IMAGE: A MAP OF THE STARS OF THE ORION CONSTELLATION

Print ISSN: 2514-863X Online ISSN: 2514-8648

JournalPreview

London Journal of Research in Computer Science and Technology

Volume 22 | Issue 1 | Compilation 1.0

London Journals Press

WWW.JOURNALSPRESS.COM

Journal Content
In this Issue

London Journals Press

Neural Network Implementation for Lane Tracking in Self-Driving Cars

Shadi Alawneh & Ayomide Yusuf

Oakland University

ABSTRACT

The process of detecting and tracking lane lines is very crucial to the development of self-driving cars. For a self-driving car to successfully drive from one location to another, it must be able to detect and track lanes with minimal to no errors. Lane tracking is a computationally intensive task that needs an efficient implementation to meet the real-time requirements in self-driving cars. A self-driving car relies on the lane markings present on the road to drive safely from one point to another, so the visibility of the lane markings is important to avoid accidents. In situations where we have faded lane lines, obstructed lane, or no lane, it will be very difficult for a self-driving car to navigate safely. Few available algorithms have been able to address these issues efficiently. This research paper is aimed at addressing these problems by developing an algorithm using trained neural networks model to track lanes in most road conditions and implementing it on the NVIDIA Jetson TX2 to meet the real-time requirements of self-driving cars.

Keywords: lane detection and tracking, neural network, NVIDIA jetson TX2, self-driving cars.

Classification: DDC Code: 006.32, LCC Code: QA76.87

Language: English

London Journals Press

LJP Copyright ID: 975811
Print ISSN: 2514-863X
Online ISSN: 2514-8648

London Journal of Research in Computer Science and Technology

Volume 22 | Issue 1 | Compilation 1.0

Neural Network Implementation for Lane Tracking in Self-Driving Cars

Shadi Alawneh[α] & Ayomide Yusuf[σ]

ABSTRACT

The process of detecting and tracking lane lines is very crucial to the development of self-driving cars. For a self-driving car to successfully drive from one location to another, it must be able to detect and track lanes with minimal to no errors. Lane tracking is a computationally intensive task that needs an efficient implementation to meet the real-time requirements in self-driving cars. A self-driving car relies on the lane markings present on the road to drive safely from one point to another, so the visibility of the lane markings is important to avoid accidents. In situations where we have faded lane lines, obstructed lane, or no lane, it will be very difficult for a self-driving car to navigate safely. Few available algorithms have been able to address these issues efficiently. This research paper is aimed at addressing these problems by developing an algorithm using trained neural networks model to track lanes in most road conditions and implementing it on the NVIDIA Jetson TX2 to meet the real-time requirements of self-driving cars.

Keywords: lane detection and tracking, neural network, NVIDIA jetson TX2, self-driving cars.

Author α: Electrical and Computer Engineering Department, Oakland University, 115 Library Drive, 248370-2242, Michigan, USA.
σ: Electrical and Computer Engineering Department, Oakland University, 115 Library Drive, 248370-2242, Michigan, USA.

I. INTRODUCTION

Making driving safer and more comfortable has been the goal of engineers in the automotive industry over the years. Many advance features have been introduced and implemented in cars over the years to achieve this goal. One of these features is the Anti-Lock Braking System (ABS), a safety anti-skid system used to prevent wheels from locking up during breaking by maintaining tractive contact with the road surface. Electronic Stability Control (ESC) is another feature introduced by automotive engineers; ESC is a computerized technology that improves car stability by detecting and reducing loss of traction. Rear view cameras were introduced to prevent minor collisions between cars. Airbags were also used to protect occupants in collisions. Recently, some Advanced Driving Assistant Systems (ADAS) such as Lane Keeping System (LKS), Cross Traffic Alert (CTA), Cross Traffic Breaking System (CTB), Cruise Control (CC), Adaptive Cruise Control (ACC), Blind Spot Information System (BLIS), Automatic Emergency Breaking (AEB), Active High Beam Control (AHBC) and many others, have been successfully integrated into cars for safety and comfort.

In addition to these new features, millions of dollars have been spent on campaigns to inform the public of the dangers of drunk driving, failure to buckle up, and texting while driving [22]. Despite the new features added to cars for safety and the money spent on campaigns for safe driving, the number of people that die on roads and highways increases every year. Based on a report by the National Safety Council, a non-profit organization that works closely with federal auto-safety regulators, 42,200 people died in accidents involving motor vehicles in 2016 [19], a six percent increase from 2015.

Some of the key factors that often play a major role in car accidents are:

- *Distracted Driving:* Distracted driving has been identified as one of the major causes of accidents. Drivers are typically distracted by using their cell phones, talking to people in the

car, greeting friends in other cars, eating while driving, singing to songs and many other activities that take their attention from the road. According to research condu- cted by the National Safety Council, 25% of car accidents in America are related to the use of cell phones [11].

- *Fatigue of the Driver:* Fatigue of the driver is another major cause of accidents. A tired driver might have impaired judgment and/or reduced time to react on the road. In 2014, the NHTASA's FARS database shows 864 fatalities were caused by drowsy driving [8].
- *Drunk Driving:* Driving while under the influence of alcohol is the most highly reported cause of car accidents. National Highway Traffic Safety Administration (NHTSA) reported that alcohol is considered a cause for over four-hundredth of automobile accidents. 44,574 drivers in USA that were killed in accidents involved alcohol in 2013 [11].
- *Over-Speeding:* Centers for Disease Control and Prevention (CDC) noted in 2013 that over-speeding contributed to twenty ninth of all fatal crashes, claiming 9,613 lives. About 1 out of 3 car accident-related deaths involved over-speeding [11].

According to the statistics above, the major cause of accidents are human errors. The government, in its effort to make driving safe has put in place sanctions and measures, yet car accidents still claim more lives each year. Engineers and scien- tist are exploring the option of building intelligent cars, mostly called self-driving cars, to prevent human errors in accidents.

Intelligent driving has been the interest of both academic and commercial spheres over the past few years. It has been discovered that the development of efficient algorithms is the key factor in developing self-driving cars. The prospects of the application of computer vision and deep learning in self-driving applications has provided solutions to some key problems in self-driving applications. Despite the interesting results achieved, the problems of lane detection and tracking, vehicle control, mapping, scene perception, and localization in self-driving cars remain a challenge that has not been fully solved. Many of the available self driving cars are driven in a supervised environment, which is not a good representation of the real world. There is countless number of variables and conditions that must be captured in the development and testing of self driving cars before they become available to the public. Among these issues, lane tracking is one of the important features in self driving car development that requires additional attention. Many algorithms have been proposed and imple- mented for lane tracking [6]. Lane tracking requires significant computing power, making it difficult for some of the existing algorithms to meet the real-time requirements of tracking lanes on the road. In addition, some of the existing algorithms are not robust enough to handle different weather and road conditions, like faded lane markings, obstructions on lanes, or an absence of lane markings.

This research work is aimed at solving these problems. An efficient algorithm was developed on GPU using the NVIDIA Jetson TX2 to meet the real-time needs of lane tracking. The algorithm is implemented in a robust way to handle different road conditions. The algorithm is then optimized by developing a neural network model that can track lanes by learning from the output of the algorithm.

II. PAPER OUTLINE

The rest of the paper is organized as follows. Section 3 provides a general overview of the self driving car architecture. Section 4 describes the challenges in lane detection and tracking. Section 5 summarizes the novel contributions of this research work. Section 6 describes a literature review of self driving cars, explore available implementations and setbacks in these implemen tations, and suggest potential improvements.

Section 7 provides a brief overview of Graphical Processing Units and the NVIDIA Jetson TX2 used for the implementation process. The implementation of the proposed algorithm is described in Section 8. The evaluation of the implementation is evaluated in Section 9.

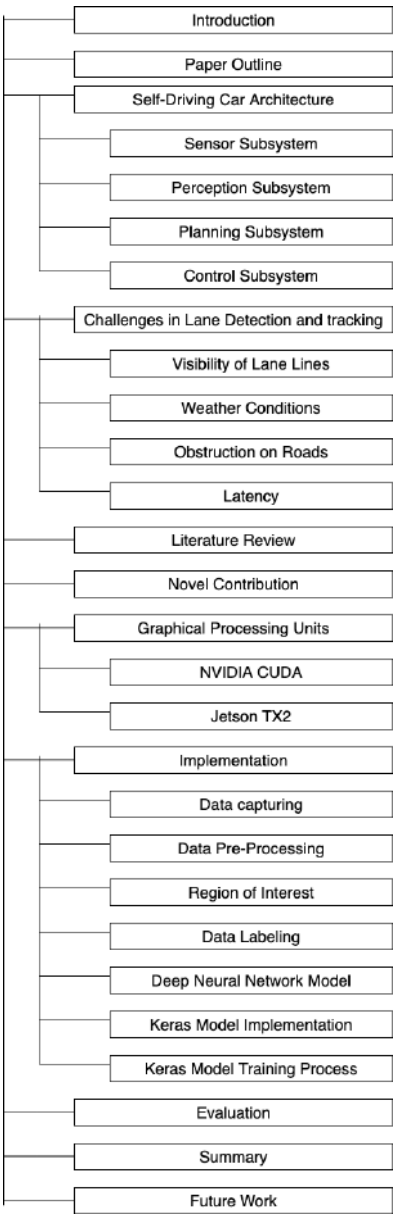

Fig. 1: The entire layout of paper

Summary and potential future works are presented in Sections 10 and 11. The entire layout of the paper is shown as a chart in Fig. 1 to help the readers understand the structure of the paper.

III. SELF DRIVING CAR ARCHITECTURE

Self driving cars mostly have various subsystems that are integrated to form a full functioning system. Some of the commonly used subsystems are: Sensor, Perception, Planning, and Control subsystems.

3.1 Sensor Subsystem

Sensors are hardware components used by self driving cars to gather data about their environment. Just like humans use their sense organs (e.g. ears, eyes) to perceive their environments, self-driving cars use different sensors to capture and perceive their environment. Some of the popular sensors used in self driving cars are Lidar, Radar, Cameras, and GPS sensors. Self driving engineers usually combine data from these sensors through a technique called sensor fusion, the fused data is then passed on to the perception sub-system where the self driving car extract useful information that helps it to navigate without accident.

3.2 Perception Subsystem

The perception subsystem processes the fused data from the sensor sub system into a well structured information that can later be used for path planning and control. This is the stage where a self driving car analyzes its environments. Perception subsystem can be further divided into two underlying subsystems; detection and localization.

1. The detection subsystem is responsible for understanding the surrounding environment. The detection subsystem includes software components like lane detection and tracking, traffic sign and traffic light classification, object detection and tracking, and free space detection.
2. The localization subsystem is responsible for using sensor and map data to estimate with minimal error the vehicles precise location.

3.3 Planning Subsystem

The data that has been processed by the perception subsystem is used by the planning subsystem for the self driving car to plan its path. Some of the components of the Planning subsystem are route planning, prediction, behavioral planning, and trajectory planning which are described below.

1. Route planning is where high level decision about the path a self-driving should take between two points on a map; for example, which roads, highways, or freeways to take.
2. Prediction component estimates what actions other cars might take in the future; for

example, if a self driving car wants to merge to highway, it needs to know if the approa- ching car in the lane it wants to merge in will either change to another, slow down for it to merge, or speed up.

3. Behavioral planning component determines what behavior the vehicle should exhibit at any point in time. It determines the action of a self driving car at each point, for example changing lanes, accelerating or decelerating, stop- ping at a traffic light or intersection, or making a left turn onto a new street
4. The trajectory planning plots the precise path the self driving car should follow.

3.4 Control Subsystem

The last subsystem we have is the control subsystem. This system ensures the self-driving car follows the path generated from the planning sub system. This subsystem usually includes components like PID controllers, model predictive controllers, or some other controllers. The control subsystem is responsible to send acceleration, braking, and steering commands to the self-driving car.

