Mini-Project Report On

RAKSHA -Road Accident Prediction Using Data Fusion Technique

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

By

Elina Joy (U2003076)
Hemi Rose Sajeev(U2003093)
Lakshmi Thampi Anoopkumar (U2003123)
Merin Jose (U2003133)

Under the guidance of
Ms Tripti C

Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology (Autonomous)
Rajagiri Valley, Kakkanad, Kochi, 682039

July 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039

CERTIFICATE

This is to certify that the mini-project report entitled ***"RAKSHA -Road Accident Prediction Using Data Fusion Technique"*** *is a bonafide work done by* ***Ms. Elina Joy(U2003076),Ms. Hemi Rose Sajeev (U2003093), Ms. Lakshmi Thampi Anoopkumar (U2003123), Ms. Merin Jose (U2003133)****, submitted to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2022-2023.*

Dr. Preetha K. G.	**Mr. Uday Babu P.**	**Ms. Tripti C**
Head of Department	Mini-Project Coordinator	Mini-Project Guide
Dept. of CSE	Asst. Professor	Asst. Professor
RSET	Dept. of CSE	Dept. of CSE
	RSET	RSET

ACKNOWLEDGEMENT

We are extremely honoured to thank **Prof (Dr) P. S. Sreejith**, Principal, RSET, and **Dr. Preetha K. G.**, Head of Department of Computer Science and Engineering, for providing us with the opportunity to undertake our mini-project,"Raksha-Road Accident Prediction Using Data Fusion Technique".

We express our gratitude to our Project Coordinator and Class-In-Charge, **Mr. Uday Babu P**, Asst.Professor, Dept. of Computer Science and Engineering, for his support and timely help whenever it was required. He has been a great mentor without whom this project would have been incomplete.

We extend our sincere and heartfelt thanks to our guide **Ms. Tripti C**, Asst.Professor, Dept. of Computer Science and Engineering, for taking the time and effort to review our work and providing valuable advice and feeback from time to time.

Last but not the least, we would like to express our sincere and heartfelt gratitude to our family and friends for their constant support throughout this entire journey.

Elina Joy

Hemi Rose Sajeev

Lakshmi Thampi Anoopkumar

Merin Jose

ABSTRACT

RAKSHA - Road accident prediction using Data Fusion Technique aims to develop an advanced safety system for self-driving vehicles that leverages data fusion techniques to enhance risk assessment and accident prevention. The system integrates data from multiple sensors, including ultrasonic distance sensors and temperature sensors, using a data fusion algorithm. This algorithm combines and processes the sensor data to provide a more accurate and comprehensive understanding of the vehicle's surroundings and environmental conditions. The estimation of data using sensor fusion is crucial for making informed decisions related to risk assessment and accident prevention. The data fusion algorithm utilizes advanced techniques such as Kalman filtering. Based on the fused sensor data, the system calculates the accident probability by considering various factors such as the distance to obstacles, vehicle speed, and environmental conditions. This probabilistic assessment provides valuable insights into the likelihood of potential accidents occurring. Furthermore, the system determines the severity level of potential accidents by employing advanced decision algorithms, such as the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS). n situations where the severity level is identified as high, the system activates an alert system to proactively communicate and warn nearby vehicles. This alert system, implemented using UDP (User Datagram Protocol), facilitates quick and reliable message transmission between vehicles. By sending alert messages, the system promotes cooperative communication among vehicles, enabling them to take appropriate evasive actions and mitigate potential accidents. Thus the proposed system , by effectively processing and fusing sensor data, assessing severity, and enabling timely communication, strives to contribute to the development of safer and more efficient self-driving transportation systems.

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 General Background

In recent years, road accidents have posed significant risks to public safety, necessitating the development of advanced systems for accident prediction, severity assessment, and timely alert generation. The occurrence of road accidents is influenced by various factors such as driver behavior, road conditions, vehicle conditions, and environmental factors.Road accidents have far-reaching consequences, both in terms of human lives and economic impact. According to global statistics, millions of people lose their lives or suffer injuries in road accidents each year. It is imperative to address the issue of road accidents to protect lives and promote societal well-being.

Understanding the causes and implications of road accidents involving autonomous vehicles is crucial for improving their technology and ensuring public trust.This project,designed for autonomous vehicles, aims to address these challenges by incorporating ultrasonic sensors to monitor vehicle presence and proximity to obstacles for accurate accident prediction, along with temperature sensors to gather road condition and weather data to enhance accident prediction accuracy integrated with Kalman filter to enhance sensor reliability and the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) algorithm to predict the severity.

The entire working of the system is based on a simple electronic circuitry arrangement which senses the surrounding conditions to make prediction about the possibility of occurence of an accident , the result of which is taken for TOPSIS execution integrating information from a set of data, to predict the severity of the accident(if occured), which then alerts the neighbouring vehicles around 100 metres apart.

1.2 Objectives

- Incorporate ultrasonic sensors to monitor vehicle presence and proximity to obstacles for accurate accident prediction.
- Utilize temperature sensors to gather road condition and weather data to enhance accident prediction accuracy.
- Integrate the Kalman filter to improve the accuracy and reliability of sensor data.
- Employ the TOPSIS algorithm for severity prediction and multi-criteria decision-making.
- Develop an alert system that generates timely notifications to authorities and drivers based on predicted accidents and severity levels.

1.3 Motivation

The project is driven by a strong motivation to enhance road safety and mitigate the consequences of accidents. By accurately predicting road accidents, assessing their severity, and generating real-time alerts, the project aims to prevent accidents, save lives, and optimize resource allocation. The system's integration with autonomous vehicles and its potential to improve traffic management further highlights its significance. Through this project, we contribute to advancements in road safety measures, leveraging cutting-edge technologies and algorithms to create a safer and more secure road environment for everyone.

1.4 Existing Systems

Previous research in road accident prediction and severity assessment has explored various approaches, including sensor-based systems and data analysis techniques. Some studies have utilized ultrasonic sensors to detect the presence of vehicles and proximity to obstacles, while others have leveraged temperature sensors to monitor road conditions and weather patterns. Additionally, the Kalman filter has been employed to improve sensor data accuracy, and the TOPSIS algorithm has been used for multi-criteria decision-making. However, there is a need for an integrated framework that combines these com-

ponents to enhance the accuracy and effectiveness of road accident prediction, severity assessment, and alert systems.

1.5 Summary of Report

The primary goal of the project is the development of an integrated system for road accident prediction, severity prediction, and alert generation. By incorporating ultrasonic and temperature sensors, along with the Kalman filter and TOPSIS algorithm, the system leverages real-time data to accurately predict accidents, assess their severity, and generate timely alerts. The project aims to enhance road safety, optimize resource allocation, and enable real-time decision-making. The outcomes contribute to advancements in intelligent transportation systems and provide valuable insights for improving road safety measures.

Chapter 2

LITERATURE SURVEY

2.1 Automatic Road Accident Detection using Ultrasonic Sensor

Automatic Road Accident Detection Using Ultrasonic Sensor is a research paper that proposes an innovative system to enhance road safety. The system utilizes an ultrasonic sensor to measure distances from nearby objects, such as vehicles or obstacles. When the sensor detects a potential accident situation based on predefined thresholds, the system triggers immediate alerts by integrating GPS and communication modules. The main objective of the system is to reduce response time and promptly notify relevant authorities, enabling them to provide timely assistance and mitigate accident risks. The integration of real-time accident detection and quick alerting mechanisms has the potential to significantly improve road safety and contribute to the overall goal of reducing road accidents and their associated casualties.

By leveraging ultrasonic sensors and intelligent alerting mechanisms, the proposed system aims to detect accidents promptly, allowing authorities to respond quickly to provide assistance. The system's effectiveness in reducing response time and improving road safety would depend on successful implementation, addressing potential drawbacks such as limited detection range, environmental interference, and false alerts. Further research and development in this area could refine the system and pave the way for its practical application, potentially saving lives and making our roads safer for all users.

2.2 Ultrasonic Distance Detector Using Arduino

The project "Ultrasonic Distance Detector Using Arduino" by Mansi Sirumalla is an implementation of an ultrasonic distance measuring system using an Arduino microcontroller. The project utilizes an ultrasonic sensor to measure the distance between the sensor and an object in front of it. The Arduino processes the data from the sensor and calculates the distance based on the time taken for the sound waves to bounce back. The measured distance is then displayed on an output device, such as an LCD or through a serial monitor interface. This simple yet effective project showcases the application of Arduino and ultrasonic sensors for distance measurement and can have practical uses in various fields, including robotics, automation, and object detection.

2.3 Application of TOPSIS Method for Decision Making

The TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) method is a multi-criteria decision-making technique used to rank alternatives based on their performance against a set of criteria. It is a widely employed method in various fields, including management, engineering, economics, and environmental studies. The TOPSIS method aims to find the best compromise solution among a set of alternatives by comparing them to ideal and anti-ideal solutions.

The basic steps involved in the TOPSIS method,according to the paper are as follows:

Identify Criteria: Determine the criteria that are relevant to the decision-making problem. These criteria should be measurable and represent the different aspects of the alternatives being evaluated.

Normalize the Decision Matrix: Convert the raw data of the alternatives and their performance on each criterion into a standardized form, usually by normalization, to bring them to a common scale.

Construct the Weighted Normalized Decision Matrix: Assign weights to the criteria to reflect their relative importance. Multiply each criterion's normalized value by its corresponding weight to create a weighted normalized decision matrix.

Determine the Ideal and Anti-Ideal Solutions: In the weighted normalized decision matrix, identify the best and worst values for each criterion. The ideal solution represents

the maximum value for benefit criteria and the minimum value for cost criteria, while the anti-ideal solution is the opposite.

Calculate the Euclidean Distances: Calculate the Euclidean distances of each alternative from the ideal and anti-ideal solutions. The Euclidean distance represents the relative similarity of each alternative to the ideal and anti-ideal solutions for each criterion.

Calculate the Relative Closeness: Compute the relative closeness of each alternative by comparing its distance to the ideal solution with the sum of its distances to the ideal and anti-ideal solutions. The relative closeness ranges from 0 to 1, with higher values indicating better rankings.

Rank the Alternatives: Rank the alternatives based on their relative closeness values. The alternative with the highest relative closeness value is considered the best compromise solution.

The TOPSIS method provides a systematic approach for decision-makers to evaluate multiple alternatives with multiple criteria, helping them make well-informed decisions by considering various factors simultaneously. It has found applications in various domains, such as project selection, supplier selection, product evaluation, and resource allocation, among others.

It's important to note that the effectiveness of the TOPSIS method depends on the proper identification of criteria, appropriate weight assignment, and the quality of data used in the decision-making process. Additionally, like any decision-making method, TOPSIS has its own assumptions and limitations, and its application should be carefully tailored to the specific context and requirements of each decision problem.

2.4 Kalman Filter and Its Application

Kalman filter is a minimum-variance estimation for dynamic systems and has attracted much attention with the increasing demands of target tracking. Various algorithms of Kalman filter was proposed for deriving optimal state estimation in the last thirty years. This paper briefly surveys the recent developments about Kalman filter (KF), Extended Kalman filter (EKF) and Unscented Kalman filter (UKF). The basic theories of Kalman filter are introduced, and the merits and demerits of them are analyzed and compared.

