SECURITY PATROL ROUTES AND CHECKPOINTS OPTIMIZATION WITH MATLAB TO MINIMIZE BLIND SPOTS AND ENHANCE THE SAFETY

YASSVINDHRAN A/L MARIMUTHU

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

SECURITY PATROL ROUTES AND CHECKPOINTS OPTIMIZATION WITH MATLAB TO MINIMIZE BLIND SPOTS AND ENHANCE THE SAFETY

YASSVINDHRAN A/L MARIMUTHU

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Electronics Engineering Technology with Honours

Faculty of Electronics and Computer Technology and Engineering
Universiti Teknikal Malaysia Melaka

2025

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek	:	SECURITY PATROL ROUTES AND CHECKPOINTS OPTIMIZATION WITH MATLAB TO MINIMIZE BLIND SPOTS AND ENHANCE THE SAFETY
Sesi Pengajian	:	2024/2025

Saya YASSVINDHRAN A/L MARIMUTHU mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

☐	**SULIT***	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD***	(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan.
/	**TIDAK TERHAD**	

Disahkan oleh:

(TANDATANGAN PENULIS) (COP DAN TANDATANGAN PENYELIA)

Alamat Tetap:

VIGNESWARA RAO GANNAPATHY
Pensyarah Kanan
Jabatan Teknologi Kejuruteraan
Fakulti Teknologi Dan Kejuruteraan Elektronik Dan Komputer
Universiti Teknikal Malaysia Melaka

Tarikh : 7/2/2025 Tarikh : 7/2/2025

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I declare that this project report entitled "SECURITY PATROL ROUTES AND CHECKPOINTS OPTIMIZATION WITH MATLAB TO MINIMIZE BLIND SPOTS AND ENHANCE THE SAFETY" is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	
Student Name	:	YASSVINDHRAN A/L MARIMUTHU
Date	:	7/2/2025

APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Electrical Engineering Technology with Honours.

Signature	:	
Supervisor Name	:	VIGNESWARA RAO A/L GANNAPATHY
Date	:	7/2/2025
Signature	:	
Co-Supervisor Name (if any)	:	
Date	:	

DEDICATION

To my beloved mother, V PUSPA RANI, and father, MARIMUTHU.

ABSTRACT

In this project, we are using MATLAB to plan security patrol routes and checkpoints better, determining the optimal routes for these patrols to go and the location for their checks is a challenging but crucial task. By data analysis and optimization techniques, we ensure that patrols efficiently cover critical areas and adjust to evolving security requirements. Most of the time, the security checkpoints are usually not well-planned, requiring guards to travel the same paths repeatedly, time, and energy. Due to poorly designed patrol routes and checkpoints, guards can miss crucial locations like ATMs or hidden corners, leaving these places open to potential threats. The guards may not even be aware that they are missing these areas. The efficiency of security measures may be prevented by guards unknowingly ignoring these areas of risk because they are unaware of the security threats. As a result of that, this degrades overall safety because there is no systematic method and there is not a clear and organized approach. Thus, it is important to develop a systematic approach using MATLAB to optimize patrol routes, ensuring comprehensive coverage while minimizing blind spots. This project aims to develop a program using MATLAB takes input parameters such as the layout of the area to be patrolled and generates optimized patrol routes to enhance surveillance coverage while reducing vulnerable areas. By leveraging computational geometry, graph theory, and optimization techniques, the program aims to improve the efficiency and effectiveness of security patrols, thereby enhancing overall security measures in various environments. By using MATLAB to plan smarter patrol routes, guards will save time and energy because they will not have to backtrack or miss important spots anymore. This means they will be able to cover all the critical areas, like ATMs and hidden corners, making those places safer from potential threats and enhance safety.

