INERTIAL DETECTION OF UNUSUAL DRIVING EVENTS FOR SELF-DRIVING

BY HAIRONG WANG

A thesis submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Marco Gruteser

and approved by

_____

_____

_____

New Brunswick, New Jersey

JANUARY, 2019

ABSTRACT OF THE THESIS

INERTIAL DETECTION OF UNUSUAL DRIVING EVENTS FOR SELF-DRIVING

by HAIRONG WANG

Thesis Director: Marco Gruteser

While modern self-driving vehicles have shown their impressive capabilities towards offering new mobility to millions of people, it still remains challenging to build fully dependable and safe self-driving systems. To ensure the dependability of automated driving, the self-driving system is not only required to understand common road situations, but also widely different unusual events (e.g., objects on the roadway, pedestrian crossing highway, deer standing next to the road, etc.), which are very rare but more likely to cause unanticipated accidents.

To detect unusual events, existing approaches seek to collect them by driving millions of miles with self-driving prototypes. But there still remains uncertainty because of limited miles covered. In contrast, this thesis proposes automatic unusual driving events identification algorithms, which can detect unusual cases through inertial sensing from in-vehicle devices in human-driven vehicles. This approach can be scaled to a much larger number of vehicles and thereby cover

larger driving distances. The approach involves monitoring human driver reactions based on inertial sensors (e.g., accelerometer and gyroscope) and demonstrating that they are useful indicators of unusual driving events.

Our inertial detection approach includes three stages. At first, we apply a potential emergency period (braking and swerving) detection to detect a sudden driver reaction, which reflects situations that challenge human drivers. Then we utilize three features to extract different properties from the detected periods. Finally, we propose a feature fusion method to fuse the features in an accuracy-driven way. Therefore, by extracting the features from the potential unusual periods, we can detect hazardous events from a large data set automatically. Besides, in order to improve the efficiency of processing a large scale of dataset, we also provide an alternative approach to process data in parallel pipelines in the cloud.

We evaluate whether the inertially identified driving events match events that are manually labeled as unusual based on more than 120 hours of real world driving data. The result shows the proposed fusion method outperforms the baseline methods with an 82% accuracy improvement for braking events as well as a 94% accuracy improvement for swerving events. To improve the dependability of automated driving systems, such detected events could be used in simulation tests to gradually refine self-driving software.

Acknowledgements

At first, I would like to say many many many thanks to my advisor, Prof. Marco Gruteser. To show how thankful I am must be the most difficult part to write in this thesis. Thank you for your advice, patience, encouragement and all the time you have spent on mentoring me. Sometimes I am not satisfied with what I am done, but you always help me think of the next steps with plenty of patience. Sometimes I am even disappointed with myself, but you are the one that always trusts in me. Your kindness and wisdom have given me countless courage and motivation. Thank you for admitting the girl one and a half years ago. There is still a long journey for her, but with your endless encouragement, she has been filled with confidence.

I would also like to thank Dr. Yingying Chen and Dr. Jorge Ortiz for taking time from their busy schedules and being my defense committee members. I would like to thank Hongyu for offering me so much advice and tutoring during the thesis. It's exciting that I can be a co-author in our accepted paper. Thank you for your great work. Also, I would like to thank Ivan Seskar, for all the patient mentoring during the summer intern. Special thanks to my intern partners - Parul and Da, I am so lucky to meet you in this lovely place.

Lastly, thank my boyfriend Luyang for all the love and accompany. Whenever I met difficulties or I felt frustrated, I am always brave enough because I know there is someone with whom I can share all my feelings and find a shelter to stay.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

Advances in autonomous vehicle technologies have recently helped bring self-driving cars to the focus of public interest. More and more self-driving car models have been released by different industries and organizations [1] [2] [3] [4]. While modern self-driving vehicles have shown their impressive capabilities towards offering new mobility to millions of people, the safety level that can be achieved by current self-driving vehicles is still unclear. There have been several reports about traffic accidents caused by self-driving or semi-automated cars [5] [6] [7], which are typically caused by unusual situations that have not appeared in the regular training and testing process. Therefore, it is still challenging to provide truly safe and reliable self-driving systems in the near future.

To improve the reliability, the self-driving system is not only required to understand common road situations, but also those unusual events (e.g., objects on the roadway, pedestrian crossing highway, deer standing next to the road, etc.) which are very rare but more likely to cause unanticipated accidents. Therefore, scaling the road dataset to billions of miles of driving and obtaining large amounts of unusual events and corner cases is highly desirable to generate more reliable algorithms.

Researchers have developed different methods for self-driving data collection. Existing approaches seek to collect them by driving millions of miles with self-driving prototypes. Since the miles self-driving cars accumulate are limited, it is

challenging to collect a sufficiently large dataset in a short period with this approach. To overcome the challenge, this motivates collecting a catalog of unusual driving events that represents unusual situations expected in billions of miles of driving to accelerate the development of truly dependable self-driving systems. We are also looking into an efficient way which can rely on low-cost in-vehicle devices with scalability and flexibility. Such a comprehensive set of unusual events and corner cases can be more easily obtained by scaling data collection to large numbers of minimally instrumented (camera-equipped) human driven vehicles, as previously advocated by BigRoad [8]. This, however, would still require identifying the unusual events in such a vast data set to create test cases for proving grounds or simulators.

To identify unusual events and corner cases, manual inspection seems a straightforward and possible approach. However, it requires plenty of extra effort, amplifies privacy concerns, and increases storage and networking overhead for collecting all data. Therefore, automatically identifying unusual driving events remains a challenge.

Some of the driving data collected on the road rely on the cameras or radar [9, 10]. However, to automatically detect unusual events by processing those frames or data captured by cameras/radars is very difficult. Also, the data from radars are more likely to focus on the distances between objects rather than the emergency of road situations. It is still challenging in finding the applicable indicators to automatically detect those corner cases which automated driving technology is interested in.

We propose automatic unusual driving events identification algorithms based on inertial sensors, which can detect unusual situations through in-vehicle algorithms and can be easily scaled for wide deployment. Our approach involves monitoring human reaction from large human driving data based on the readings

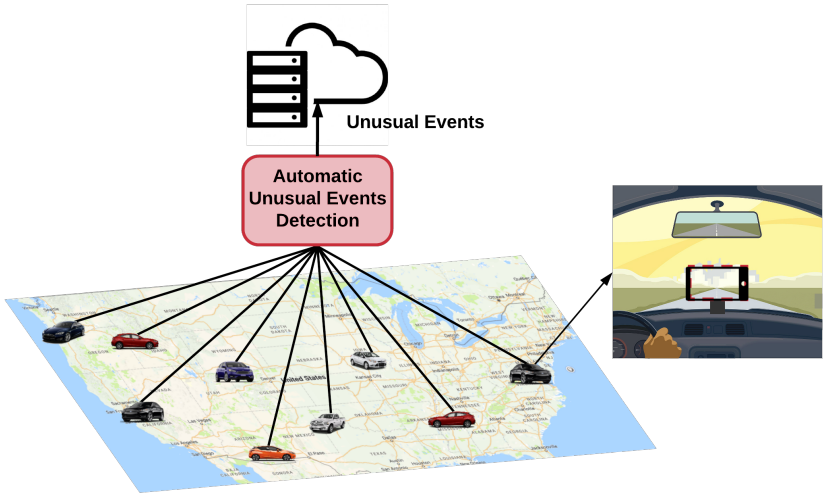


Figure 1.1: System overview of unusual driving events identification

from embedded sensors (e.g., accelerometer and gyroscope), which are useful indicators in sensing vehicle dynamics as well as driver reactions. In particular, it detects sudden driver reactions (e.g., hard braking and swerving), since situations that challenge human drivers are also likely to be interesting test cases for automated vehicles. In detail, a three stage inertial sensing approach is proposed to detect unusual braking and swerving events.

In our three stage approach, we first apply a potential emergency period (braking and swerving) detection to identify candidates and then we use three different features to extract the properties of candidates. At last, we propose a fusion method to combine the advantage of those features. Therefore, by filtering the features from the potential unusual periods, we can easily detect hazardous events from a large data set automatically. By comparing the performance of our method with a strawman solution, we can conclude that this represents an efficient and practical approach to detect unusual events.