Kalman filter has been widely applied in the fields of orbit calculation, target tracking and navigation, such as calculations of spacecraft orbit, tracking of maneuvering target and positioning of GPS. Further more, it also plays an important role in the fields of integrated navigation and dynamic positioning, sensor data fusion, microeconomics, and especially in the field of digital image processing and the current hot research fields like pattern recognition, image segmentation and image edge detection.

Comparison

The research paper[4] specifies only a single level filtration method.Various filtering methods that exist currently have demerits like the complex structure of algorithms, lack real-time tracking and reliability. We should strengthen the combination of theory and practical application of Kalman filter to improve its practicability.

Complexity: Implementing and managing separate components can lead to increased overall system complexity, especially when dealing with data synchronization and communication between the modules.

Data Inconsistency: Operating components independently might result in data inconsistencies or time lags between the modules, especially when handling real-time data.

Communication Overhead: When components operate independently, there may be additional communication overhead between them, leading to increased latency and complexity in data exchange.

Resource Consumption: Each component requires its own computational resources, memory, and potentially separate microcontrollers or hardware, leading to higher resource consumption compared to an integrated solution.

Maintenance Overhead: Managing and maintaining multiple components might be more challenging than dealing with a unified solution, as any updates or changes could impact multiple modules.

Chapter 3

System Analysis

3.1 Expected System Requirements

The expected system requirements of our project RAKSHA include:

3.1.1 Hardware Requirements

- Arduino UNO.
- DHT11 temperature sensor.
- HC-SR04 ultrasonic sensor.
- Vehicle with a compatible screen or display for visual alertspeakers for ae alerts to inform the driver of potential hazards.

3.1.2 Software Requirements

- Arduino 1.8.16 IDE for code development and uploading to Arduino UNO.
- Python as the primary programming language for algorithm implementation and data processing.
- Specific sensor libraries for data acquisition and processing.
- UDP server setup and implementation for the vehicle screen.
- Firebase for data storage and real-time communication.

3.1.3 End-User Requirements

- Vehicle with a functioning display or screen for receiving alert messages.

- Proper integration of the Arduino and sensors with the vehicle's electronics.
- Basic understanding of vehicle electronics and wiring for installation.
- An internet connection is required for data transmission and updating the alert system, ensuring the latest safety information is available to the driver.

3.2 Feasibility Analysis

3.2.1 Technical Feasibility

The technical feasibility of our project is well-demonstrated through the successful integration and implementation of various hardware and software components.

The core hardware components used include the Arduino UNO micro-controller, the DHT11 temperature sensor, and the HC-SR04 ultrasonic sensor. Arduino UNO, being a widely used and versatile microcontroller, provides a reliable platform for interfacing with different sensors and actuators. The DHT11 temperature sensor accurately measures ambient temperature, while the HC-SR04 ultrasonic sensor precisely calculates distance based on ultrasonic waves, both of which are crucial for our project's functionality.

On the software side, Python programming language was employed to develop the algorithms for data manipulation, real-time data processing, and decision-making. Python's ease of use, extensive libraries, and compatibility with Arduino made it an excellent choice for our project. Additionally, the Arduino 1.8.16 IDE played a significant role in programming the Arduino Uno and uploading the code.

The successful integration and synchronization of these hardware and software elements showcase the technical feasibility of our project. The system performs its intended tasks efficiently and accurately, capturing real-time data from the sensors, processing it through the algorithms, and generating meaningful insights. This lays a strong foundation for potential future enhancements and scalability. Moreover, the utilization of widely available and accessible technologies ensures that the project can be easily replicated and deployed in various applications beyond its current scope.

3.2.2 Operational Feasibility

The operational feasibility of our project is highly promising, ensuring the successful implementation of Arduino UNO, DHT11 temperature sensor, and HC-SR04 ultrasonic sensor. The system's simplicity and user-friendly nature allow for easy installation, configuration, and maintenance, enabling seamless adoption by developers and end-users alike. With real-time data processing capabilities provided by the Arduino UNO, the system offers accurate and timely information on the vehicle's surroundings and temperature conditions, critical for safe navigation and adapting to diverse weather scenarios. The utilization of widely used programming languages like Python with the Arduino IDE further enhances the operational feasibility, empowering developers to leverage their existing expertise effectively.

Overall, our project presents a robust and efficient operational solution, making it highly feasible for enhancing autonomous vehicle safety and functionality.

3.2.3 Economic Feasibility

The economic feasibility of our project is promising, primarily due to its potential to enhance safety and efficiency in autonomous vehicles. By implementing a smart system that incorporates Arduino UNO, DHT11 temperature sensor, and HC-SR04 ultrasonic sensor, the project offers a cost-effective approach to creating autonomous vehicles. These hardware components are affordable and widely available, reducing the development cost significantly.

Moreover, the software requirements, including the Arduino 1.8.16 IDE and VS Code, are free to use, making them accessible to developers without incurring additional expenses.

3.3 Hardware Requirements

The following are the hardware requirements to develop our project RAKSHA.

3.3.1 Arduino UNO

The Arduino UNO is a widely used open-source micro-controller board based on the ATmega328P micro-controller. It is one of the most popular boards in the Arduino family

due to its simplicity, versatility, and ease of use. The Arduino UNO board comes with digital input/output pins, analog input pins, PWM (Pulse Width Modulation) pins, and more, making it suitable for a wide range of applications. It can be easily programmed using the Arduino IDE, allowing users to upload code and interact with various sensors and actuators.

The Arduino UNO will serve as the central processing unit for the autonomous vehicle, interfacing with various sensors and actuators to control its movements and collect data.

3.3.2 HC-SR04 Ultrasonic Sensor

The HC-SR04 is a popular ultrasonic distance sensor that uses sonar to determine the distance between the sensor and an obstacle. It consists of an ultrasonic transmitter and receiver, which work together to measure the time taken for the ultrasonic waves to bounce back from an obstacle after being emitted. By calculating the time of flight of the ultrasonic waves, the sensor can accurately determine the distance of the obstacle from the sensor.

The HC-SR04 is capable of measuring distances in the range of 2cm to 400cm with an accuracy of about 3mm.

In our project, the HC-SR04 ultrasonic sensor is used to detect obstacles in the path of the autonomous vehicle, helping it to navigate safely and avoid collisions.

3.3.3 DHT11 Temperature Sensor

The DHT11 is a basic and cost-effective digital temperature and humidity sensor. It comes with a calibrated digital output that provides accurate temperature and humidity readings. The sensor is capable of measuring temperatures in the range of 0°C to 50°C with an accuracy of ±2°C and humidity in the range of 20 to 90 percentage RH with an accuracy of ±5 percentage.

The DHT11 communicates with the Arduino UNO via a single-wire digital interface, making it easy to integrate into projects.

In our project, the DHT11 temperature sensor is used to monitor the ambient temperature and provide this information to the Arduino UNO.

3.4 Software Requirements

The following are the software requirements to develop our project RAKSHA.

3.4.1 Arduino IDE 1.8.16

The Arduino 1.8.16 IDE is an integrated development environment specifically designed for programming Arduino boards, including the Arduino UNO used in our project. It provides a user-friendly interface that simplifies the process of writing, compiling, and uploading code to the Arduino micro-controller. The IDE is open-source and freely available, making it accessible to a wide range of users, from hobbyists and students to professional developers and engineers. Additionally, it offers a vast library of pre-written code examples and community-contributed libraries, allowing developers to easily implement complex functionalities and interact with a wide range of sensors and actuators, crucial for the successful execution of our project.

In our project, we utilized the Arduino IDE to program the interactions between the Arduino UNO and the HC-SR04 ultrasonic sensor and DHT11 temperature sensor.

Its built-in library offered a range of functions that simplified the process of reading data from the sensors and processing the information. We leveraged these functions to implement the logic for calculating the distance measured by the ultrasonic sensor and obtaining the temperature value from the DHT11 sensor. The IDE's syntax highlighting and auto-completion features ensured that our code was free of syntax errors and logical mistakes before uploading it to the Arduino UNO.

Its ease of use, extensive library, and real-time debugging capabilities significantly contributed to the successful implementation of our project.

3.4.2 Visual Studio Code (VS Code)

Visual Studio Code (VS Code) is a lightweight and versatile source code editor developed by Microsoft. It is widely used by developers due to its simplicity, extensibility, and cross-platform compatibility. VS Code provides an extensive set of features, including intelligent code completion, debugging, Git integration, and support for various programming languages through extensions. Its user-friendly interface and powerful functionalities make it a preferred choice for software development across different domains.

In our project, Visual Studio Code (VS Code) played a central role as the primary code editor, with Python as the chosen programming language. With its powerful support for Python development and a plethora of Python-related extensions, VS Code proved to be an indispensable tool throughout our project. Its user-friendly interface and robust features, such as intelligent code completion, syntax highlighting, and code linting, greatly improved code quality and productivity. The integrated debugging capabilities facilitated swift identification and resolution of any programming issues, ensuring smooth progress during the development phase.

It provided an efficient and streamlined development environment empowering our team to deliver a robust and effective solution.

Chapter 4

Methodology

4.1 Proposed Method

- Read the sensor data from sensor
- Check for probability of accident occurrence using a threshold value.
- If the readings exceeds the threshold value, we determine the severity of accident.
- Alerts the systems nearby about the occurrence of the accident.

4.1.1 Sensor Data Collection

In this module, we collect data from HC SR04 Ultrasonic sensor and DHT11 Temperature sensor (Fig 4.1).This data is later sent through kalman filtering algorithm to cancel out the noise from the data.

Kalman Filtering

The Kalman filter(Fig 4.2) estimates the state of the system at each time step based on two main sources of information:

Process Model: A model that describes how the state of the system evolves over time. In the case of vehicle movement, this could be a simple kinematic model based on speed and acceleration.

Measurement Model: A model that relates the state of the system to the sensor measurements. In the case of an ultrasonic sensor, this could be a direct mapping from the distance to the measured value.

Prediction Step: At each time step, the Kalman filter predicts the state of the system based on the process model. This prediction is associated with uncertainty.