ABSTRAK

Dalam projek ini, kami menggunakan MATLAB untuk merancang laluan ronda keselamatan dan kawalan sempadan dengan lebih baik, menentukan laluan optimum untuk ronda ini dan lokasi pemeriksaan mereka adalah tugas yang mencabar tetapi penting. Dengan analisis data dan teknik optimasi, kami memastikan bahawa ronda-ronda secara efisien meliputi kawasan-kawasan penting dan menyesuaikan dengan keperluan keselamatan yang berkembang. Kebanyakan masa, kawalan keselamatan biasanya tidak dirancang dengan baik, memerlukan pengawal untuk melakukan perjalanan di laluan yang sama berulang kali, masa, dan tenaga. Kerana laluan ronda dan kawalan yang kurang berkualiti, pengawal boleh terlepas lokasi penting seperti ATM atau sudut-sudut tersembunyi, meninggalkan tempat-tempat ini terdedah kepada ancaman yang berpotensi. Pengawal mungkin tidak sedar bahawa mereka melepaskan kawasan-kawasan ini. Kecekapan langkah-langkah keselamatan mungkin dihalang oleh pengawal secara tidak sengaja mengabaikan kawasan-kawasan risiko ini kerana mereka tidak menyedari ancaman keselamatan. Akibatnya, keselamatan keseluruhan terjejas kerana tidak ada kaedah sistematik dan tidak ada pendekatan yang jelas dan teratur. Oleh itu, penting untuk membangunkan pendekatan sistematik menggunakan MATLAB untuk mengoptimumkan laluan ronda, memastikan liputan yang komprehensif sambil meminimumkan titik buta. Projek ini bertujuan untuk membangunkan program menggunakan MATLAB yang mengambil parameter input seperti susunan kawasan yang akan dipatrol dan menghasilkan laluan ronda yang dioptimumkan untuk meningkatkan liputan pengawasan sambil mengurangkan kawasan yang rentan. Dengan menggunakan geometri komputasi, teori graf, dan teknik optimasi, program ini bertujuan untuk meningkatkan kecekapan dan keberkesanan ronda keselamatan, dengan itu meningkatkan langkah-langkah keselamatan keseluruhan dalam pelbagai persekitaran. Dengan menggunakan MATLAB untuk merancang laluan ronda yang lebih bijak, pengawal akan menjimatkan masa dan tenaga kerana mereka tidak perlu kembali atau melepaskan tempat-tempat penting lagi. Ini bermakna mereka akan dapat meliputi semua kawasan penting, seperti ATM dan sudut-sudut tersembunyi, menjadikan tempat-tempat tersebut lebih selamat daripada ancaman yang berpotensi dan meningkatkan keselamatan.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, VIGNESWARA RAO A/L GANNAPATHY for his precious guidance, words of wisdom and patient throughout this project.

I am also indebted to Universiti Teknikal Malaysia Melaka (UTeM) for the financial support through enables which enables me to accomplish the project. Not forgetting my fellow colleague, friends for the willingness of sharing his thoughts and ideas regarding the project.

My highest appreciation goes to my parents, parents in-law, and family members for their love and prayer during the period of my study. An honourable mention also goes to seniors for all the motivation and understanding.

Finally, I would like to thank all the staffs at the UTeM, fellow colleagues and classmates, the faculty members, as well as other individuals who are not listed here for being co-operative and helpful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

δ	-	Voltage angle
	-	
	-	
	-	
	-	
	-	
	-	
	-	

LIST OF ABBREVIATIONS

V	-	Voltage
	-	
	-	
	-	
	-	
	-	
	-	
	-	

LIST OF APPENDICES

APPENDIX	TITLE	PAGE

CHAPTER 1

INTRODUCTION

1.1 Background

In today's security landscape, ensuring the safety of individuals and assets is a top priority across various environments, including campuses, industrial facilities, urban areas, and transportation hubs. Traditional security measures often rely heavily on human patrols to monitor and secure these areas. However, the effectiveness of these patrols can be compromised by inefficient route planning and poorly designed checkpoints, leading to significant blind spots in surveillance coverage.

