AN INTELLIGENT DEVICE TO ANALYZE AND DETECT DRIVER ALERTNESS USING MEDIAPIPE AND RASPBERRY PI

Richard Feng[1], Jonathan Sahagun[2]

[1]Troy High School, 2200 Dorothy Ln, Fullerton, CA 92831
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Distracted driving is an important problem that plagues the world, and is the cause of thousands of deaths in the United States. Distracted driving has cost an estimate of $44.2 billion to the US in damages. A way I can help solve this problem is creating a device that can detect if the driver is distracted. This device contains an Arducam Day n Night 1080p camera, a Raspberry Pi 3, and a buzzer. The device utilizes python and Mediapipe, a library which can locate important landmarks on the driver's face [1]. During development, I had to face many challenges such as the driver possibly looking down subtlety with their eyes, or the camera being positioned in a weird way. During experimentation, I tested the delay, and accuracy of the device. After 35 trials, the device is accurate around 94%, and the delay is around 9 seconds. Overall, this device can be used by everyone who wants to be kept responsible and alert while driving.

KEYWORDS

MediaPipe, Raspberry Pi, AI, Driving

1. INTRODUCTION

With the reliance on technology growing everyday, it's understandable that drivers would occasionally sneak a glance at their phones every so often. But this moment of distraction can cause the driver their life. Distracted driving kills around 3,000 people each year, which makes up 8% - 9% of fatal car accidents [2]. Distracted driving is not only a problem for the drivers, but it also endangers the lives of surrounding drivers, pedestrians, and cyclists. Around 600 pedestrians and cyclists were killed by distracted driving during 2020 [2]. Ever since the creation of the first automobile, it was possible to become distracted. This problem has only grown with the creation of the smartphone. A study done in the United States shows that 13% of drivers text and drive, most of which are 18 - 24 years old [3]. It can be expected that this percentage will only go up as smartphones become more accessible and even easier to use. In addition to civilian casualties, city infrastructure can also be damaged due to crashes from distracted driving. The damage from car crashes has cost America $340 billion in damages, and costs taxpayers in the United States $30 billion in 2019, which is an additional $230 in taxes for every household [4]. The amount of damages caused by distracted driving can be calculated by analyzing the amount of police reported crashes of 2020, which was 5.25 million [6]. 13% of all police reported car accidents were reported as distraction related, so through assuming all car crashes in America

David C. Wyld et al. (Eds): NSEC, InWeS, CSIP, BMLI, CACIT, EDUIT -2023
pp. 19-27, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131502

being reported by the police, distracted driving can cause up to $44.2 billion [5]. Obviously this is not the most accurate estimation, but it gives perspective to the severity of this issue.

AdeptDriver tackles the distracted driving issue by using realistic car footage for a simulation, and asks the driver questions about the road. Although the simulation is realistic and addresses the problem clearly, it doesn't directly give the driver feedback while they're actually driving. My device is able to give real time feedback to the driver, which I view as more important. As for the device professors from India and South Africa created, its biggest flaw is the inability to perform at night. The accuracy of the device also isn't as high as mine, with an accuracy of 87.9 compared to 94.3 respectively. And as for the program centers that promote driving without phone use, its problem is similar to that of AdeptDriver, people can easily become distracted again when a text pops up, or when someone is calling, making them forget everything they learned. By having a very accurate AI model, and delivering instant feedback, my device can improve on these shortcomings [10].

I propose a device that can track the drivers facial movements, and will alert the driver with a distinct pinging noise if they're detected to be distracted or not. This device will help remind the driver to focus on the road. Because of how fast everything is on roads, a split second of distraction can cause severe consequences. This solution works well because of its immediate feedback on the driver. Setting up the device is also very easy, just requiring a car vent and car charger. The camera can be hooked up to the car vent, and the device can just be powered by the car battery. As long as the camera has a clear view of the driver, it will always alert them whenever they're determined to be distracted. Because of its simplicity, it's incredibly easy to set up and start driving. There have been many attempts at trying to solve this issue, such as apps that can lock your phone, or lessons on educating people about how dangerous this issue can be. However, each has its own downsides, as an app that locks the phone can prevent the driver from using any mapping software, and there are more ways to become distracted than being on your phone while driving. Teaching people the consequences of distracted driving is definitely a positive thing, but it ignores the fact that distracted drivers can get into accidents because of the fact that they're distracted. An issue comes up while driving, and people can forget about the consequences, and tunnel vision on the distraction.