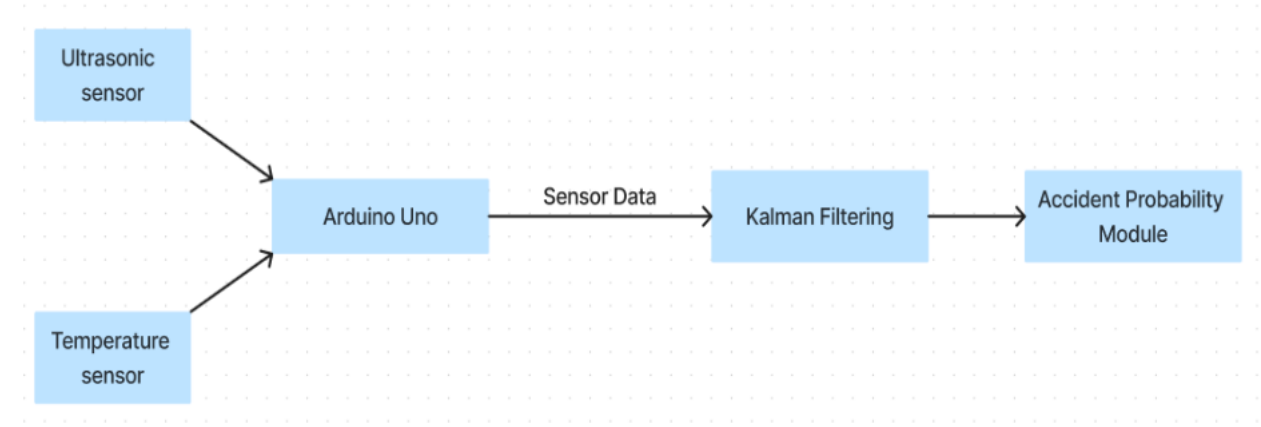

Figure 4.1: Sensor data collection

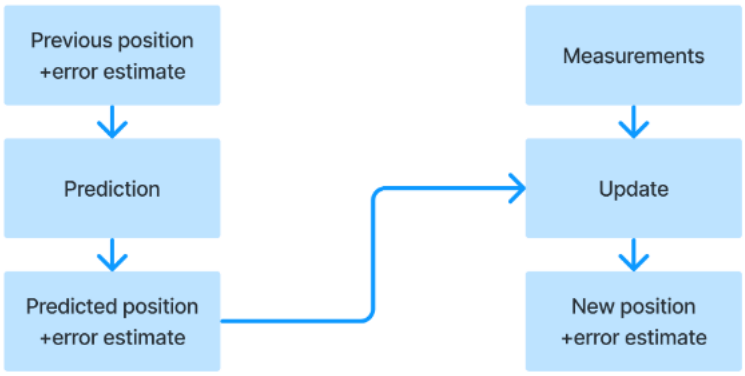

Figure 4.2: Kalman filtering

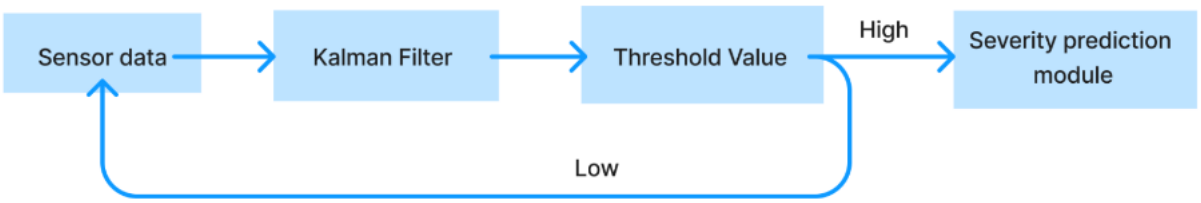

Figure 4.3: Accident probablity module

Measurement Update Step: After receiving a new sensor measurement, the Kalman filter compares the predicted state with the actual sensor measurement. It then updates the state estimate based on the Kalman gain, which balances the confidence in the prediction and the confidence in the measurement. The Kalman filter reduces the estimation error by using the best possible combination of the prediction and measurement.

Filtering Ultrasonic Sensor Data: By applying the Kalman filtering algorithm to the ultrasonic sensor data, you can effectively reduce noise and uncertainties in the measurements. The filtered data will provide a more reliable estimate of the true distance between the vehicle and nearby objects, which is crucial for accident detection and prevention.

4.1.2 Accident Probablity Module

The Accident probablity module(Fig 4.3) takes the filtered data from the Kalman filter and processes it.We keep a threshold value to detect high probablity accidents.If the intervehicular distance is less than or equal to 30 units, the system considers it highly likely that an accident might occur, and the probability is set to the maximum value.If the distance is greater than 30 and less than or equal to 50 units the likelihood of an accident is moderate.If the distance is greater than 75 and less than or equal to 100 units, there is a lower probability of an accident.If the distance is greater than 100 units,the system considers it safe, there is no probability of an accident occurring.

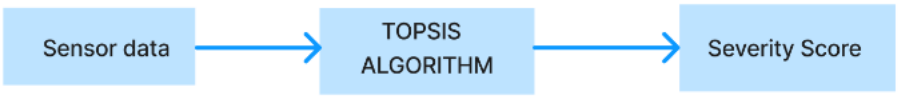

Figure 4.4: Accident severity module

4.1.3 Accident Severity prediction

The accident severity prediction module takes data from real time data set and predict the severity based on the weighted and non-weighted topsis score as shown in fig 4.4.

TOPSIS

TOPSIS stands for Technique for Order of Preference by Similarity to Ideal Solution. It is multicriteria based decision-making method.Given below is the topsis algorithm step by step: Input: Let there be 'm' real time tuples under consideration. Each alternative is evaluated based on 'n' criteria (C1, C2, ..., Cn). Step 1:Normalize the Decision Matrix: The input decision matrix 'X' is normalized to bring all the criteria on the same scale. This step ensures that each criterion contributes equally to the final ranking. The normalization is done by dividing each element of the matrix by the square root of the sum of squares of the corresponding column.

Step 2:Multiply by Weights: After normalization, the normalized matrix is multiplied by the weights vector to get a weighted normalized matrix,in case of weighted topsis. The weights represent the relative importance of each criterion in the decision-making process.

Step 3:Determine Ideal and Anti-Ideal Solutions: The ideal best and ideal worst solutions are identified for each criterion. The ideal best solution contains the maximum values for each column, and the ideal worst solution contains the minimum values.

Step 4:Calculate Euclidean Distances: Euclidean distances are calculated for each alternative with respect to both the ideal best solution and the ideal worst solution. The distance represents the similarity of an alternative to the ideal and anti-ideal solutions for each criterion.

Step 5:Calculate TOPSIS Scores: The TOPSIS score for each alternative is calculated

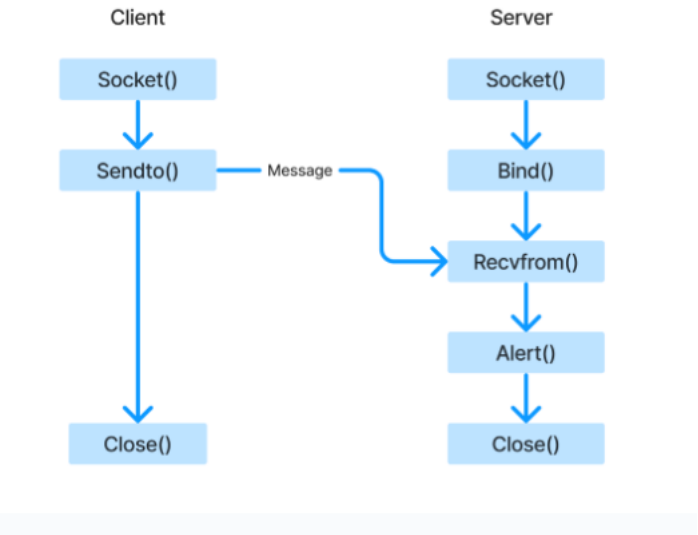

Figure 4.5: Alert System

by dividing the distance to the ideal worst solution by the sum of distances to the ideal best and ideal worst solutions. The TOPSIS score represents how close an alternative is to the ideal solution compared to the anti-ideal solution.

Based on the weighted and non-weighted topsis score, we determine the accident severity. If the severity is high the alert system is activated.

4.1.4 Alert System

On high severity of accidents the alert system(fig 4.5) is activated and message is sent to the nearby cars of the range(eg:100m). The protocol used is UDP(User datagram Protocol). Initially at client, the socket is created . Then message is defined,encoded and send to all the nearby vehicles. At the server side, the socket is created and the IP address and port address is bound to it.On the occurrence of the accident an alert system is activated an the server will receive a message to reroute.

Chapter 5

System Design

Draw usecase diagrams, activity diagrams, sequence diagrams etc. Add a brief explanation for each UML diagram.

5.1 Architecture Diagram

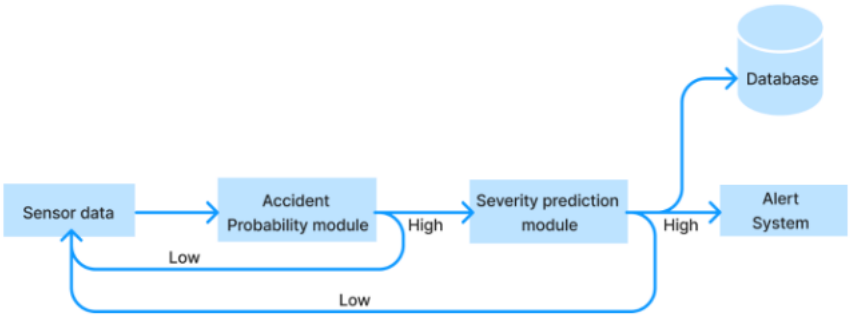

Figure 5.1: Architecture diagram

5.2 Sequence diagram

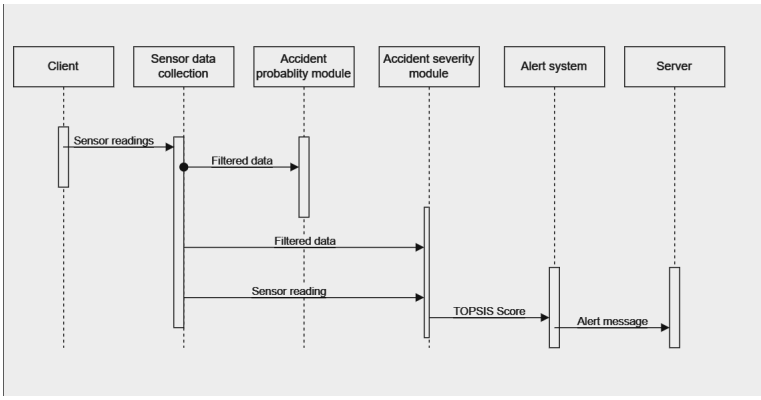

Figure 5.2: Sequence diagram

5.3 Usecase diagram

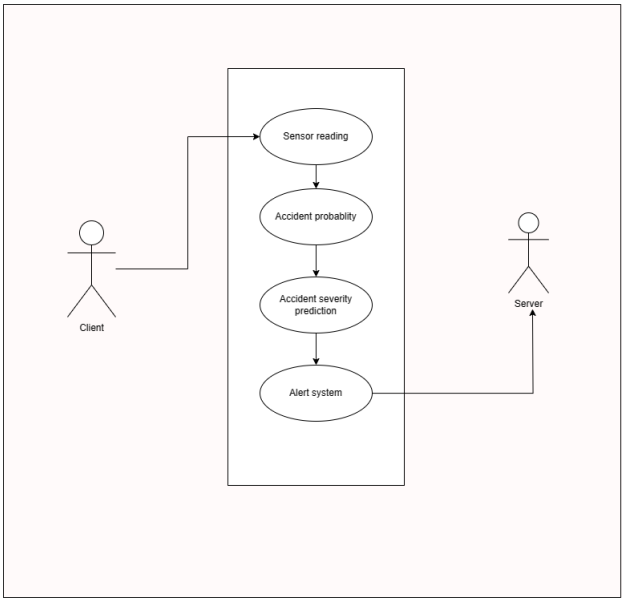

Figure 5.3: Usecase diagram

Chapter 6

System Implementation

6.1 Data Collection

This module establishes communication with the Arduino board and reads data from its sensors (ultrasonic and temperature). The system needs to collect real-time data from the Arduino's sensors to monitor the distance and temperature. The Serial Communication Module handles the communication protocol and data retrieval from the Arduino, allowing the system to continuously update sensor readings.

