3CV3S: Cloud-enabled Cooperative Car Video Share and Search System

Ming-Fong Tsai

Department of Electronic Engineering, National United University, Taiwan
mingfongtsai@gmail.com

Abstract

Driver liability in a road accident can be clarified through the assistance of a car video record. Hence, a variety of devices and functions have recently emerged in order to provide related services to vehicle drivers. Such devices or functions tend to include HD video recording, wide angle capturing range and night vision support. However, in circumstances when the vehicle impact takes place out of sight or is obscured, the above-mentioned technologies are not able to clarify the accident liability. In this paper, a Cloud-enabled Cooperative Car Video Share and Search System (3CV3S) is proposed in which vehicle information is uploaded periodically to the cloud. This information includes position, heading and timestamp. When an accident occurs, the recorded information can be used to search for and request related car video from neighbouring cars. In addition, an interface is implemented to allow drivers to utilise the proposed 3CV3S functions, that is, uploading and obtaining car videos, via smartphone.

Keywords: Cooperative, Cloud

1 Introduction

In many circumstances, neighbouring vehicles and pedestrians contribute to the occurrence of road accidents via behaviour such as running through a red light and moving into a blind spot [1]. In order to avoid such accidents, a variety of active driving safety assistance systems have been developed for in-vehicle use, including functions such as lane assist systems, predictive pedestrian protection and autonomous emergency braking [2]. After detecting obstacles and moving objects based on either camera or radar sensors, these systems are permitted to take control of vehicle steering, braking and acceleration in case the driver is in danger. In addition to active systems, a passive driving safety assistance system has also been developed that utilises wireless communication technology in order to broadcast messages to nearby vehicles. A vehicle will broadcast a warning message when it is in or poses a danger, such as employing an emergency brake or having a flat tire. Nearby vehicles that receive this warning message will note the situation and thus avoid a potential rear-end crash. In another work [3], an algorithm is proposed to enhance the research on forward collision avoidance systems. The proposed algorithm has the ability of tracking vehicles in a moving platform and in complicated road environments effectively and accurately. However, although the mentioned forms of driving assistance system aim to avoid collisions, when an accident occurs they are unable to clarify driver liability. As a result, in recent years, drivers have been using car video recorders to record their movement path [4]. Such recorders are typically installed in the vehicle such that the whole front view is captured, with the recorded videos then used to recreate an accident scene, thus clarifying driver liability as long as the crucial information is not out of sight.

A variety of different functions have been developed as part of car video recorder research. Chen et al. [5] propose a safety system for the PAPAGO P3 driving recorder that includes features such as lane and vehicle detection, while Baek et al. [6] present a vehicle driving state system featuring on-board diagnostics such that more vehicle-related information is recorded. However, none of the above-mentioned driving assistance systems are capable of clarifying accident liability. Hence, a browsing and retrieval system for driving information is proposed in [7]. Unfortunately, the latter research focused on self-driving recorder data retrieval, which is also not capable of clarifying accident liability when the accident occurs in a blind spot. Kuo et al. [8] thus proposed a road map utilising a Global Positioning System (GPS) which is able to obtain the relative locations of neighbouring cars in a vehicular network. The effective retrieval of car video records through matching information with neighbouring cars is researched in [4], with the authors presenting a video retrieval algorithm that adopts feature matching and text motion matching in order to acquire the desired video segments. However, image processing time is required to implement the algorithm, while video segments are usually encoded via data compression. Furthermore, the matching of scale-

[*]Corresponding Author: Ming-Fong Tsai; E-mail: mingfongtsai@gmail.com
DOI: 10.3966/160792642018081904003

invariant feature transforms and text in order to retrieve the relevant videos can be highly complex. For the purpose of providing an effective cooperative driving video retrieval platform, this study enhances work previously outlined in [9], proposing a 3CV3S. The system is designed to retrieve related car videos when there is a need to clarify accident liability, without the need of going through a complex process such as extraction of image processing time from the compressed video. Vehicles are equipped with radar sensors such that neighbouring cars can be detected, with useful vehicle information such as position, heading, vehicle type, speed and acceleration broadcast to other vehicles through VANET. The information received by neighbouring cars is then processed by a car video retrieval algorithm. In this way useful car videos can be determined and the user may send a request to the video owner to retrieve the car video.