Besides, in order to improve the efficiency in processing a large-scale dataset, we also provide an alternative approach to design a parallel streaming data processing pipeline and create a prototype implementation on the Cloud Dataflow

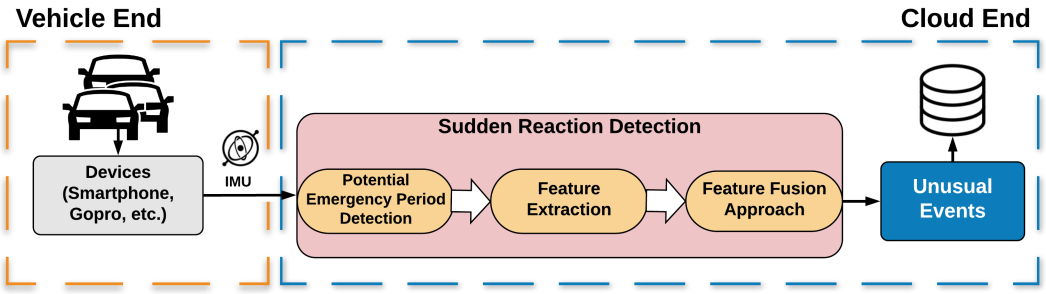


Figure 1.2: Proposed system flow for unusual events identification

service. This reduces the running time and improves efficiency of our system. Overall, we utilize inertial sensors and the parallel pipeline to achieve a scalable and reliable implementation of unusual event detection for massive datasets. The contribution of our work can be summarized as follows:

- Introducing a scalable unusual events collection approach for self driving research and development, which employs human driven vehicles, instead of highly-instrumented vehicles.
- Analyzing more than 120 hours of driving data to evaluate the accuracy of unusual events identification and demonstrating the potential of these detected events to improve the performance of self-driving algorithms.
- Introducing a fusion method which takes full advantage of different features, improving accuracy and performance of detection over strawman solutions.
- Building parallel pipelines for processing large-scale datasets, which improves efficiency greatly.

Thesis statement:

Our low-cost and scalable approach for sudden reaction detection is able to automatically detect unusual driving events from inertial sensors both in batch and streaming modes with high accuracy improvement compared with baseline inertial algorithms.

The outline for subsequent chapters are discussed as follows:

Chapter 2 presents related work in detecting unusual events including camera-based, OBD-based and IMU-based approaches. Current status of self-driving systems will also be discussed, conveying a concise view of self-driving development background.

Chapter 3 describes the implementation of our detecting algorithms. Three stages: potential emergency period detection, feature extraction and feature fusion approach of our detection algorithm are illustrated in detail.

Chapter 4 discusses the mechanisms of cloud processing design for two different modes batch and streaming pipeline. The processing workflow will be demonstrated, revealing the scalability and reliability that it brings.

Chapter 5 presents the evaluation for our sudden reaction detection including the corner cases we have extracted and comparison between features and our fusion method.

Chapter 6 gives the conclusion.

Chapter 2
Related Work

Detecting unusual events is of great importance to driving assistant system developments. Most of the industries pay more attention to unusual events to evaluate the safety performances of their self-driving algorithms. This thesis intends to utilize inertial sensing readings as the indicators of unusual events detection. The background of this will be clarified in three aspects: existing approaches in detecting vehicle dynamics, current technology of self driving and the background related to parallel computing.

2.1 Detecting Vehicle Dynamics

There has been active work on detecting vehicle dynamics while driving both in the research field and in the commercial field [11, 12, 13]. The existing related approaches can be divided into the following categories: camera-based, inertial measurement unit (IMU) based, OBD-based and radar based approaches.

Camera-based approach

At present, some commercial applications have launched on the market such as Mobileye [14] and iOnRoad [15]. Some stereo cameras or mono cameras can provide longitudinal support to estimate the distances to the objects or pedestrians on the road and help drivers keep a safe distance to prevent collisions. However, the performances of cameras are seriously limited when the visibility of road is poor.

To overcome the limitations of objective conditions, some work focus on more

sophisticated instrument. Michael et al. [9] proposed an infrared camera for night view systems, to help drivers recognize dangerous situations, such as pedestrians, animals or obstacles on the road, which performs well even in poor visibility. Furthermore, both research and industry recently take efforts to develop driving assistant applications on smartphones. In [10], the authors developed a mobile phone application named CarSafe to detect drowsiness by observing the drivers' eye movement to alert the drivers with the phone's front camera. [16] and [17] are able to utilize on-board cameras to capture unusual cases, but only cover a subset of challenging situations such as abnormal pedestrian movements and unclear drivable roads.

However, it is undeniable that the camera-based systems are at the cost of high computational power. Additional efforts should be made such as image rectification, disparity image calculation and complex stereo calibration algorithms. Besides, all these camera-based methods require placement for built-in cameras.

OBD-based approach

On-board diagnostics (OBD) systems provide real-time data in addition to a standardized series of diagnostic trouble codes, which help rapidly identify and remedy malfunctions within the vehicle. An inexpensive OBD-II port adapter can be used to obtain the speed of vehicle, which can be read in smartphones via a Bluetooth connection [11].

Radar-based approach Despite of the OBD utilization, radar system is also popular equipment which has been used for a long time. At present, radar systems are applied widely in self-driving vehicles. Radar is computationally lighter than a camera. It can work under various weather which utilized reflection to see obstacles in the behind. The authors of [18] demonstrate that radar is often identified as the best suited sensor for automated braking and pedestrian safety functionality.

IMU-based approach

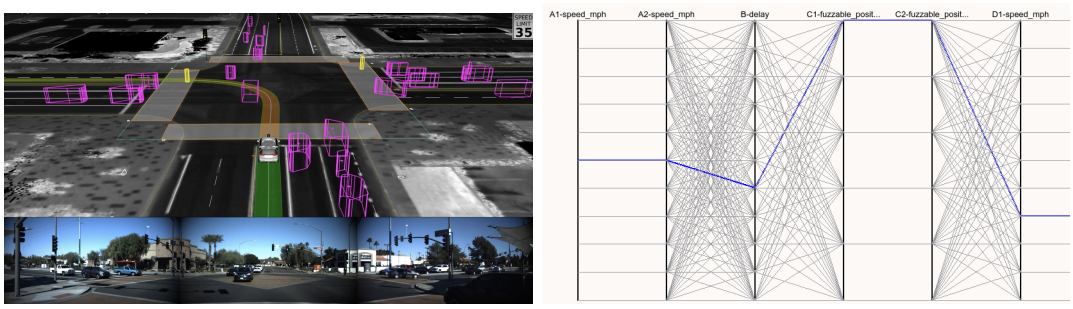
Inertial measurement unit achieves comparable results to the methods above in a much lower cost without influence of the visibility of road. The readings of inertial sensors facilitate a broad array of different research directions to driving safety [12, 19, 20, 21] and road monitoring [22]. For example, sensor readings, i.e., gyroscope and accelerometer from smartphones or other devices can be used to determine driver phone use, which can facilitate many driving safety applications [11]. In addition, there are studies on solving driving safety issues by detecting dangerous steering wheel motions [23, 24].

In [19], the authors propose a novel system, MIROAD, that uses Dynamic Time Warping (DTW) and smartphone-based sensors to detect and recognize drivers' potentially-aggressive actions without external processing. Similarly with MIROAD, Dai et al. [25] use an installed program on the mobile phone to collect and compute accelerations obtained from its accelerometer and orientation sensors to detect abnormal or dangerous driving maneuvers. V-Sense system, which is proposed in [22], is a vehicle steering detection middleware to detect how a vehicle is steered and then alarming drivers in real time. However, such algorithms could only detect known aggressive driving patterns, while not diverse human driver reactions which are naturally performed during emergency periods.

Compared with the research listed above, our research provides a new insight in sensing vehicle dynamics, that uses inertial sensor readings to detect unusual events from sudden human reactions which might cause dangerous accidents. Such detected events could be used for the self-driving companies in simulation tests to gradually refine self-driving software.

2.2 Current Self-Driving Technologies

Detecting unusual events is of great significance to automated driving system developments, since the analysis of such corner cases will be helpful to improve



(a) Carcraft (b) Fuzzing

Figure 2.1: Waymo's simulation software and a fuzzing process to alter variables.

current algorithms [26, 27, 28, 29]. Taking a close look at how advanced self-driving companies work on their models will give a better understanding of the usability of unusual driving events.

In its safety report, Waymo revealed to the public details of the technology for training their self-driving cars [30]. In total, their testing procedure includes three steps: simulation testing, closed-course testing and public road testing. For simulation testing, Waymo built a software called Carcraft which can simulate the virtual scenarios while driving. Each day, 25,000 virtual self-driving vehicles drive up to eight million miles in simulation, refining old skills and testing new maneuvers that help them navigate the real world safely. Crucially, the virtual miles focus on those interesting miles in which the self driving software might learn something new. For now, those interesting miles refer to the most challenging situations that their self driving cars encounter on public roads [31].

The simulation testing also includes thousands of variations on the traffic through a process called fuzzing. By altering the simulated factors, the software can be refined gradually and can more easily handle any complicated situations. Then Waymo validates their new software by private track testing and public testing. And then the cycle begins again. By testing and validating multiple times, their software will have a more comprehensive understanding of those unusual events.

Therefore, such unusual situations play a significant role in automated driving technologies, which are the most interesting parts that should be attached importance in training and testing process. Thus, our work is useful towards improving safety and reliability of self-driving transportation.

2.3 Cloud Computing

As a result of the pursuit for scalable and secure networks benefits, cloud computing has become the forefront of current technology trends [32]. An increasing number of companies are getting involved into this field and obviously have benefited significantly from it [33, 34]. Cloud computing has made it possible for even a small business to afford such reliable and scalable processing capabilities, which seems the preferred solution, on the other hand, for large companies extending their infrastructure or launching new innovations.