The experiments I ran were to test the delay, and accuracy of the device. In order to test the delay, I would have the camera analyze my face, become distracted on purpose, and start a stopwatch. When the device detected that I was distracted, the buzzer started to ring, and the stopwatch would be paused. As for the accuracy of the device, I would glance down at my phone every so often, and check if the device can catch it. If the device catches my distractedness, it earns a score of 1/1. If it doesn't it earns a score of 0/1. For both experiments, I ran around 35 trials for an accurate result. For the delay experiment, there was an average delay of 9.36 seconds, and is largely due to the processing power of the Raspberry Pi. As for the accuracy of the device, the device is accurate 94.3% of the time, which far exceeded my expectations.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Catch Driver's Faces

Because the camera is how the device will view the driver, there would've needed to be a strategic way to set up the camera so it faces the driver directly, while not obstructing the vision of the driver. One way to solve this problem would've been to attach the camera to the roof of the

car, but that would create an undesirable angle, as the camera would need to be facing down on the driver. One way to achieve my goal would be to attach a vent mount on the camera, and place the camera in front of the driver vent. This would work greatly because the camera is positioned low enough to not be distracting while also having a good view on the driver.

2.2. How the Computer Would Know if the Driver is Distracted

The Raspberry Pi acts as the computer in this device, and is responsible for all the computing done [11]. The Raspberry Pi hosts all of the code, and it determines whether or not the driver is distracted. The biggest problem with this project was determining how the computer would know if the driver is distracted. The easiest way to solve this is by using pose estimation, more specifically mediapipe. Mediapipe allows the computer to detect landmarks on the driver's face, such as where the nose, eyes, and chin is [12]. By doing this, I can analyze patterns in people's faces whenever they look down, or some other behavior that would hint towards being distracted. The computer would then be programmed using the mediapipe library to look for these behaviors, and notify the driver whenever it's detected.

2.3. The buzzer

The buzzer in this device allows for communication to the driver, as whenever the driver is detected to be distracted, the buzzer will sound. The buzzer will ping in intervals, and it only stops after the driver is detected to be focused on the road again. The buzzer is also very customizable for its cheap price, as it can play a wide variety of different notes. If the driver is detected to be somewhat distracted, the buzzer will have a countdown, and once the countdown is finished and the driver is still semi-distracted, the buzzer will also start alerting the driver.

3. SOLUTION

The three major components that make up this device are the camera, Raspberry Pi, and buzzer. The camera is an Arducam Day & Night Vision. The camera is 1080p, and can operate in darkness. It is connected to the Raspberry Pi 3, and feeds it data from the driver's face. The Raspberry Pi acts as the computer in the device, and determines if the driver is distracted or not given the data from the camera. A buzzer is connected to the Raspberry Pi, and will produce a buzzing sound whenever the Raspberry Pi detects the driver to be distracted [13]. The device starts when a face is detected with the camera. The camera feeds the Raspberry Pi information to the driver's face. Whenever the driver is detected to be distracted, the buzzer gets activated, and alerts the driver. As mentioned previously, the buzzer can still activate if the driver is detected to be semi-distracted for too long. This cycle repeats until the device is turned off, or until no face is detected. All the devices in this project were specifically picked for their efficiency while still being cheap. Altogether, this project cost around $75, with the camera and Raspberry Pi both costing $35, and a buzzer costing $5. The code in this device was written in python, and utilizes Mediapipe, a machine learning library that analyzes the driver's face. It also uses libraries such as numpy for mathematical calculations, pygame for creating a window of the driver's face, and the time library for creating countdowns.