Although the car video retrieval algorithm is essentially implementable in the car video recorder, it should be kept in mind that the latter is an embedded system with a generally limited computational resource. A complex algorithm will extend the loading time of the video recorder and thus should be kept simple in design. However, Cloud Computing can overcome this situation. In such a case, the algorithm is implemented outside the video recorder, with the integration of car video recorder and cloud omitting inter-vehicular information transfer and instead uploading it to the cloud. All computational processes are performed by the cloud as well. A Geographic Information System (GIS) also plays a role in the video retrieval algorithm such that a correct decision can be made. Indeed, the research presented in [10] states that cloud computing can be applied to overcome challenges in the application of GIS, providing a brief evaluation of cloud computing with respect to GIS and also proposing a multi-tiered architecture for a GIS cloud system. A cloud-based interface that handles data of smart cars efficiently was designed in [11]. However, the integrated service system is focused on the collection of information from smart cars rather than driving videos. Some studies have proposed a traffic video sharing platform [12], which shares a similar function as our proposed work. However, the driving video and GPS track data are required to be uploaded manually. The data are not uploaded periodically; there might be insufficient data when a user tries to retrieve the relevant data from the platform. Furthermore, the video must be in the database in order to be retrieved, playing the role of burden for the server. In the system proposed in the present paper, each vehicle uploads its information to the cloud periodically, including timestamp, position, heading and speed. However, the uploading of a car video is optional so that the quantity of uploaded data can be controlled. This means the 3CV3S database requires fields for path information, which is not the case for video recorder data. In order to search for related videos, the user sends a request message to the system. By providing the timestamp, position and heading of the vehicle, the video search algorithm will be able to search for neighbouring vehicles that may have critical videos recorded when the accident took place. When a critical video is found, if the video has not already been uploaded to the system an upload request will be sent to the video owner.

This paper is structured in 4 sections, including the present introduction section. Section 2 outlines the proposed method. Section 3 presents the implementation of 3CV3S. In the final section, conclusions and future work are discussed.

2 Cloud-enabled Cooperative Car Video Share and Search System

2.1 System Overview

Figure 1 demonstrates an example of a car accident in which the crash occurs in the victim vehicle's car video recorder blind spot. It can be seen that even though Car 1 is equipped with a car video recorder, driver liability cannot be clarified by its video record since it is hit from the rear by Car 2, which is out of the video recorder's line of sight. If Car 2 leaves the scene rapidly, the perpetrator may not be identified. However, the accident also takes place just ahead of Car 3, with the whole scenario recorded by Car 3's video recorder. Hence, if Car 3 is able to share its recorded car video, Car 1 will be able to identify the perpetrator of the accident [13]. In the present paper, a 3CV3S is proposed that aims to provide a platform for the sharing and searching of car videos in case of accident. Through the utilisation of the cloud, both the uploaded car videos and the periodically uploaded vehicle information can be stored and remain available at all times, making 3CV3S more reliable. As 3CV3S purposely serves as a cooperative platform; the utilisation of cloud computing is essential in order to handle information from numerous users.

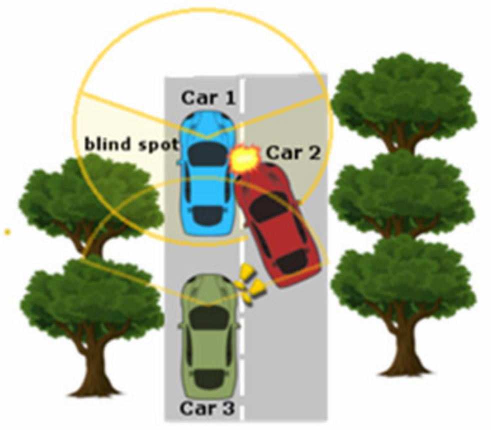


Figure 1. Blind spot scenario

A system overview of the proposed 3CV3S is shown in Figure 2. The system can be divided into the following four main parts: the upload of path information, video requesting, video upload request, and the upload of car video. The upload of path information updates the movement path of the vehicle and includes fields such as vehicle ID, timestamp and GPS information. The ID field provides user identification information, which does not involve any privacy issues, while the timestamp field records the time of the vehicle available at a specific position. Finally, the GPS field includes the position, speed and heading of the vehicle at that particular moment. The timestamp and GPS data play a particularly important role in the 3CV3S video search service, as this information is used to search for related car videos. In the video requesting process, users request related car videos by uploading timestamp and GPS information to the 3CV3S cloud, with 3CV3S then using this information to search for vehicle data with similar timestamp and GPS values. For example, a user may select data falling within the range of a distance of 3 metres and time of 10 seconds. As mentioned above, although a vehicle's timestamp and GPS information will be available in the 3CV3S database, the corresponding video might not be available because video upload is optional. Thus, during the next stage, when the required videos are targeted, 3CV3S will send request messages to the video owners to prompt them to upload the corresponding videos. It is then up to the video owner to decide whether to provide the video or not. If the video owner decides to share the car video, the user who requested the video will be able to download it after the owner successfully uploads the video.