This module is responsible for establishing communication with the Arduino board and reading data from its sensors, which are the ultrasonic distance sensor and the temperature sensor. The module uses the serial library, which provides tools for working with serial connections in Python. It starts by defining the port and baud rate for the communication with the Arduino (arduino_port and arduino_baudrate). Then, it creates a serial connection with the Arduino using serial.Serial(arduino_port, arduino_baudrate). The time.sleep(2) call allows a 2-second delay to ensure that the Arduino has enough time to establish a connection after the serial port is opened. The arduino.write(b'r') and arduino.write(b't') commands send the characters 'r' and 't' to the Arduino, respectively, indicating a request for ultrasonic and temperature data. The Arduino responds with the requested data, which is then read from the serial port using arduino.readline(). The received data is parsed to extract the ultrasonic distance and temperature, and these values are stored in the variables ultrasonic_distance and temperature, respectively.

6.2 Accident Probability Module

6.2.1 Kalman Filter

In this module a basic Kalman filter is implemented to filter and estimate the state variables, which are the ultrasonic distance and temperature values. The Kalman filter is a recursive algorithm that estimates the state of a system based on noisy sensor measurements and a mathematical model of the system. The Kalman filter parameters are set up in this module:

dt: Time step or sampling interval.

A: State transition matrix, which predicts the next state based on the current state.

C: Measurement matrix, which maps the state to the measurement space.

Q: Process noise covariance matrix, representing the uncertainty in the process model.

R_ultrasonic and R_temperature: Measurement noise covariances, representing the uncertainty in the sensor measurements.

The update_kalman(z_ultrasonic, z_temperature) function takes the ultrasonic distance and temperature as inputs and updates the Kalman filter to obtain filtered values for these variables. The Kalman filter consists of two steps: Prediction and Measurement Update. In the Prediction step, the filter predicts the next state based on the current state using the state transition matrix and updates the covariance matrix accordingly. In the Measurement Update step, the filter incorporates the sensor measurements and corrects the state prediction using the Kalman gain. The filtered ultrasonic distance and temperature are returned as filtered_ultrasonic and filtered_temperature, respectively. The filtered_ultrasonic distance is given to the calculate_accident_probability() function.

6.2.2 Calculate Accident Probability Module

This module defines a function calculate_accident_probability(distance) that calculates the probability of an accident based on the given distance. The function takes the distance as input and uses predefined thresholds to determine the accident probability: If the distance is less than or equal to 30 cm, the probability is set to 100%. If the distance is between 30 cm and 50 cm, the probability is set to 75%. If the distance is between 50 cm and 75 cm, the probability is set to 50%. If the distance is between 75 cm and 100 cm, the probability is set to 25%. If the distance is greater than 100 cm, the probability is set to

0%. The calculated accident probability is returned. If the accident probability is high (greater than 80%), the TOPSIS Algorithm Module is executed to rank alternatives based on sensor data.

6.3 Severity Prediction Module

The system requires a method to evaluate severity on various criteria. This module implements the TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) algorithm, a multi-criteria decision-making method. The topsis(X, weights, impact) function takes three inputs:

X: The decision matrix containing alternative solutions and their performance values for each criterion.

weights: The weight vector representing the relative importance of each criterion.

impact: The impact vector representing the desired direction (positive or negative) of each criterion.

The steps in TOPSIS Algorithm:

- Normalize the Decision Matrix:The decision matrix X(read from the dataset) contains rows representing alternative solutions and columns representing different criteria. In this step, the function normalizes the decision matrix by dividing each element in a column by the Euclidean norm of that column. This normalization ensures that all criteria are treated on the same scale and have equal influence in the subsequent calculations.

- Weighted Normalization:After normalizing the decision matrix, the function multiplies each element in the normalized matrix by the corresponding weight from the weight vector. The weight vector (weights) represents the relative importance of each criterion. Higher weights indicate more significant criteria, and lower weights indicate less importance.

- Determine Ideal and Negative-Ideal Solutions:The impact vector (impact) specifies whether each criterion should be maximized or minimized to achieve the ideal solution. Based on the impact vector, the function identifies the ideal solution (maximum values for each column) and the negative-ideal solution (minimum values for

each column).

- Calculate Euclidean Distances:For each alternative (row) in the weighted normalized matrix, the function calculates the Euclidean distances to both the ideal and negative-ideal solutions. The Euclidean distance is a measure of similarity between the alternative and the ideal or negative-ideal solutions.

- Calculate TOPSIS Scores:The TOPSIS score for each alternative is calculated as the ratio of the distance to the negative-ideal solution over the sum of distances to the ideal and negative-ideal solutions. A higher TOPSIS score indicates a better-ranked alternative, while a lower score indicates a relatively worse-ranked alternative.

In this system if any real time tuple has a TOPSIS score greater than or equal to 0.5, the system sends an accident severity message to nearby vehicles using UDP communication (UDP Communication Module).

6.4 Alert System

The system needs to communicate important alerts, such as high accident severity, to nearby vehicles or relevant parties. The UDP Communication Module enables the script to send messages to a nearby servers for further dissemination or action.

6.4.1 Client

The system defines a function udp() that implements UDP (User Datagram Protocol) communication to send a message to a server and receive a response. The function first prepares the message to be sent to the server. The message to be sent is "Accident ahead please redirect," which is stored in the variable msgFromClient. The message is converted to bytes using str.encode(msgFromClient) so that it can be sent as data in the UDP packet. The server address and port to which the message will be sent are defined in the serverAddressPort variable. In this case, the server address is set to "0.0.0.0", which means the server will listen on all available network interfaces on the host machine. The port number is set to 8080. A UDP socket is created and is configured to use IPv4 (AF_INET) and UDP (SOCK_DGRAM) for communication. The message is sent to the server using the created UDP socket. The UDPClientSocket.sendto(bytesToSend,

serverAddressPort) call sends the encoded message bytes to the specified server address and port.

6.4.2 Server

The server first gets its own IP address using socket.gethostname(). It would then receive the message at its socket and then decode it. It then creates a pop-up window using Tkinter to display the alert message. A voiceover for the message is also made .The server keeps on running in the loop until manually terminated, so that it can receive messages from any client.

6.5 Firebase Configuration Module

This module is responsible for configuring the connection to the Firebase Realtime Database using the pyrebase library. The firebase_config dictionary contains the necessary credentials and information to connect to the Firebase database (e.g., API key, project ID, database URL). The pyrebase.initialize_app(firebase_config) function initializes the connection to the Firebase database using the provided configuration. The database reference is obtained by firebase.database() to interact with the Firebase Realtime Database. When an accident occurs, the processed data, including sensor readings, filtered values, accident probability, and TOPSIS rankings, is pushed to the Firebase database.

Chapter 7

Testing

7.1 Unit Testing

Unit tests focus on verifying the functionality of individual components or units of code in isolation.

Unit Test 1: Sensor Data Processing

Description: Verify the correct reading and processing of sensors (e.g., ultrasonic sensor ,temperature sensor) in the sensor data collection.

Test Case: Verify the handling of invalid or out-of-range sensor values.

Unit Test 2: Severity Prediction Model

Description: Validate the accuracy of the severity prediction model in the system.

Test Case: Check the model's behavior with different input combinations.

Unit Test 3: Alert System

Description: Test the alert system's functionality to deliver real-time notifications to nearby drivers and emergency services.

Test Cases: Ensure that alerts are sent promptly and accurately based on severity predictions.

7.2 Integration Testing

Integration Test 1: Data Flow and Processing

Description: Validate the end-to-end data flow and processing between the accident prediction and severity prediction modules.

Test Cases:

- Verify that the processed data from the accident prediction module is correctly used as input for the severity prediction module.

- Test the communication and data exchange between the two modules.

  Integration Test 2: Alert System Integration

  Description: Test the integration of the alert system with the accident and severity prediction modules.

  Test Case: Ensure that the alert system receives accurate predictions from severity prediction module.

7.3 System Testing

The entire system is tested as a whole to ensure that all integrated components work together as expected

End-to-End Functionality:

Description: Test the end-to-end functionality of the entire system, including accident prediction, severity prediction, and the alert system.

Test Cases:

- Simulate real-time sensor data input and validate the system's ability to predict potential accidents accurately.
- Verify that the severity prediction accurately assesses the severity of detected accidents based on sensor data.
- Ensure that the alert system sends timely and accurate notifications to nearby drivers and emergency services based on severity predictions.

7.4 Acceptance Testing

Acceptance tests focus on ensuring that the entire system meets the specified requirements and user expectations.

Acceptance Test 1: Accident Prediction Accuracy

Description: Validate the overall accuracy and effectiveness of the accident prediction module.

Test Cases: Evaluate the system's ability to correctly predict potential accident scenarios.

Acceptance Test 2: Emergency message transmission

Description: Simulate emergency response scenarios to test the system's ability to measure accident severity.

Test Cases: Evaluate the timeliness and accuracy of the alert system's notifications to nearby vehicles.

Chapter 8

Results

The integration of road accident prediction, severity prediction and alert system can have the followinng results:

- Road accident prediction involves using various data sources, such as historical accident data, real-time sensor data to estimate the likelihood of an accident occurring in a specific location or along a particular route.

- Severity prediction focuses on assessing the potential impact and seriousness of an accident when it occurs. It involves analyzing various factors like vehicle speed, collision force, angle of collision, engine temperature conditions to determine the severity of the accident.

- The alert system is designed to notify relevant parties about potential accidents or hazardous situations. It can be integrated into vehicles, road infrastructure, and connected systems. When an accident is predicted or detected, the alert system can automatically send warnings to nearby drivers.

Figure 8.1: Alert system interface

Chapter 9

Risks and Challenges

1. Sensory accuracy:The data collected by ultrasonic and temperature sensors must be precise to ensure accurate predictions. Any deviations in sensor readings can result in false alarms or missed alerts, which defeats the purpose of the system. To mitigate this risk, we can use redundant sensors and calibration techniques to ensure accuracy.

2. Data processing:The amount of data generated by the sensors can be overwhelming, and processing it in real-time requires significant computing power. Furthermore, the data must be processed quickly to provide timely alerts

3. Sensor damage: Due to overuse the sensors can be damaged overtime.

4. Network connectivity: In the case of areas with poor network connectivity, the transmission and reception of alert messages will adversely affect the system.

Chapter 10

Conclusion

Sensor fusion plays a pivotal role in the development and advancement of autonomous vehicles, and its importance is expected to grow significantly with increasing research in this field. As autonomous vehicles become a reality, sensor fusion emerges as a key technology that enables these vehicles to navigate safely and efficiently in complex and dynamic environments. This advanced technique involves the integration and analysis of data from multiple sensors to make accurate and reliable decisions, making it essential for predicting and preventing accidents in autonomous vehicles.