Some companies like Netflix increasingly rely on cloud services for their online operations, which allows them to concentrate on their core business ideas [35]. The companies working on social media also migrates their online services to cloud providers, since the cloud infrastructure with a massive storage can easily hold and manage the messages and information. In addition, by providing higher processing power and sophisticated tools, cloud computing enables data scientists to tap into massive data to analyze it for patterns and insights, find correlation, apply complicated algorithms, make predictions and analyze in data backed decision making.

There are multiple kinds of big data tools, some focusing on batch processing such as Hadoop [36] and some on real-time data processing including Kafka [37] and Cassandra [38]. These platforms provide an alternative for processing and managing massive organizational data. From our perspectives, the big data analytics provided by the cloud will be an essential tool for processing sensor data

when considering the goal of a large-scale collection. Streaming pipelines will help when the real-time driving sensor data is continuously tapped in.

Most frameworks only have one specialty, while Google Cloud Dataflow service, which is utilized in this thesis, is multifunctional. It integrates both of the stream and batch modes, allowing users to build pipelines, monitor their execution and analyze data on the cloud. It reports that Dataflow has replaced MapReduce gradually inside the Google with a fact that Dataflow shows a better performance in handling multipetabyte datasets [39].

We believe that such powerful and hyper-scale infrastructure will allow efficient processing of massive data , which simplifies the procedure in data management and analyzing. Thus, the sensor data can be uploaded and processed on the cloud automatically and large-scale unusual driving events will be identified more easily.

Chapter 3
Detection Algorithms

In this chapter, we mainly introduce our design and implementation of the automatic driving corner cases detection approach based on inertial sensors. Our approaches: **Sudden Reaction Detection** uses three steps: candidate period detection, feature extraction and feature fusion approach. We detail them below.

3.1 Overview

Since driving miles with self-driving prototypes are limited, we are looking forward to a more scalable solution which can be applied on large numbers of vehicles to detect the interesting miles. Our method starts from the identification of human reaction, since the unusual situations that surprise a human driver are more likely to challenge an automated driving system than those "boring" driving situations. Because of the deployment on conventional human-driven vehicles that allows reaching the necessary scale much more quickly, the system can make use of detailed measurements of the human driver's sudden steering and braking reactions to road events.

The unusual situations which trigger human driver's reactions like hard braking and high speed swerving usually involve large accelerations and angular speed. Such features could be captured by the accelerometer and gyroscope of an Inertial Measurement Unit (IMU) available in many phones, cameras, and cars. The detector is triggered when the feature score exceeds a threshold, which can be chosen as a percentile of the feature value. Therefore, to identify such sudden reaction

events, we propose a three-stage inertial sensing detection technique leveraging the IMU potentially available within vehicles. The three-stage detection mechanism includes candidate period detection, feature extraction and feature fusion approaches.

To further filter out human driver's unusual driving events, IMU's reading should be evaluated in terms of *Amplitude* and *Urgency*. *Amplitude* refers to the peak values in the IMU's reading, as the moments which have significant acceleration and angular speed will be more likely considered as unusual. *Urgency* quantify the time window in which there are one or multiple peaks, since the movements like high speed swerving and hard braking usually peak readings within in short time period.

In our three-stage detection, as unusual events (e.g., animals running across the road and other vehicles sudden inserting) often involves sudden reactions of drivers, such as hard braking or sudden steering, the system first detects these candidate period based on the inertial sensor readings. In the second step, the system extracts various features, trying to indicate the unusuality of detected human reactions. Finally, a fused model is designed to combine the properties of each feature in an accuracy-driven way.

3.2 Candidate Period Detection

Human drivers tend to perform emergency reaction like hard braking or swerving to avoid accidents, when unusual situations happen. This motivates the first stage of our detection mechanism to identify candidate periods in terms of braking and swerving events which have relatively large readings on accelerometer and gyroscope.

Without lose of generality, the pose of IMU within the vehicle should be assumed unknown. Therefore, coordinates alignments will be performed to project

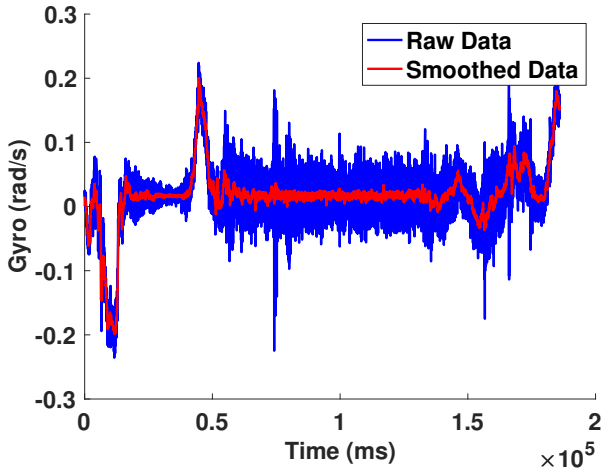
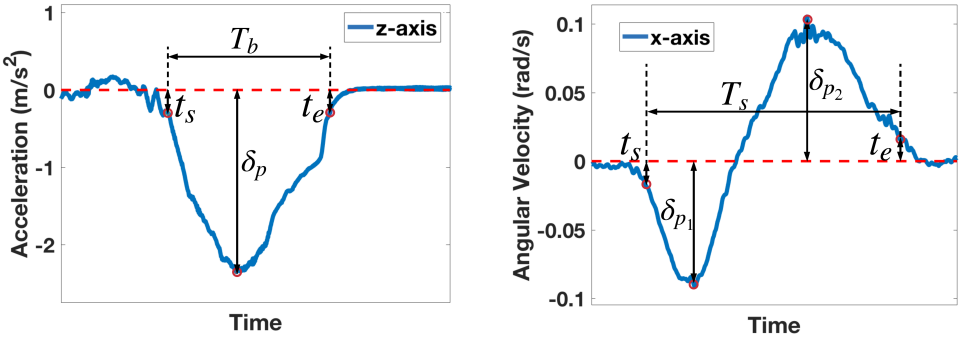


Figure 3.1: Data illustration before and after smoothing



(a) Braking Event

(b) Swerving Event

Figure 3.2: The accelerometer and gyroscope readings on sample braking and swerving event.

IMU's reading from its own coordinates system to vehicle's coordinates system. As vehicle will only get gravity while stationary and will have dominate acceleration component in driving direction, vehicle's coordinates system could be determined by a period of acceleration [11].

Besides, we utilize a low pass filter to remove the noise from the raw IMU's reading caused by vehicle vibrations and bad road conditions. A example with comparison between raw gyroscope data and smoothed data is shown in Figure 3.1.

Braking Events Detection.

Generally speaking, when a driver braking, a large acceleration can be observed in the opposite direction of the driving direction. Figure 3.2(a) shows the acceleration reading trace of a braking event, in which a negative bump can be observed due to the press of braking pedal. Thus, such bumps observed on acceleration reading can be used to identify braking events.

In order to accurately capture the bumps that are actually caused by braking events, a peak detection method is applied first to find all the negative peaks of the accelerometer readings on the driving direction, whose value is defined as δ_p to quantify the amplitude of braking event as shown in figure 3.2(a). A threshold $min\_braking$ is used to remove noisy peaks such as the ones caused by road bump vibrations. The noisy peaks will be effectively filtered out other confounding movements such as slight vibrations caused by road bumps.

After each effective peak of braking events is detected, the system searches forward and backward to find two positions , which are the first point of which the acceleration value is less than a threshold $time\_thres$. It is used to avoid acceleration noise caused by vehicle fluctuations and improve the overall detection accuracy. We define the the starting point and ending point of this event as t_s and t_e respectively, as illustrated in figure 3.2(a). Besides, the difference between t_s and t_e is defined as the duration (T_b) of a braking event.

Algorithm 1 Braking Event Detection

Input: A (z axis Accelerometer readings), $min\_braking$, $time\_thre$

valleys = all the local minima δ_p where $|A| > min\_braking$ **for** *each valley* **do**

(Searching backward) **if** $|A| < time\_thre$ **then**

Record the start time t_s

end

(Searching forward) **if** $|A| < time\_thre$ **then**

Record the end time t_e

end

return *braking* with δ_p, t_s, t_e

end

Swerving Events Detection.

During swerving events, a driver usually first turns the steering wheel to one direction quickly and then turn back to the other direction. This action will result in two consecutive bumps in opposite directions of the gyroscope readings, as shown in Figure 3.2(b). Therefore, we capture such characteristics by thresholding peaks on gyroscope reading for peak detection, and then filter out effective swerving based on the time interval between two opposite direction bumps. Similar with the brake detection, δ_p, t_s, t_e are defined as the amplitude of the first peak, starting point and ending point of swerving events.

To detect swerving events and distinguish such events with two consecutive turns, for each pair of consecutive bumps in opposite directions, we calculate the time interval between the ending time of the front bump and the starting time of the rear bump, as illustrated in Figure 3.2(b). If the interval is smaller than a threshold, this pair of consecutive bumps can be quantified as a swerving event.