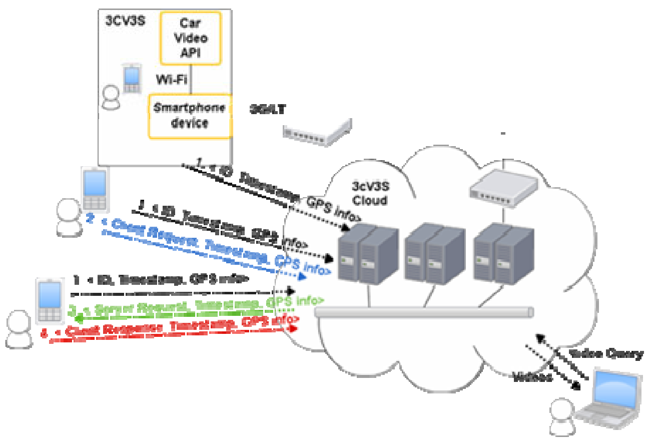

3CV3S
Car Video API
Wi-Fi
Smartphone device
3G/LT
3cV3S Cloud
1 < ID, Timestamp, GPS info>
1 < ID, Timestamp, GPS info>
2 < Client Request, Timestamp, GPS info>
1 < ID, Timestamp, GPS info>
3 < Server Request, Timestamp, GPS info>
4 < Client Response, Timestamp, GPS info>
Video Query
Videos

Figure 2. System overview

2.2 Retrieval Method

The retrieval of car videos is based on timestamp and GPS information. In 3CV3S, as it is compulsory to upload this information periodically, the system is able to identify related videos as well as the video owners when a user makes a video search request. However, the frequency of video uploads is defined by users themselves, with the car video recorder's video files able to be uploaded either at a configured frequency or only upon request by other users.

An example of retrieval method application is shown in Figure 3, in which Car 1 and Car 2 have similar GPS heading values. Since Car 2's position is behind Car 1, the GPS value of Car 2 is thus also behind that of Car 1. Therefore, if the driver of Car 1 sends a video request in that situation, the video recorder data of Car 2 will be targeted, with 3CV3S then requesting the owner of Car 2 to upload the car video. In Figure 3, Car 3 also has an approximately similar GPS value to that of Car 1, with the exception of the opposite heading value. Hence, even if Car 1 sends a request, the car video recorder data of Car 3 will not be requested by the system. In the proposed method, the timestamp deviation is set to ±60 seconds, the position deviation to ±10 metres and the heading deviation to an angle of ±90°. These settings are also configurable by the user, who can obtain a more accurate result by setting stricter limits (although fewer videos will be retrieved).

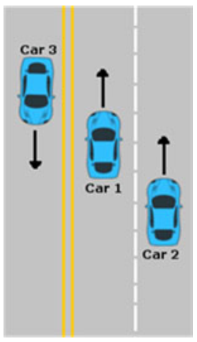


Figure 3. Example of retrieval method scenario

In order to acquire accurate position deviation and heading deviation data, a vector graph is used to compute the relationship between cars. The construction of the vector graph for the given scenario is shown in Figure 4, in which the coordinate of Car 1 is *(x_1, y_1)* and that of Car 2 is *(x_2, y_2)*. The distance between Car 1 and Car 2 can be calculated through equation (1), where d is the distance:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1}$$

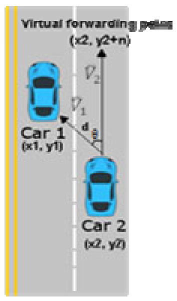


Figure 4. Constructing vector graph

However, since the values of x and y represent the longitude and latitude of the cars, the correct formula with which to calculate the distance between the two cars should be the Haversine formula, in which the radius of the Earth (R) is considered, as shown in equation (2):

$$d = 2R \times \sin^{-1}\left(\sqrt{\sin^2\left(\frac{x_2 - x_1}{2}\right) + \cos x_1 \cos x_2 \sin^2\left(\frac{y_2 - y_1}{2}\right)}\right) \quad \textbf{(2)}$$

In order to compute the heading deviation, two vectors are needed. The first vector, $\overline{V_1}$, is the direction from Car 2 to Car 1 and is obtained by $\overline{V_1} = (x_2 - x_1, y_2 - y_1)$. Similarly, $\overline{V_2}$ is the heading direction from Car 1 to Car 2, where $\overline{V_2} = (x_2 - x_2, y_2 - n) = (0, y_2 - n)$. Referring to the formula of the angle between two vectors (3), the angle θ can be obtained using equation (4).

$$\cos\theta = \frac{\vec{V_1} \cdot \vec{V_2}}{\left|\vec{V_1}\right|\left|\vec{V_2}\right|} \quad \textbf{(3)}$$

$$\begin{aligned}\theta &= \cos^{-1}\left(\frac{(x_2 - x_1)\cdot 0 + (y_2 - y_1)\cdot(y_2 - n)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \cdot \sqrt{0^2 + (y_2 - n)^2}}\right) \\ &= \cos^{-1}\left(\frac{(y_2 - y_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}\right)\end{aligned} \quad \textbf{(4)}$$

Different values of d and θ, which can be adjusted by users, determine the cars to which the system will send a video upload request.