One of the primary challenges with traditional patrol route planning methods is their built difficulties and potential to human mistake. Guards may find themselves repeatedly getting around the same paths, overlooking critical areas or failing to adapt to evolving security needs. This inefficiency not only wastes valuable resources such as time, fuel, and energy but also leaves vulnerabilities unaddressed, potentially exposing areas to security threats.

The existence of blind spots in patrol routes poses a significant risk, as these areas may be exploited by perpetrators to carry out illegal activities undetected. Blind spots can result from various factors, including inadequate surveillance coverage, poor lighting, or inefficient patrol routes. Addressing these blind spots requires a systematic approach to route optimization that ensures comprehensive coverage while minimizing the risk of overlooking critical areas.

This is where my project comes into play. By leveraging MATLAB's powerful capabilities in data analysis, optimization, and simulation, my project aims to develop a systematic approach to optimize security patrol routes and checkpoints. The objective is clear to enhance security measures by improving the efficiency and effectiveness of patrol routes, thereby reducing blind spots and enhancing overall safety.

Through this project, we seek to develop a program that takes into account key input parameters, such as the layout of the area to be patrolled and generates optimized patrol routes. By analyzing data and employing optimization techniques, the program will ensure that patrols efficiently cover critical areas while adapting to changing security requirements. By minimizing the risk of overlooking crucial locations such as ATMs or hidden corners, this project aims to enhance safety by systematically addressing blind spots in surveillance coverage.

Overall, this project is significant in its potential to transform security patrol planning and execution. By optimizing patrol routes and checkpoints, it can help security personnel make better use of their resources and respond more effectively to security threats, ultimately contributing to a safer environment for individuals and communities alike.

1.2 Project Relation to Current Issues

In today's world, security concerns are at the forefront of societal challenges. Whether in bustling campuses area, urban areas, commercial districts, or residential neighborhoods, the need for strong security protocols is crucial. However, traditional methods of patrolling and surveillance often fall short in efficiently covering all areas of potential risk, leaving blind spots that can be exploited by evil parties. With the rise of various security threats, ranging from theft and vandalism to more severe incidents like terrorism, there is an urgent need to optimize security patrol routes and checkpoint locations.

Moreover, in an era where resources are increasingly limited, it is important to maximize the efficiency of security operations. This includes minimizing fuel consumption, reducing time spent on patrols, and streamlining the deployment of security personnel. By leveraging computational techniques and optimization algorithms within MATLAB, this project directly addresses these resource optimization challenges, offering a systematic approach to enhancing security while minimizing resource wastage.

Furthermore, the project aligns with greater technology advancements and trends towards making use of data analytics. As cities and organizations worldwide embrace smart technologies and data analytics to improve efficiency and safety, the optimization of security patrol routes and checkpoint locations represents a critical application of such technologies. By utilizing the power of MATLAB for this purpose, the project exemplifies the project showing how advanced computer methods can be used in real-world security operations.

Generally, the project's relevance lies not only in its immediate impact on enhancing security measures but also in its alignment with greater trends towards technological innovation and efficiency optimization. By addressing current security challenges through the lens of computational methods, it contributes to the ongoing efforts to create safer and more resilient communities in an ever-evolving security landscape.

1.3 Problem Statement

Inefficient security patrol routes and poorly located checkpoints pose significant challenges in ensuring comprehensive surveillance coverage, leading to potential blind spots and compromised safety in various environments. Current approaches often lack systematic optimization, resulting in wasted resources, missed critical areas, and increased potential to security threats. Guards may unknowingly ignore certain areas due to poorly designed patrol routes, causing the risk of security breaches. Consequently, there is a pressing need to develop a systematic and efficient method for optimizing security patrol routes and checkpoint locations to enhance safety and surveillance coverage while minimizing blind spots. This project aims to address these challenges by leveraging MATLAB to analyze area layouts, optimize patrol routes, and dynamically adjust checkpoints, ultimately improving the efficiency and effectiveness of security patrols in various environments.