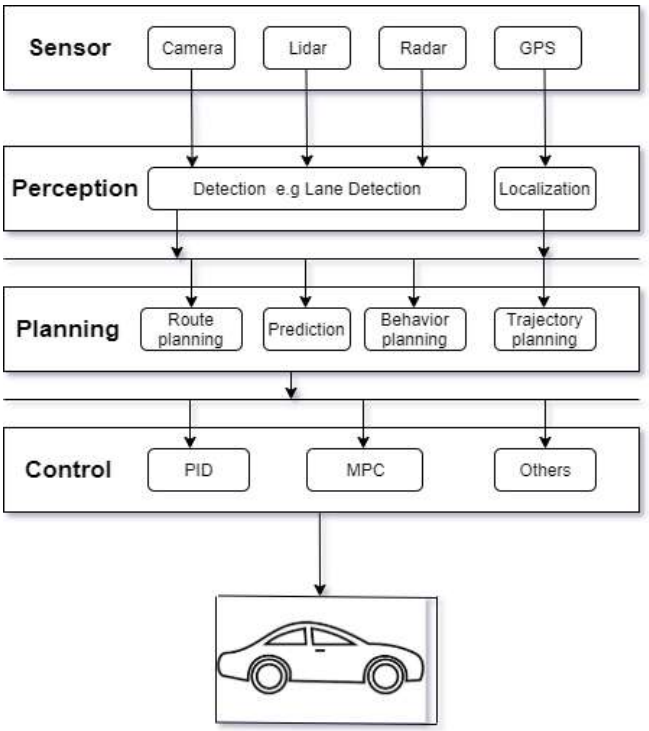

Fig. 2: An example of self-driving car architecture

We show how these systems interact with each other to form the architecture of a self-driving car in Fig. 2. This research paper is focused on the perception subsystem which is the most challenging in designing self-driving cars. There are many components in the perception subsystem; our interest is in the lane detection and tracking.

IV. CHALLENGES IN LANE DETECTION AND TRACKING

Lane detection and tracking is one of the components in the perception subsystem. It primarily uses frames from the video feed taken by the camera; hence its input is from the camera in the sensor subsystem. The lane lines on the road are used to guide cars to move from one point to another without going off track i.e. drive only on the drivable path. It is usually easy for human drivers to follow lane markings on the road or even use intuition if the lane markings are not visible. The task is now to make self-driving cars do the same job as human or even better. Self-driving cars must detect and track lane lines in all conditions with minimal or no errors to avoid driving off- course. Lane detection and detection might be easy for human drivers but difficult for self-driving, some of the challenges of lane detection and tracking are as follow.

4.1 Visibility of Lane Lines

Visibility of the lane lines is the major challenge in lane detection and tracking. Lane lines must be clearly visible with no obstruction for a self-driving car to detect and track it. Most roads have faded lane lines, worn-out lane lanes, obstructions, while some have no lane markings. If self-driving cars are built with assumption that lane lanes will be present and clearly visi- ble all times, they will crash on some roads. Most self-driving car manufactures are testing their cars in a controlled environ- ment on good roads with visible lane markings. For self-driving cars to be accessible for all on roads, this issue must be solved.

4.2 Weather Conditions

Lane detection and tracking heavily depends on the data captured by cameras in self-driving cars.

Cameras need clear sight to capture lane lines on the road. There are some weather conditions that makes visibility so hard that human drivers sometimes can't see through. Weather conditions like heavy rainfall, snowstorm, fog, and sometimes exhaust from other vehicles will make it almost impossible for a self-driving car to detect and track lane lines. The challenge is to make self-driving car handle these conditions and still drive safe.

4.3 Obstruction on Road

Unwanted materials or objects on the road can make it difficult for self-driving car to track lane lines successfully. Also, there might be leaves on the road covering lane lines during fall season. This condition is not common, but it needs to be accounted for because one occurrence of it might lead to accident that will take human life.

4.4 Latency

Latency is a major challenge in designing a self-driving. A millisecond delay in response time might cause a serious damage. A self-driving car must be able to capture, process, and make decisions based on every lane lines in a frame at a very fast rate. There are several ways to solve this problem; one way is to have used a very power core in the car for processing videos from the camera, it is also possible to use more than one core in the car, this is called multicore. Graphical processing unit is the mostly widely use device. It usually has thousands of cores that can be executed in the parallel which makes it a better choice for image processing in self-driving cars.

V. LITERATURE REVIEW/RELATED WORK

According to the report in [24, 20], it has been argued that humans are generally not good at driving. This is usually true when humans are driving under the influence of drugs, driving with partial attention on the road, or drunk driving [14]. Even if humans can't drive safe all the time, developing a self-driving car that drives better that an average human is hard to come by [4]. Self-driving cars cannot drive better than an average human until they are able to sense the environment on their own, perform computations, and make better decisions. One advantage a self-driving car has over humans is that it cannot be distracted. Adversely, it cannot also make decisions outside what it has been trained to handle. Therefore, the task is to develop self-driving cars that make the same or better decisions than humans can. A major achievement was accomplished in the 2007 DARPA Urban Challenge, when six autonomous cars successfully navigated an urban region to the finish line, out of eleven total cars. The first-place finisher could travel at an average speed of 15mph [16].

It has been estimated that by 2050, self-driving cars will be widely accepted and implemented compared to human-driven cars. This estimation is due to the increase in research and investments into self-driving cars. Fully autonomous cars are expected to be widely accepted in 2035 [15, 28].

The following are the features of a fully autonomous car as described by the Society of Automotive Engineers (SAE):

1. A fully autonomous car must be able to finish a journey from a location to another with no input from a human except for defining the destination at takeoff.
2. A fully autonomous car must be able to successfully drive on all road's conditions.
3. A fully autonomous car must always be able to adhere to all traffic rules.

The need for and expansion of self-driving cars has been controversial. Detractors contend that if we can arrive at a place where humans always obey all traffic rules, we should not invest valuable resources into the development of self-driving cars. However, supporters of autonomous cars believe that setting up a system where humans obey traffic rules always is nearly impossible. Instead, driving can become safer and more enjoyable if we develop self-driving cars mimic human driving while obeying all traffic rules and regulations. Besides saving lives and properties, the widespread adoption of self-driving cars is anticipated to have many positive benefits including: reducing traffic congestion by a factor of 10 [5], helping to cut down the number of cars on the road, increasing speed limits, increasing

leisure time and productivity during commutes [23], and removing restriction for vulnerable groups [25].

Some notable research work has been carried to develop effective lane tracking algorithms for self-driving cars. In this paper, we explore past work in this area, identify its limitations, and provide solutions for the gaps. Qui et al. [21] used Gabor filter, an edge extraction method to develop algorithm that detects and tracks lanes in real-time. In their study, the features in the lane markings were enhanced using the Gabor filter to make it more visible, and Hough Transform was used to detect the lane markings under the restriction of the polar angle. The researchers used the location of the previous frame in the video to determine the region of interest for the next frame. They tested their algorithm on multi-lane and multi-scene roads with fast moving cars and were able to track the lane markings. Despite their positive contributions, one major setback with this implementation is that it cannot handle roads in bad conditions. Their algorithm would fail on a road with no lane markings, obstructed lanes, or faded lanes.

In another study, Lee et al. [13] used simple filters as well as Kalman filters to detect and track lanes. The focus of their work was to develop an algorithm for the Lane Departure Warning System. They successfully designed a commercially ready lane departure warning system on an IMX6Q board. They implemented their algorithm in real-time and also proposed a lane detection and tracking solution. Their method yielded 97% lane detection accuracy at daytime and 95% accuracy at nighttime. They were also able to run the algorithm at 15 frames per second. Although they achieved good results, their algorithm would not work on a road that is not in good condition or is without visible lane markings.

Kun et al. [28] developed a spline-based system for multilane detection and tracking. They used the extended Kalman filter tracking with their new method 'Catmull-Rom spline' to detect and track lane lines. They used the Kalman filter for the robustness of their model. They didn't make the assumption about the parallelism and the shape of the lane markings, and as such, their algorithm was able to get detailed information from the road. The algorithm was also able to detect all lane markings visible to the camera. To evaluate the performance of their implementation, they drove a test vehicle on roads with different conditions, which included worn out lane markings, construction sites, narrow corners, and exits and entries of the highways. The results of their implementation were significant; however, their implementation was not robust enough to include roads with no lane markings at all.

Chanho and Ji-Hyun [12] designed lane detection and tracking system for real-time applications. They were able to provide solutions for major problems in lane detection and tracking such as, poor visibility in bad weather, lack of clarity on lane markings, light reflections, and shadows. Researchers were also able to use an efficient region of interest to reduce the high noise and the calculation time. They used a line clustering with scan-line tests, gradient cue, and color cue to verify the characteristics of the lane markings. The accumulated statistical data was used to separate false lane and real lane markings. They designed their system to be able to handle bad weather conditions to some degree; however, their system would still fail in the absence of lane markings.

Assiq et al. [1] designed a vision-based algorithm for lane detection in self-driving cars. The images of the road were first converted to grayscale, followed by the use of a noise reduction technique to remove unnecessary information from the image. Additionally, Canny edge detection was used to detect the edges of the lane markings, while Hough Transform was used to draw lines on the lane marking edges. The lane boundaries were first detected and subsequently, the information was used to predict lane markings position. Assiq et al. [1] were able to create an algorithm robust enough to handle various lighting conditions and run in real-time. Their algorithm reportedly performed well on roads with or without lane markings and curved and straight roads in different lighting conditions. The result of their implementation provided promising results;

nevertheless, it cannot meet the real-time requirements of self - driving cars.

In a work related to Assiq et al. [1], Bounini et al. [3] developed an algorithm that uses lane boundaries and painted lines to track lanes for intelligent and autonomous vehicles. They used techniques like Canny edge detection, Hough Transform, least-square method, and Kalman filter. They used Hough Transform to initialize the algorithm at each time needed; least-square method and Kalman filter to reduce the adaptive region of interest. The downside of their work is the fact that their algorithm runs only on the CPU which cannot meet the real-time needs of self-driving cars.

A real-time Lane Detection and tracking was developed by Hajjouji et al. [9] on FPGA. They were able to detect lane lines by using Sobel operator with an adaptive threshold and Hough Transform. They used Kalman filter for tracking the detected lane lines. In order to simplify the Coordinate Rotation Digital Computer (CORDIC) algorithm, they used the gradient directions of the edge state, which made their algorithm meet the real-time requirements of tracking lane lines. The result from evaluating their algorithm shows the implementation can effectively detect and track lanes in different lighting conditions. Their result is encouraging; however, the algorithm cannot detect and track lane lines effectively in different road conditions, including those with faded lane lines, obstructed lane lines, and other factors that affect the visibility of lane lines.

In our recently published paper, we presented a GPU implementation for Lane Tracking in Self-Driving Cars [26]. Our Implementation was based on techniques such as Image Capturing, Gray scale conversion, Edge detection, Noise Reduction, Line Detection, Region of Interest, and Lane Boundary Extraction. Our implementation was able to detect and track lane lines on roads with different conditions. We were also able to track the lane-lines in real-time by using NVIDIA GPU for our implementation. We measured the processing time for the CPU and GPU implementations and recorded speedup of 20X. The performance of the algorithm was tested with real road data (video) and the accuracy was up to 95%. The idea of this implementation is to extract the lane boundaries and use a mathematical model to map the lane boundaries to lane line locations. In our current research, we made this process more robust by developing a neural network model that is trained by the mathematical model. The model is then used instead of the mathematical model to make the implementation more robust.

VI. NOVEL CONTRIBUTIONS

There have been a lot of research works done in the literature on lane tracking, some of which address specific problems and provide solutions. Our contribution to this topic is to address the challenges explained in section 4 of this paper. We implemented an algorithm that can detect and track lane lines when lines are visible, and accurately predict where lane lines should be when they are not visible or obstructed. We were able to handle some weather conditions like when visibility is not too bad, but situations like snowstorm and minimal to zero visibility were not covered. We solved the latency by using the NVDIA Jetson TX with GPU for the implementation. Generally, we implemented a neural network model that can detect and track lane lines in most conditions, specifically, when lane lines are not present.

VII. GRAPHICAL PROCESSING UNITS

A graphical processing unit is a chip that is typically used for rendering images, by performing rapid mathematical calculations [27]. It accelerates the creation of images in a frame buffer by rapidly manipulating and altering memory using its specialized electronic circuit [10]. GPUs are designed for speeding-up applications that are based on image processing. Some of the differences between GPUs and CPUs are as follow:

1. CPUs are powerful for latency-oriented applications while GPUs are powerful for throughput-oriented applications.
2. The Arithmetic Logic Units of CPUs are designed to reduce operation latency. GPUs

Arithmetic Logic Units are designed to be energy efficient with long latency.
3. CPUs use large caches to convert long latency memory accesses to short latency cache. GPUs are designed with small caches to help boost memory throughput.