2.3 System Flowchart

In 3CV3S, the movement trajectory (path history) of each vehicle is uploaded to the cloud periodically, for example at a frequency of 1 Hz, including information such as GPS position, speed, heading direction and timestamp. The movement trajectory of each vehicle is stored by the system cloud and can be referred back to at any time when requested. With reference to the system flowchart of 3CV3S shown in Figure 5, the 3CV3S cloud is always on standby for user requests to search for a car video. For example, assume that an accident has occurred and the victim decides to search for car videos that recorded the scene. A search request, including vehicle position, heading and timestamp, is sent to the 3CV3S cloud; the 3CV3S video retrieval algorithm then filters and selects the required video recorder data based on the provided information, with the corresponding vehicle of the selected video recorder data being the neighbouring vehicle of the accident victim when the accident took place. Then, the 3CV3S cloud transmits an upload request to the selected video recorder data's owner, who has the right to decide whether to share the video with the cloud or not. In addition, it is possible to retrieve multiple videos if there are multiple data that satisfy the retrieval conditions.

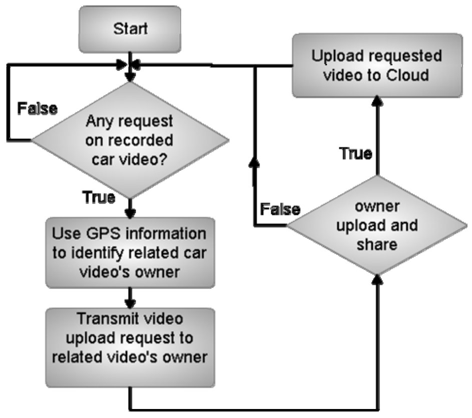

Start
Upload requested video to Cloud
False
Any request on recorded car video?
True
True
Use GPS information to identify related car video's owner
False
owner upload and share
Transmit video upload request to related video's owner

Figure 5. System flowchart

3 System Implementation

3.1 Implementation Concept

3CV3S is composed of two main parts. First, an Android application, which can be easily installed in a smartphone device, serves as the 3CV3S client side. Second, the 3CV3S cloud, which is a web service in cluster form, supported by 8 servers. In the present paper, both parts are implemented. While the Android application uploads the GPS information to the web server at a frequency of 1 Hz, video data are stored via a built-in camera sensor. The application is also able to listen to requests from the web server, which prompt the user to upload video data. The web server is in turn able to receive GPS information from each smartphone device, as well as search for neighbouring cars' videos when the user inputs the GPS information via the system website.

System architecture is shown in Figure 6. In addition to being responsible for uploading GPS information periodically, the Android application is the client side with which the user can send a query request for videos to the cloud platform. The event handler, which distinguishes the message sent by the client side, is compulsory because there are two actions that the client can perform: First, the upload of GPS and vehicle information (car ID, timestamp…etc.), which upon receipt is stored in the database, and second, a query request. In the latter case, the car ID of the request sender, as well as the requested information such as timestamp, is transferred to the Location-based Query Engine. Although the Query Engine enables all neighbouring cars to be found based on location and timestamp, the initial result will be too general and contain a large amount of unrelated data. For example, a neighbouring car with a different heading and direction may not have captured the critical scene on camera. Hence, all the candidate cars are then processed by the video retrieval algorithm such that those neighbouring cars that potentially recorded the critical video scene can be identified. The system then sends a request message to the selected car owners to prompt them to upload their car videos to the 3CVCS cloud. Figure 7 shows the software stack of 3CV3S, for which Hadoop is used as the distributed computing platform and MongoDB as the database. The video retrieval algorithm and data analytic functions are implemented in the MapReduce programming model. The GPS track and uploaded driving videos of vehicles are stored in the cloud; these collected data can be used for further work, such as, tracing and studying the behaviour of users, performing graphical analysis, and determining a high accident rate area. If sufficient bandwidth is available, further services such as real-time video sharing may be applied.