1.4 Project Objective

The primary objective of this project "Security Patrol Routes and Checkpoints Optimization with MATLAB to Minimize Blind Spots and Enhance Safety" is:

a) Collect and analyze current data on patrol routes, checkpoints, and security incidents, presenting findings in graphical form for insight into existing patrol patterns and potential blind spots at UTeM.

b) Use optimization techniques to adjust patrol routes dynamically based on security needs and environmental changes, while conducting comprehensive testing to ensure efficient coverage and reduce blind spots.

1.5 Scope of Project

The scope of the project titled "Security Patrol Routes and Checkpoints Optimization with MATLAB to Minimize Blind Spots and enhance the safety" includes the following key areas:

a) **Data Collection and Analysis:**

- Gather data on the layout of the area to be patrolled, including maps, blueprints, or in the digital app.
- Analise existing patrol routes and checkpoints, if any, to identify inefficiencies and blind spots.

b) **Algorithm Development:**

- Develop algorithms within MATLAB to optimize patrol routes and checkpoint locations.

- Implement computational geometry techniques to determine the most efficient paths for patrols to cover the entire area while minimizing overlap and blind spots.

c) **Optimization Techniques:**

- Incorporate optimization techniques to adjust patrol routes dynamically based on changing security requirements or environmental factors.
- Integrate real-time data feeds or sensors to adapt patrol routes in response to detected security threats or anomalies.

d) **Documentation and Deployment:**

- Document the project thoroughly, including algorithms and methodologies

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Literature review, being the foundation of this report, demands the inclusion of its most crucial points for comprehensive insight and relevance. For my Bachelor's project, I'm diving into a bunch of books and articles related to my topic. This is called a literature review. Basically, I'm checking out what other people have said about my topic already. By doing this, I can figure out what's already known and what still needs to be explored. It's like building a map of what's out there in the world of ideas. This helps me see where my project fits in and what new things I can bring to the table. So, my literature review is like laying the groundwork for my project by seeing what's been done before and figuring out what I can add.

2.2 Enhancing Security Patrol Routes and Checkpoints Optimization with MATLAB

The optimization of security patrol routes and checkpoints using MATLAB encompasses a variety of methodologies rooted in computational geometry, graph theory, and advanced optimization techniques. Initially, the area of interest is mapped out, with critical locations and potential threats identified as nodes in a graph. MATLAB's robust suite of functions and toolboxes, such as the Optimization Toolbox and the Mapping Toolbox, facilitate the creation of these spatial models. The use of algorithms like the shortest path algorithm, Dijkstra's algorithm, and the Traveling Salesman Problem (TSP) solver enables the determination of the most efficient routes. These algorithms ensure that each node is visited in the least amount of time and distance, reducing redundancy and enhancing coverage. Additionally, constraints such as patrol frequency, guard availability, and specific security needs are incorporated into the optimization process to generate practical and actionable patrol schedules.

Moreover, MATLAB's ability to handle large datasets and perform real-time analysis is instrumental in adapting to dynamic security environments. Advanced data analysis techniques, including machine learning algorithms, are employed to assess historical incident data and predict potential security threats. This predictive capability allows for the adjustment of patrol routes in response to evolving security requirements. Furthermore, MATLAB's visualization tools provide a clear representation of the optimized routes, enabling security managers to easily interpret and implement the suggested patrol paths. By continuously analyzing and refining these routes based on real-time data, MATLAB ensures that the patrol strategies remain effective and responsive to new threats, thereby enhancing overall security management.

2.3 Patrol Routes and Checkpoints Optimization with MATLAB:

Patrol routes and checkpoints are crucial for keeping places safe, but if they're not planned well, important areas might get missed. This project uses MATLAB, a powerful tool for solving complex problems, to make patrol routes smarter and decide where checkpoints should go. By analyzing data and using smart algorithms, we aim to make sure security guards cover all the important spots efficiently. This helps improve safety in different places by making sure no areas are left unprotected.