Also, the duration (T_s) of a swerving event is defined using the difference between the start of the front bump and end of the rear bump.

Algorithm 2 Swerving Event Detection

Input: G (x axis Gyroscope readings), $min\_swerving$, $time\_thre$,$dura\_thre$

positive bumps = all the positive bumps with δ_p, t_s, t_e, where $|\delta_p| > min\_swerving$

negative bumps = all the negative bumps with δ_p, t_s, t_e, where $|\delta_p| > min\_swerving$ **for** *each positive bump* **do**

  for *each negative bump* **do**

    if $t_{s2} - t_{e1} < dura\_thre$ **then**

      return *swerving* with δ_p, t_{s1}, t_{e2}

    end

  end

end

3.3 Feature Extraction

Based on the detected candidate periods, we first describe a **Strawman solution** using *amplitude* of sensor readings as feature to detect unusual events:

Amplitudes. Due to the large accelerations or gyroscope readings while unusual situations, using amplitudes of such reading as features seems to be an intuitive solution. Specifically, the sudden braking events and swerving events can be detected based on a threshold δ_p value. Ideally, the top 5 or 10 percentage of the sorted candidate periods should have a high detection accuracy for unusual events.

- **Amplitudes of acceleration** Drivers tend to brake when facing unusual events, which might reach a high deceleration. By filtering the accelerometer readings on the direction of driving, calculating the amplitudes of acceleration during a brake will be an intuitive way. The detected braking events should be sorted descendingly based on their amplitude value of acceleration referring to δ_p in Figure 3.2(a).

- **Amplitudes of angular velocity** Similarly, a driver tends to perform a hard swerving to avoid hitting unexpected obstacles lying on the road. By processing gyroscope readings and finding the amplitudes of angular velocity, we might be able to find out unusual events from detected swerving events. Similarly we should sort swerving events based on the amplitudes of gyroscope (δ_p) and filter the corner cases in high-amplitude driving period. The difference from amplitudes in braking is that there are two peaks detected in a swerving event thus we will have two peaks' amplitudes on opposite directions. In this case, we only utilize the first amplitude of the peak to quantify the amplitudes of the swerving event, since the first reaction is likely to be the case that we are more interested in.

However, this solution does not work well in practice because the majority of braking events with large acceleration and swerving events with large angular velocity are usual events including normal braking when facing red traffic lights and normal swerving like lane changes. Considering the urgency of unusual events, we propose **Derivative-based** and **Duration-based** features to further characterize detected events.

We now discuss intuition and methodology of how to extract the urgency features for both braking events and swerving events:

Derivatives. Unusual events do not always come with high-amplitude braking. Estimating the urgency of a braking or swerving event is also important to determine unusual events.

To this end, we calculate the derivatives of acceleration and gyroscope to represent the urgency of a braking event and a swerving event.

- **Derivatives on accelerometer readings** Instead of considering the specific amplitudes of acceleration, determining how fast the driver brakes matters. It requires us to calculate the derivatives of acceleration to show how

urgent a braking event might be. To calculate derivatives, we use the difference of amplitudes of every five sample points to divide the difference of time. A derivative value at m can be calculated through Equation 3.1. Here, A represents the readings on z-axis of accelerometer and t represents the time.

$$d_b(m) = \frac{A(m+5) - A(m)}{t_{m+5} - t_m} \tag{3.1}$$

- **Derivatives on gyroscope readings** Calculating the derivatives on gyroscope readings during swerving events is also an efficient way to detect sudden swerving events. Sometimes we pay more attention to how quickly drivers react to unusual events which can be reflected via the derivatives. The calculation is similar with the braking events. We also calculate the derivatives for each sample point during the whole event, as shown in Equation 3.2. Here G is the x-axis readings of gyroscope.

$$d_s(m) = \frac{G(m+5) - G(m)}{t_{m+5} - t_m} \tag{3.2}$$

Duration. As a sudden braking or swerving event often happens in a short moment, we can also use duration to depict the urgency of unusual events. As shown in Figure 3.2, the duration of a braking or swerving event are represented by T_b or T_s correspondingly, which is equal to the interval between t_s and t_e.

- **Duration of a braking event** As a sudden braking event often happens in a moment, we can also use duration to depict the urgency of unusual events. Duration of a braking event should be sorted ascendingly to allow focusing on rapid braking, which is more likely to be an unanticipated event.

$$T_b(m) = t_e(m) - t_s(m) \tag{3.3}$$

- **Duration of a swerving event** Apparently, a sudden swerving event takes much shorter than a normal swerving event. So calculating the duration of a swerving event referring to T_s in Figure 3.2(b) seems an intuitive approach when considering an evaluation in terms of urgency. Since there are two peaks in a detected swerving event, to calculate the duration, we use the difference between the start time of the first bump and the end time of the second bump.

$$T_s(m) = T_{2e}(m) - T_{1s}(m) \tag{3.4}$$

3.4 Feature Fusion

Based on preliminary results, we find that the duration based and derivative based approaches work better than the Strawman solutions that rely on the amplitude of sensor readings. However, the detected unusual events from one approach tend to have less shared events with other approaches. To effectively take advantage of all three features, we propose a feature fusion mechanism to combine the three extracted features (amplitude-based, duration-based and derivative-based) in order to get better performance for filtering unusual events. We design a *accuracy driven weight assignment* method to assign weights to different features based on their detection accuracy. The principle underlying this method is illustrated in Equation 3.5.

$$\begin{aligned} f_{fusion} &= \sum_i w_i f_i \\ w_i &= \frac{n_{f_i}}{\sum_i n_{f_i}} \end{aligned} \tag{3.5}$$

The fused feature value (f_{fusion}) of a candidate period is equal to the sum of each feature value (f_i) multiplied by its weight (w_i). The value of each feature is normalized to a Gaussian distribution with the same mean and variance. The weight of each feature (w_i) is calculated based on the detection accuracy.

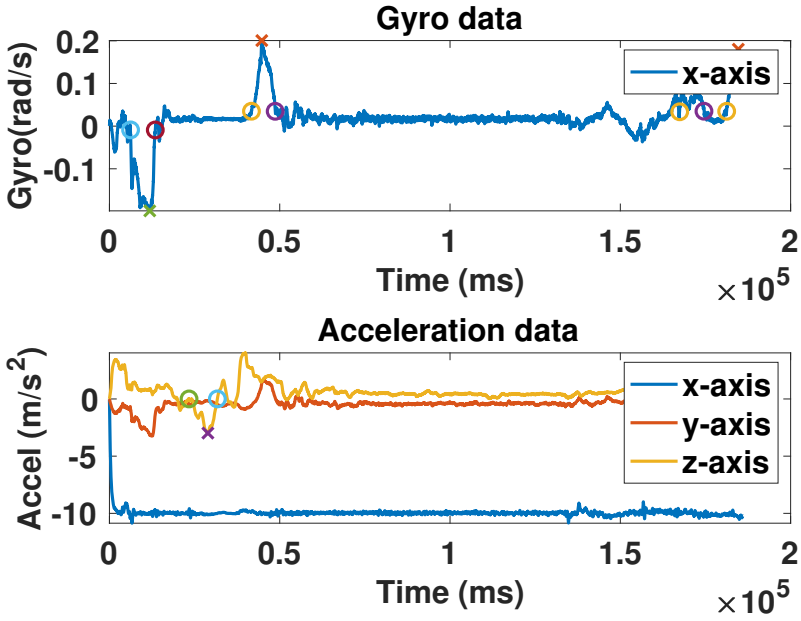


Figure 3.3: Accelerometer and gyroscope readings in the same video

Specifically, for each feature, we extract the top 5% potential unusual events and determine the actual detected unusual events for each feature (n_{f_i}) based on the manual review. We divide n_{f_i} by the sum of detected unusual events for all features to calculate the corresponding weight (w_i). With this method, the system assigns a higher weight for features with better detection accuracy, while assigning a lower weight to the one with worse detection accuracy.

3.5 Implementation

We implemented the algorithms to process inertial data from Gopro HERO5 cameras. The recorded videos contain the associated inertial data. We extracted accelerometer, gyroscope and GPS readings using Go scripts. Based on the comparison between the events in videos and accelerometer/gyroscope readings, we can determine the approximate inertial coordinates. By tracking the bumps in accelerometer readings when the braking events and turns happen, those events can be detected on each axis. As shown in Figure 3.3, the bumps on Z-axis are

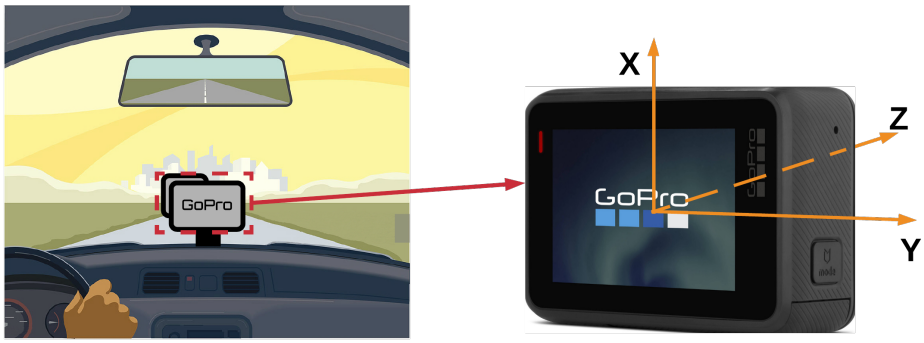


Figure 3.4: Inertial sensor coordinates of GoPro

related with braking, the bumps on Y-axis are related with the turns and the acceleration on X-axis is usually fluctuating around $-10m/s^2$ which implies the direction is correspondent to the gravity. In terms of gyroscope coordinates, we detect bumps on X-axis when turns occur. Thus, the approximate coordinates of the inertial sensor in GoPro are determined.