Computer Engineers often integrate a GPU with a CPU on the same board, either in the motherboard of a personal computer, or on a graphics card [18]. As the processing power of GPUs increase, it became a widely accepted choice for other computing-intensive tasks not related to graphic processing. GPUs were used in modelling and scientific calculations, providing computing power needed for machine learning and artificial intelligence software.

7.1 Nvidia Cuda

In 2007, NVIDIA developed and released Computer Unified Device Architecture (CUDA) [18]. CUDA is a comprehensive hardware and software architecture for Generaln Purpose computing on Graphics Processing Units (GPGPU). NVIDIA started exploring GPGPU and High Performance Computing (HPC) after developing CUDA. Their design involved combining large programmability, performance, and ease of use.

CUDA supports two different types of programming interfaces:

1. Device level APIs: This involves the use of GPGPU standard DirectX. It uses the high-level shader language (HLSL) to implement compute shaders.
2. OpenCL: OpenCL is a standard that was developed by Khronos group. OpenCL is used for writing OpenCL kernels.

Device level APIs and Open CL are usually implemented on GPUs from different vendors because they are independent of GPU hardware. Another device level approach which directly uses the driver is CUDA programming. CUDA programming gives the programmer enough control but is complex to use [26]. This approach gives the programmers the ability to make use of native support for high-level languages such as C, C++, FORTRAN, Java, and Python, reducing code complexity and development costs.

7.2 Jetson TX2

Applications such as lane detection and tracking in automotive require a lot of computing power. This constraint coupled with the real-time requirement makes lane detection and tracking a difficult task. Lane detection and tracking in real-time in self-driving cars necessitate the use of a super-computer to effectively deploy these processes. Embedding a super-computer in a self-driving car is not feasible in terms of cost. The introduction of GPUs provides an alternative to super-computers. GPUs are of most value for image processing implementations because the type of calculations they are designed for is like those calculations used in image processing. Most information that's utilized in image processing is delineated within the kind of matrices, and GPUs are well-known for high computational performance on matrix operations.

GPUs in comparison to CPUs are much more fitting to perform matrix operations and other types of advanced math-ematical transformations [17]. For this reason, lane tracking runs faster on a GPU as opposed to a CPU. The implementa- tion in this research work is therefore done on NVIDIA Jetson TX2. Jetson TX2 is an embedded system-on-module that has quad-core ARM Cortex-A57, 8GB 128 - bit LPDDR4, and an integrated 256-core Pascal GPU.

As reported by NVIDIA at the time of the development of Jetson TX2, it was considered the fastest and most power efficient embedded computing system [17]. It runs on 7.5-watts and is best known as a supercomputer. Jetson TX2 can be integrated into a wide range of products, with minimal effort due to its variety of standard interfaces. Jetson TX2 is most useful for developing computer vision and deep learning applications; it runs on Linux. We used Jetpack for our implementation. Jet- pack comprises of the essential libraries required to program a Jetson TX2 device.

VIII. IMPLEMENTATION

The programming language used for developing the algorithm in this research work is Python. Python is the most utilized programming language in artificial intelligence applications. It encompasses a great number of inbuilt libraries for artificial intelligence and machine learning. The algorithm was implemented on CPU with an Intel i5-6330 processors running at 2.5GHz and on a Jetson TX2 device with 256 CUDA cores. The algorithm was implemented on both platforms to measure the performance advantage of using GPUs for lane detection and tracking. The implementation was tested using a large data set of videos taken from cameras attached to the front end of a car. This dataset was provided by Berkeley Deep Drive [2]. The dataset contains videos of distinctive properties. The detailed description of the implementation is described in the following sections.

8.1 Data Capturing

The primary input to the system is a sequence of colorful images taken from a camera attached to a non-stationary car. These images are frames extracted from real-time video taken by the camera. The camera is typically placed at the front-view mirror within the car for a good angle of vision. The images are extracted from the video in real-time and saved for processing in the on-board computer memory. These images are then fetched frame by frame from the memory for processing. The process of saving, fetching, and processing video frames must be done in real-time to avoid unexpected behaviors in a self-driving car.

The data used for training the neural network model used in this research was taken with a camera attached to the front view of a moving car. The camera is attached in a way to capture the drive-able area on the road, the road boundaries, lane lines when visible, obstruction on lane lines, and other features that will be useful for training the model. The video data will later be fed into the neural model frame by frame for training process.

8.2 Data Pre-Processing

The input to the neural network model is a sequence of video frames and the expected result for each frame is the round boundaries. These road boundaries will then be used to estimate the location of the lane lines. We need to train the model with a labeled data, which means we need to extract the lane boundaries from our training data. There is extra information in the data that are not required which must be removed. The data might also have noise that reduces the accuracy of the implementation. Data pre- processing is the process of removing extra details and noise in video frames.

Noise exists in most everyday applications, including in computer vision systems. Gaussian Blur is a technique in image processing that is used to blur images, a process which we undertook to remove the noise in our implementation. It is widely used in graphics software to lower image noise and cutdown details. The result of this blurring on an image creates a smooth blur that is like viewing the image through a translucent screen. In computer vision, Gaussian smoothing is mostly implemented during the pre-processing stage to enhance image structures at different scales.

8.3 Region of Interest

The frames from the video captured by the onboard camera contain a great deal of data, but we only need information about the road boundaries. For this reason, a region of interest is used to narrow down the searching range for features on the road. ROI comprises of the major information needed to be implemented to lower computation cost of unwanted data. This process is mostly used to determine the region of interest in the video frame. This was accomplished by drawing a polygon along the area that is needed in the entire video frame. This is a critical task as it determines if the selected area will contain the necessary data to detect the lane boundaries.

8.4 Data Labeling

Data labeling is a machine learning technique used for tagging data with labels used for training

models. The process typically involves taking a set of unlabeled data and argument each piece of the data with meaningful tags. We have a set of unlabeled video frames from the camera attached to the car, this is the primary input to our implementation. In order to train our model with the video frames, we need to label them. For each video frame, we will label it with the locations of the boundary pixels. Doing this, we can teach our model how to locate boundary pixels in any video frame.

To estimate the road boundary data used for labeling the video frames, we used the edge detection technique. Edge detection is a common computer vision technique used for finding the boundaries of object within an image. The boundaries of a road are the sharp contrasts between the road surface and painted lines or other kinds of non-pavement surfaces. The edges in an image provide the sharp contrasts. It is therefore important to use edge detection to isolate the boundaries of the road. This process also lessens the learning data by simplifying the image considerably. There are many edge detection algorithms which have been implemented including Canny edge detection, which is considered one of the best edge detection algorithms. Our implementation is based on Canny edge detection. Canny edge detection is specifically used for detecting the edges in an image while disposing of other data. The maxima of the partial derivative of the image function M in the direction orthogonal to the edge direction, and smoothing the signal along the edge direction is derived by:

$$G\sigma = 1/(2\Pi\sigma 2)\exp[(x2 + y2)/(2\sigma 2)] \quad (1)$$

where,

$$n = (\Delta G * M)/(|\Delta G * M|) \quad (2)$$

Canny edge detection implements a multi-stage algorithm to detect wide ranges of edges in images. The Open CV implementation of Canny edge detection was used in this work. It requires two parameters in addition to the input image, a low and high threshold that determines if an edge should be included. The threshold at a point defines the change in intensity at that point. All the points above the high threshold will be included in the resulting image, while the points between the threshold values will only be included if the edges next to them are high threshold or included in the resulting image. Edges below the low threshold are terminated. The recommended low: high threshold ratios are 1:2 or 1:3.

8.5 Deep Neural Network Model

In our previous paper [26], the left and right road boundaries were used for calculating the location of the left and right lanes using a mathematical method we formulated. This mathematical calculation is applied to each pixel in the frame to estimate the location of an equivalent pixel that corresponds to a lane marking. This process is time consuming and might reduce the accuracy of the result. To solve this problem, a deep neural network was developed to replace the mathematical model. The network was trained using the data collected from our previous implementation.

The deep learning model used in our research is Keras [7, 7], a Tensorflow API. Keras is a high-level API that is mostly used to build and train deep learning models. Keras is popular for its use in fast prototyping, advanced research, and production. It has the following advantages:

1. Keras interface is consistent and simple. It is optimized for common use cases.
2. Models developed with Keras are modular and composable.
3. Keras models are easily extendable. It is easy to create new layers on existing models.

8.6 Keras Model Implementation

The Keras model applied in this research project is used to implement a regression. The model we developed has two densely connected hidden layers, with one output layer as shown in Fig. 3. The two connected hidden layers are used to feed in the left and right edges of the road, while the output layer gives the right location of the lane marking. The model was trained for 1000 epochs before we achieved a minimal error in the

predictions. Each input node is connected to each output node.

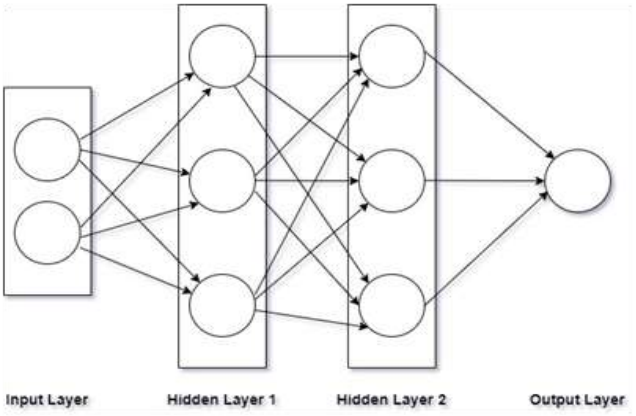

Fig. 3: Structure of the neural model implemented

8.7 Keras Model Training Process

The Keras model is used to implement a linear regression model that is trained to predict an output based on some sets of inputs. The inputs for this model are the left and right boundaries of the road and the output, the right and left lanes, are then estimated based on the assumed lane width.

The data used to train this model is extracted from the video. The true values of the road boundaries are extracted and the right and left lanes are recorded. The data is then saved in a file of ".data" extension. To avoid the model being trained with corrupted data points, the data is first cleaned. Pandas, an open source library for data manipulation and analysis is used for data extraction, analysis, and cleaning.

The data collected is then separated into two sets-a training and a testing set. The training dataset is used to train the model after it is created; this set comprises 80% of the total data. The testing dataset is used to evaluate the performance and accuracy of the model after training; 20% of the total dataset is used for testing the model. The joint distribution of some of the data is shown in Fig. 4.

The model is then created and built after data analysis and optimization. The Keras Tensorflow API is used to create a sequential mode with two densely connected, hidden layers, with one output

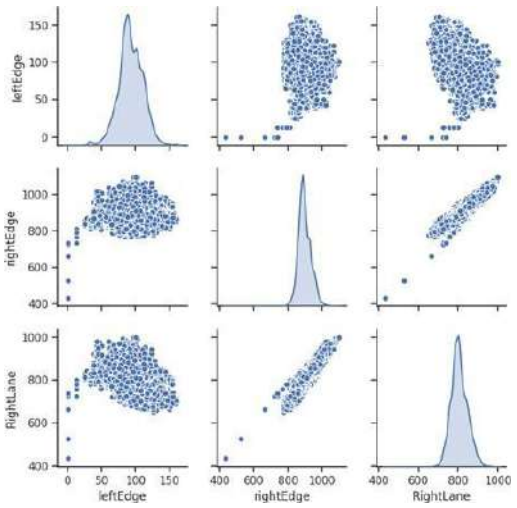

Fig. 4: Joint distribution of a few pairs of columns from the training

layer. The mean absolute error of the training training process was recorded at ± 2.0 pixel as shown in Fig. 5. This result was achieved by changing the parameters that was used to train the model.

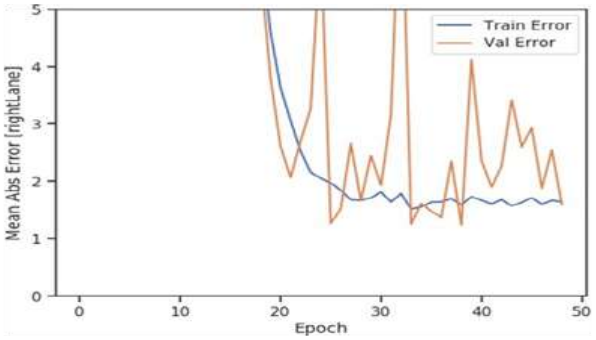

Fig. 5: Model mean absolute error

IX. EVALUATION

For performance evaluation, we implemented our algorithm on a GPU (Jetson TX2) and CPU (HP Intel i5-6330 CPU 2.5GHz). We used Python programming language for the implementation. In order to measure the accuracy of the algorithm, a large database of images and videos taken in different road conditions was used [28].