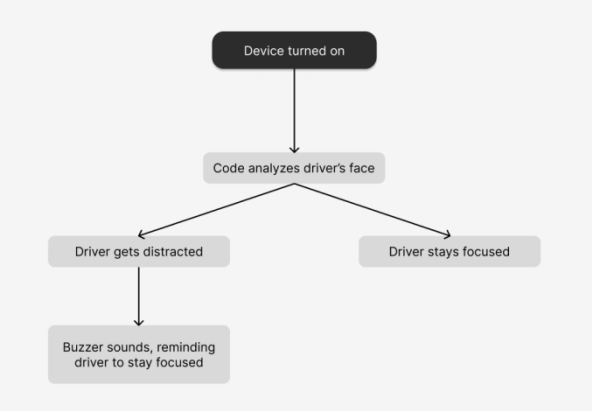

Figure 1. Overview of the solution

The Raspberry Pi is the device that allows everything to work in this project. As mentioned previously, mediapipe was used in the creation of this device. Mediapipe allows the device to find landmarks on the person's face, such as the nose, eyes, and chin. After receiving that information, the Raspberry Pi can use math to know where the middle point of the face is, and create lines to better visualize the face.

```
# draw on image
if results.face_landmarks:
    theta, phi = drawFaceLandmarks(annotated_image, results.face_landmarks, 45 ,0)
    code = isDistracted(theta, phi)
    if code == -1:

        print("distracted")
        buzzer.start(cycle)

        if distracted:
            distracted_durration = time.time() - distracted_start_time
        else:
            distracted = True
            distracted_start_time = time.time()
            distracted_durration = 0

    elif code == 1:
        print("semi distracted")

        if distracted:
            distracted_durration = time.time() - distracted_start_time
        else:
            distracted = True
            distracted_start_time = time.time()
            distracted_durration = 0

        if distracted_durration > distracted_durration_threshold:

            buzzer.start(cycle)


    if code == 0:
        print("not distracted")
        buzzer.stop()
        distracted = False
```

Figure 2. Screenshot of code 1

This code classifies the driver's face given the information of their face. Theta and phi are the values that determine where the driver is looking. Theta is a number between 0 - 180, 0 meaning looking up, and theta means looking up. Phi is a number between 0 - 360, and determines which direction the driver is looking. 0 means looking ahead. Calculations are done to phi to make it either positive or negative, positive means looking left, and negative means looking right. Given theta and phi, it is run through the isDistracted function, which combines those two variables and gives it sense. The output of isDistracted gives us code, which classifies the driver. If the driver is distracted, the buzzer starts making noise, and a distracted duration starts. If the driver is seen to be semi-distracted, the distraction duration also starts, and if it crosses the threshold (2 seconds), the buzzer starts. Nothing happens if the driver is detected to not be distracted.

The camera is the device that feeds information to the Raspberry Pi. After being attached to the car vent, the camera first looks for a face to analyze. After finding one, it uses mediapipe to find the landmarks, and math to determine where the driver is looking. The information fed to the Raspberry Pi is updated every frame.