The inertial processing can be implemented directly in vehicles or in the cloud. We focuses on the latter and describe this implementation next.

Chapter 4
Cloud Processing Design

This chapter clarifies how we utilize the full-managed service: Google Cloud Dataflow to transform and process large-scale data, simplifying the mechanics of parallel data processing as well as improving efficiency and scalability. In this chapter, general overview of cloud processing design will be presented at first and this is followed by our implementation of the inertial algorithm in both batch and streaming mode.

4.1 Background

Commonly, our unusual event identification algorithm can be easily implemented and processed on the mobile end. Naive batch processing can require long time for a large dataset. Now we also provide an alternative solution for cloud processing, which allows a long-period data persistence with only one-time uploading. It will further allow users to reprocess the data in the future with even more complicated algorithms. The approach of cloud processing will accelerate processing duration and optimize data management in terms of large quantity of dataset. For real-time data processing, such a practical platform is also applicable in achieving real-time unusual events detection which provides possibilities for different kinds of user cases.

Cloud Dataflow is a unified programming model providing powerful service in developing and executing wide varieties of parallel data processing patterns. It enables developers to build up processing pipelines, transform and analyze

their data both for batch mode and streaming analytic amongst its capabilities. Dataflow allows users only concentrating on the logic composition of data executing job. With Google Cloud Pub/Sub, Dataflow can also take data in publish-and-subscribe mode which enables real-time data processing and analyzing.

In our system, streaming and batch data modes are both achieved in processing inertial sensor data for sudden reaction detection using Cloud Dataflow in a straightforward way. For the streaming pipeline, we use Cloud Pub/Sub to ingest IMU data from real-time driving events. For the batch pipeline, IMU data is imported from Cloud Storage. Both of the input data will be processed using Dataflow and results will be stored to Cloud Storage or Cloud BigQuery in two pipeline modes as well. At last, unusual events will be easily filtered after analyzing and will be stored in cloud.

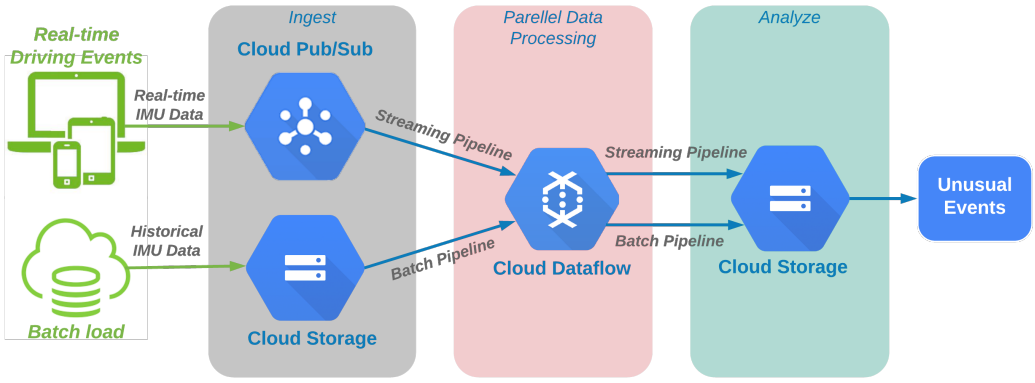


Figure 4.1: Overview of cloud processing design

4.2 Batch Processing

Cloud Dataflow

Cloud Dataflow builds a *pipeline* which represents a data processing job. It handles data of varying patterns and structure using a format called *PCollection*,

a parallel extension of a Java collection whose elements can be spread over multiple physical machines. Dataflow also includes a library of parallel transforms, or *PTransforms*, which allow high-level programming of processing operations using basic templates, including performing mathematical computations on data, converting data from one format to another and etc.. In addition, it supports diverse sources and sinks for *pipeline I/O* which provides read and write transforms for a number of common data storage types. The service optimizes processing tasks – for example, by reducing multiple tasks into single execution passes with equal reliability and expressiveness.

To process different elements in parallel, Cloud Dataflow will automatically partition data and distribute them to different compute engine instances. Then the aggregation operation will combine the data across the dataset, including data which may be spread across multiple workers. It will generate a workflow graph based on the created *PCollection* and applied *PTransforms* with optimized resource usage. Cloud Dataflow also include automatic tuning features, such as autoscaling and dynamic work rebalancing, to optimize data distribution and resource allocation.

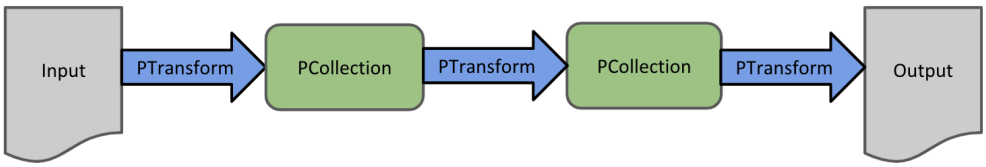


Figure 4.2: A linear pipeline with three sequential transforms

Our dataset includes data from 1430 videos in total. The data from each video involves its acceleration and gyroscope readings respectively, formatted in *CSV* documents. We will show how we process the sensor readings using Dataflow to achieve sudden reaction detection on those videos.

Pipeline Flow Design

Figure 4.3 shows the mechanics of how we utilize *PTransforms* to ingest the input, and how we transform the *PCollection* to the required results. Based on the objective that we are trying to process the videos in parallel, our *PCollection* should be a distributed dataset which holds a single trip as a single element. This allows fully parallel processing, where the processing time for all the trips in 1430 videos could be the same as the time to process one trip. In practice Cloud Dataflow chooses a lower degree of parallelism by default.

To achieve this, we divide our batch processing pipeline into two steps: dataset organizing and parallel computing. Since the Dataflow *TextIO.Read* transform returns a *PCollection* of Strings, each corresponding to one line of an input text file, transforms must be done to turn it into the reasonable format. To alter the lines of data to trips, we need to apply *PTransforms* to organize our dataset at first. Then when the *PCollection* of trips is ready, we can continue to working on the algorithms that process each element and write the results.

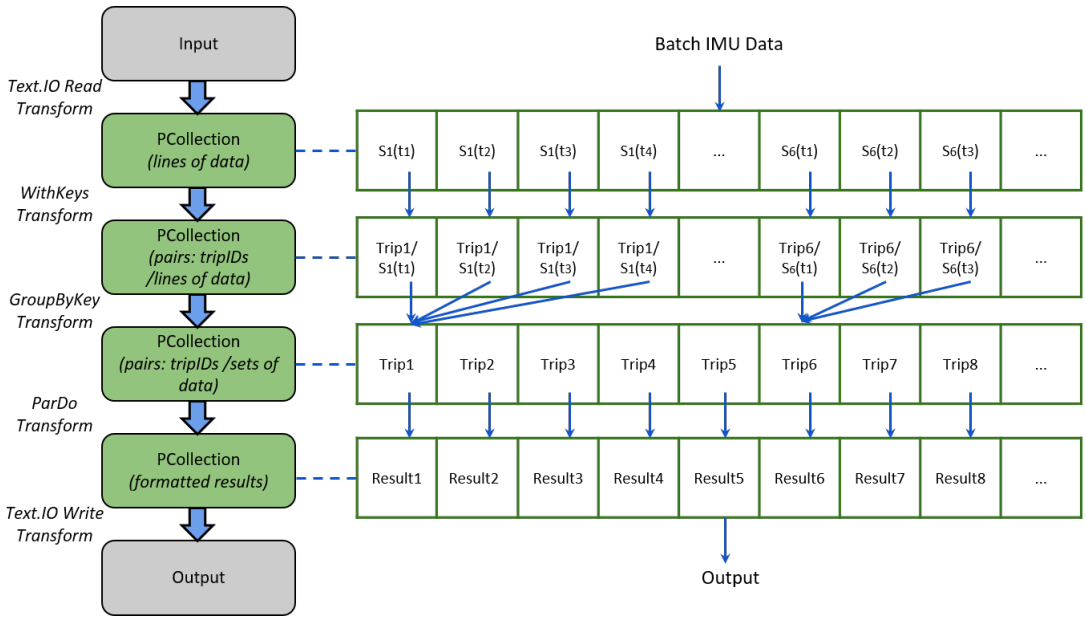


Figure 4.3: *PCollection* with transforms

- Dataset organizing

  At first, we read IMU data from Google Cloud Storage using a transform

called *TextIO.Read*, which reads the text file and returns a *PCollection* of String. Each String in the resulting *PCollection* represents one line from the text file. Then we use the *WithKeys* Transform to assign keys: tripID to each element in *PCollection*, in order to prepare for combining the elements from one trip file to a single element of *PCollection*. *GroupByKey* transform, groups all the lines of data into a set based on the keys. Thus, we transform a *PCollection* of lines into a *PCollection* of tripID/sets of data pairs. Each element in the *PCollection* represents a single trip which allows us to parallel process all the trips.