In our project the integration of accident prediction, severity prediction, and an alert system using the TOPSIS algorithm creates a robust and proactive road safety ecosystem. This system not only helps prevent accidents by identifying potential hazards butminimizes accident severity. By leveraging advanced technologies and data-driven approaches, this comprehensive system takes significant strides towards making roads safer, reducing accidents, and saving lives. Its potential to mitigate road accidents and improve overall traffic management makes it a valuable tool for authorities and policymakers seeking to enhance road safety and create safer road environments for all road users.

10.0.1 Future Scope

- Real-time Traffic Management Integration: Integrating the system with real-time traffic management systems can optimize traffic flow in accident-prone areas. The project can expand to include dynamic traffic rerouting to prevent traffic congestion and reduce the probability of accidents.

- Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) Communication: Implementing V2V and V2I communication can enable vehicles to exchange real-time

data and warnings with each other and the surrounding infrastructure. This enhanced communication can further improve the accuracy and responsiveness of accident predictions and alerts.

- Autonomous Vehicle Integration: As autonomous vehicles become more prevalent, the project can be adapted to integrate seamlessly with autonomous driving systems. The system can provide additional safety measures for autonomous vehicles, contributing to the wider adoption of autonomous transportation.

- Multimodal Transportation Safety: The project can be extended to include other modes of transportation, such as bicycles and pedestrians. By incorporating data from various transportation modes, the system can address safety concerns for all road users.

References

[1] R.M. Zulqarnain, M. Saeed, N. Ahmad, F. Dayan, B. Ahmad "Application of TOPSIS Method for Decision Making". In International Journal of Scientific Research in Mathematical and Statistical Sciences Volume-7, Issue-2, pp.76-81, April (2020)

[2] O. Nedjmedine, M.Tahar, "Analysis of road accident factors using Decision Tree Algorithm: a case of study Algeria", 2022 5th International Symposium on Informatics and its Applications (ISIA), pp.1-6, 2022.

[3] S. Malik, H. El Sayed, M. A. Khan and M. J. Khan, "Road Accident Severity Prediction — A Comparative Analysis of Machine Learning Algorithms," 2021 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT), Dubai, United Arab Emirates, 2021, pp. 69-74, doi: 10.1109/GCAIoT53516.2021.9693055.

[4] .J. Z. Sasiadek and P. Hartana, "Sensor data fusion using Kalman filter," Proceedings of the Third International Conference on Information Fusion, Paris, France, 2000, pp. WED5/19-WED5/25 vol.2, doi: 10.1109/IFIC.2000.859866.

[5] irumalla, Mansi. (2021). Ultrasonic Distance Detector Using Arduino. SSRN Electronic Journal. 10.2139/ssrn.3918137.

[6] U. Khalil, A. Nasir, S. M. Khan, T. Javid, S. A. Raza and A. Siddiqui, "Automatic Road Accident Detection Using Ultrasonic Sensor," 2018 IEEE 21st International Multi-Topic Conference (INMIC), Karachi, Pakistan, 2018, pp. 206-212, doi: 10.1109/INMIC.2018.8595541.

[7] Q. Li, R. Li, K. Ji and W. Dai, "Kalman Filter and Its Application," 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), Tianjin, China, 2015, pp. 74-77, doi: 10.1109/ICINIS.2015.35.

Appendix A: Base Paper

International Journal of Scientific Research in Mathematical and Statistical Sciences
Volume-7, Issue-2, pp.76-81, April (2020)

Research Paper

E-ISSN: 2348-4519

Application of TOPSIS Method for Decision Making

R. M. Zulqarnain [1*], M. Saeed [2], N. Ahmad [2], F. Dayan [2], B. Ahmad [4]

[1]School of Mathematics, Northwest University, Xian 710127, China.
[2]School of Science, Department of Mathematics, University of Management and Technology Lahore, Punjab, Pakistan.
[3]Department of Mathematics and Statistics, The University of Lahore, Pakistan.

**Corresponding author: ranazulqarnain7777@gmail.com, Tel: +86-13028563376*

Available online at: www.isroset.org

Received: 16/Feb/2019, Accepted: 10/Apr/2020, Online: 30/Apr/2020

***Abstract*—**In this paper, we discuss the order preference by similarity ideal solution (TOPSIS) method with basic concepts and determine the TOPSIS algorithm. Secondly, we construct a graphical model for the TOPSIS method by using the TOPSIS algorithm. Finally, we use the developed method for decision making in our daily life. In this work, we use the TOPSIS method for the selection of a car by using hypothetical data and examined that the civic is the best automotive car according to given parameters.

Keywords— Multiple Criteria Decision Making (MCDM), TOPSIS, Positive Ideal Solution (PIS), Negative Ideal Solution (NIS)

I. INTRODUCTION

Decision Making is the best procedure to choose a superlative alternative from all feasible alternatives. Almost in all other issues, the overall number of criteria because decision making the general alternatives is pervasive. Such criteria normally contrast one another so there might be no way out satisfying all criteria simultaneously. To deal with such problems the decision-makers want to solve the MCDM problem. There are different methods to solve MCDM problems. One of them presented by Hwang and Yoon in [1] is known as a TOPSIS to solve the MCDM problem with many alternatives. The core concept of this technique is that the chosen alternative should have the smallest geometrical distance from the PIS and the largest geometrical distance from NIS [2].

Nowadays this technique used in different fields of life such as energy [3–7] medicine [2,8–10] engineering and manufacturing systems [11–16] safety and environmental fields [17–22] chemical engineering [5,23,24] and water resources studies [5,19,23,25]. Chen & Hwang extend the idea of the TOPSIS method and presented a new model for TOPSIS[26]. Zulqarnain et al. developed the graphical model of the TOPSIS method and used for the selection of a medical clinic for the diagnosis of disease [27]. Moreover, to solve uncertain data Chen extended the TOPSIS for Group Decision Making in the fuzzy atmosphere [28] and used the newly proposed method for decision making. The importance weights of multi-criteria and alternative rating w.r.t. these criteria were treated as linguistic variables, evaluated by a group of decision-makers. To facilitate the decision making in a fuzzy environment many researchers extended the TOPSIS technique reported in the literature [3,4,18,19,25,29–35,6,8,11–15,17]. The author's developed the idea of generalized interval-valued fuzzy soft matrices (IVFSM) in [36]. Zulqarnain et al [37] used the trapezoidal fuzzy numbers by Sanchez's approach for disease identification. The usage of interval numbers is too a significant enhancement of [38–40]. The extension of TOPSIS under fuzzy data has been used to express the prospect of achievement for pancreatic transplantation [8]. A decision-making method on IVFSM introduced in [41] and the authors provided the application of IVFSM [42] and comparative study with a fuzzy soft matrix in [43].

Mahmoodzadeh et al. developed a technique for the project assortment by combining fuzzy AHP and TOPSIS methods and used the upgraded technique to calculate the weights of each criterion at first and then the TOPSIS algorithm was engaged for ranking the projects to be selected [44]. The authors faced some difficulties to determine the accurate value of the elements of the decision matrix, such as their values were considered as intervals, to overcome these difficulties they extended the TOPSIS method with interval data in [38]. Several approaches have been established for MCDM problems, in [45] the authors provided a proper guideline of how and which method could be used for MCDM problems according to the situation.

The authors extended the TOPSIS to Atanassov intuitionistic fuzzy set and proposed the algorithm of extended TOPSIS for multi-attribute group decision-making problem in [46]. The idea of multiple attribute intuitionistic fuzzy group decision-making algorithm was introduced in [46]. Many researchers worked on the TOPSIS method and used in medical diagnosis and for decision making in different fields of life reported in the literature [47–50].

The following paper is organized as follows. In section II, we study and discuss some basic concepts of the TOPSIS method and present the classical TOPSIS algorithm. We proposed the graphical model for the TOPSIS method by using the TOPSIS algorithm. In section III, we use the TOPSIS method for decision-making and choose the best car by using hypothetical data. lastly, the conclusion is made in section IV.

THE EFFICIENCY OF TOPSIS [51]

First, it is important to discuss the efficiency of the TOPSIS method. The calculation time of the TOPSIS method rises slightly when the number of criteria increases to 16. However, the point to be noted is that the time does not exceed 10 seconds see [51].

The TOPSIS is examined by changing the number of criteria and users, and it was determined that the efficiency is high when the number of users is under 320 and the number of criteria is not more than 16.

TOPSIS Method

Hwang and Yoon [1] developed a technique to resolve MCDM known as the TOPSIS method. To support the shortest Euclidean distance, they proposed the PIS and NIS and each criterion needs to be maximized or minimized. They claimed that the TOPSIS method helps rank alternatives closeness which based on optimum ideal solution and obtained the maximum level from available alternatives. The best alternative has rank one and the worst alternative approaches rank zero. For every alternative, there is an intermediate ranking between the best answer extremes. An identical set of choice criteria permits correct weighting of relative disease and therefore the optimum disease is alarming which needs attention. Here are presented the steps for the TOPSIS technique. TOPSIS views an MCDM problem with m-alternatives as a geometric system with m points in the n-dimensional space [52]. The core concept of this technique is that the chosen alternative should have the smallest geometrical distance from the PIS and the largest geometrical distance from the NIS [53]. To apply TOPSIS [54], a common assumption is that criteria should be either monotonically increasing or decreasing so that PIS and NIS can be easily identified.

II. CLASSICAL TOPSIS ALGORITHM

Step 1: Establishment of DM

Construct the decision matrix as follows

$$DM = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_p \end{array} \begin{array}{c} \begin{array}{cccc} R_1 & R_2 & \dots & R_q \end{array} \\ \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1q} \\ c_{21} & c_{22} & \dots & c_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ c_{p1} & c_{p2} & \dots & c_{pq} \end{bmatrix} \end{array}$$

Where l is the alternative index ($l = 1, 2, ..., q$); n is the number of potential sites and m is the criteria index ($m = 1, 2, ..., p$).

The elements $R_1, R_2, ..., R_q$ of the DM define the criteria while $A_1, A_2, \ldots, A_p$ defining the alternatives.

Step 2: Calculation of the Normalized Decision Matrix (NDM)

To represent the relative performance of the alternatives the NDM constructed as follows.

$$NDM = L_{lm} = \frac{c_{lm}}{\sqrt{\sum_{l=1}^{q} c_{lm}^2}}$$

Step 3: Determination of the Weighted Normalized Decision Matrix (WNDM)

By multiplying every element of each column of NDM got a weighted decision matrix.

$$V = V_{lm} = W_m \times L_{lm}$$

Step 4: Identification of the PIS and NIS

The PIS (I^+) and the NIS (I^-) are defined concerning the weighted decision matrix as follows

$$\text{NIS} = I^- = \{V_1^-, V_2^-, \ \dots, V_q^-\}, \text{ where:}$$

$$V_m^- = \{(\text{mini}(V_{lm}) \text{ if } m \in J); (\text{maxi } V_{lm} \text{ if } m \in J')\}$$

Where J' is associated with the non-beneficial attributes and

J is associated with beneficial attributes.

Step 5: Separation Distance from PIS and NIS of each alternative

$$S_l^+ = \sqrt{\sum_{m=1}^{p} (V_m^+ - V_{lm})^2} \ ; l = 1, 2, \ \dots, \ q$$

$$S_l^- = \sqrt{\sum_{m=1}^{p} (V_m^- - V_{lm})^2} \ ; l = 1, 2, \ \dots, \ q$$

Where, l = Alternative index,

m = Criteria index.