2.3.1 Importance of Patrol route Optimization

Patrol route optimization is crucial for enhancing the efficiency and effectiveness of security operations. Optimized patrol routes ensure comprehensive surveillance coverage, reducing the risk of leaving critical areas unchecked and preventing potential security breaches. Strategic route planning maximizes resource utilization, ensuring that manpower, time, and fuel are used efficiently, which in turn reduces operational costs and increases the overall productivity of security personnel. By analyzing data on historical incidents and threat levels, routes can be designed to prioritize high-risk areas, improving the likelihood of preventing or quickly responding to security threats.

Additionally, the integration of advanced technologies and data analysis tools, such as MATLAB, enables dynamic and adaptive patrol strategies. These tools allow for real-time adjustments to patrol routes as security threats evolve, ensuring that security measures remain effective and relevant. This adaptability is particularly important in today's rapidly changing security landscape, where static patrol routes may quickly become ineffective. Overall, patrol route optimization is a key component of modern security management, offering significant benefits in resource efficiency, threat mitigation, and the ability to adapt to changing security needs, ultimately enhancing the protection of the areas being safeguarded.

2.3.2 General Methods

The application of MATLAB in optimizing security patrol routes and checkpoints involves several key methodologies that are grounded in computational geometry, graph theory, and optimization techniques. This section explores the general methods employed in using MATLAB for this purpose, highlighting the steps and algorithms commonly utilized in the process.

I. Computational Geometry:

Computational geometry plays a crucial role in understanding the spatial layout of the area to be patrolled. This involves mapping out the physical environment, identifying critical locations, and defining the boundaries of the patrol area. Using

MATLAB, one can create detailed geometric models of the patrol area, which serve as the foundation for route optimization

Key techniques in computational geometry include:

Voronoi Diagrams: These are used to partition the patrol area into regions based on proximity to predefined points, which can represent checkpoints. This helps in ensuring that each region is adequately covered.

Delaunay Triangulation: This technique is used to create a network of interconnected points, which can represent potential paths for patrol routes. It ensures that the network is efficient and covers the area comprehensively.

II. Graph Theory:

Graph theory provides the framework for modeling the patrol area as a network of nodes (checkpoints) and edges (paths). This allows for the application of various algorithms to determine the optimal routes.

III. Commonly used graph theory algorithms include:

Shortest Path Algorithms: Algorithms such as Dijkstra's or A* are employed to find the shortest paths between checkpoints, ensuring minimal travel distance and time.

Eulerian and Hamiltonian Paths: These are used to create routes that either cover all edges (paths) or all nodes (checkpoints) without repetition, ensuring comprehensive coverage without redundancy.

2.3.3 Optimization Techniques for Security Patrol Routes:

Numerous optimization techniques have been applied to optimize security patrol routes and checkpoints, aiming to minimize blind spots and enhance safety. One common approach is the use of metaheuristic algorithms such as genetic algorithms (GAs), particle swarm optimization (PSO), and simulated annealing (SA). For instance, Sun et al. (2019) employed a GA to optimize routing design and fleet allocation for freeway service patrols, demonstrating superior performance over simulated annealing methods. Similarly, Shi and Huang (2022) proposed an improved genetic and ant colony hybrid algorithm for optimizing patrol routes of intelligent vehicles, achieving significant reductions in the number of turns and iterations compared to traditional approaches.

• Genetic Algorithms (GA) with MATLAB

MATLAB provides a robust platform for implementing Genetic Algorithms through its Global Optimization Toolbox. The 'ga' function in MATLAB allows users to define optimization problems with various constraints and objectives. Researchers can input the patrol area, set constraints like maximum patrol time, and define fitness functions that minimize distance traveled or maximize coverage. Studies such as Rathore et al.'s work on vehicle routing leverage MATLAB's GA capabilities to solve complex routing problems efficiently. The toolbox also supports custom mutation and crossover functions, which can be tailored to specific patrol scenarios.