The videos were taken from various locations such as high-ways, country roads, and suburban areas. The quality of lane markings in the videos was not consistent: some included full markings, dashed markings, and no markings at all. The roads are not all straight; sharp curves were also included,

especially at the entries and exits of highways and country roads. Additionally, some videos have different weather and lighting conditions. Examples of lane detected in a video with our implementation is showed if Fig. 6 and Fig. 7.

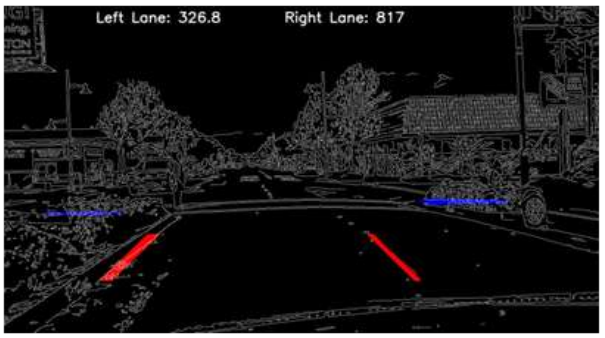

Fig. 6: Left and Right Lane Estimation with lane boundaries in Grayscale

Fig. 7: Left and Right Lane Estimation with lane boundaries in Original Image

The performance of the algorithm was evaluated by comparing the actual left and right lane locations to the predicted location for the left and right lanes. We recorded 92% accuracy for the left lane and 96% accuracy for the right lane.

X. SUMMARY

In our research, we developed a deep neural network model for detecting and tracking lane markings in real-time. The model was first trained by using the extracted road boundaries and the estimated lane markings. It was also trained for 50 epochs until a minimal error was recorded. The estimated lane markings used to train the model was obtained from the implementation in our previous research. In that study, we processed the videos in frames. We performed the following operations on each frame to predict the location of the lane markings.

1. Data Pre-processing: This process involves removing noise from the video frames.
2. Region of Interest: This is the step where additional information of the road not needed are discarded.
3. Data labeling: This is the stage where we identify the right and left boundaries of each video frame.
4. Training Model: This is the final process where we use the labeled data to train the neural network model.

The implementation was able to handle up to 30fps for a resolution of 1280x720 pixels. The performance of our algorithm was also evaluated by comparing the actual location of the lane markings to the predicted lane markings and we recorded 92% accuracy for the left lane and 96% accuracy for the right lane.

FUTURE WORK

Although the results obtained in this are encouraging, there is still a need to improve future implementations. The accuracy of the implementation can be improved significantly if sensors like RADAR and LIDAR are used in conjunction with the camera, a method known as sensor fusion. There are situations where the camera by itself will not be able to perform well, especially in extreme weather conditions such as fog. Using the information from RADAR/LIDAR alongside the data from the camera will make the implementation more robust.

Sensor fusion is an intelligent process that combines data from different sensors with the aim of improving an application or system performance. It enables the resulting information from different sources have less uncertainty that would be possible when sources are used individually.

REFERENCES

1. O. O. Khalifa Abdulhakam. AM. Assidiq, M. R. Islam, and S. Khan. Real time lane detection for autonomous vehi- cles. In Kuala Lumpur,

editor, International Conference on Computer and Communication Engineering, 2008.
2. Berkeley. Berkeley deep drive.
3. F. Bounini, D. Gingras, V. Lapointe, and H. Pollart. Autonomous vehicle and real time road lanes detection and tracking. In IEEE Vehicle Power and Propulsion Conference (VPPC), Montreal, 2015.
4. M. Buehler, K. Lagnemma, and S. Singh. The darpa urban challenge, cambirdge: Springer. 2009.
5. Houston Chronicle. Get ready for automated cars, 2012.
6. A. Davies. Oh look, more evidence humans shouldn't be driving, 2015.
7. Google. Tensor flow.
8. H. Guan, W. Xingang, and W. Wenqi. Real-time lane-vehicle detection and tracking system. In Chinese Control and Decision Conference (CCDC), Yinchuan, 2016.
9. I. E. hajjouji, A. E. mourabit, Z. Asrih, S. Mars, and B. Bernoussi. Fpga based real-time lane detection and tracking. In International Conference on Electrical and Information Technologies ICEIT. Morocco, 2016.
10. D. B. Kirk and W. Hwu. Programming Massively Parallel Processors. Elsevier, 2009.
11. Sbf Lawyers. The main causes of car accidents are the result of driver error, 2017.
12. C. Lee and J.-H. Moon. Robust lane detection and tracking for real-time applica- tions. IEEE Transactions on Intelligent Transportation Systems, PP(99):1–6, 2018.
13. D.-K. Lee, J.-S. Shin, J.-H. Jung, S.-J. Park, and S.-J. Oh. Real-time lane detection and tracking system using simple filter and kalman filter. In Robust Lane Detection and Tracking for Real-Time Applications, Milan, 2017. Ninth International Conference on Ubiquitous and Future Networks (ICUFN).
14. F. Lex, D. E. Brown, M. Glazer, W. Angell, S. Dodd, B. Jenik, J. Terwilliger, J. Kindelsberger, L. Ding, S. Sea- man, H. Abraham, A. Mehler, A. Sipperley, A. Pettinato, B. Seppelt, L. Angell, and B. Mehler and. and bryan, "large-scale deep learning based analysis of driver beha- vior and interaction with automation," 19 november 2017. [online]. Available: [Accessed, 19:2018, June.
15. Report linker. Autonomous vehicles: Advanced driver assistance systems and the evolution of self-driving functionality: Global market analysis and forecasts, 2015.
16. Jerome Lutin. Not if, but when: Autonomous driving and the future of transit. Journal of Public Transportation, 21(92-103):2375–0901, 2018.
17. NVIDIA. Jetson tx2.
18. NVIDIA. Nvidia cuda architecture.
19. U. S. Department of Transportation. National motor vehicle crash casual survey, 2018.
20. World Health Organization. Global status report on road safety, 2015.
21. Y. Feng Q. Gao and L. Wang. A real-time lane detection and tracking algorithm. In IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference, Chengdu, 2018.
22. New York Times. U.s. traffic deaths rise for a second straight year, 2017.
23. C. Urmson and W. Whittaker. Self-driving cars and the urban challenge. IEEE Intelligent Systems, 23(2):66–68, 2008.
24. T. Vanderbilt. Why do we drive the way we do (and what it says about us), 2008.
25. Watermark. Driverless cars: Exciting opportu- nity or futuristic dream?
26. A. Yusuf and S. Alawneh. Gpu implementation for automatic lane tracking in self-driving cars. In WCX SAE World Congress Experience, Detroit, USA, 2019.
27. A. Yusuf and S. Alawneh. A survey of gpu implementations for hyperspectral image classification in remote sensing. Canadian Journal of Remote Sensing, 44(5):532–550, 2019.
28. K. Zhao, M. Meuter, C. Nunn, D. Muller, S. Muller Schneiders, and J. Pauli. A novel multi-lane detection and tracking system. Alcada de Henares, 2012. in 2012 IEEE Intelligent Vehicles Symposium.

This page is intentionally left blank

Maintenance Assisted by Artificial Intelligence (MAAI)

Cécile Lidwine Inès Nlemba, Emmanuel Tonye & Alphonse Binele Abana

University of Yaounde I

ABSTRACT

This work presents the realization of a maintenance device assisted by artificial intelligence. This system makes it possible to improve the supervision of the network, the analysis of technical feedback, and the correlation of those feedbacks to reduce incident resolution times and better plan or initiate interventions. More precisely, our system is built up based on the ZABBIX package for monitoring, the Alexa package for the conversational agents as well as on some Artificial Intelligence bricks such as AVS, IoT, AWS... It is added to the list of possible solutions and acceleration levers for operators 'growth by reducing the time to detect faults in the network.

Keywords: artificial intelligence, maintenance task, supervision, incident.

Classification: DDC Code: 006.3, LCC Code: Q335

Language: English

London Journals Press

LJP Copyright ID: 975812
Print ISSN: 2514-863X
Online ISSN: 2514-8648

London Journal of Research in Computer Science and Technology

Volume 22 | Issue 1 | Compilation 1.0

Maintenance Assisted by Artificial Intelligence (MAAI)

Cécile Lidwine Inès Nlemba[α], Emmanuel Tonye[σ] & Alphonse Binele Abana[ρ]

ABSTRACT

This work presents the realization of a maintenance device assisted by artificial intelligence. This system makes it possible to improve the supervision of the network, the analysis of technical feedback, and the correlation of those feedbacks to reduce incident resolution times and better plan or initiate interventions. More precisely, our system is built up based on the ZABBIX package for monitoring, the Alexa package for the conversational agents as well as on some Artificial Intelligence bricks such as AVS, IoT, AWS... It is added to the list of possible solutions and acceleration levers for operators 'growth by reducing the time to detect faults in the network.

Keywords: artificial intelligence, maintenance task, supervision, incident.

Author α : Telecom Paris Tech, France.
σ ρ: University of Yaounde I, Cameroon.

I. INTRODUCTION

The infrastructures of telecommunications operators include an increasing number of equipment of the most varied in terms of voice, data, video services, etc. Declined by business line, these are telematics systems that rely on IT resources, the scope of which is constantly expanding, with the interconnection of intranets, the internet, distributed and heterogeneous systems (telephony, Wifi/3G/4G network, base stations, switches, etc.) and ever newer technologies. The proper functioning of such systems requires many human resources to ensure their maintenance, which is generally computer-assisted. Artificial intelligence devices such as conversational agents, also called chatbots, aim to improve the computer-assisted maintenance system, less wasted time, autonomy, continuity of service, and better efficiency. Monitoring is essential for the proper conduct of maintenance activities. It is the process of ensuring the availability and performance of the equipment of a telematics system at all times. Nowadays, there are several monitoring solutions, like ZABBIX, the availability and performance of the equipment of a telematics system. There are also several monitoring solutions, like ZABBIX [1], CENTREON, NAGIOS, PRTG (Paessler Router Traffic Grapher), and others, making it possible to optimize the functioning of infrastructures at all levels (integrity, performance, availability, etc.). Our work contributes to the maintenance assisted by artificial intelligence, which combines the use of conversational agents and the judicious choice of monitoring tools to equip itself with a real maintenance assistant driven by artificial intelligence and allowing to increase productivity, improve the employee experience and help engineers and technicians in the field.

II. GENERAL INFORMATION ON ARTIFICIAL INTELLIGENCE

Artificial Intelligence (AI) is defined as a set of theories and techniques implemented to produce machines capable of simulating human intelligence, therefore, it corresponds to a group of concepts and technologies more than to a constituted autonomous discipline.

A chatbot can work well without artificial intelligence and natural language recognition. It is the case of all the bots that offer conversations that only include closed or multiple-choice questions. However, to simulate a human conversation and amplify the performance of chatbots, it is necessary to equip them with artificial intelligence. AI is therefore an additional

London Journal of Research in Computer Science and Technology

component of the chatbot. By integrating AI into our chatbot, we will be able to manage things like:

- The context
- Mimicry. The data and the exchanges with humans will allow the chatbot to acquire automatisms and train itself to have as natural a conversation as possible about maintenance tasks.
- Automatic speech recognition. Automatic speech recognition (often referred to as voice recognition) is the computer technology that will allow the assistant to analyze the human voice picked up by using a microphone and transcribe it into text before proceeding.
- Speech synthesis. It is the computer technology that will allow our system to create artificial speech from a typed text.

III. ARTIFICIAL INTELLIGENCE MODEL

Artificial intelligence simulates the human intelligence, which is based on the brain. Companies like Microsoft, Apple, Google, AMAZON, and others have made artificial intelligence solutions based on artificial neural networks. These are integrated into chatbots known on the market as Cortana, Siri, Google Now, and Alexa.