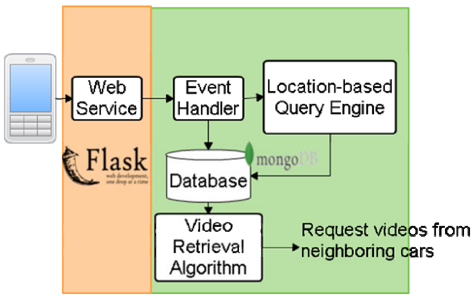

Web Service
Event Handler
Location-based Query Engine
Flask
mongoDB
Database
Video Retrieval Algorithm
Request videos from neighboring cars

Figure 6. System architecture of the cloud platform

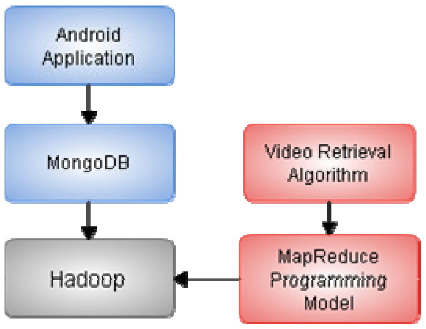


Figure 7. 3CV3S software stack

3.2 Implementation Considerations

In general, Apache Hadoop is considered a popular framework for the establishment of cloud platforms, being a reliable and scalable open-source software programme. In Apache Hadoop, the distribution process of large data sets across clusters of computers is allowed via the use of a simple programming model known as MapReduce [14]. Apache Hadoop is also able to scale up servers to the thousands, as well as having the ability to detect and recover software failures. There are several modules of Apache Hadoop, including the Hadoop Distributed File System (HDFS) for high throughput access to data, the Hadoop YARN [15] framework for job scheduling and cluster resource management, and the Hadoop MapReduce programming model, which can handle the parallel processing of large volumes of data. In order to integrate with the Apache Hadoop framework, a scalable, distributed database known as HBase [16] has been developed. However, the HBase database system is less convenient when dealing with geographic location queries, such as those employed in searching for neighbouring cars in 3CV3S. Thus, HBase is not a suitable database to be implemented in 3CV3S in order to maintain system simplicity and performance. Instead, MongoDB [17] was chosen for use in 3CV3S, as it is a document-oriented NoSQL database and supports geographic query, which is essential for effective 3CV3S performance.

4 System Results

The interface of the 3CVCS client side Android application contains a Record button and an Upload button; a more detailed review can be found in [9]. When the user clicks the Record button, the system will start to upload the self-supplied GPS information to the web server at a frequency of 1 Hz. Simultaneously, the recorded video data will be stored as separate files in the smartphone device every 30 seconds. If the Upload button is then clicked, the recorded video files will be uploaded to the web server. The same Upload button can be pushed when an upload request is received from the server. The 3CV3S website mainly shows a map, on which users can either input their longitude and latitude or select a location directly; again, further detail can be found in [9]. The web server then uses the user-input GPS information to search for neighbouring cars' video data from the database. In default mode, the timestamp deviation is set to ±60 seconds, the position deviation to ±10 metres and the heading deviation to an angle of ±90°. Search results are then marked with a location tag on the map. If the timestamp deviation, position deviation and heading deviation are configured in a strict manner, fewer videos will meet the requirements and thus fewer results will be found. When the results are displayed, users can click on any tag to obtain the associated video data. If the selected video is not yet uploaded, an upload request will be sent to the video owner.

The cloud platform is formed by 8 hp servers with quad core Intel i7 CPU and 4GB RAM, while the web application framework is Flask and the implemented database is MongoDB. Flask is a lightweight web application framework written in Python and based on the WSGI toolkit and the Jinja2 template engine. Flask was selected for 3CV3S because it maintains the flexibility of Python, providing a simple template and thus saving time in web application development. In this implementation, the query response time of the MongoDB database when there are approximately 100,000 records is less than 1 second. As an experiment, the query result was tested by applying different timestamp, position and heading deviations. In order to carry out this experiment, the traffic situation was simulated as indicated in Figure 8. A total of 1,000 cars were randomly generated to drive on a map consisting of 9 intersections, with a distance of 50 metres between each intersection. Each car took 20 seconds to drive from the entrance to the exit of the map, while the total simulation time was 20 minutes. In order to simplify the simulation, the cars drove straight through the intersections without changing direction. Approximately 15~20 cars were present on the map at any one time.