Step 6: Relative Closeness to the Ideal Solution.

The relative closeness of the ideal solution is computed as

$$C_l = \frac{S_l^-}{(S_l^+ + S_l^-)} \ , 0 \le C_l \le 1$$

Step 7: Ranking of Preference Order

The ranking is done based on the values of C_l the higher value of the relative closeness has a high rank and hence the better performance of the alternative. Rank the preference in descending order to compare the better performances of alternative.

Figure 1. Graphical Model for TOPSIS

III. APPLICATION OF TOPSIS METHOD

A person wants to choose the car for his family. For the selection of best automotive car, he hires a team of three experts (decision-makers) such as (l = 3) represented by D = {D_1, D_2, D_3}. Firstly, the experts select four best automotive cars showroom in the city as follows A = {Civic, Corolla, Swift, Hyundai} and decide the four evaluation criteria represented by C = {Style, Safety, Fuel Efficiency, Expanses} for selection of one of the best automotive car out of four cars.

$$C = \begin{cases} \text{benefit criteria} \\ \text{Cost criteria} \end{cases}$$

$$J_1 = \begin{cases} X_1: & \text{Style} \\ X_2: & \text{Safety} \\ X_3: & \text{Fuel Efficiency} \end{cases} \quad J_2 = \{X_4: \text{Expenses}$$

SOLUTION BY TOPSIS

TOPSIS method will be illustrated with the help of a car selection problem. Here the set of alternatives is A = {Civic, Corolla, Swift, Hyundai} and the set of evaluation criteria is C = {Style (St), Safety (Sa), Fuel Efficiency (FE), Expanses (Exp)} are given.

Step 1: Construction of a Decision Matrix

The decision matrix is given in the following table.

Table 1: Decision Matrix D = $[x_{ij}]_{m\times n}$

Cars	St	Sa	FE	Exp
Civic	7	9	9	8
Corolla	8	7	8	7
Swift	9	6	8	9
Hyundai	6	7	8	6

Step 2: Normalization

By using the following formula, we get

Table 2: Calculating $\sqrt{\sum_{i=1}^{m} x_{ij}^2}$

Cars	St	Sa	FE	Exp
Civic	49	81	81	64
Corolla	64	49	64	49
Swift	81	36	64	81
Hyundai	36	49	64	36
$\sum_{i=1}^{m} x_{ij}^2$	230	215	273	230
$\sqrt{\sum_{i=1}^{m} x_{ij}^2}$	15.17	14.66	16.52	15.17

To normalize the decision matrix dividing each entry by $\sqrt{\sum_{i=1}^{m} x_{ij}^2}$

Table 3: Normalized Decision Matrix R = $[r_{ij}]_{m\times n}$

	St	Sa	FE	Exp
Civic	0.46	0.61	0.54	0.53
Corolla	0.53	0.48	0.48	0.46
Swift	0.59	0.41	0.48	0.59
Hyundai	0.40	0.48	0.48	0.40

Step 3: Computation of the weight matrix

The weights assigned by the experts (decision makers) to the criteria are given by the matrix

W = [w_1 (Security) = 0.1, w_2 (Environment) = 0.4, w_3 (Qualified staff) = 0.3, w_4 (Expenses) = 0.2]$^{\text{Transpose}}$

Step 4: Computation of WNDM Ŕ = $[r'_{ij}]_{m\times n}$

To get WNDM, multiplying each column of NDM in Table 3 by weights w_j, of weight vector computed in the step 3.

Table 4: Weighted Normalized Decision Matrix Ŕ = $[r'_{ij}']_{m\times n}$

Weights w_j	0.1	0.4	0.3	0.2
A	St	Sa	FE	Exp
Civic	0.046	0.244	0.162	0.106
Corolla	0.053	0.192	0.144	0.092
Swift	0.059	0.164	0.144	0.118
Hyundai	0.040	0.192	0.144	0.080

Step 5: The calculation of PIS and NIS

To find the PIS A*

Table 5: Positive Ideal Solution

	Benefit Criteria ∈ J^+			Cost Criteria∈J^-
A	St	Sa	FE	Exp
Civic	0.046	0.244=v_2^*	0.162=v_3^*	0.106
Corolla	0.053	0.192	0.144	0.092
Swift	0.059=v_1^*	0.164	0.144	0.118=v_4^*
Hyundai	0.040	0.192	0.144	0.080

Therefore A* = {0.059, 0.244, 0.162, 0.080}

To find the NIS A^-

Table 6: Negative Ideal Solution

	Benefit Criteria $\in J^+$			Cost Criteria$\in J^-$
A	St	Sa	FE	Exp
Civic	0.046	0.244	0.162	0.106
Corolla	0.053	0.192	0.144	0.092
Swift	0.059	0.164 = v_2'	0.144=v_3'	0.118
Hyundai	0.040=v_1'	0.192	0.144	0.080=v_4^*

Therefore A^- = {0.040,0.164,0.144,0.118}

Step 6: Determine the separation measures for each alternative
Calculating separation from PIS A*

Table 7: Calculation of S_i^*

	St	Sa	FE	Exp	$\sum_{j=1}^{n}(v_j^* - v_{ij})^2$	S_i^*
Civic	$(0.046-0.059)^2$	$(0.244-0.244)^2$	$(0)^2$	$(0.026)^2$	0.000845	0.029
Corolla	$(0.053-0.059)^2$	$(0.192-0.244)^2$	$(-0.018)^2$	$(0.012)^2$	0.003208	0.057
Swift	$(0.053-0.059)^2$	$(0.164-0.244)^2$	$(-0.018)^2$	$(0.038)^2$	0.008186	0.090
Hyundai	$(0.053-0.059)^2$	$(0.192-0.044)^2$	$(-0.018)^2$	$(0)^2$	0.003389	0.058

Calculating separation from NIS A^-

Table 8: Calculation of S_i'

	St	Sa	FE	Exp	$\sum_{j=1}^{n}(v_j^* - v_{ij})^2$	S_i'
Civic	$(0.046-0.040)^2$	$(0.244-0.164)^2$	$(0.018)^2$	$(-0.012)^2$	0.006904	0.083
Corolla	$(0.053-0.040)^2$	$(0.192-0.164)^2$	$(0)^2$	$(-0.026)^2$	0.001629	0.040
Swift	$(0.053-0.040)^2$	$(0.164-0.164)^2$	$(0)^2$	$(0)^2$	0.000361	0.019
Hyundai	$(0.053-0.040)^2$	$(0.192-0.164)^2$	$(0)^2$	$(-0.038)^2$	0.002228	0.047

Step 7: Computation of RCC to the ideal solution C_i^*
RCC to the ideal solution C_i^* is computed as follows

$C_1^* = \frac{S_i'}{S_i' + S_i^*} = \frac{0.083}{0.112} = 0.74$ (Best)

Similarly, we can get

$C_2^* = 0.41$
$C_3^* = 0.17$
$C_4^* = 0.45$

Hence "Civic" is the best automotive car with the above evaluation criteria.

IV. CONCLUSION

In this paper, we discuss the TOPSIS method in detail and constructed a graphical model for the TOPSIS method. We used the TOPSIS method for the selection of the best automotive car by using hypothetical data and examined that Civic is the best car according to the above selected parameters.

REFERENCES

[1] C. Hawng, K. Yoon, "Multiple Attribute Decision Making: Methods and Applications," A State of the Art Survey 1. *Berlin Heidelberg Springer*, **1981**.

[2] Y. Bi, D. Lai, H. Yan, "Synthetic evaluation of the effect of health promotion : Impact of a UNICEF project in 40 poor western counties of China," *R. Soc. Public Heal*, vol. **124**, pp. **376–391**, **2010**.

[3] I. Chamodrakas, D. Martakos, "A utility-based fuzzy TOPSIS method for energy-efficient network selection in heterogeneous wireless networks," *Appl. Soft Comput. J*, vol. **11**, no. **4**, pp.

3734–3743, **2011**.

[4] R. M. Zulqarnain, M. Saeed, B. Ali, N. Ahmad, L. Ali, S. Abdal, "Application of Interval Valued Fuzzy Soft Max-Min Decision Making Method," *International Journal of Mathematical Research,* vol. **9**, no. **1,** pp. **11-19, 2020.**

[5] M. Behzadian, S. K. Otaghsara, M. Yazdani, J. Ignatius, "A state-of the-art survey of TOPSIS applications," *Expert Syst. Appl*, Vol. **39**, no. **17**, pp. **13051–13069**, **2012**.

[6] F. Dayan, M. Zulqarnain, H. Naseer, "A Ranking Method for Students of Different Socio Economic Backgrounds Based on Generalized Fuzzy Soft Sets," *Int. J. Sci. Res*, Vol. **6**, no. **9**, pp. **691–694**, **2017**.

[7] L. Yang, J. Deuse, "Multiple-attribute decision-making approach for an energy-efficient facility layout design," *Publ. by Elsevier B.V*, vol. **3**, pp. **149–154**, **2012**.

[8] G. La, G. Aiello, C. Rastellini, R. Micale, L. Cicalese, "Multi-Criteria Decision Making support system for pancreatic islet transplantation," *Expert Syst. Appl*, Vol. **38**, no. **4**, pp. **3091–3097**, **2011**.

[9] T. Chen, "A signed-distance-based approach to importance assessment and multi-criteria group decision analysis based on interval type-2 fuzzy set," *Knowl Inf Syst*, Vol. **35**, pp. **193–231**, **2013**.

[10] R. Kuo, Y. Wu, T. Hsu, "Integration of fuzzy set theory and TOPSIS into HFMEA to improve outpatient service for elderly patients in Taiwan," *J. Chinese Med. Assoc*, Vol. **75**, pp. **341–348**, **2012**.

[11] K. Im, H. Cho, "A systematic approach for developing a new business model using morphological analysis and integrated fuzzy approach," *Expert Syst. Appl*, Vol. **40**, no. **11**, pp. **4463–4477**, **2013**.

[12] K. K. damghani, S. Sadi-nezhad, M. Tavana, "Solving multi-period project selection problems with fuzzy goal programming based on TOPSIS and a fuzzy preference relation," *Inf. Sci. (Ny)*, Vol. **1**, **2013**.

[13] M. Nakhaeinejad, N. Nahavandi, "An interactive algorithm for multi-objective flow shop scheduling with fuzzy processing time through resolution method and TOPSIS," *Int J Adv Manuf Technol*, no. **July**, **2012**.

[14] H. Hosseini, A. S. Milani, "An improvement of quantitative strategic planning matrix using multiple criteria decision making and fuzzy numbers," *Appl. Soft Comput. J*, Vol. **12**, no. **8**, pp. **2246–2253**, **2012**.

[15] A. Taleizadeh, S. Taghi, A. Niaki, "A hybrid method of Pareto TOPSIS and genetic algorithm to optimize multi- product multi-constraint inventory control systems with random fuzzy replenishments A Hybrid Method of Pareto , TOPSIS and Genetic Algorithm to Optimize Multi-Product Multi-Constr," *Math. Comput. Model*, **2009**.

