vehicles enter and leave the monitored area by one or more roads**, and usually more than one camera is required due to the size of the monitored area.

- One of the challenges in implementing camera-based traffic monitoring systems is their **computational complexity**.

i. Sophisticated artificial intelligence algorithms, which often process high-resolution images from cameras in real-time, have high resource usage. Due to this fact, they cannot be implemented in constrained embedded systems or even in single-board computers.
ii. Naturally, the usage of images with lower resolution minimizes resources required for image processing algorithms, but affects the image quality and hinders some of the applications, including automatic number plate recognition (ANPR).

- Therefore I propose **a multi-camera tracking system** for an intersection or a road stretch.
- We assume that the monitored area has a finite number of entrances and exits.
- Moreover, the cameras can be deployed so that all cars entering the area are detected and fields of views (FoVs) of neighbouring cameras overlap partially. This ensures that when a **vehicle leaves one camera's FoV it is already in the FoV of its neighbour**.
- In such deployment, when a vehicle is processed by a camera, the information about its detection, classification or tracking can be transferred to neighbouring camera. This may simplify image processing operations on the neighbouring cameras and enable multi-camera tracking.
- Information about the vehicles can be transferred in different ways but since it is needed locally (e.g., tracking) the **use of local communication** is preferred.
- Allowing device-to-device communication gives the ability to quickly exchange information about the vehicles and **removes the necessity to include central decision unit** in the system.

To address the requirements of this use-case I propose a Neuroflow system, which has two unique features:

1. multi-camera vehicle tracking with number plate processing, implemented directly in the camera embedded system,

2. local communication mechanism between cameras which allows to hand-over information about the vehicles.

These features allow building a distributed vehicle tracking system which follows the Edge Computing paradigm–all decisions are taken locally by cameras and only the final tracking of a vehicle is stored in the central system.

Reduced complexity of the proposed image processing algorithms allows building a system based on single board computers which are relatively low-cost and power-efficient.

Communication mechanism between the cameras is based on the ISM-band wireless communication, with added reliable network protocol to provide expected Quality of Service metrics. The system can be used for long-term monitoring as well as for temporary deployments (which are installed only for a limited time and then moved to another location).

3. What are the technologies used in the functional PoC?

- Implementation of single camera tracking algorithms (tracking by detection) that can be run in the camera embedded system (edge processing).

- Multiple camera tracking that does not require central processing.

- Vehicle tracking approach that is based on the number plate recognition and can efficiently operate with low resolution images.

- The use of low-power ISM radio for local communication between the cameras in order to implement multi-camera tracking.

Multi-target multi-camera (MTMC) tracking is more universal but also more challenging. In MTMC systems vehicles need to be tracked within the FoV of each camera and also between two or more cameras.
Existing solutions to the task either match a vehicle between the cameras using the number plates, re-identification (ReID) and feature matching, or cameras with overlapping FoVs.

Our approach is based on edge processing where detection and tracking algorithms are fit into the low-power embedded system of the camera.

It was possible thanks to the design of a dedicated detection algorithm that uses **proprietary, hourglass-like convolutional neural network**, and *a system of cameras with overlapping FoVs*. This set of features gives us the opportunity to create a multi-camera tracking system with edge processing.

Hackathon Stage-UI/Design/Code Submission:

1. DB structure & explanation of the key features and algorithms:

SYSTEM ARCHITECTURE:

Purpose of the system:

- The required measurements are a part of a short-term traffic survey, the designed system should be easy to install and configure, and operate efficiently on batteries for the requested time (usually 24–48 h). This simplifies the deployment as the process of getting proper permissions for AC-power supply is often difficult and time-consuming. On the other hand, energy-efficiency of a system is an important feature, also in long-term evaluation.

- In all the scenarios, vehicle tracking is one of the key features of the system, to provide flow-related data, detect anomalies (e.g., vehicle stopped in unusual place) and identify vehicles. This can be used for information/planning, as well as road space retention enforcement.

Architecture:

- The vehicle tracking system is composed of a measuring system and a central server that collects information from the measuring system and provides it to the clients.

- Single server can communicate with a number of independent measuring systems deployed in different areas.

- Each measuring system can be composed of up to 16 FLOW cameras. The cameras are responsible for image acquisition, vehicle detection, and tracking.

- Cameras are based on Jetson Nano platform which is a power efficient single board computer capable of running artificial intelligence algorithms at the edge in real time.