- Parellal computing

  After organizing dataset, a *ParDo* transform is used to perform potential emergency period detection and extract the features as the output. Finally, *TextIO.Write* method allows users to write the results to Cloud Storage as well.

The whole pipeline of our sudden reaction detection runs 12min 49s, which reduces processing duration significantly comparing with sequential file-by-file processing. Even with large scale dataset, our sudden reaction detection can be executed and monitored under Dataflow with higher efficiency.

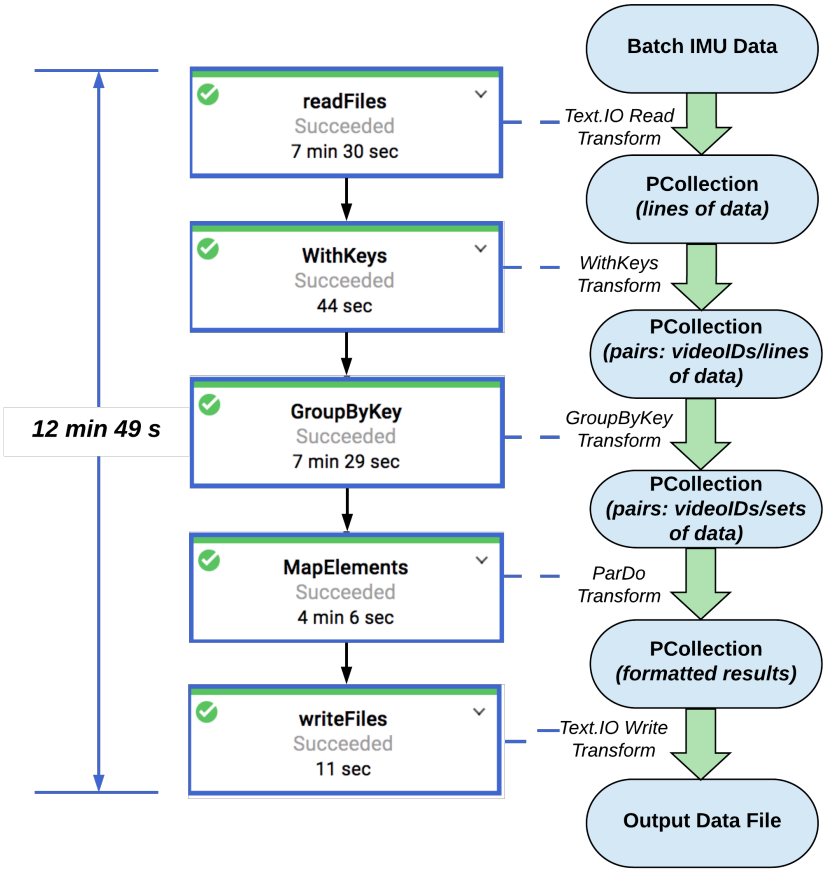


Figure 4.4: Batch Pipeline Design

4.3 Streaming Processing

Cloud Pub/Sub

In order to build streaming pipeline, instead of ingesting data directly from the batch storage, we have to alter to other solutions to read the input. Cloud Pub/Sub is such a powerful tool when it comes to streaming processing.

Cloud Pub/Sub is a reliable and scalable tool for stream analytics and event-driven computing systems. It ingests streams from events and delivers them to Cloud Dataflow for processing. Cloud Pub/sub allows secure and durable communication between different independent applications, that helps developers

quickly integrate their workflows and data streaming.

To send messages using Pub/Sub, it relies on two key concepts: topic, which is the entity that represents a feed of messages, and subscription, the entity that represents an interest in receiving messages on a particular topic. In practice,a publisher will create messages or data and publish them to a topic. A subscriber will receive the messages on the subscription.

Pipeline Flow Design

The overall streaming pipeline flow is similar with the batch one. There are two main differences between them. First, without using Cloud Storage, all the streaming inertial sensor data must be uploaded to the cloud in a real time way. Instead of using *TextIO.Read*, we use *PubsubIO.Read* to read data from a topic. We also implement the function to automatically send the stream data to a topic from the mobile end. Note that we don't create subscriber function here since Dataflow automatically creates and manages a subscription behind the scenes. Thus, the *PubsubIO.Read* transform continuously reads from a Pub/Sub stream and returns an unbounded *PCollection* of Strings that represent the data from the stream. Additionally, each element in the resulting *PCollection* is encoded as a UTF-8 string by default.

The other main difference is that we add an important transform called *Window Transform* on the data read by *PubsubIO* before any other parallel transforms. By defining a window on the streams, Dataflow transforms that aggregate multiple elements, such as GroupByKey and Combine, will work implicitly on each defined window. In particular, they process each *PCollection* as a succession of multiple finite windows, though the entire collection itself may be of unlimited or infinite size. There is another related concept called *Triggers* which is used to determine when to "close" each finite window. Using a trigger can help to refine the window transform, especially in dealing with late-arriving data or to provide early results.

After adding a Window transform with correlated triggers, it assigned the elements in an unbounded *PCollection* to one or more windows, and each individual window contains a finite number of elements. Then, different with grouping data in a same trip in batch processing, the streaming *GroupByKey* transform will group the elements in a same window. Then *ParDo* function will process on a per-window basis. In our system, we don't actually test the streaming approach on real world vehicles, but simulate the process with our batch dataset. It shows the competitive performances compared with the processing on mobile end in terms of detection accuracy.

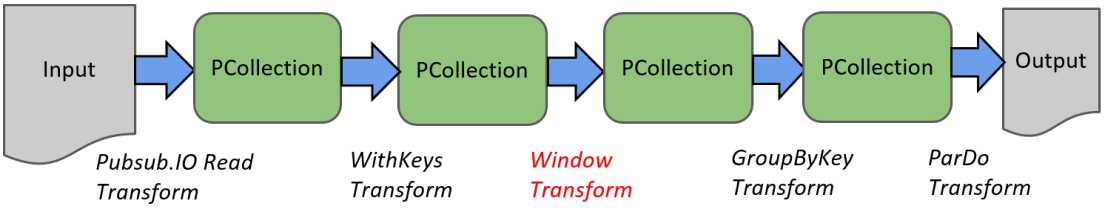


Figure 4.5: Streaming Pipeline Design

4.4 Implementation

Manage Cloud Storage

Before parallel processing through Dataflow, we first need to upload inertial sensor readings to Cloud Storage. Here we use the *gsutil* tool, a Python application that allows users to access Cloud Storage from command line.

We first use *gsutil* to create buckets, which are the basic containers to hold data. Then the *gsutil cp* command is used to upload an object to buckets. Here we upload all the inertial sensor data including accelerometer and gyroscope data to the bucket. Besides, the *gsutil* command-line tool also allows to download objects from buckets and list the detailed information of objects or buckets.

Hence, the dataset has been stored in Cloud Storage which only takes a short period to upload and can be processed anytime. For further grabbing the input,

gsutil uses the prefix *gs://* to indicate a resource in Cloud Storage. Specifically, we use *gs://[BUCKET_NAME]/[OB-JECT_NAME]* to access any data we want.

Chapter 5

Evaluation

Here comes the evaluation chapter. We will first discuss the dataset for our batch processing and then present our detection accuracy of our fusion approach and strawman solution. Then we will illustrate the detailed comparison results between all the features.

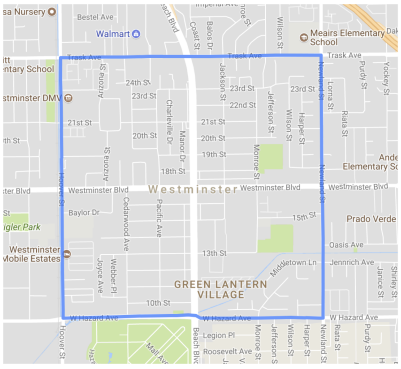
5.1 Dataset Description

The sudden reaction detection is evaluated with a dataset including 120 hours of videos collected in Los Angeles, CA, including 1430 videos in overall. In this dataset, we use GoPros mounted at the bottom center under the windshield to record the full driver's front view videos with 1280×720 resolution at $30Fps$.