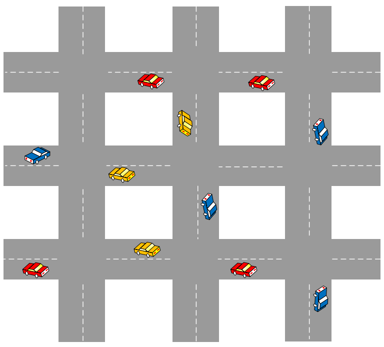
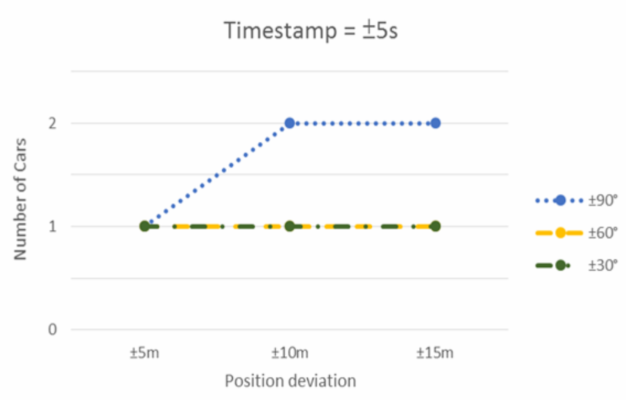
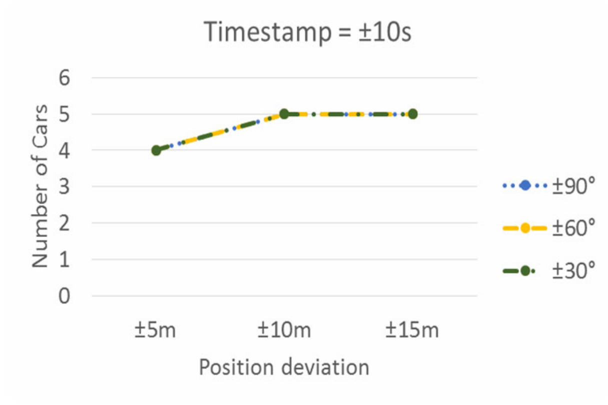


Figure 8. The simulation scenario

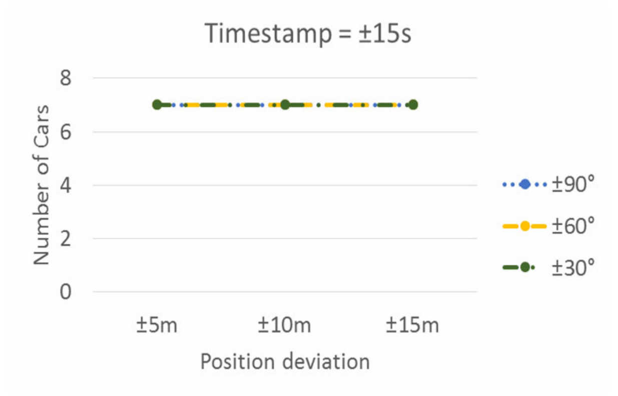
The number of cars with critical videos found by the proposed video retrieval algorithm is shown in Figures 9(a)~(d), for applied timestamp deviations of ±5, ±10, ±15 and ±20 seconds, position deviations of ±5, ±10 and ±15 metres, and heading deviations of ±30°, ±60° and ±90°. Analysis of these figures reveals that number of cars found increases with both greater timestamp deviation and position deviation values. However, in the experiment, the number of cars found with critical videos did not increase when the heading deviation increased. This is because the distance between each intersection was 50 metres. In the experiment for which the result is shown in Figure 9, the position deviation ranged between ±5 ~ ±15 metres. Therefore, only cars that drive across the intersection where an accident takes place will be selected. In this situation, an increase in the heading deviation will not increase the number of cars selected by the algorithm. Figure 10 illustrates that when a position deviation of between ±20 and ±60 metres is used, the heading deviation value will have a greater influence on the result when the position deviation is set to ±60 metres. This is due to the fact that cars driving across intersections adjacent to the accident location can be found by the video retrieval algorithm when the heading deviation is set to ±90°, as shown in Figure 11. However, if the position deviation is less than 50 metres, those vehicles driving through adjacent intersections and for whom the accident scene is not visible will not be seen as candidates to upload videos.



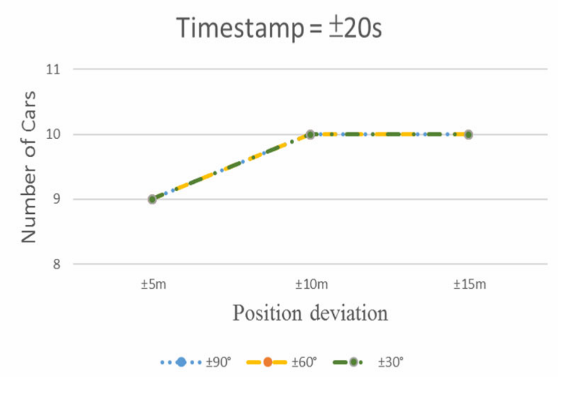
(a) Timestamp deviation: ±5s



(b) Timestamp deviation: ±10s



(c) Timestamp deviation: ±15s



(d) Timestamp deviation: ±20s

Figure 9. Simulation results showing the number of cars found by the 3CV3S video retrieval algorithm for different time deviation, position deviation and heading deviation values