These AI solutions are Cognitive Multi-Agent Systems in which one agent translates the voice request into text, and another understands and interprets the message resulting from this translation. An agent uses the keywords recognized in the interlocutor's intentions to search for information in Wikipedia according to the business logic, which can be weather, location, and in our case maintenance tasks. Once the information is found, the reverse operation is performed until the response is translated from text format to voice format.

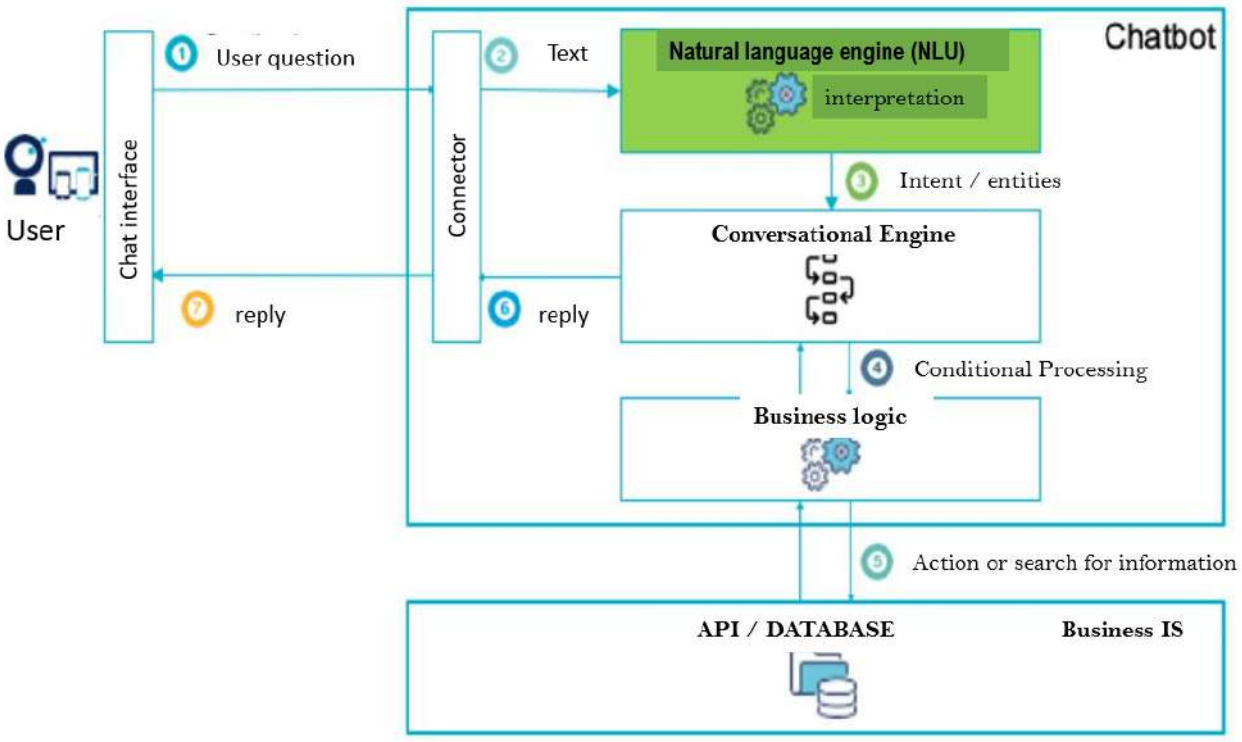

Figure 1: The operating principle of a multi-cognitive agent system [2]

Cortana, Siri, Google Now, and Alexa are all Level 3 chatbots because they use intent detection as a natural language understanding technique. They understand simple sentences with error tolerance, and their voice responses are contextual. However, our interest was focused on Alexa, because we can embed it in a connected speaker like the Raspberry Pi [3]. It is a generalist multi-agent cognitive system that we can specialize in a specific domain by adding skills.

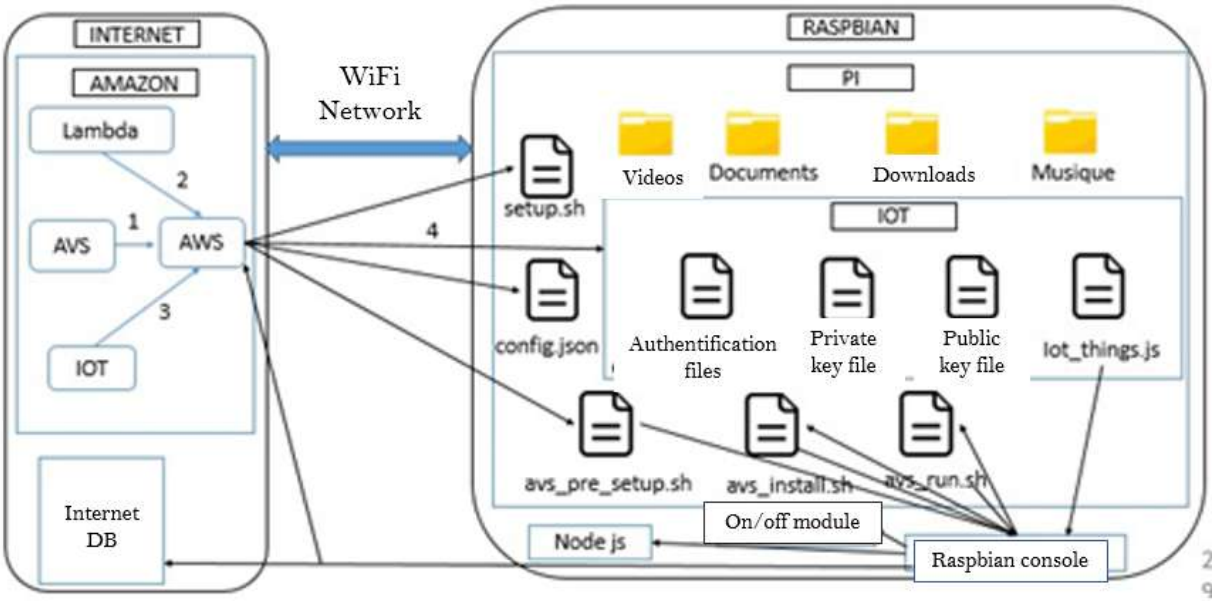

Figure 2: Integration of Alexa in the Raspberry Pi

IV. MONITORING TOOLS

There are many monitoring software on the market. Still, for the realization of our maintenance task tool, we have chosen ZABBIX as the solution to embed in the Raspberry Pi because it is compatible with many platforms, it has a large community, it is open-source, it has a lot of features (generate network maps, make graphics, display problems on the network and notify engineers in case of failure, etc.) and it uses data encryption.

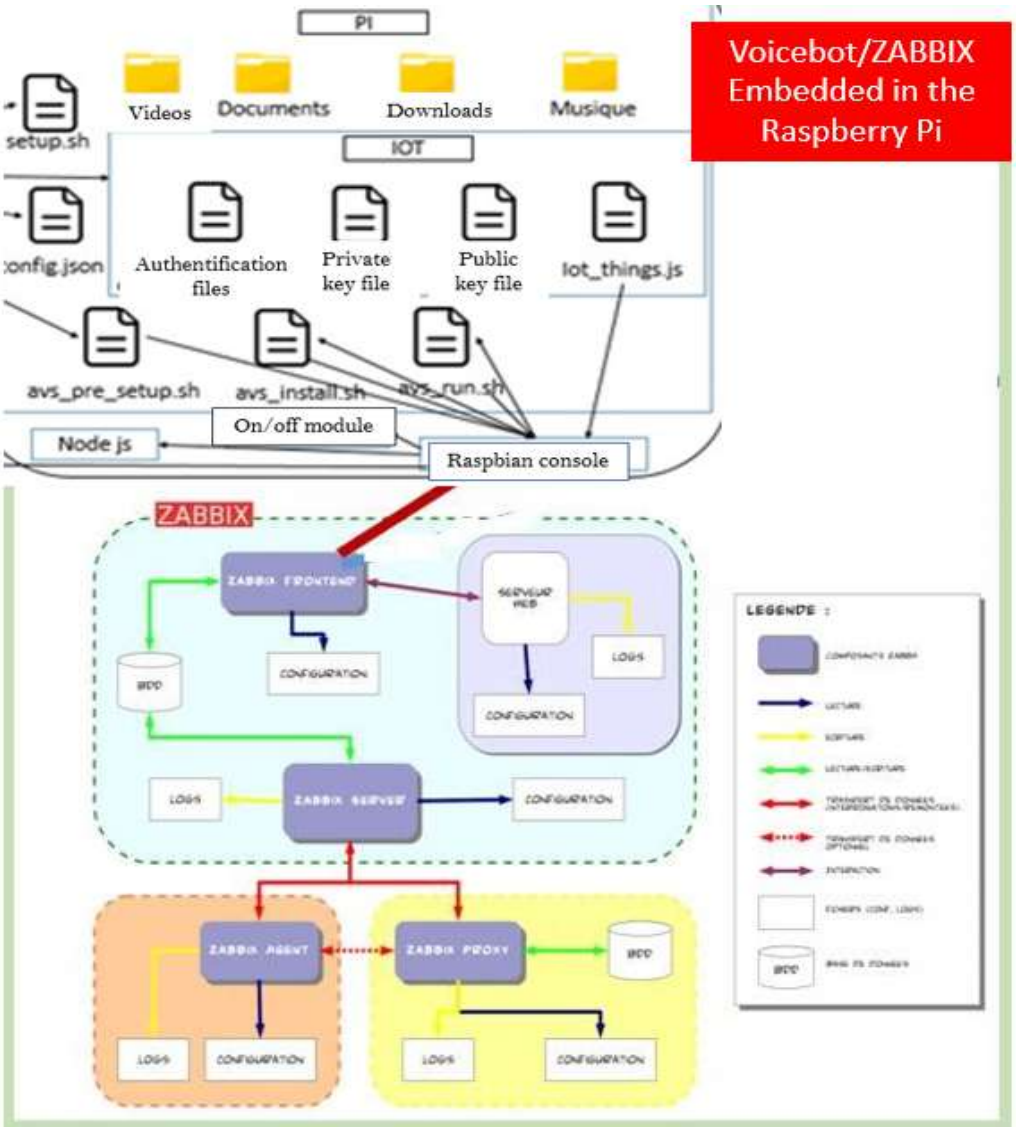

Figure 3: Integration of Zabbix in the Raspberry Pi

V. MODELING OF THE ASSISTANT

The device we call MAAI (Maintenance Assisted by Artificial Intelligence) allows us to obtain real-time information on the state of network equipment, automatically alert engineers and technicians in the event of a problem and even prevent possible network failures, and remedy them. These functionalities are performed thanks to the combination of ZABBIX' relevance, and the innovative nature of voice chatbots.

5.1 MAAI Artificial Intelligence Bricks

The three components of Alexa's artificial brain that we are going to embed in the Raspberry Pi to achieve our AI model are:

- *AVS (Alexa Voice Service):* it brings the intelligence of speech recognition, and synthesis.
- *IoT (Internet of Things):* it brings the intelligence to transform the Raspberry Pi into a connected object that can detect voice interactions, and bring out the responses audibly. It is also this module that ensures the security of the transmission of information.
- AWS (Amazon Web Services) allows us, thanks to its Lambda function module, to add skills to this artificial brain so that it can obtain information on the operating state of the operator's network equipment.