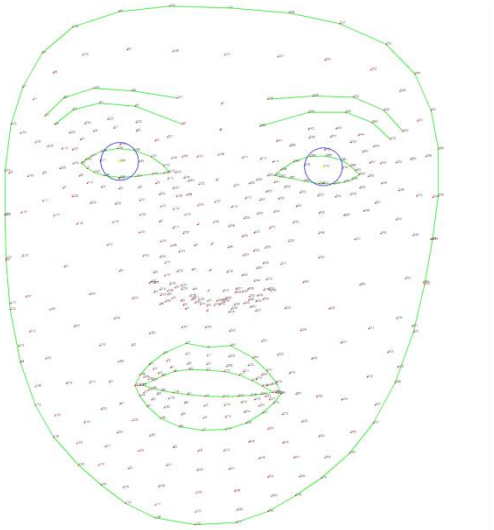

Figure 3. The face shape

```
def drawFaceLandmarks(image, face_landmarks, offset_angle = 0, offset_incline = 0):
    # returns theta and phi
    width = image.shape[1]
    height = image.shape[0]

    forehead = face_landmarks.landmark[10]
    x1 = round(forehead.x * width)
    y1 = round(forehead.y * height)
    z1 = round(forehead.z * width)

    chin = face_landmarks.landmark[152]
    x2 = round(chin.x * width)
    y2 = round(chin.y * height)
    z2 = round(chin.z * width)

    # Draw Line down the middle of the face
    if show_image:
        cv2.line(image, (x1, y1), (x2, y2), (0, 255, 0), thickness=1)

    # Midde point of face
    mid = findMidPoint(x1, y1, x2, y2)

    mid_x = mid[0]
    mid_y = mid[1]
    mid_z = (z1+z2)/2

    # Draw Middle Point of Face
    if show_image:
        cv2.circle(image, mid, 5, (255, 0, 255), 2)

    nose = face_landmarks.landmark[4]
    nose_x = round(nose.x * width)
    nose_y = round(nose.y * height)
    nose_z = round(nose.z * width)
```

Figure 4. Screenshot of code 2

The function drawFaceLandmarks finds the important landmarks on the drivers face, and stores them as variables for later use. The important landmarks being stored here are the forehead, chin, and nose. The mediapipe labels each important landmark of the face as a number, with the forehead being landmark 10, the chin being landmark 152, and the nose being landmark 4. Given those three landmarks, even more information can be gathered on the driver, such as finding the midpoint of the face given the x and y coordinates of the driver's forehead and nose. A line can also be created to better visualize it. Theta and Phi then use the positions of the forehead, nose, and chin to calculate where the driver is looking. Every frame, the camera takes a picture, feeds the image to the Raspberry Pi, the position of the landmarks are updated, and the driver's status is determined. This repeats until the Raspberry Pi is shut off, or until a face is no longer detected.

Although the buzzer may not be the most important part, it still serves an important purpose of giving information to the driver. Whenever the driver is seen to be distracted, the buzzer sounds, alerting the driver to focus on the road. The sound only stops after the driver is focused again.