- Jetson Nano is the simplest in Jetson Family, equipped with Quad-Core ARM Cortex-A57 MPCore processor, 128-core NVIDIA GPU and 4 GB of memory capable of executing 472 GFLOPs.

- Cameras are equipped with a GPS receiver that provides them with location and accurate time synchronization that is used for time-stamping of vehicle detections.

- In this setup, we were able to run the system for around 24 h, when powered from a 21 Ah battery.

- The actual runtime depends on system workload, that is, the number of vehicles that need to be detected and traced.

Each camera can use two types of communication: local and global.

1. **Local communication** is designed to be used for bidirectional communication between the cameras in order to exchange setup information, statuses and, most importantly—information about vehicles being tracked.
2. The communication is based on a proprietary communication protocol utilizing 868 MHz frequency band. It was designed specifically to meet the requirements of the vehicle tracking system, especially low latency.

- Information about detected vehicles allows a set of cameras to collect information about the route that every vehicle takes through the area covered by the measurement system.

- The camera that spots a vehicle leaving its FoV broadcasts the information about the vehicle to all cameras in the system.
- The camera that picks up a vehicle entering its FoV, correlates the information received over the radio with its own detection results and continues tracking the object. This procedure continues until the vehicle leaves the monitored area.
- The last camera that detects a vehicle leaving its FoV sends this information to the gateway camera.
- The gateway camera forwards tracking information to the external server.

The **global communication** is available in gateway cameras to enable communication with the Internet and the server.
- It is primarily used for collecting information about tracked vehicles but also for remote management and monitoring of the measuring systems.
- In the experimental setups all cameras were equipped with global communication (using LTE) in order to collect raw images for offline verification of the detection and tracking algorithms.

- The central server is responsible for storage of the information from the cameras and provides data to the clients.

The architecture using both local and global communication was designed to meet specific requirements of the tracking system.

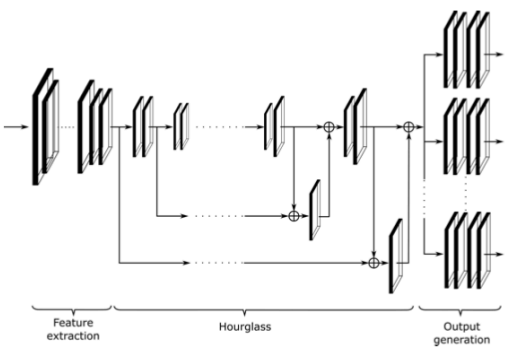
- Processing video images in the cameras offloads the communication interface, as high resolution video streams do not need to be transmitted.
- The proposed edge processing also improves the overall privacy, as images may never leave the camera and can be removed as soon as anonymous tracking information is extracted and the vehicle leaves the camera's FoV.

- Extracted tracking information **contains information about random vehicle id, timestamp, tracking event type, vehicle path and parameter** that estimates the credibility of tracking.
- This information i**s below 30 bytes** that need to be transmitted between the cameras of the measurement system.
- Because there is not much data to transmit, the local communication does not require high data throughputs.
- Instead, it should ensure reliable and low latency communication, as this reduces processing delay in the chain of cameras that track the vehicle.

Vehicle Detection & Tracking:
The systems use tracking by detection for both single-camera tracking and multiple-camera tracking.

For **single-camera tracking**, vehicles detected on each video frame are assigned a bounding box, location, and a timestamp.
- Detection results from subsequent snapshots (frames) are fed to the tracking algorithm.
- The algorithm links the vehicles detected at the snapshots, taking into account locations of the bounding boxes, timestamps and the physics of the vehicle movement (e.g., speed).
- As a result, each camera assigns each vehicle a unique identifier and a timestamped history of vehicle locations within the camera's field of view.

General structure of the neural network used for vehicle detection:



Multi-camera tracking benefits from the timestamps and the fact that the fields of view of the cameras overlap.