5.2 Wizard sequence diagram

Figure 4 describes the interactions between the different elements of our system

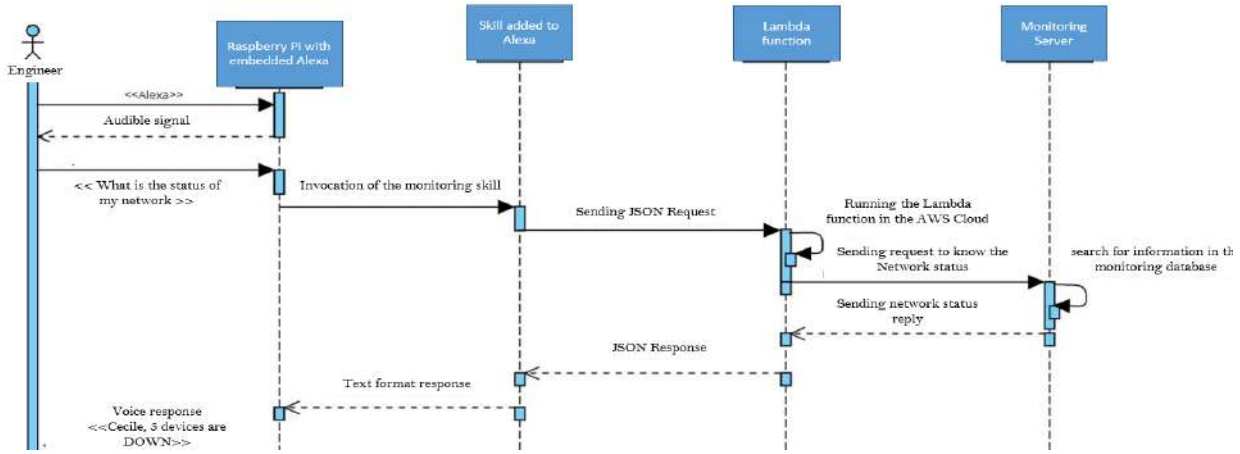

Figure 4: Wizard Sequence Diagram

5.3 MAAI architecture

The environment used for the realization of the MAAI prototype is composed of:

- A PC that allowed us to develop our monitoring skills thanks to the AWS Lambda tools (to write and host the function in Node.JS 12).
- 7-inch LCD touch screen connected to the Raspberry Pi 4B and in which we have embedded the Debian operating system, the Zabbix 5.0 monitoring solution, the components of the brain of Alexa, in particular, AVS and Telegram on the command line as well as Python 3 for the automation of alert feedback.
- Our two devices communicate by Internet and monitor the operator's network via the latter's intranet.

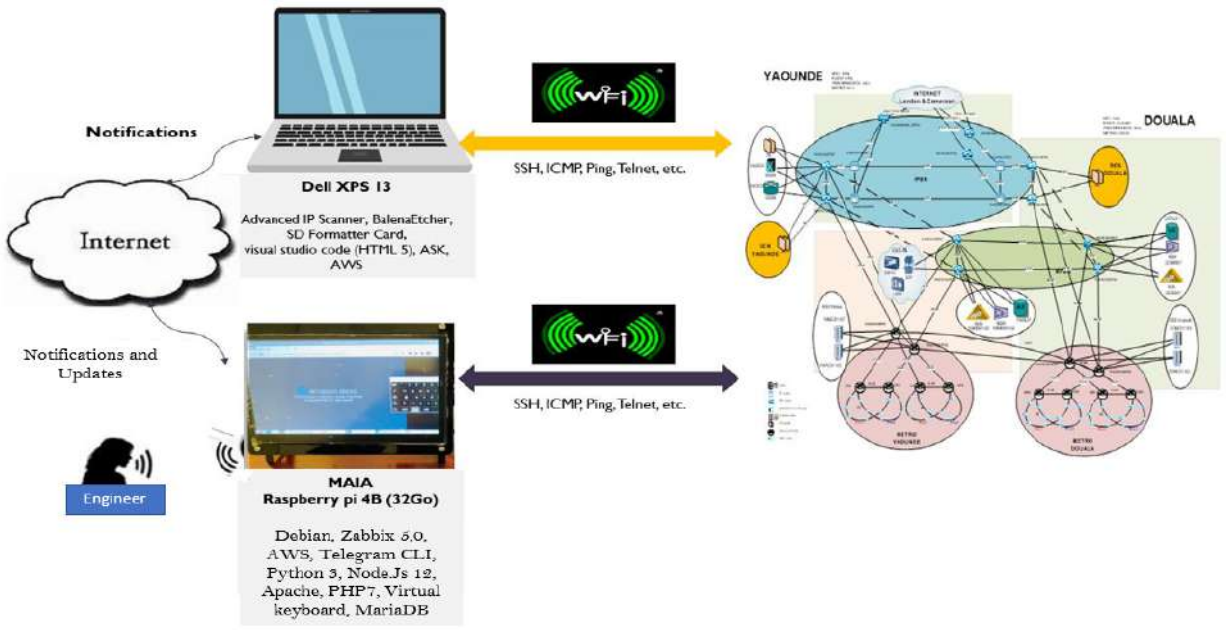

Figure 5: MAAI physical architecture: Case of the NeXttel operator [4]

5.4 Wizard implementation flowchart

The approach adopted to realize our solution is as follows in figure 6

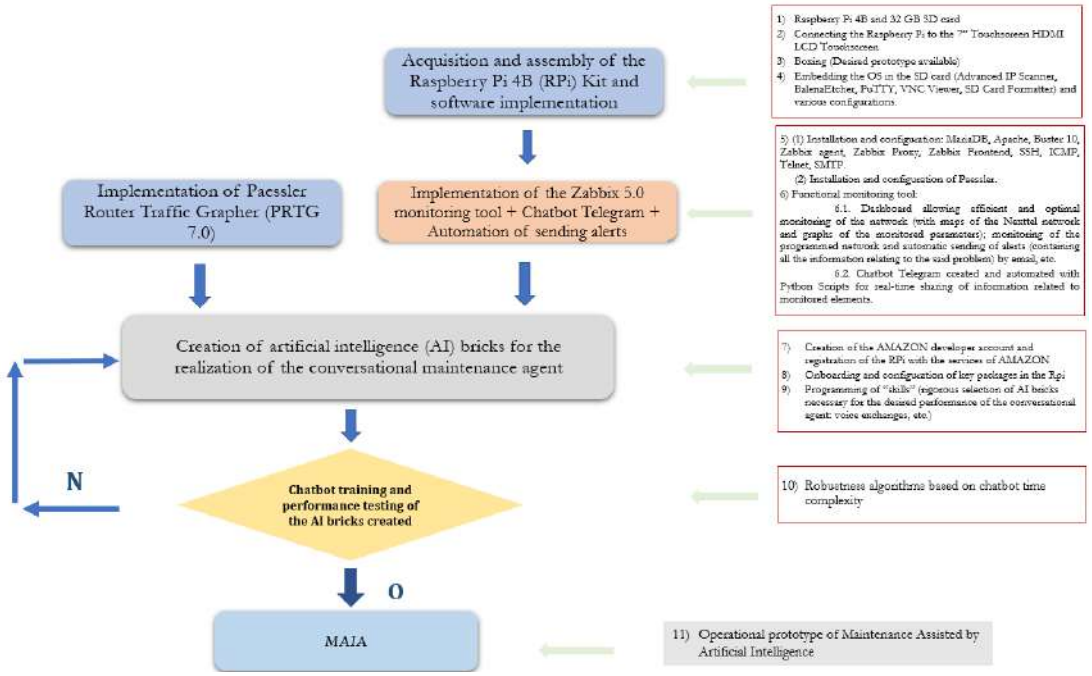

Figure 6: Realization steps

VI. PROTOTYPE

The operational prototype of Maintenance tasks Assisted by Artificial Intelligence (MAAI) has three main functions:

1. Vocal monitoring assistant with whom we can exchange information in real-time on the state of the network, and automatically alerts us by email and via a Telegram message in the event of a problem with a piece of equipment.
2. Tablet because we have embedded a virtual keyboard
3. Nano computer has all the capabilities and functions of a personal computer.

Figure 4: MAAI prototype

VII. CONCLUSION

Our MAAI solution, which uses advances in speech recognition and synthesis, will increase productivity by allowing engineers to save the time spent typing certain commands manually to obtain network status and by rendering information accessible everywhere.

REFERENCES

1. Cécile Lidwine I. NLEMBA. Design and production of a chatbot for artificial intelligence-assisted maintenance of the nexttel network. Master's memory, Ecole Nationale Supérieure Polytechnique de Yaoundé, 2021.

2. Olivier EZRATTY. *The uses of artificial intelligence*. ISSN 2680 - 0527, 2019.
3. John Paul Mueller, Mining Amazon Web Services: building applications with the Amazon API, Sybex, 2004.
4. Emmanuel TONYE, Alphonse BINELE. *Signaling in mobile telecommunications networks: Architecture, protocols and services*. AMAZON, 2018.
5. Nathan Liefting, Brian van Baekel, Zabbix 5 IT Infrastructure Monitoring Cookbook: Explore the new features of Zabbix 5 for designing, building, and maintaining your Zabbix setup [1ed.], Packt Publishing, 2021.
6. Amit Konar, Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain [1ed.], CRC Press, 2000.
7. Francesca Rossi, Peter van Beek and Toby Walsh (Eds.), Handbook of Constraint Programming [1st ed], Elsevier, 2006

This page is intentionally left blank

Pressure Dependence of the Critical Temperature of Hg-Based and Bi-Based Superconductors

Thaipanya Chanpoom

ABSTRACT

This research, we propose to study the pressure dependence of the critical temperature by using one-band model include the vHs density of states. We also set up the theoretical model to investigate the analytical formula of depend on pressure of cuprates superconductors. We find that the critical temperature increases as increasing pressure. We have compared the results with the experimental data of Hg-based and Bi-based superconductors that can fit well on Hg-based and Bi-based superconductors.

Keywords: critical temperature, pressure dependence, vHs density of states, cuprates superconductors.

Classification: DDC Code: 150.195, LCC Code: BF408

Language: English

London Journals Press

LJP Copyright ID: 975813
Print ISSN: 2514-863X
Online ISSN: 2514-8648

London Journal of Research in Computer Science and Technology

Volume 22 | Issue 1 | Compilation 1.0

Pressure Dependence of the Critical Temperature of Hg-Based and Bi-Based Superconductors

Thaipanya Chanpoom

ABSTRACT

This research, we propose to study the pressure dependence of the critical temperature by using one-band model include the vHs density of states. We also set up the theoretical model to investigate the analytical formula of T_c depend on pressure of cuprates superconductors. We find that the critical temperature increases as increasing pressure. We have compared the results with the experimental data of Hg-based and Bi-based superconductors that can fit well on Hg-based and Bi-based superconductors.

Keywords: critical temperature, pressure dependence, vHs density of states, cuprates superconductors.

Author: Program of physics and General Science Faculty of Science and Technology Nakhon ratchasima Rajabhat University, Nakhonratchasima 30000, Thailand.

I. INTRODUCTION

Since the discovery of high temperature superconductors, the study on pressure dependence of the critical temperature is useful to find the new superconductors with higher T_c. Both of the low and high temperature superconductors, the ambient pressure is the effect to the critical temperature. For the low temperature superconductor Al, Gonzalez- Pedreros and Baquero reported that the critical temperature decreased as increasing pressure [1] while sulfur hydride superconductor, the critical temperature reach to $200K$ under high pressure $56GPa$ [2]. The pressure is one of parameter that strongly affect to the critical temperature of cuprates superconductors [3], it is a key can understand the basic mechanism of high temperature superconductivity. It was found that the critical temperature increased when applied the external pressure to cuprates super- conductors. The materials do not super- conductor can be induced to superconducting state by ambient pressure. The external pressure can be induced the charge carrier from the charge-reservoir block to the CuO_2 layer, it made T_c depended on pressure and number of holes in cuprates. The change of T_c by applied pressure is attributed to the electron density of sates at the Fermi level. The critical temperature of $HgBa_2Ca_2Cu_3O_{8+\delta}$ increased to 164K under high pressure conditions [4]. For the bismuth compound superconductors, T_c increased as increased pressure with the pressure coefficient rate of $3K/GPa$, T_c reaches a maximum value and then decreases with increasing pressure above $1.2GPa$ [5]. The pressure dependence of T_c for $Tl-2122$ and $Tl-2223$, T_c increased at a moderate of $1.8K/GPa$ and $2.4K/GPa$ respectively while $Tl-2021$ shown that a negative value of the pressure coefficient [6]. The mercury based high temperature superconductor $Hg-1223$ was the three-layer compound, the superconductivity occurred below $133.5K$ [7] and T_c increased as increasing pressure with the pressure coefficient $dT_c/dP \approx 1.8K/GPa$ [8]. The T_c increased as pressure indicated that superconductivity at higher temperature could be stabilized under high-pressure conditions. The critical temperature T_c of $Hg-1223$ was shown to $150K$ at $11GPa$ [9], $153K$ at $15GPa$ [10] and $153K$ at $23.5GPa$ [11]. The highest T_c of $Hg-1223$ can be achieved to $164K$ at $45GPa$ [4]. The

London Journal of Research in Computer Science and Technology

critical temperature of Bi-based and Tl-based superconductors increased as increased pressure with the positive pressure coefficient [12]. Tissen et al. observed that T_c increased from $6K$ at ambient pressure to about $8.9K$ at $20GPa$ for $2H-NbS_2$ superconductor [13]. The pressure dependence of T_c can be derived from weak-coupling BCS model [14] or in strong-coupling Eliashberg model [15].