• Multi-Agent Systems (MAS) with MATLAB

MATLAB's Simulink and Stateflow environments are well-suited for modeling and simulating Multi-Agent Systems. The AnyLogic software mentioned in the literature can integrate with MATLAB for complex simulations. Researchers like Yanyun Fu et al. have used MAS to create dynamic and adaptive patrol strategies. MATLAB's capabilities in handling real-time data and its extensive libraries for communication protocols enable the implementation of MAS where agents can dynamically update their routes based on new information. The ability to simulate different agent behaviors and interactions helps in fine-tuning the coordination and efficiency of patrol units.

• Particle Swarm Optimization (PSO) with MATLAB

MATLAB’s Global Optimization Toolbox also supports Particle Swarm Optimization through the particleswarm function. This function is particularly useful for optimizing multi-objective problems encountered in patrol route planning. Researchers such as Mario Peñacoba et al. utilize MATLAB to simulate and optimize robot trajectories using PSO. MATLAB allows for the customization of swarm behavior and fine-tuning of parameters like inertia, cognitive, and social coefficients. Visualization tools in MATLAB, such as 3D plots and animations, help in analyzing the movement and convergence of particles, providing insights into the optimization process

2.4 Data Analysis

Data analysis plays a pivotal role in the optimization of security patrol routes and checkpoints through MATLAB, facilitating insights into historical incidents, threat patterns, and patrol efficacy. MATLAB's robust data analysis capabilities enable the processing and interpretation of extensive datasets, crucial for informing optimization algorithms. This involves analyzing historical data to pinpoint high-risk areas and times, which informs prioritization in patrol routes. Additionally, real-time data integration from surveillance feeds and sensors allows for dynamic adjustments to patrol routes in response to emerging threats. Before implementation, simulation tools in MATLAB enable virtual testing of optimized routes under diverse scenarios, ensuring effectiveness and identifying potential improvements. Overall, by leveraging computational geometry, graph theory, optimization techniques, and data analysis, MATLAB empowers security agencies to refine patrol strategies, enhance coverage, mitigate blind spots, and elevate overall security measures effectively.

2.5 Simulation-Based Studies on Security Patrol Optimization

Simulation-based studies have also contributed significantly to the optimization of security patrol routes and checkpoints. For example, Nan et al. (2022) developed a dynamic path planning algorithm based on an improved particle filter optimization technique to handle dynamic obstacles and task changes encountered by patrol robots in power monitoring systems. Their simulation results demonstrated the effectiveness of the proposed algorithm in real-time planning and security fulfillment. Additionally, Verma et al. (2022) proposed a decentralized coordination algorithm for patrolling and target tracking in the Internet of Robotic Things (IoRT), utilizing dynamic waypoints and self-triggered communication mechanisms. Their simulations in the Robot Operating System (ROS) and Gazebo validated the performance of the proposed solution in patrolling and target tracking tasks.

2.6 Challenges and Opportunities

Optimizing security patrol routes and checkpoints using MATLAB presents several challenges. The complexity of real-world scenarios, including varying threat levels and unpredictable incidents, makes accurate modeling and simulation difficult and computationally intensive. Additionally, the availability and quality of data pose significant obstacles, as comprehensive and accurate datasets are crucial for effective optimization but are often difficult to obtain due to privacy concerns and security restrictions. Integrating MATLAB-based solutions with existing legacy systems used by security agencies adds another layer of complexity, involving compatibility issues and the need for seamless real-time data exchange. Furthermore, while simulations can provide valuable insights, real-world validation remains a critical challenge, as factors like human behavior and environmental conditions can significantly impact the performance of optimized routes.

Despite these challenges, there are significant opportunities in this field. The use of advanced algorithms and data analysis techniques in MATLAB can lead to more efficient and effective patrol strategies, reducing resource wastage and improving coverage. The integration of real-time data analytics and machine learning can enhance the adaptability of patrol routes to evolving security threats. Moreover, the ability to simulate and visualize different scenarios provides valuable tools for planning and decision-making. As computational power and data availability continue to improve, the potential for implementing and scaling these solutions in real-world settings increases, offering enhanced security management capabilities and more proactive threat mitigation.