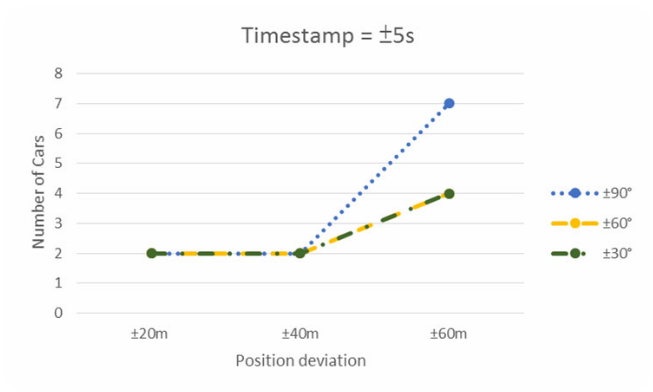


Figure 10. Simulation result for a position deviation of between ±20 ~ ±60 metres

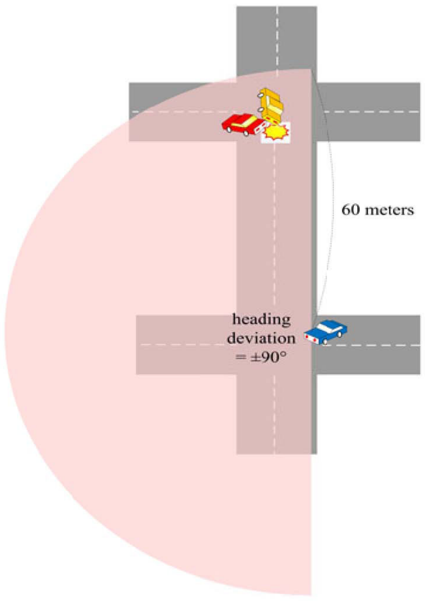


Figure 11. Effectiveness of heading deviation for ±60 metres position deviation

In 3CV3S, the car video retrieval method is integrated with cloud computing technology due to the latter's high level of efficiency and reliability. Reliability is particularly important for 3CV3S because users must be able to retrieve all related video recordings. System reliability was tested in a further experiment in which servers were manually shut down one by one at an interval of 10 minutes. In this experiment, the timestamp deviation and heading deviation were ±20 s and ±30°, respectively, while the position deviation varied between ±5 and ±15 metres. The experimental result shown in Figure 12 reveals that the recorded videos were still available when either one, two or three servers were not available, thus proving the data reliability of 3CV3S thanks to the implementation of cloud technology. Fault tolerance is achieved via HDFS, in which datanodes send heartbeats periodically to the namenode. Therefore, the namenode recognises the availability of all datanodes. Whenever a datanode is detected as unavailable by the namenode, a backup mechanism such as block replication will be launched in order to ensure data availability. In the experiment, data blocks were replicated three times in HDFS. In order to detect an offline datanode before losing all data blocks, the offline datanode must be detected within 20 minutes. This is because the time interval in which the datanodes go offline is 10 minutes. Datanode timeout is configured by the parameters "dfs.namenode.heartbeat. recheck.interval" and "dfs.heartbeat.interval" in the configuration files.

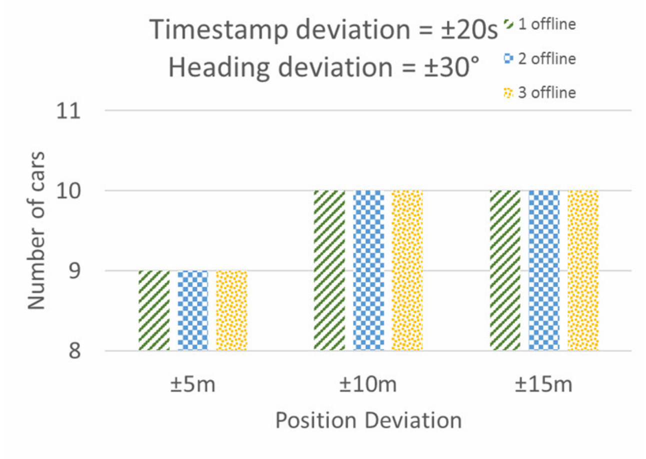


Figure 12. Simulation result obtained when machines fall offline one by one at an interval of 10 minutes

5 Conclusion

The number of drivers who install video recorders in their vehicles will increase as the problem of accident liability clarification is paid more attention. Here we present 3CV3S, a system that enables users to search for and share videos with neighbouring cars. 3CV3S is implemented in a smartphone device that serves as the client side, with the 3CV3S cloud including a web server, a distributed database and a video retrieval method. The client side uploads vehicle information such as position, heading and timestamp periodically, while the recorded video is uploaded when requested by another user. Each user's car information is recorded in the 3CV3S cloud, with a video retrieval algorithm employed to search for critical videos when GPS information and user-defined conditions are provided. In future work, 3CV3S will be integrated with another NoSQL database system, HBase, in order to fulfil Big Data requirements. Furthermore, an active safety method will be designed aimed at protecting real-time information collected and uploaded to the cloud. On the other hand, seeking to provide a better video uploading experience, an improvement will be made such that the quality of the video upload will change dynamically according to available bandwidth.