The pressure effects on T_c can consider with various parameters such as the phonon frequency and the density of states. The both phonon and non-phonon interactions were employed for investigated the pressure dependence of T_c [16], it was found that increasing electron-phonon coupling constant and non-electron-phonon coupling constant will increase the T_c of MgB_2 superconductor [17]. Chanpoom used the one-band model to explained T_c depended on pressure of Tl-based and Bi-based superconductors, the results can be fitted well on the experimental data of $Bi-2212, Bi,-2223, Tl-1223$ and $Tl-2223$ [12].

The van Hove singularity (vHs) density of states is a logarithmic density of states of electron in a periodic potential in two dimensions system [18]. In high temperature superconductors have evidence of suddle points which yield a logarithmic divergence density of states. In cuprates superconductors, the Fermi energy lies close to singularity [19]. The cuprates superconductors are the quasi-two-dimensional superconductor that existence of the vHs density of states in the band structure [20]. The vHs density of states is important issue and more suitable for use to study mechanism of cuprates superconductors. Houssa et al. studied the influence of vHs density of states on the thermal conductivity of YBCO superconductors, It is shown that the data can be reproduced by considering the effect of position of vHs density of states on the thermal conductivity peak observed below T_c [21]. Bok and Bouvier suggested that the existence of vHs density of states play an important role on many properties in superconducting and normal state of high temperature superconductors [22]. The strength of vHs density of states play strong influence on T_c and relate to the highest of T_c [23]. The observation of extended vHs density of states in $YBa_2Cu_4O_8$, it was show that the singularity related to CuO_2 planes and located about $19meV$ below the Fermi energy at Y point in the Brillouin zone [24]. In H_3S superconductor, the vHs density of states show higher T_c than H_2S which vHs density of states is absent [25]. The vHs density of states related to the highest values of T_c and could reduce the isotope effect coefficient [26] while in $YBCO$ superconductors it can be increased the isotope coefficient as increasing T_c [27].

This research, we use one-band model to study the pressure dependence of the critical temperature. The mediate of Cooper pair is phonon. We also set up the theoretical model to investigated the pressure dependence of the critical temperature. The analytical formula of T_c was derived from considering the effect of vHs density of states. We have compared the results with the experimental data of Hg-based and Bi-based superconductors.

II. MODEL AND CALCULATION

This work investigate the analytical formula of T_c depend on pressure. We use the one-band model include the vHs density of states to derive T_c's equation. The procedure start from the single-particle excitation energy $E_{k'}$ that was in the form $E_{k'}=\sqrt{\varepsilon_{k'}^2+\Delta_{k'}^2}$ where $\varepsilon_{k'}$ is the band dispersion measure

from the Fermi level and $\Delta_{k'}$ is the superconducting gap. The pairing of Cooper pair is phonon mediate to superconductivity. The energy gap with vHs density of states at finite temperature is given by

$$\Delta_k(T) = -\frac{1}{2}\sum_{k,k'}^{V_{k,k'}\left(\frac{\Delta_{k'}(T)}{E_{k'}}\right)} \ln\left|\frac{E_F}{\varepsilon}\right| \tanh\left(\frac{E_{k'}(T)}{2T}\right) \tag{1}$$

here $V_{k,k'}$ is the effective pairing interaction potential that keeping to be constant and define $h=1, k_B=1$. The gap equation taken into account the vHs density of states near the Fermi energy could be obtained as

$$\frac{1}{\lambda} = \int_0^{\omega_D} \ln\left|\frac{E_F}{\varepsilon}\right| \frac{\tanh\left(\frac{\sqrt{\varepsilon_{k'}^2+\Delta_{k'}^2(T)}}{2T}\right)}{\sqrt{\varepsilon_{k'}^2+\Delta_{k'}^2}} d\varepsilon \tag{2}$$

here λ is the coupling constant, E_F , ω_D are Fermi energy and Debye cutoff energy respectively. At critical temperature, we set superconducting gap to zero, $\Delta_{k'}(T_c)=0$, Eq.(2) become

$$\frac{1}{\lambda} = \int_0^{\omega_D} \ln\left|\frac{E_F}{\varepsilon}\right| \frac{\tanh\left(\frac{\varepsilon}{2T_c}\right)}{\varepsilon} d\varepsilon \tag{3}$$

The variation of the critical temperature on pressure can be found by differentiation of Eq.(3) with respect to pressure as

$$\frac{d(1/\lambda)}{dP} = \frac{d\left[\int_0^{\omega_D} \ln\left|\frac{E_F}{\varepsilon}\right| \frac{\tanh\left(\frac{\varepsilon}{2T_c}\right)}{\varepsilon} d\varepsilon\right]}{dP}$$

$$= \tanh\left(\frac{\omega_D}{2T_c}\right)\ln\left(\frac{E_F}{\omega_D}\right)\frac{d\ln\omega_D}{dP} + \int_0^{\omega_D} \frac{\tanh\left(\frac{\varepsilon}{2T_c}\right)}{\varepsilon} d\varepsilon \frac{\partial \ln E_F}{\partial P}$$

$$-\frac{1}{2T_c}\int_0^{\omega_D} (\ln E_F)\sec h^2\left(\frac{\varepsilon}{2T_c}\right) d\varepsilon \frac{\partial \ln T_c}{\partial P} + \frac{1}{2T_c}\int_0^{\omega_D} (\ln \varepsilon)\sec h^2\left(\frac{\varepsilon}{2T_c}\right) d\varepsilon \frac{\partial \ln T_c}{\partial P} \tag{4}$$

The $\int_0^{\omega_D} \frac{\tanh(\varepsilon / 2T_c)}{\varepsilon} d\varepsilon$ term of Eq.(4) yields $\ln(1.14\omega_D / T_c)$. On limit $\omega_D >> T_c$, the approximate value of $\tanh(\omega_D / 2T_c) \approx 1$, Eq.(4) write as

$$-\frac{1}{\lambda}\frac{d\ln\lambda}{dP} = \ln\left(\frac{E_F}{\omega_D}\right)\frac{d\ln\omega_D}{dP} + \ln\left(\frac{1.14\omega_D}{T_c}\right)\frac{\partial \ln E_F}{\partial P}$$

$$-\frac{1}{2T_c}\int_0^{\omega_D}(\ln E_F)\sec h^2\left(\frac{\varepsilon}{2T_c}\right)d\varepsilon\frac{\partial \ln T_c}{\partial P} + \frac{1}{2T_c}\int_0^{\omega_D}(\ln \varepsilon)\sec h^2\left(\frac{\varepsilon}{2T_c}\right)d\varepsilon\frac{\partial \ln T_c}{\partial P} \quad (5)$$

The third term on the right hand of Eq.(5), we calculate on limit $\omega_D >> T_c$, the approximate value is

$$-\frac{1}{2T_c}\int_0^{\omega_D}(\ln E_F)\sec h^2\left(\frac{\varepsilon}{2T_c}\right)d\varepsilon \simeq -\ln E_F$$

The fourth term on the right hand of Eq.(5), we consider on limit $\omega_D >> T_c$ and use $\int_0^{\omega_D} \frac{\tanh(\varepsilon / 2T_c)}{\varepsilon} d\varepsilon \simeq \ln(1.14\omega_D / T_c)$, the approximate value of this term is as follows

$$\frac{1}{2T_c}\int_0^{\omega_D}(\ln \varepsilon)\sec h^2\left(\frac{\varepsilon}{2T_c}\right)d\varepsilon \approx \ln\omega_D - \ln\left(\frac{1.14\omega_D}{T_c}\right)$$

then Eq.(5) can be written in the following form

$$-\frac{1}{\lambda}\frac{d\ln\lambda}{dP} = \ln\left(\frac{E_F}{\omega_D}\right)\frac{d\ln\omega_D}{dP} + \ln\left(\frac{1.14\omega_D}{T_c}\right)\frac{\partial \ln E_F}{\partial P}$$

$$-\ln E_F\frac{\partial \ln T_c}{\partial P} + \left[2T_c\ln\omega_D - \ln\left(\frac{1.14\omega_D}{T_c}\right)\right]\frac{\partial \ln T_c}{\partial P} \quad (6)$$

This work, we consider E_F / T_c and ω_D / T_c are constant then the variation of the critical temperature on pressure of Eq.(6) yields

$$\frac{d\ln T_c}{dP} = \frac{1}{\ln\left(\frac{1.14E_F}{T_c}\right)}\left\{\ln\left(\frac{E_F}{\omega_D}\right)\frac{d\ln\omega_D}{dP} + \ln\left(\frac{1.14\omega_D}{T_c}\right)\frac{\partial \ln E_F}{\partial P} + \frac{1}{\lambda}\frac{d\ln\lambda}{dP}\right\} \quad (7)$$

We define $\phi = \frac{\partial \ln\lambda}{\partial \ln V}$ and $\phi_F = \frac{d\ln E_F}{d\ln V}$. From the mode Gruneisen parameter $r_G = -\frac{\partial \ln\omega_D}{\partial \ln V}$ and the Bulk modulus $B_0 = -\frac{\partial P}{\partial \ln V}$, we get $\frac{d\ln\lambda}{dP} = -\frac{\phi}{B_0}$, $\frac{d\ln E_F}{dP} = -\frac{\phi_F}{B_0}$ and $\frac{d\ln\omega_D}{dP} = \frac{r_G}{B_0}$, then Eq.(7) can

be written in the form

$$\frac{d\ln T_c}{dP}=\frac{1}{\ln\left(\frac{1.14E_F}{T_c}\right)}\frac{r_G}{B_0}\left\{\ln\left(\frac{E_F}{\omega_D}\right)-\frac{1}{\lambda}\frac{\phi}{r_G}-\ln\left(\frac{1.14\omega_D}{T_c}\right)\frac{\phi_F}{r_G}\right\} \tag{8}$$

From Eq.(8), the variation of the critical temperature on pressure varies on the Fermi energy and other parameters. For MgB_2 superconductor, the pressure dependence of the critical temperature depend on the change of the density of states in the Fermi level, the quasi harmonic phonon frequencies and the Coulomb pseudopotential [28-30]. For cuprates superconductors, T_c depend on pressure and number of holes in unit cell. The electron-phonon coupling constant and non-electron-phonon coupling constant can be increased the T_c on the pressure dependence of T_c scenario.

III. RESULTS AND DISCUSSION

This research study the pressure dependence of the critical temperature by using one-band model and taken into account the vHs density of states. The analytical formula of critical temperature depend on pressure can be derived from Eq.(8), we get the critical temperature equation as

$$T_c=\exp\left[\frac{1}{\ln\left(\frac{1.14E_F}{T_c}\right)}\frac{r_G}{B_0}\left\{\ln\left(\frac{E_F}{\omega_D}\right)-\frac{1}{\lambda}\frac{\phi}{r_G}-\ln\left(\frac{1.14\omega_D}{T_c}\right)\frac{\phi_F}{r_G}\right\}P\right] \tag{9}$$

The vHs density of states is observed in cuprates superconductors that can be applied for explained many physical properties in almost two-dimensional high temperature superconductors. To illustrate the critical temperature depended on pressure and how the effect of vHs density of states on T_c. We computed the T_c equation as show that in Eq.(9). For Hg - based superconductors, based on Eq.(9), we plot T_c versus P with the set of parameter values as $E_F=3,500$, $r_G=2.2$, $B_0=100$, $\omega_D=750$, $\phi,\phi_F=-1$ and $\lambda=0.35$. On the absent pressure, the critical temperature of these compounds are $T_c=98K$, $T_c=108.8K$, $T_c=126.8K$, $T_c=133K$ and $T_c=134K$ for $Hg-1201$, $Hg-1212(B)$ $Hg-1212(A)$, $Hg-1223(A)$ and $Hg-1223(B)$ respectively. Based on the above set of parameters, we showed that the graphs on figure 1. The trend of graphs are similar, the critical temperature increased as increasing pressure with positive value of the pressure coefficient dT_c/dP.