[16] L. I. Tong, C. H. Wang, H. C. Chen, "Optimization of multiple responses using principal component analysis and," *Int J Adv Manuf Techno*, Vol. **27**, pp. **407–414**, **2005**.

[17] R. A. Krohling, V. C. Campanharo, "Fuzzy TOPSIS for group decision making: A case study for accidents with oil spill in the sea," *Expert Syst. Appl*, Vol. **38**, no. **4**, pp. **4190–4197**, **2011**.

[18] X. Wang, H. K. Chan, "A hierarchical fuzzy TOPSIS approach to assess improvement areas when implementing green supply chain initiatives," *Int. J. Prod. Res*, Vol. **51**, no. **10**, pp. **3117–3130**, **2013**.

[19] Y. Kim, E. S. Chung, S. M. Jun, S. U. Kim, "Prioritizing the best sites for treated wastewater instream use in an urban watershed using fuzzy TOPSIS," *Resour. Conserv. Recycl*, Vol. **73**, pp. **23–32**, **2013**.

[20] P. Li, H. Qian, J. Wu, J. Chen, "Sensitivity analysis of TOPSIS method in water quality assessment: I. Sensitivity to the parameter weights," *Environ. Monit. Assess*, Vol. **185**, no. **3**, pp. **2453–2461**, **2013**.

[21] A. Awasthi, S. S. Chauhan, S. K. Goyal, "A multi-criteria decision making approach for location planning for urban distribution centers under uncertainty," *Math. Comput. Model*, Vol. **53**, no. **1–2**, pp. **98–109**, **2011**.

[22] M. J. Ostad-Ahmad-Ghorabi, M. Attari, "Advancing environmental evaluation in cement industry in Iran," *J. Clean. Prod*, Vol. **41**, pp. **23–30**, **2013**.

[23] P. Li, J. Wu, H. Qian, "Groundwater quality assessment based on rough sets attribute reduction and TOPSIS method in a semi-arid area, China," *Env. Monit Assess*, Vol. **184**, no. **Sep**, pp. **4841–4854**, **2012**.

[24] Y. Sun, Z. Liang, C. Shan, H. Viernstein, F. Unger, "Comprehensive evaluation of natural antioxidants and antioxidant potentials in Ziziphus jujuba Mill . var . spinosa (Bunge) Hu ex H . F . Chou fruits based on geographical origin by TOPSIS method," *Food Chem*, Vol. **124**, no. **4**, pp. **1612–1619**, **2011**.

[25] D. Yong, "Plant location selection based on fuzzy TOPSIS," *Int J Adv Manuf Technol*, Vol. **28**, no. **Aug**, pp. **839–844**, **2006**.

[26] S. J. J. C, C. L. Hwang, "Fuzzy Multiple Attribute Decision Making Methods," *Springer-Verlag Berlin Heidelberg*, Chapter 5, **1992**.

[27] R. M. Zulqarnain, S. Abdal, B. Ali, L. Ali, F. Dayan, M. I. Ahamad, Z. Zafar, "Selection of Medical Clinic for Disease Diagnosis by Using TOPSIS Method," *Int. J. Pharm. Sci. Rev. Res*, Vol. **61,** no. **1**, pp. **22-27, 2020.**

[28] C. T. Chen, "Extensions of the TOPSIS for group decision-making under fuzzy environment," *Fuzzy Sets Syst*, Vol. **114**, no. **1**, pp. **1–9**, **2000**.

[29] M. Anisseh, F. Piri, M. Reza, "Fuzzy extension of TOPSIS model for group decision," *Artif Intell Rev*, Vol. **38**, pp. **325–338**, **2012**.

[30] L. Dymova, P. Sevastjanov, A. Tikhonenko, "An approach to generalization of fuzzy TOPSIS method," *Inf. Sci. (Ny)*, Vol. **238**, no. **February**, pp. **149–162**, **2013**.

[31] S. Rouhani, M. Ghazanfari, M. Jafari, "Evaluation model of business intelligence for enterprise systems using fuzzy TOPSIS," *Expert Syst. Appl*, Vol. **39**, no. **3**, pp. **3764–3771**, **2012**.

[32] M. Zulqarnain, F. Dayan, "Choose Best Criteria for Decision Making Via Fuzzy Topsis Method," *Math. Comput. Sci*, Vol. **2**, no. **6**, p. **113**, **2017**.

[33] M. Zulqarnain, F. Dayan, "Selection Of Best Alternative For An Automotive Company By Intuitionistic Fuzzy TOPSIS Method," *Int. J. Sci. Technol. Res*, Vol. **6**, no. **10**, pp. **126–132**, **2017**.

[34] T. Kaya, C. Kahraman, "Multicriteria decision making in energy planning using a modified fuzzy TOPSIS methodology," *Expert Syst. Appl*, Vol. **38**, no. **6**, pp. **6577–6585**, **2011**.

[35] F. Cavallaro, "Fuzzy TOPSIS approach for assessing thermal-energy storage in concentrated solar power (CSP) systems," *Appl. Energy*, Vol. **87**, no. **2**, pp. **496–503**, **2010**.

[36] F. Dayan, M. Zulqarnain, "On Generalized Interval Valued Fuzzy Soft Matrices," *Am. J. Math. Comput. Model*, Vol. **3**, no. **1**, pp. **1–9, 2018.**

[37] R. M. Zulqarnain, X. L. Xin, B. Ali, S. Abdal, A. Maalik, L. Ali, M. I. Ahamad, Z. Zafar, "Disease identification using trapezoidal fuzzy numbers by Sanchez's approach," *Int. J. Pharm. Sci. Rev. Res*, Vol. **61,** no **1**, pp. **13-18, 2020.**

[38] G. R. Jahanshahloo, F. H. Lotfi, A. R. Davoodi, "Extension of TOPSIS for decision-making problems with interval data : Interval efficiency," *Math. Comput. Model*, Vol. **49**, no. **5–6,** pp. **1137–1142**, **2009**.

[39] L. Dymova, P. Sevastjanov, A. Tikhonenko, "A direct interval extension of TOPSIS method," *Expert Syst. Appl*, no. **March**, **2013**.

[40] Z. Yue, "An extended TOPSIS for determining weights of decision makers with interval numbers," *Knowledge-Based Syst*, Vol. **24**, no. **1**, pp. **146–153**, **2011**.

[41] M. Zulqarnain, M. Saeed, "A New Decision Making Method on Interval Valued Fuzzy Soft Matrix (IVFSM)," Br. J. Math. Comput. Sci, Vol. **20**, no. **5**, pp. **1–17, 2017**.

[42] M. Zulqarnain, M. Saeed, "An Application of Interval Valued

Fuzzy Soft Matrix in Decision Making Problem," *Sci. Int*, Vol. **28**, no. **3**, pp. **2261–2264**, **2016**.

[43] M. Zulqarnain, M. Saeed, M. F. Tbassum, "Comparison Between Fuzzy Soft Matrix (FSM) and Interval Valued Fuzzy Soft Matrix (IVFSM) in Decision Making," *Sci. Int*, Vol. **28**, no. **5**, pp. **4277–4283**, **2016**.

[44] M. S. Mahmoodzadeh, J. Shahrabi, M. Pariazar, "Project selection by using a fuzzy AHP and topsis technique," *World Acad. Sci. Eng. Technol*, Vol. **1**, no. **6**, pp. **270–275**, **2007**.

[45] M. Velasquez, P. T. Hester, "An Analysis of Multi-Criteria Decision Making Methods," *Int. J. Oper. Res*, Vol. **10**, no. **2**, pp. **56–66**, **2013**.

[46] D. Li, "Extension of the TOPSIS for multi-Attribute group Decision making under Atanassov IFS environments," *Int. J. Fuzzy Syst. Appl*, Vol. **1**, no. **4**, pp. **47–61**, **2011**.

[47] R. M. Zulqarnain, S. Abdal, A. Maalik, B. Ali, Z. Zafar, M. I. Ahamad, S. Younas, A. Mariam, F. Dayan., "Application of TOPSIS Method in Decision Making Via Soft Set," *Biomed. J. Sci. Tech. Res*, Vol. **24**, no. **3**, pp. **18208–18215**, **2020**.

[48] M. Sevkli, S. Zaim, A. Turkyılmaz, M. Satır, "An Application of Fuzzy Topsis Method for Supplier Selection," *IEEE Int. Conf. Fuzzy Syst*, no. **July**, **2010.**

[49] M. Zulqarnain, F. Dayan, M. Saeed, "TOPSIS Analysis for The Prediction of Diabetes Based on General Characteristics of Humans," *Int. J. Pharm. Sci. Res*, Vol. **9**, no. **7**, pp. **2932-2939, 2018**.

[50] S. Eraslan, "A Decision Making Method via TOPSIS on Soft Sets," J. New Results Sci, Vol. **8**, pp. **57–71**, **2015**.

[51] D. Y. Cheng, K. M. Chao, C. C. Lo, C. F. Tsai, A user centric service-oriented modeling approach, Vol. **14**, no. **4**. **2011**.

[52] M. R. Fathi, H. Zarei Matin, M. Karimi Zarchi, S. Azizollahi, "The Application of Fuzzy TOPSIS Approach to Personnel Selection for Padir Company, Iran," *J. Manag. Res*., Vol. **3**, no. **2**, pp. **1–14, 2011**.

[53] E. Triantaphyllou, B. Shu, S. N. Sanchez, T. Ray, "Multi-Criteria Decision Making : An Operations Research Approach," *Encycl. Electr. Electron. Eng*, Vol. **15**, no. February, pp. **175–186**, **1998**.

[54] W. Sa, "The mean error estimation of TOPSIS method using a fuzzy reference models," *J. Theor. Appl. Comput. Sci*, Vol. **7**, no. **3**, pp. **40–50**, **2013**.