Acknowledgments

The author would like to thank the Ministry of Science and Technology of the Republic of China for financially supporting this research under grant number MOST 105-2221-E-035-065, MOST104-2221-E-035-021 and MOST103-2218-E-035-013.

References

[1] Y. Chen, C. Yang, Vehicle Red-light Violation Detection base on Region, *IEEE International Conference on Computer Science and Information Technology*, Chengdu, China, 2010, pp. 700-703.

[2] H. Kopetz, S. Poledna, Autonomous Emergency Braking: A System-of-systems Perspective, *IEEE Conference on Dependable Systems and Networks Workshop*, Budapest, Hungary, 2013, pp. 1-7.

[3] P. Liu, W. Li, Y. Wang, H. Ni, On-road Multi-vehicle Tracking Algorithm based on An Improved Particle Filter, *IET Intelligent Transport Systems*, Vol. 9, No. 4, pp. 429-441, May, 2015.

[4] K.-H. Lee, J.-N. Hwang, J.-H. Yoo, K.-H. Choi, Effective Car Video Retrieval using Feature Matching in a Mobile Video Cloud, *IEEE International Conference on Distributed Smart Cameras*, Hong Kong, China, 2012, pp. 1-6.

[5] W. Chen, L. Jian, S. Kuo, Video-based on-road Driving Safety System with Lane Detection and Vehicle Detection, *IEEE International Conference on ITS Telecommunications*, Taipei, Taiwan, 2012, pp. 537-541.

[6] S.-H. Baek, D.-W. Jeong, Y.-S. Park, H.-S. Kim, M.-J. Kim, J.-W. Jang, Implementation Vehicle Driving State System with OBD-II, MOST Network, *17th Asia-Pacific Conference on Communications*, Sabah, Malaysia, 2011, pp. 709-714.

[7] M. Naito, C. Miyajima, T. Nishino, N. Kitaoka, K. Takeda, A Browsing and Retrieval System for Driving Data, *IEEE Intelligent Vehicles Symposium*, San Diego, CA, 2010, pp. 1159-1165.

[8] C.-I. Kuo, P.-C. Wang, C.-H. Lin, C.-K. Shieh, M.-F. Tsai, Implementation of Radar Map Using GPS in Vehicular Networks, *International Computer Symposium*, Hualien, Taiwan, 2012, pp. 663-672.

[9] M. F. Tsai, S. W. Huang, C. M. Chiu, L. D. Sun, C. K. Shieh, Cooperative Car Video Share and Search System, *Applied Mechanics and Materials*, Vols. 764-765, pp. 853-857, May, 2015.

[10] M. A. Bhat, R. M. Shah, B. Ahmad, Cloud Computing: A Solution to Geographical Information Systems (GIS), *International Journal on Computer Science and Engineering*, Vol. 3, No. 2, pp. 594-600, February, 2011.

[11] Y. Xu, J. Yan, A Cloud-based Design of Smart Car Information Services, *Journal of Internet Technology*, Vol. 13, No. 2, pp. 317-326, March, 2012.

[12] S.-M. Yuan, C.-Y. Chiang, S.-B. Yang, Y.-L. Chen, A Traffic Video Searching and Sharing Platform based on Smart Wearable Devices, *IEEE International Conference on Consumer Electronics*, Las Vegas, NV, 2014, pp. 85-86.

[13] C.-S. Li, H.-C. Chao, IPv6 Auto-configuration VANET Cross Layer Design based on IEEE 1609, *IET Networks*, Vol. 1, No. 4, pp. 199-206, December, 2012.

[14] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, E. Baldeschwieler, Apache Hadoop Yarn: Yet Another Resource Negotiator, *ACM Symposium on Cloud Computing*, Santa Clara, CA, 2013, pp. 1-16.

[15] L. George, *HBase: The Definitive Guide*, O'Reilly Media, 2011.

[16] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, R. E. Gruber, Bigtable: A Distributed Storage System for Structured Data, *International Conference on USENIX Symposium on Operating Systems Design and Implementation*, Seattle, WA, 2006, pp. 1-14.

[17] P. Membrey, E. Plugge, T. Hawkins, *The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing*, Apress, 2010.

Biography

Ming-Fong Tsai received the Ph.D. degree from the Department of Electrical Engineering, Institute of Computer and Communication Engineering, National Cheng Kung University, Taiwan. He is currently an Assistant Professor with the Department of Electronic Engineering, National United University, Taiwan. His current research interests include Internet of Things, Vehicular Communications and Multimedia Communications.