2.7 ARTICLES REVIEW

According to (Yanyun Fu, Tsinghua University, Beijing, Yiping Zeng, Deyong Wang, Hui Zhang, Yang Gao and Yi Liu), this study addresses the challenge of planning routes for security patrols in smart communities. It combines a multi-agent simulation model with a genetic algorithm (GA). The GA evolves the routes to find optimal ones, while the multi-agent model sets constraints and evaluates routes. Using a GIS map for realistic environment representation, the system simulates patrol tasks using the Anylogic platform. Results show the multi-agent system effectively finds optimal routes, making the planning process clear and dynamic[1]. In research by (Maite Dewinter, Christophe Vandeviver, Tom Vander Beken and Frank Witlox), this review paper explores methods to solve the Police Patrol Routing Problem (PPRP), highlighting the complexity of balancing proactive and reactive police duties. It starts with existing literature on dynamic vehicle routing problems (DVRP) and identifies a gap specifically focused on PPRP. Various methods are discussed, including genetic algorithms (GA), linear programming, routing policies, Markov Decision Processes, and online stochastic combinatorial optimization. Hybrid GA, routing policies, and local search are deemed most effective for PPRP.[2]

A study was conducted by (Dongming Zhao, Huimin Yu, Xiang Fang, Lei Tian and Pengqian Han) this paper proposes a path planning algorithm for intelligent patrol cars, used in environments like data centers. The approach combines multi-objective Cauchy mutation cat swarm optimization (MOCMCSO) with the artificial potential field method (APFM) to optimize path length and turning angles. Simulations and experiments in a data center show this method achieves a balance between short paths and smooth navigation, outperforming other optimization methods.[3] In accordance with (Yirui Jiang, Hongwei Li, Binbin Feng, Zekang Wu, Shan Zhao, and Zhaohui Wang), the paper proposes a model to optimize the allocation and routes of city inspectors in Zhengzhou, China, aiming to minimize response times and the number of inspectors needed. It uses a priority-patrol-and-multi objective genetic algorithm (DP-MOGA) to classify patrol segments and develop optimal routes. Numerical experiments confirm the algorithm’s effectiveness in reducing response times and stabilizing performance across different scenarios.[4]

In consonance with (Qiu Mingyue, Zhang Xueying,2023), this study develops precise patrol routes for police by combining ant colony optimization and genetic algorithms, addressing the declining effectiveness of traditional patrols. The ant colony algorithm first determines the shortest routes, which are then refined using the K-means algorithm and optimized further with genetic algorithms to ensure accurate and effective patrol routes[5]. According to (Omer Ozkan, Muhammed Kaya,2021), this research focuses on optimizing UAV routes for border security, specifically the Turkey-Syria border. A Genetic Algorithm Based Matheuristic (GABM) is developed to minimize the number of UAVs needed and optimize their routes. The approach combines genetic algorithms with mathematical modeling to improve efficiency, significantly reducing UAV numbers and flight distances in simulations.[6]

In proportion to (Cesar Guevara, Janio Jadán, César Zapata, Luis Martínez, Jairo Pozo and Edison Manjarres, 2019), this article proposes a dynamic route generation model for police patrolling using K-means clustering to identify critical patrol points and Google Maps API for route design. Tested within Ecuadorian territory, the model showed promising results in the testing phase and led to the development of an Android-based mobile application for police use.[7]