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
buzzerPin = 18
GPIO.setup(buzzerPin, GPIO.OUT)
buzzer = GPIO.PWM(buzzerPin, 261.63)
cycle = 50
```

Figure 5. Screenshot of code 3

This code sets up the buzzer and allows the Raspberry Pi to communicate to it properly, through specifying the pins it's connected to. In order to have the code communicate to the raspberry pi properly, rpi.gpio is imported to connect the code with the buzzer. Rpi.gpio is a python library that allows python to connect to the gpio pins on the Raspberry Pi. Because the buzzer is connected through those pins, it's needed in the code. Along with rpi.gpio, time is also imported to start countdowns on when the buzzer should be activated. When the driver is distracted, a countdown starts, and once the countdown is over and the driver is still distracted, the buzzer will be activated. As mentioned earlier, the buzzer can still be activated if the driver is semi-distracted. The only difference now is that the countdown is now longer before the device will react.

4. EXPERIMENT

4.1. Experiment 1

A possible blind spot in my program is the AI being too delayed to sense when the driver is distracted. It is important that this doesn't happen so the buzzer alerts the driver in time.

I can test the delay of the AI by purposefully becoming distracted in front of the camera, and record the delay of when I become distracted to when the buzzer sounds. I will have the camera pointed towards me, and a phone on my lap with a stopwatch open. By periodically looking down and starting the stopwatch, I can track when I start to become distracted, and when the buzzer sounds. There will not be any control data, and because I can repeat this process multiple times, I can run many trials for an accurate result, and I'll stop when the results are all similar.

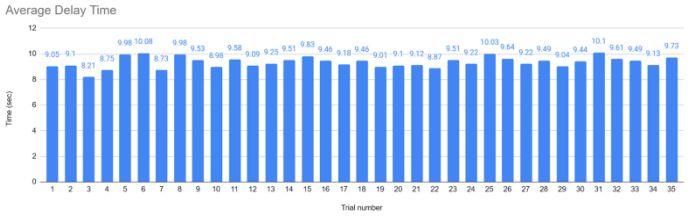

Figure 6. Figure of experiment 1

The mean of the data is 9.36, and the median is 9.44. The range of the data is between 8.21 and 10.08.This means that whenever a driver is looking down and not focused, the device will take around 9.36 seconds to notify the driver. Because this experiment was hand timed, there will be some errors. This delay is mainly due to the processing power of the Raspberry Pi, as its cheap price and size doesn't allow for extremely fast calculations. An important part of this design is developing a good product with a cheap price. Although Iwasn't surprised that there was a delay, the fact that the delay was around 9 seconds long was a little bit concerning. The most important part of my development now is focusing on how to reduce the delay without replaying the Raspberry Pi 3 with a better model, such as a Raspberry Pi model 4 or Google Dev Board.

4.2. Experiment 2

Another potential blind spot in this device is not accurately detecting if the driver is distracted or not. This is probably the most important part of the device, and it is vital that it works properly.

Testing the accuracy of the AI can be done by subtlety glancing at my phone while I'm pretending to be driving. If the AI detects my distractedness, then it will earn a score of 1/1. If the AI fails to do so, it will earn a score of 0/1. I will repeat this process many times until I have an accurate number that can represent how accurate the AI is. At the end of the experiment, I will add up all the times the AI scored a point, and divide it by the number of trials done [14]. This will give a decimal that can gauge how accurate the AI is. The closer to one, the more accurate it is, and the closer it is to zero, the less accurate it is.

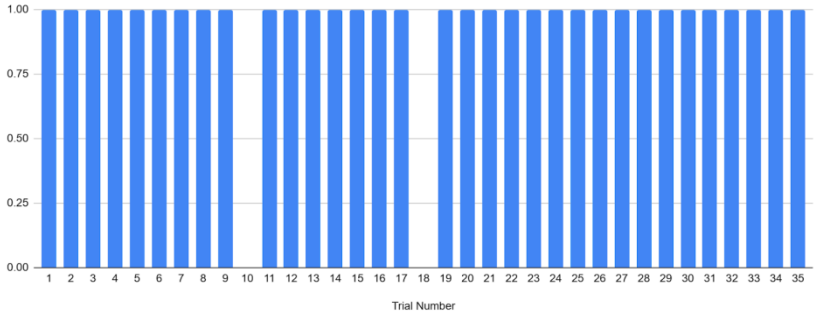

Figure 7. Figure of experiment 2

Throughout this experiment, I found that the device was excellent at detecting whenever the driver was distracted. Out of 35 trials, the average of the data is 0.943, meaning that the device is accurate 94.3% of the time. Although the device was excellent at detecting if the driver was distracted, it wasn't as accurate when determining if the driver was focused or not, as there would be times when I was looking ahead, and the buzzer would keep making noises. I think this is mostly because the code makes it very specific on where the driver must be looking in order to be focused on the road, which is why the code catches distraction most of the time. The times where the device wasn't able to catch me being distracted was when I positioned my head to face the side windows, with my eyes looking down. Although the AI can detect where my eyes are looking, it's hard to do so when my head is facing the sides, where it's harder to track where my eyes are looking.

5. RELATED WORK

A paper written by the CEO of Adept Driver tackles the same problem my device attempts to solve [7]. Their solution is through creating a driving simulation, and asking the driver to answer certain questions about the road, such as clicking on which cars that are relevant to their driving. This solution uses real car footage, so it's very realistic. But a simulation is not reality, and it's hard to apply training to the actual driving experience. Something can come up, drivers can become distracted and forget about the risks. This is why my device focuses on giving spontaneous feedback to alert the driver quickly. My device focuses on real-time driving, while Adept Driver uses simulation footage.