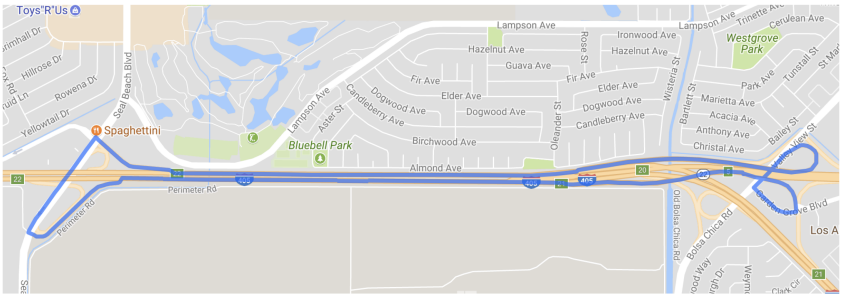
The dataset is collected by ten different drivers under different road situations, including urban roads, highway roads in both daytime and nighttime as shown in Figure 5.1. The driving ranges and routes are also shown in Figure 5.2. The shape curved in 5.2(a) is the range in where the drivers collect urban road data. 5.2(b) shows the routes of highway driving data.

(a) Urban road (b) Highway road (c) Nighttime-view road

Figure 5.1: Different road situation samples



(a) Urban Road Map



(b) Highway Map

Figure 5.2: Experiment driving route for the Los Angeles dataset.

Metadata Extraction

To extract inertial sensing data from embedded sensors in Gopros, we follow the tutorials on its official website, in which case we are able to acquire all the necessary sensor readings from the videos. The details of sensing data which can be extracted are listed as follows: (a) 400 Hz 3-axis gyroscope readings, (b) 200 Hz 3-axis accelerometer readings, (c) 18 Hz GPS position (latitude/longitude/altitude/speed), (d) 1 Hz GPS timestamps, (e) 1 Hz GPS accuracy (cm) and fix (2d/3d), (f) 1 Hz temperature of camera.

5.2 Unusual Event Detection Accuracy

Our system can achieve high unusual event detection accuracy for both sudden reaction detection and unfamiliar view detection, as shown in Table 5.1. Setting 98 percentile of the fused feature value as the threshold, sudden reaction detection could achieve 53.16% and 63.16% accuracy for unusual braking events and swerving events respectively.

Our evaluation shows the system can detect unusual events from a large dataset only leveraging the accelerometer and gyroscope readings from the in-vehicle motion sensor. The results also demonstrate that the proposed detection method outperforms the Strawman detection approach and has a much higher precision to extract unusual events.

Methods	Accuracy (%)	Strawman (%)
Sudden Reaction Detection for Braking Events	53.16	29.11
Sudden Reaction Detection for Swerving Events	63.16	31.58

Table 5.1: Unusual event detection accuracy versus strawman solution accuracy for two methods.

To demonstrate the performance of the sudden reaction detection method, we compare the proposed feature fusion detection method with the Strawman solution as well as the approaches which filter unusual events based on the derivative and duration feature individually. Specifically, detected potential emergency periods are sorted in terms of the amplitude of accelerations, derivative of accelerations, duration of braking events, and the fused value of all the features respectively for braking event.

We evaluate all the extracted features among which only the duration of braking events is ranked descendingly and the other three are ascendingly.

Different from other features, we use 5-fold cross validation to evaluate the

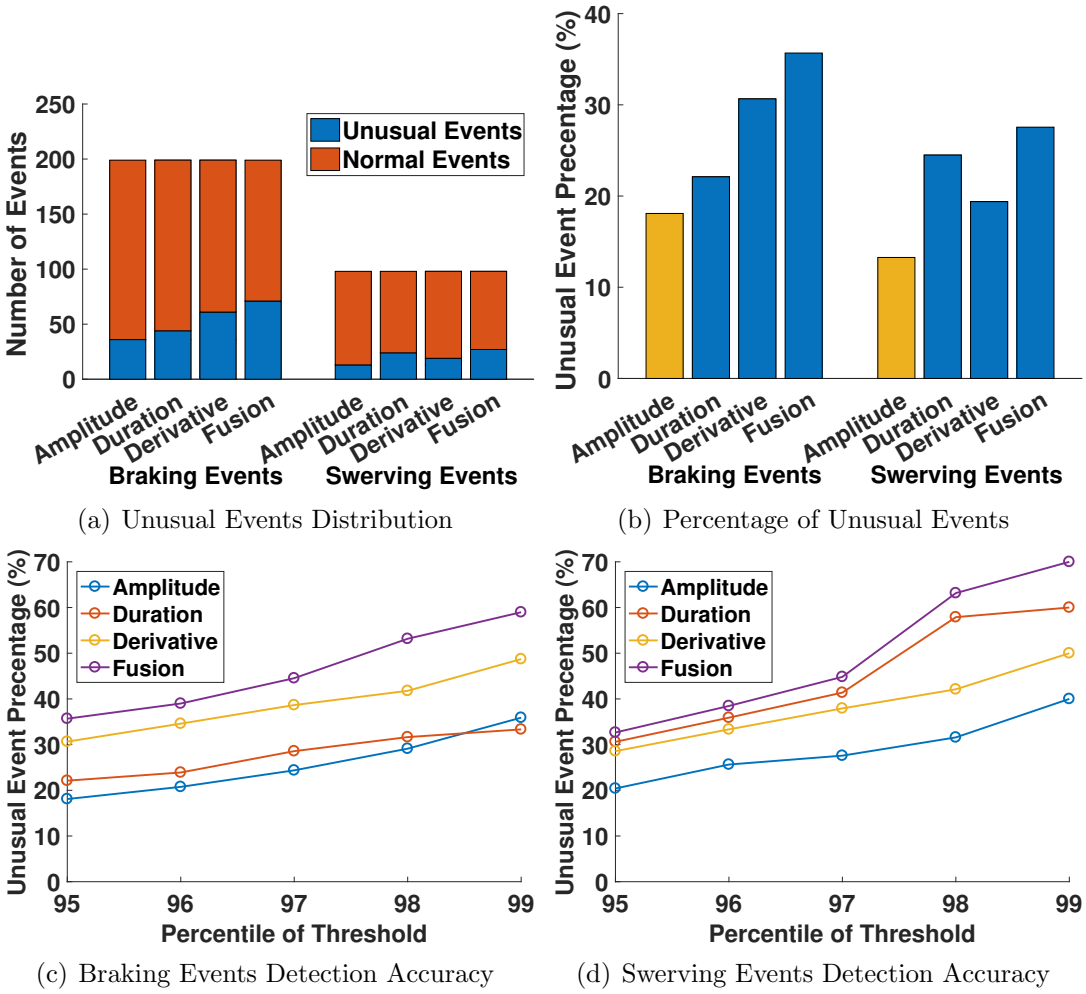
performance of the feature fusion method by generating weights from four subsets and estimate feature fusion value of the other subset.

The whole dataset is randomly partitioned to five subsets. Four subsets are assigned as training set to generate weights for each features, and the remaining subset uses the weights to calculate the feature fusion value. We repeat this subset assignment five times to use each subset as test set once, thus all braking events will have a feature fusion result.

Then, unusual braking events are identified by thresholding the four metrics values with a threshold from top 95 percentile to 99 percentile. Same process also applied for swerving detection based on gyroscope reading.

We find 3987 braking events and 981 swerving events in total over the 120 hours driving data. Thresholding the 95th percentile of braking and swerving feature values will filter out 199 braking events and 98 swerving events respectively, The number of unusual events among them are shown in Figure 5.3(a) and the corresponding percentage is shown in Figure 5.3(b). The unusual events are manually labeled in terms of whether the driver was surprised to perform sudden reactions, and sample unusual events are showed in Figure 5.4. Those samples include: (1) the driver avoids a black plastic bag flying across the front view so the driver takes a hard lane change, (2) the driver turns the steer wheel quickly since another vehicle interrupts to his lane emergently, (3) the driver takes a hard brake since he was disturbed by the strong sunlight and does not see the front vehicle clearly.

Among 199 detected braking events, feature fusion approach extracts 71 unusual braking events, which surpasses the amplitude (36 detected), duration (44 detected) and derivative (61 detected) based approaches. Similarly, 27 sudden swerving events are detected out of 98 chosen swerving events with the feature fusion approach, which is better than other approaches (13, 24 and 19 sudden swerving events detected correspondingly). Since the length of unusual events



(a) Unusual Events Distribution

(b) Percentage of Unusual Events

(c) Braking Events Detection Accuracy

(d) Swerving Events Detection Accuracy

Figure 5.3: Evaluation results of sudden reaction detection.

captured by feature fusion is 0.25 hours out of 120 hours driving, our sudden reaction detection can largely save the bandwidth by only uploading detected unusual situations.