- When a vehicle leaves the camera's field of view, the camera sends the tracking information to its neighbours.
- Based on timestamps, the camera that picks up tracking is able to decide which vehicle (from the previous camera) it is tracking.
- Because unique identifiers are generated randomly and are not related to the vehicle identity (e.g., the number plate or, VIN) the multi-camera tracking does not expose privacy.
- Inter-camera tracking can be either run directly on the cameras or on the central server.
- For the first scenario the picking camera correlates information received over the radio with the results of its own detection.
- Because the FoVs partially overlap, the picking camera detects the vehicle a moment before the vehicle leaves the previous camera FoV and as it broadcasts the tracking information.
- This pattern continues and successive cameras accumulate information about the tracked vehicles.
- Operation of the system is similar in the centralized scenario except that all the cameras send their individual tracking information to the server.
- The server performs inter-camera tracking based on the timestamps and the information about the parameters of the measurement system, in particular—the FoVs of all the cameras in the system.

Local Low-power Radio Communication:

- Cameras communicate with each other using 868 MHz ISM band.
- The 2.4 GHz was briefly tested at the beginning of the project but the limited communication range and high susceptibility to traffic-generated radio noise proved them unusable for the required application.
- On the other hand, in 868 MHz band we were able to achieve communication range between cameras in the order of several hundred meters as well as packet reception rates close to 100%.
- The drawback of 868 MHz communication is the available bandwidth, with the default set to 50 kbps and the maximum being 200 kbps (but at the cost of data reliability and communication range).

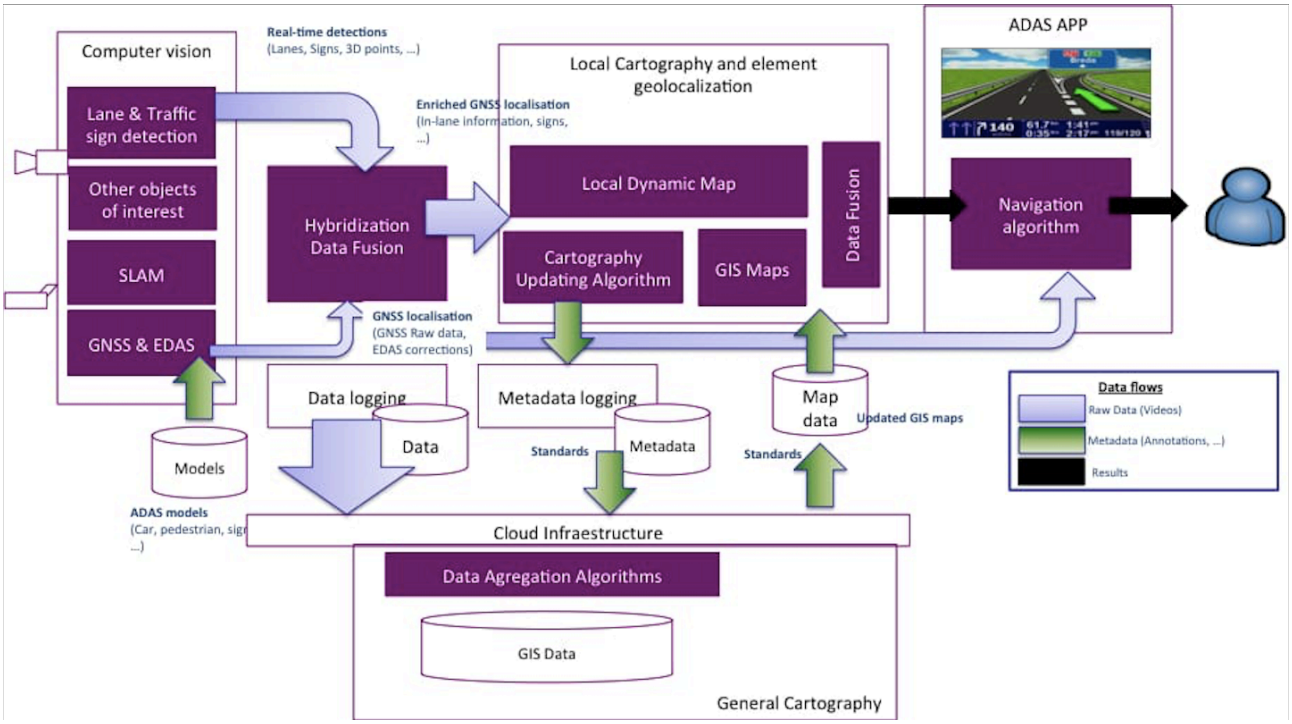
2. UI screen layout & Submit the code (pdf of file or URL of video)

EVALUATION IN PILOT LOCATIONS:
Local Low-power communication:
Vehicle Detection:
Single-Camera Tracking:
Multi-Camera Tracking:

- The proposed image processing approach, including detection and tracking, can be executed on low-power embedded devices in cameras.
- Presented solution is based on edge-computing paradigm—data is processed locally by cameras in order to perform single camera tracking.
- Moreover, we have proposed a dedicated low-power radio communication to exchange data between the cameras, in machine-to-machine manner.
- This ability is used to exchange vehicle tracking information between neighbouring cameras in order to implement multi-camera vehicle tracing.
- The energy usage of the system is low—camera embedded device consumes 20 W at maximum, and can run on 21 Ah battery pack for almost 24 h. This enables our devices to be powered from batteries and deployed in different locations where external power supply is not available or problematic.

————————-xx————————-

3. Moving Lane Guidance & Lane-Level Navigation:



Lane Level Navigation allow it to recreate the road with HD precision. Sub-meter level accuracy identifies which lane a vehicle is in, allowing CommutLink Maps to give highly detailed and timely driving guidance.

Lane guidance is, for many drivers out there, an essential part of the navigation experience. Especially when driving on roads they haven't explored before, that is, as the lane assistance can help them always use the proper lane for the direction they're following.

CommutLink Maps optimizing the lane-change guidance at complex intersections. On the way, in response to road constructions, accidents and other road incidents, as well as road conditions such as vehicle merging or lane narrowing, it can also provide users with early warning through lane-level image guidance and dynamic voice broadcast, making driving safer and more reliable. More at ease.

The first is to intelligently recommend the optimal lane. This is a function that is very useful for both novice novices and "veteran drivers".

It is also a function that uses multi-camera surveillance video tracking and positioning & low power communication in edge processing in depth.

This function can automatically recommend the most suitable lane for you based on the recognition of the current road traffic conditions and travel routes made by Cyclists, Buses or Priority-Vehicle users, making each lane change safer, more efficient, and more comfortable.

In addition, this function can also identify special roads including emergency lanes, bus lanes, etc., and promptly remind you to leave the current lane, so as to cause traffic violations to deduct points and fines, which is very humane.

The second new feature-more navigation perspectives, is that the current navigation software will automatically pop up more accurate road signs in scenes such as intersections, thinking that this new feature of CommutLink Maps is completely "It's a good deal," but in fact, the opposite is true. This combination of CommutLink AI navigation technology and computer vision positioning technology makes the navigation software smarter.

The last trick: the navigation interface is refined. This feature can improve the navigation experience of mobile phones. It can be said to be epic. The experience that can only be achieved by high-end autonomous driving technology is first realized on ordinary smart phones, so that users do not change The car can also get a smart car-like experience.

The detailed navigation interface is to fully present the road condition information around the vehicle in the navigation interface, as much as possible to achieve 100% restoration of the real scene of the road, not only speed limit signs, high-speed guardrails, and road lines Color and performance can be perfectly restored to the navigation interface, greatly enhancing the amount of information in the navigation interface, allowing the driver to control the vehicle from a global perspective.

Not only that, CommutLink Maps can also display the traffic information in the super-view area on the navigation interface. Scenes such as road construction and vehicle ingress can all inform the driver in advance, which greatly enhances driving safety.

CommutLink Maps adheres to the sub-meter-level high-precision positioning of the IVATS system, and develops precise and extremely fast lane positioning and navigation systems to advance together with the IVATS system.

And in order to be able to achieve low-latency communication, CommutLink specifically established a cooperation with Reliance Jio, becoming the first map manufacturer in the industry to support India Mobile's spatio-temporal information services, and based on this, provide users with ultra-low-latency and extremely reliable network communication solutions. Even in a weak network environment, a stable communication link can still be achieved.

In general, the second-generation "sub-meter" lane-level navigation of CommutLink Maps is a function worthy of the word "great". It not only provides users with an epoch-making brand-new intelligent navigation experience, but also greatly improves the driver's experience. The driving experience makes every trip more comfortable and safer; and through cooperation with the country's cutting-edge technology, it has promoted the universalization and popularization of national science and technology, and strengthened India’s national pride and technological self-confidence.

Such a significant technological development indicates that the mobile navigation market, which is still not easy to use, is about to usher in a disruptive market change.

Therefore, we can say responsibly that CommutLink Maps deserves to be regarded as a technological leader in the domestic navigation market.