Appendix B: Sample Code

```
from json import JSONDecoder

import serial
import time
import pyrebase
import numpy as np

# Establish serial communication with Arduino
arduino_port = 'COM6'  # Update with the correct serial port of your Arduino
arduino_baudrate = 9600
arduino = serial.Serial(arduino_port, arduino_baudrate)
time.sleep(2)  # Allow time for Arduino to establish connection

# Firebase configuration
firebase_config = {
    'apiKey': "AIzaSyAJ2-ljlBntiJIrVFoPFKgzeUfu8ZOuxYA",
    'authDomain': "raksha-648b6.firebaseapp.com",
    'databaseURL': "https://raksha-648b6-default-rtdb.firebaseio.com",
    'projectId': "raksha-648b6",
    'storageBucket': "raksha-648b6.appspot.com",
    'messagingSenderId': "737863321421",
    'appId': "1:737863321421:web:d7808918d77d0778839904"
}

# Initialize Firebase
firebase = pyrebase.initialize_app(firebase_config)
database = firebase.database()

# Kalman Filter parameters
dt = 0.1  # Time step
A = np.array([[1, dt], [0, 1]])  # State transition matrix
C = np.array([[1, 0], [0, 1]])  # Measurement matrix
Q = np.array([[1e-4, 0], [0, 1e-4]])  # Process noise covariance
```

```
R_ultrasonic = 0.1  # Measurement noise covariance for ultrasonic sensor
R_temperature = 0.5  # Measurement noise covariance for temperature sensor

# Initialize state variables
x = np.array([[0], [0]])  # Initial state (ultrasonic and temperature)
P = np.eye(2)  # Initial state covariance matrix

# Function to update the Kalman filter and return the filtered data
def update_kalman(z_ultrasonic, z_temperature):
    global x, P  # Declare x and P as global variables

    # Prediction step
    x_pred = np.dot(A, x)
    P_pred = np.dot(np.dot(A, P), A.T) + Q

    # Measurement update step
    z = np.array([[z_ultrasonic], [z_temperature]])
    y = z - np.dot(C, x_pred)
    S = np.dot(np.dot(C, P_pred), C.T) + np.array([[R_ultrasonic, 0], [0, R_temperature]])
    K = np.dot(np.dot(P_pred, C.T), np.linalg.inv(S))

    x = x_pred + np.dot(K, y)
    P = np.dot(np.eye(2) - np.dot(K, C), P_pred)

    # Return filtered data (ultrasonic and temperature)
    filtered_ultrasonic = x[0][0]
    filtered_temperature = x[1][0]

    return filtered_ultrasonic, filtered_temperature

def calculate_accident_probability(distance):
    if distance <= 30:
        return 100.0  # Set a high probability when the distance is less than the threshold
```

```
67      elif distance > 30 and distance <= 50 :
68          return 75.0  # Set a probability of 75% when
    the distance is between 75% and 100% of the
   threshold
69      elif distance > 50 and distance <= 75:
70          return 50.0  # Set a probability of 50% when
    the distance is between 50% and 75% of the
   threshold
71      elif distance > 75 and distance <= 100:
72          return 25.0  # Set a probability of 25% when
    the distance is between 25% and 50% of the
   threshold
73      elif distance > 100:
74          return 0.0  # Set 0% probability when the
   distance is above 25% of the threshold
75
76
77
78
79 # Perform TOPSIS algorithm on the data
80 import numpy as np
81 import csv
82
83 def topsis(X, weights, impact):
84      # Normalize the decision matrix
85      normalized_matrix = X / np.sqrt(np.sum(X**2,
   axis=0))
86
87      # Multiply normalized matrix by weights to get
   weighted normalized matrix
88      weighted_matrix = normalized_matrix * weights
89
90      # Determine the ideal and negative-ideal
   solutions
91      ideal_best = np.max(weighted_matrix, axis=0)
92      ideal_worst = np.min(weighted_matrix, axis=0)
93      # Calculate the Euclidean distances to the ideal
    and negative-ideal solutions
94      dist_best = np.sqrt(np.sum((weighted_matrix -
   ideal_best)**2, axis=1))
95      dist_worst = np.sqrt(np.sum((weighted_matrix -
```

```
 95 ideal_worst)**2, axis=1))
 96
 97     # Calculate the topsis score
 98     topsis_scores = dist_worst / (dist_best +
    dist_worst)
 99     if np.any(topsis_scores >= 0.5):
100         pass
101
102     # Rank the alternatives based on relative
    closeness
103     rankings = np.argsort(topsis_scores)
104     return  rankings,topsis_scores
105
106 def topsisnowt(X):
107     # Normalize the decision matrix
108     normalized_matrix = X / np.sqrt(np.sum(X**2,
    axis=0))
109
110     # Determine the ideal and negative-ideal
    solutions
111     ideal_best = np.max(normalized_matrix, axis=0)
112     ideal_worst = np.min(normalized_matrix, axis=0)
113     # Calculate the Euclidean distances to the ideal
     and negative-ideal solutions
114     dist_best = np.sqrt(np.sum((normalized_matrix -
    ideal_best)**2, axis=1))
115     dist_worst = np.sqrt(np.sum((normalized_matrix-
    ideal_worst)**2, axis=1))
116
117     # Calculate the topsis score
118     topsis_scores = dist_worst / (dist_best +
    dist_worst)
119     if np.any(topsis_scores >= 0.5):
120         pass
121
122     # Rank the alternatives based on relative
    closeness
123     rankings = np.argsort(topsis_scores)
124     return  rankings,topsis_scores
125 import socket
126 def udp():
```

```
127     import socket
128 
129     msgFromClient = "Accident ahead please redirect"
130 
131     bytesToSend = str.encode(msgFromClient)
132 
133     serverAddressPort = ("192.168.102.205", 8080)
134 
135     bufferSize = 1024
136 
137     # Create a UDP socket at client side
138 
139     UDPClientSocket = socket.socket(family=socket.
    AF_INET, type=socket.SOCK_DGRAM)
140 
141     # Send to server using created UDP socket
142 
143     UDPClientSocket.sendto(bytesToSend,
    serverAddressPort)
144 
145     msgFromServer = UDPClientSocket.recvfrom(
    bufferSize)
146 
147     msg = "Message from Server {}".format(
    msgFromServer[0])
148 
149     print(msg)
150 
151 
152 # Main loop
153 class CustomEncoder:
154     pass
155 
156 
157 try:
158     # Read ultrasonic sensor distance from Arduino
159     arduino.write(b'r')  # Send command to Arduino
    to request distance data
160     ultrasonic_data = arduino.readline().decode().
    strip()
161     ultrasonic_distance = float(ultrasonic_data.
```

```
split(":")[1].split("cm")[0].strip())
    print("Distance:", ultrasonic_distance)

    # Read temperature sensor value from Arduino
    arduino.write(b't')  # Send command to Arduino to request temperature data
    temperature_data = arduino.readline().decode().strip()
    temperature = float(temperature_data.split(":")[1].split("C")[0].strip())
    print("Temperature:", temperature)

    # Update the Kalman filter
    x, P = update_kalman(ultrasonic_distance, temperature)



    # Calculate accident probability based on the measured distance
    accident_probability = calculate_accident_probability(x)
    print("Accident Probability:", accident_probability)

    if accident_probability >80:
        # Perform TOPSIS algorithm
        #csv_file= r"C:\Users\eljoy\Downloads\sensor_raw.csv"

        weights = np.array([0.75, 0.75, 0.75, 0.25, 0.25, 0.25, 0.90])  # Adjust the weights based on your preference
        # Read decision matrix from CSV file
        csv_file1 = r"C:\Users\eljoy\Downloads\sensor_raw.csv"
        rows = []
        with open(csv_file1, 'r') as file:
            csv_reader = csv.reader(file)
            for row in csv_reader:
                rows.append(row)
```

```
191         impact=np.array([1, 1, 1, 1, 1, 1, 1])
192         # Convert the data to a NumPy array
193         X = np.array(rows,dtype=float)
194         ranks, topsis_scores = topsis(X, weights ,
    impact)
195         rank1,topsisscore1=topsisnowt(X)
196         print("TOPSIS Scores:", topsis_scores)
197         print("TOPSIS SCORE WITH NO WEIGHT:",
    topsisscore1)
198     else:
199         exit(0)
200     if np.any(topsis_scores >= 0.5 ):
201         if(np.any(topsisscore1)>=0.5):
202             print("High Accident severity.....
    message redirecting to nearby vehicles")
203             udp()
204         # Push data to Firebase database
205             data = {
206                 "Distance": ultrasonic_distance,
207                 "Temperature": temperature,
208                 "AccidentProbability":
    accident_probability,
209                 "Ranks": ranks.tolist(),
210                 "TOPSIS Scores": topsis_scores#.
    tolist()
211             }
212             print(data)
213             print(type(data))
214             JSONDecoder().decode(CustomEncoder().
    encode(data))
215             database.child("sensor_data").set(data)
216
217 except KeyboardInterrupt:
218     arduino.close()
```

```
 1 import socket
 2 import threading
 3 import winsound
 4 import pyttsx3
 5 import tkinter as tk
 6
 7 # Initialize the text-to-speech engine
 8 engine = pyttsx3.init()
 9
10 # Set the speech rate (lower value means slower
   speech)
11 speech_rate = 150
12 engine.setProperty('rate', speech_rate)
13
14 # Server IP address and port
15 SERVER_IP = socket.gethostname()  # Replace with your
    server IP address
16 SERVER_PORT = 8080  # Replace with an unused port
   number
17
18 # Create a UDP socket
19 server_socket = socket.socket(socket.AF_INET, socket.
   SOCK_DGRAM)
20 s = (SERVER_IP, SERVER_PORT)
21
22 # Bind the socket to the server IP address and port
23 server_socket.bind(s)
24
25
26 def show_alert():
27     alert_text = "Accident ahead. Please Reroute."
28     # Create a pop-up window to display the alert
   message
29     alert_window = tk.Toplevel(root)  # Use Toplevel
   () instead of Tk() for additional pop-up windows
30     alert_window.title("Alert")
31
32     # Increase the size of the pop-up window
33     alert_window.geometry("400x200")  # Width: 400
   pixels, Height: 200 pixels
34
```

```
35      alert_label = tk.Label(alert_window, text="
   Accident ahead. Please Reroute.", font=("Impact", 18
   ))
36      alert_label.pack(pady=50)
37
38      # Start the Tkinter event loop to display the pop
   -up window
39      alert_window.after(1000, lambda: say_alert(
   alert_window, alert_text))  # Pass alert_window
   instance
40
41
42  def say_alert(alert_window, alert_text):
43      # Call the function to say "Accident ahead.
   Please reroute."
44      engine.say(alert_text)
45      engine.runAndWait()
46
47      # Destroy the pop-up window after voice alert and
    2-second delay
48      alert_window.after(1500, alert_window.destroy)
49
50
51  def receive_messages():
52      while True:
53          # Receive data from the client
54          try:
55              data, address = server_socket.recvfrom(
   1024)  # Buffer size is 1024 bytes
56              data = data.decode()
57
58              # Play the beep sound 5 times (500
   milliseconds each)
59              for _ in range(5):
60                  winsound.Beep(500, 500)  # 500 Hz
   frequency, 500 ms duration
61
62              # Call the function to show the pop-up
   window
63              show_alert()
64          except KeyboardInterrupt:
```

```
            print('Server stopped.')
            break


# Start receiving messages in a separate thread
receive_thread = threading.Thread(target=
receive_messages)
receive_thread.start()

# Create the main Tkinter window
root = tk.Tk()
root.withdraw()  # Hide the main window

print('UDP server listening on {}:{}'.format(
SERVER_IP, SERVER_PORT))

# The server will keep running until manually
terminated
root.mainloop()
```

```
# include<dht.h>
dht
DHT;
int
trig = 7;
int
echo = 6;
int
timeInMicro;
int
distInCm;
# define DHT11_PIN 10

float
temperature;

void
setup()
{
    Serial.begin(9600);
pinMode(7, OUTPUT);
pinMode(6, INPUT);
}

void
loop()
{
    digitalWrite(trig, LOW);
delayMicroseconds(2);
digitalWrite(trig, HIGH);
delayMicroseconds(10);
digitalWrite(trig, LOW);

timeInMicro = pulseIn(echo, HIGH);

distInCm = timeInMicro / 29 / 2;

Serial.print("Distance: ");
Serial.println(distInCm);

// Read
```

```
42 temperature
43 sensor
44 value
45 int
46 chk = DHT.read11(DHT11_PIN);
47 temperature = DHT.temperature;
48 Serial.print("Temperature: ");
49 Serial.print(temperature);
50 Serial.println(" C");
51
52 delay(2000);
53 }
```