(a) Swerving to avoid s tire segment

(b) Braking due to a cut-in vehicle

(c) Braking due to strong sunlight

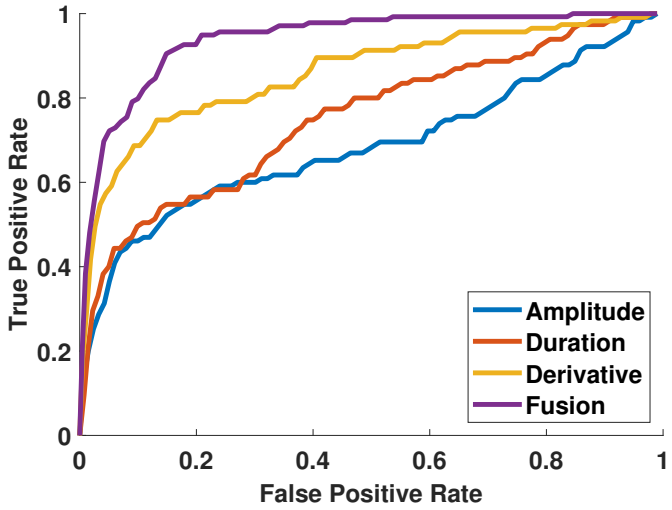
Figure 5.4: Examples of detected unusual events

To further explore the relationship between sudden reaction detection accuracy and percentile of threshold value for each method, we plot the accuracy for braking events detection in Figure 5.3(c) and swerving events detection in Figure 5.3(d).

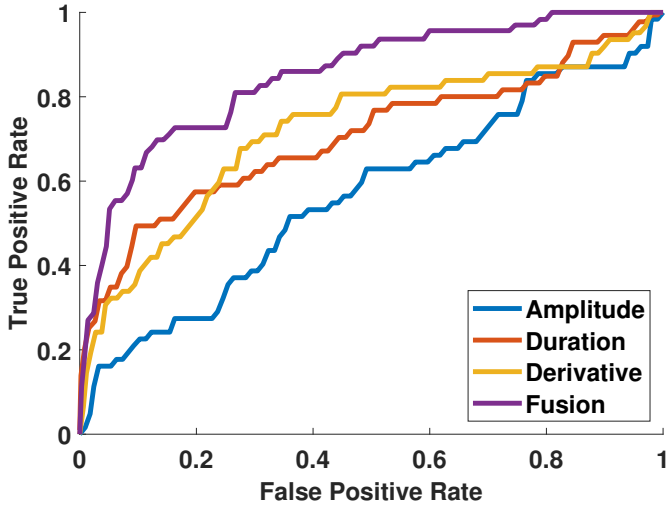
As shown in Figure 5.3(c), while the detection percentage of all four features increases, the feature fusion approach has the fastest increasing speed. The detection precision of unusual events of the fused approach reaches 60% when setting 99% percentile of threshold. In the swerving case, the fused feature shows similar performance with the duration feature, while it is still significantly better than the amplitude and derivative approaches. By selecting a different partition of overall braking and swerving events, we show that most of the unusual events are centralized on the top part after ranking them based on the features.

Besides the precision improvement, our system also achieves a high estimated recall compared to the baseline approaches. Due to the large amount of manual effort needed to label the large video dataset with ground truth, we limited labeling to 40% of events with at least one high feature value (a total of 1878 labeled events) and calculate recall over this dataset as an estimate for overall recall. Figure 5.5 shows the ROC curve of our braking events detection and sudden swerving events detection methods correspondingly. We can observe that the feature fusion approach achieves the highest area under the curve (AUC) value, which indicates the performance improvement compared to baseline approaches.

In conclusion, we can observe that as the percentile of threshold increasing, the detection accuracy all of the four methods are rising. Among the four approaches, our proposed fusion method performs better than the other three methods.



(a) Braking Events ROC Curve



(b) Swerving Events ROC Curve

Figure 5.5: ROC curve of proposed method for sudden reaction fetection.

Chapter 6
Conclusion

Our system aims to automatically identify unusual driving events to allow scaling the collection of driving data and corner cases to a much larger fleet of human-driven vehicles without requiring upload or human review of all data. The proposed system is able to capture various unusual circumstances, including hazardous event like sudden braking and swerving events through a three-stage process involving inertial sensing. The evaluation is based on more than 120 hours of real road driving data and shows that it outperforms baseline methods on unusual event with an 82% accuracy improvement over baseline for braking events and a 94% accuracy improvement for swerving events on sudden reaction detection. The event identification process requires only inertial measurements and front view driving videos, allowing collection of data from smartphones or dashcams. All the detection algorithms can be processed parallel on the cloud, achieving a scalable and reliable implementation relying on Google Cloud Dataflow service. Thus, the light-weight design and minimal infrastructure requirement of this approach will allow large-scale unusual driving events identification. We hope that an extensive dataset of driving corner cases collected with this approach would provide a better understanding of potential limitations of current systems and accelerate the development of robust automated driving technology.

References

[1] "Google creates new self-driving car company," 2016.

[2] "Some of the companies that are working on driverless car technology," 2018.

[3] "Toyota to invest $500m in uber for self-driving car programme," 2018.

[4] "Tesla just dropped full self-driving mode from the model 3," 2018.

[5] "Self-driving uber vehicle strikes and kills pedestrian," 2018.

[6] "Tesla model s reportedly on autopilot crashes into fire truck at 65 mph, no injury," 2018.

[7] "Tesla's autopilot was involved in another deadly car crash," 2018.

[8] L. Liu, H. Li, J. Liu, C. Karatas, Y. Wang, M. Gruteser, Y. Chen, and R. P. Martin, "Bigroad: Scaling road data acquisition for dependable self-driving," in *Proceedings of MobiSys 2017*. ACM, 2017, pp. 371–384.

[9] M. Grimm, "Camera-based driver assistance systems," *Advanced Optical Technologies*, vol. 2, no. 2, pp. 131–140, 2013.

[10] C.-W. You, N. D. Lane, F. Chen, R. Wang, Z. Chen, T. J. Bao, M. Montes-de Oca, Y. Cheng, M. Lin, L. Torresani *et al.*, "Carsafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 2013, pp. 13–26.

[11] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin, "Sensing vehicle dynamics for determining driver phone use," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 2013, pp. 41–54.

[12] G. Castignani, R. Frank, and T. Engel, "Driver behavior profiling using smartphones," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013, pp. 552–557.

[13] H. Chu, V. Raman, J. Shen, A. Kansal, V. Bahl, and R. R. Choudhury, "I am a smartphone and i know my user is driving," in *Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference on*. IEEE, 2014, pp. 1–8.

[14] "Mobileye," https://www.mobileye.com/en-us/.

[15] "ionroad," https://ionroad.com/.

[16] D. M. Gavrila and S. Munder, "Multi-cue pedestrian detection and tracking from a moving vehicle," *International journal of computer vision*, vol. 73, no. 1, pp. 41–59, 2007.

[17] C. Guo, S. Mita, and D. McAllester, "Robust road detection and tracking in challenging scenarios based on markov random fields with unsupervised learning," *IEEE Transactions on intelligent transportation systems*, vol. 13, no. 3, pp. 1338–1354, 2012.

[18] J. Hasch, "Driving towards 2020: Automotive radar technology trends," in *Microwaves for Intelligent Mobility (ICMIM), 2015 IEEE MTT-S International Conference on*. IEEE, 2015, pp. 1–4.

[19] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*. IEEE, 2011, pp. 1609–1615.

[20] S. Lawoyin, X. Liu, D.-Y. Fei, and O. Bai, "Detection methods for a low-cost accelerometer-based approach for driver drowsiness detection," in *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1636–1641.

[21] C. Bo, X. Jian, X.-Y. Li, X. Mao, Y. Wang, and F. Li, "You're driving and texting: detecting drivers using personal smart phones by leveraging inertial sensors," in *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013, pp. 199–202.

[22] D. Chen, K.-T. Cho, S. Han, Z. Jin, and K. G. Shin, "Invisible sensing of vehicle steering with smartphones," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '15. New York, NY, USA: ACM, 2015, pp. 1–13. [Online]. Available: http://doi.acm.org/10.1145/2742647.2742659

[23] W. Van Winsum and H. Godthelp, "Speed choice and steering behavior in curve driving," *Human factors*, vol. 38, no. 3, pp. 434–441, 1996.

[24] K. Schmidt, M. Beggiato, K. H. Hoffmann, and J. F. Krems, "A mathematical model for predicting lane changes using the steering wheel angle," *Journal of safety research*, vol. 49, pp. 85–e1, 2014.

[25] J. Dai, J. Teng, X. Bai, Z. Shen, and D. Xuan, "Mobile phone based drunk driving detection," in *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2010 4th International Conference on-NO PERMISSIONS*. IEEE, 2010, pp. 1–8.

[26] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[27] "Rambo," https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/rambo.

[28] "Comma ai research," https://github.com/commaai/res.

[29] "Cg23," https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/cg23.

[30] "Safety report waymo," 2017.

[31] "Inside waymo's secret world for training self-driving cars," 2017.

[32] "2018 cloud computing survey," 2018.

[33] "Microsoft sales lifted by cloud computing," 2018.

[34] "Cloud-computing business lifts oracle's profit – update," 2017.

[35] "Ten years on: How netflix completed a historic cloud migration with aws," 2018.

[36] "Hadoop," https://hadoop.apache.org/.

[37] "Kafka," https://kafka.apache.org/.

[38] "Cassandra," http://cassandra.apache.org/.

[39] "Google dumps mapreduce in favor of new hyper-scale analytics system."