Professors from India and South Africa developed a system that can detect the specific task the driver is performing while driving, such as eating food, drinking, or talking to the passenger [8].

This paper utilizes very advanced technologies, such as VGG-16 and ResNot-50. The solution they propose is very similar to mine, but is a bit less accurate, with an accuracy of 87.9% with ResNot-50. Although it's accuracy is not as high. Its AI is very advanced, and is more specific than mine. However, the biggest downside to this solution is that the paper never mentioned its ability to perform at night. Driving at night is usually even more dangerous than during the morning due to limited visibility, so the ability for the device to operate at night is very important.

Researchers from Texas promoted programme centers that advocated for drivers to abstain their use of phones during driving [9]. Because program centers are generally safe spaces that promote a learning environment, this idea isn't too bad. However, the biggest issue with this idea is that many drivers only go to program centers such as these after they've been in an accident. And even if drivers go there before they've been in an accident, just like the problem mentioned in AdeptDriver, drivers can get caught up in things, and the lessons learned can be forgotten. This is why I still believe a device that gives real-time feedback is more important.

6. CONCLUSIONS

A big limiter in this project was the power of the Raspberry Pi. Because keeping the device small and compact was an integral part, the "computer" of the device couldn't be too large, which limited the power of the device. The limited power would cause a small delay in the code. In addition to this, the buzzer was chosen also due to its small size. Because of this, the buzzer may not be loud enough in certain situations, such as when a car is honking at the driver. If given more time and a larger budget, the buzzer would simply be switched out with a small speaker capable of producing louder noises, and have more customizable options for the beeping noise. Changing the Raspberry Pi would be a matter of budget, as the current Raspberry Pi 3 could be replaced with the latest Raspberry Pi model, or some other device [15].

The device can be improved, but it mostly depends on the budget and size constraints. The current design works great for its cheap cost, even if there are some downsides. In the future, if given a larger budget, it would be possible to create a much better version of this device, with all the current flaws solved.

REFERENCES

[1] Mashaw, Jerry L., and David L. Harfst. "Inside the national highway traffic safety administration: Legal determinants of bureaucratic organization and performance." U. Chi. L. Rev. 57 (1990): 443.
[2] Steadman, Mindy, et al. "CU L8ter: YouTube distracted driving PSAs use of behavior change theory." American journal of health behavior 38.1 (2014): 3-12.
[3] Braitman, Keli A., and Anne T. McCartt. "National reported patterns of driver cell phone use in the United States." Traffic injury prevention 11.6 (2010): 543-548.
[4] Ranney, Thomas A., W. Riley Garrott, and Michael J. Goodman. NHTSA driver distraction research: Past, present, and future. No. 2001-06-0177. SAE Technical Paper, 2001.
[5] Diegelmann, Svenja, Katharina Ninaus, and Ralf Terlutter. "Distracted driving prevention: An analysis of recent UK campaigns." Journal of Social Marketing 10.2 (2020): 243-264.
[6] Stewart, Timothy. Overview of motor vehicle crashes in 2020. No. DOT HS 813 266. 2022.
[7] Vegega, Maria, Brian Jones, and Chris Monk. Understanding the effects of distracted driving and developing strategies to reduce resulting deaths and injuries: a report to congress. No. DOT HS 812 053. United States. Office of Impaired Driving and Occupant Protection, 2013.
[8] Vaegae, Naveen Kumar, et al. "Design of an Efficient Distracted Driver Detection System: Deep Learning Approaches." IEEE Access 10 (2022): 116087-116097.
[9] Overton, Tiffany L., et al. "Distracted driving: prevalence, problems, and prevention." International journal of injury control and safety promotion 22.3 (2015): 187-192.

[10] Zhu, Hao. "Big data and artificial intelligence modeling for drug discovery." Annual review of pharmacology and toxicology 60 (2020): 573-589.
[11] Jolles, Jolle W. "Broad-scale applications of the Raspberry Pi: A review and guide for biologists." Methods in Ecology and Evolution 12.9 (2021): 1562-1579.
[12] Lugaresi, Camillo, et al. "Mediapipe: A framework for building perception pipelines." arXiv preprint arXiv:1906.08172 (2019).
[13] Zhao, CheahWai, JayanandJegatheesan, and Son Chee Loon. "Exploring iot application using raspberry pi." International Journal of Computer Networks and Applications 2.1 (2015): 27-34.
[14] Ramesh, A. N., et al. "Artificial intelligence in medicine." Annals of the Royal College of Surgeons of England 86.5 (2004): 334.
[15] Imteaj, Ahmed, et al. "An IoTbased fire alarming and authentication system for workhouse using Raspberry Pi 3." 2017 International conference on electrical, computer and communication engineering (ECCE). IEEE, 2017.