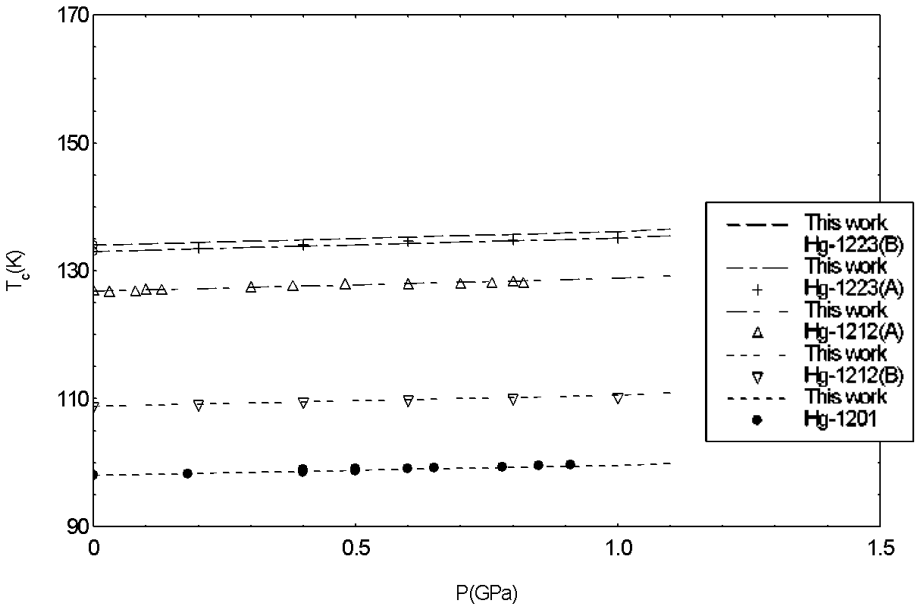

Figure 1: The variation of T_c versus P for Hg - based superconductors

From figure 1, we found that the critical temperature increased as increased pressure and can be fitted well with the experimental data of $Hg-1201$, $Hg-1212(A), Hg-1212(B)$, $Hg-1223(A)$ and $Hg-1223(B)$ superconductors. The pressure coefficient of these compounds were consistent with a value of $1.55K/GPa$, $1.72K/GPa$, $2.01K/GPa$, $2.11K/GPa$ and $2.12K/GPa$ for $Hg-1201$, $Hg-1212(B)$, $Hg-1212(A)$, $Hg-1223(A)$ and $Hg-1223(B)$ respectively. The pressure coefficient in this work agreed with the measurement data of Hg -based compounds superconductors were $1.80K/GPa$ and $1.71K/GPa$ for $HgBa_2CaCu_2O_{6+\delta}$ and $HgBa_2Ca_2Cu_3O_{8+\delta}$ respectively [31]. Our results confirmed that the mechanism of Hg -based superconductors can be explained by one-band model included the vHs density of states.

In case Bi -based superconductors, we used the same set of parameters as Hg - based. The critical temperature at zero external pressure was $T_c = 82.5K$, $T_c = 85K$ and $T_c = 102K$ for $Bi-2212(crystal)$, $Bi-2212(polycrystal)$ and $Bi,Pb-2223$ respectively. Based on Eq.(9), graph T_c versus P shown that on figure 2. The tendency of graphs were similar, the critical temperature increased as increasing pressure with positive value of the pressure coefficient.

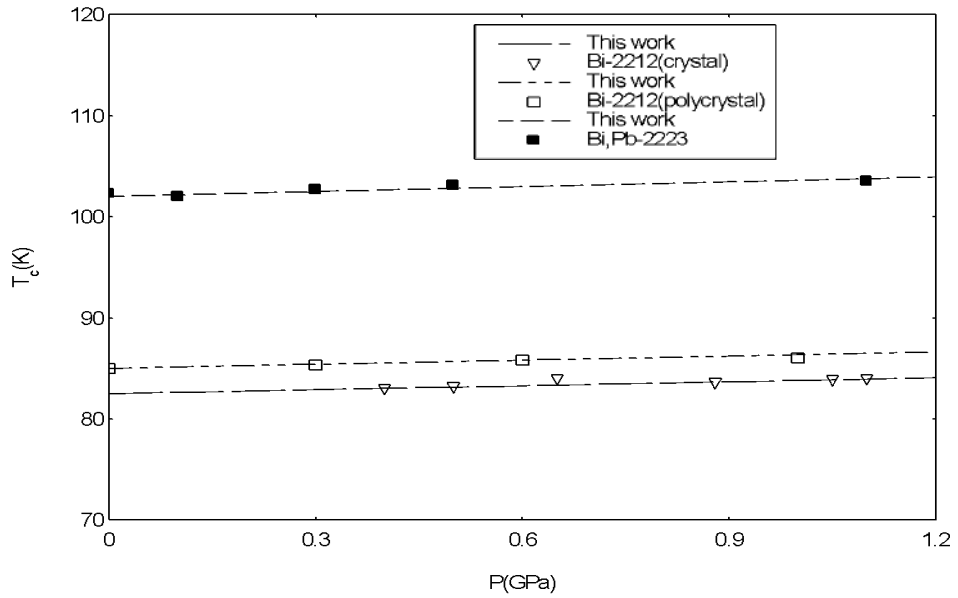

Figure 2: The variation of T_c versus P for Bi-based superconductors

From figure 2, we obtained the best fitted to the experimental data of Bi-based superconductors, $Bi-2212(Crystal)$, $Bi-2212(Polycrystal)$ and $Bi,Pb-2223$. The graphs in figure 2 were similar with the graphs for Hg-based superconductors. We found that the pressure coefficient all of compounds in figure 2 were positive values as $1.31K/GPa$, $1.35K/GPa$ and $1.48K/GPa$ for $Bi-2212(Crystal)$, $Bi-2212(Polycrystal)$ and $Bi,Pb-2223$ respectively. The positive values of dT_c/dP corresponded to the result from Klehe et al. [31], Kubiak et al. [32] and Chen et al. [33]. This work showed that the frame work of one-band model and taken into account the vHs density of states play an important role on the mechanism of Bi - based superconductors.

IV. CONCLUSION

This research study pressure dependence of the critical temperature in cuprates superconductors. The analytical formula of T_c is derived from the one-band model include the vHs density of states. On this work, the paring interaction is phonon mediate on weak coupling interaction. We have analyzed the results with the experimental data of Hg-based and Bi - based superconductors. In case of Hg -based superconductors, we found that the critical temperature increased as increasing pressure with different positive value of the pressure coefficient. For $Hg-1201$, $Hg-1212(B)$, $Hg-1212(A)$, $Hg-1223(A)$ and $Hg-1223(B)$, the pressure coefficient were $1.55K/GPa$, $1.72K/GPa$, $2.01K/GPa$, $2.11K/GPa$ and $2.12K/GPa$ respectively. These results get a good agreement with the measurement data reported from Klehe et al. [31]. The positive value of the pressure coefficient showed that the superconductivity in cuprates can be occurred at high pressure. Jover et al. suggested that the main contribution to the pressure coefficient is the intrinsic pressure effect [34]. Our results corresponded to data of high temperature superconductors at high pressure. In case of Bi-based superconductors, we found that the critical temperature increased as increasing pressure with different positive value of the pressure coefficient. The pressure coefficient for Bi-based in this work were $1.31K/GPa$, $1.35K/GPa$ and $1.48K/GPa$ for $Bi-2212(Crystal)$, $Bi-2212(Polycrystal)$ and $Bi,Pb-2223$ respectively. We get the best fit to the experimental data of Bi - based superconductors.

The trend of graphs corresponded with the report from Chanpoom [12]. Our results showed that the framework of one-band model and taken into account the vHs density of states play an important role on the mechanism of Hg -based and Bi -based superconductors.

ACKNOWLEDGEMENT

The author would like to thank Nakhon ratchasima Rajabhat University for financial support.

REFERENCES

1. G.I. Gonzalez-Pedreros, R. Baquero, Physica C **548**,132 (2018).
2. A.P. Drozdov, M.I. Eremets, I.A. Troyan, V. Ksenofontov, S.I. Shylin, Nature **525**,73 (2015).
3. Y. Iye, Chinese Journal of Physics **30(2)**, 229 (1992).
4. L. Gao, Y.Y. Xue, F. Chen, Q. Xiong, R.L. Meng, D. Ramirez, C.W. Chu, J.H. Eggert, H.K. Mao, Phys. Rev. B **50**, 4260 R (1994).
5. C.W. Chu, J. Bechtold, L. Gao, P.H. Hor, Z.J. Huang, R.L. Meng, Y.Y. Sun, Q. Wang, Y.Y. Xue, Phys.Rev.Lett. **60**, 941 (1988).
6. J.G. Lin, K. Matsuishi, Y.Q. Wang, Y.Y. Xue, P.H. Hor, C.W. Chu, Physica C **175**, 627 (1991).
7. A. Schilling, M. Cantoni, J.D. Guo, H.R. Ott, Physica C **178**,183 (1993).
8. L. Gao, Z.J. Huang, R.L. Meng, J.G. Lin, F. Chen, L. Beauvais, Y.Y. Sun, Y.Y. Xue, C.W. Chu, Physica C **213** , 261 (1993).
9. H. Takahashi, A. Tokiwa-Yamamoto, N. Môri, S. Adachi, H. Yamauchi, S. Tanaka, Physica C **218**, 1 (1993).
10. C. W Chu, L. Gao, F. Chen, Z.J. Huang, R.L. Meng, Y.Y. Xue, Nature **365**, 323 (1993).
11. M. Nunez-Regueiro, J.L. Tholence, E.V. Antipov, J.J. Capponi, M. Marezio, Science **262**, 97 (1993).
12. T. Chanpoom, Int. J. Mod. Phys. B **34**, 2050276 (2020).
13. V.G. Tissen, M.R. Osorio, J.P. Brison, N.M. Nemes, M. Garici Hernandez, L. Cario, P. Rodiere, S. Vieira, H. Suderow, Phys.Rev. B **87**, 134502 (2013).
14. J. Bardeen, L.N. Cooper, J.R. Schrieffer, Phys. Rev. **108**, 1175 (1957).
15. G.M. Eliashberg, Sov. Phys. JEPT. **12**, 1000 (1961).
16. O.A. Ogbuu, O. Abah, Physica C **519**, 100 (2015).
17. T. Chanpoom, Songklanakarin J. Sci. Technol. **43(6)**, 1604 (2021).
18. L. Van Hove, Phys.Rev. **89**, 1189 (1953).
19. J. Labbe, J. Bok, Europhys Lett. **3**,1225(1987).
20. J. Bok. J. bouvier, Journal of Superconductivity and Novel. Magnetism **25(3)**, 657 (2012).
21. M. Houssa, M. Ausloos, R. Cloots, Phys.Rev. B **56**, 6226 (1997).
22. J. Bok, J. Bouvier, Physica C **460-462**, 1010 (2007).
23. R.J. Radtke, M.R. Norman, Phys.Rev.B **50**, 9554 (1994).
24. K. Gofron, J.C. Campuzano, A.A. Abrikosov, M. Lindroos, A. Bansil, H. Ding,D. Koelling, B. Dabrowski, Phys. Rev.Lett. **3**, 3302 (1994).
25. W. sano, T. Koretsune, T. Tadano, R. Akashi, R. Arita, Phys.Rev.B **93**, 094525 (2016).
26. E. Cappelluti, L. Pietronero, Europhysics Letters, **36**, 619 (1996).
27. T. Chanpoom, Int. J. Mod. Phys. B **32**, 1850130 (2018).
28. T. Tomita, J. Hamlin, J.S. Schilling, D.G. Hinks, J.D. Jorgensen, Phys.Rev.B **64**, 092505 (2001).
29. X.J. Chen, H. Zhang, H.U. Habermeier, Phys.Rev.B **65**, 144514 (2002).
30. S. Deemyad, J.S. Schilling, D.G. Hinks, Physica C **361**, 227 (2001).
31. A.K. Klehe, J.S. Schilling, J.L. Wagner, D.G. Hinks, Physica C , **223**, 313 (1994).
32. R. Kubiak, K. Westerholt, G. Pelka, H. Bach, Y. Khan, Physica C , **166**, 523 (1990).
33. X.J. Chen, V.V. Struzkin, Y. Yu, A.F. Goncharov, C.T. Lin, H.K. Mao, R.J. Hemley, Nature **466**, 950 (2010).
34. D.T. Jover, H. Wilhelm, R.J. Wijngaarden, Physical Rev. B , **55(17)**, 11833 (1997).