As stated by (Yongxin Gao, Zhonglin Dai, Jing Yuan, 2022), the paper presents a hybrid algorithm for path planning of coal mine patrol robots, improving upon artificial fish swarm and dynamic window algorithms. By incorporating an improved genetic algorithm, the proposed method enhances path accuracy, obstacle avoidance, and overall efficiency. Simulations show reduced path length, time, and improved smoothness in the challenging underground environment[8]. Pursuant to (Cesar Guevara and Matilde Santos,2020), this study proposes an algorithm combining AI and machine learning to create police patrol routes using crime data from Quito, Ecuador. It predicts future crime hotspots and uses spatial-temporal information to design effective patrol routes through a fuzzy decision system. The results demonstrate that integrating spatial and temporal data improves route planning, enhances security, and reduces police resource expenditure.[9]

According to (Yirui Jiang, Shan Zhao, Hongwei Li, Yulu Qin and Xiaoyue Yang,2022), this paper addresses the problem of dividing areas for street patrols. It introduces a new algorithm that combines spectral clustering and simulated annealing. Tested in Zhengzhou, China, this algorithm improves patrol area assignments by considering workload and is more effective compared to other advanced algorithms[10]. In the manner of (Hai-shi Liu, Yu-xuan Sun, Nan Pan, Qi-yong Chen, Xiao-jue Guo and Di-lin Pan,2021), this study proposes a new method to improve the efficiency of border patrols using drones. It employs a whale algorithm based on chaos theory for planning drone missions in complex environments. The method effectively coordinates multiple drones, improving patrol efficiency as shown by simulation results.[11]

As reported by (Alam, Rahman, Carrillo and P. et al. Stochastic,2020), the paper discusses how multiple robots can patrol an area with limited visibility and no reliable communication. It proposes using non-deterministic patrolling paths to avoid predictability by adversaries. The method employs Markov chains and convex optimization, proving effective through simulations and physical tests.[12] In line with (Mario Peñacoba, Jesús Enrique Sierra-García, Matilde Santos, and Ioannis Mariolis,2023), this research focuses on optimizing the paths of a surveillance robot

using genetic algorithms, particle swarm optimization, and pattern search. Each method is compared in various environments, showing that pattern search works best in simple scenarios, while particle swarm optimization excels in complex ones.[13]
In accordance with (Kangjing Shi and Li Huang,2022), the study presents an improved hybrid algorithm combining genetic and ant colony optimization for intelligent vehicle path planning. It focuses on reducing the number of turns and energy consumption. The improved algorithm outperforms traditional methods, especially in reducing iterations and turns in both simple and complex grids. [14]Correspondent to (Jose, C, Vijula Grace and K.S.,2020), this paper proposes a dynamic path planning model for emergency vehicles using a VANET simulation and a new optimization algorithm, Exp-BSA. The method forecasts travel times and finds optimal paths, outperforming other methods in terms of distance, traffic density, speed, and travel time in various simulations.[15]

Concordant to (Omer Ozkan,2021), this paper develops an algorithm to use UAVs for forest fire detection, combining simulated annealing and local search with an integer linear programming model. Tested in Turkey, the algorithm uses daily fire-risk maps to optimize UAV routes, proving effective and efficient in reducing CPU times compared to a genetic algorithm.[16] In agreement to (Li Huang, MengChu Zhou, Hua Han, Shouguang Wang and Aiiad Albeshri,2024), this study proposes a learning-inspired immune algorithm for planning maritime patrol paths for multiple robots. It uses historical data to improve solution diversity and efficiency. The algorithm performs better than existing methods, generating multiple effective patrol schemes and saving time and storage space.[17]

In rapport to (Rathore, Jain and Parida,2019), this paper develops a genetic algorithm-based solution for the vehicle routing problem, focusing on optimizing warehouse locations and transportation costs in logistics operations. Tested with real-life scenarios, the model successfully determines optimal warehouse locations to meet customer demands at minimal cost.[18] Corresponding to (Xiuqiao Sun, Xiuqiao Sun, SciProfilesScilitPreprints, Jian Wang, Weitiao Wu and Wenjia Liu,2019), the research introduces a genetic algorithm to optimize patrol routing and fleet allocation for freeway service patrols. It compares overlapping and non-overlapping patrol strategies, showing that overlapping is more effective. The genetic algorithm outperforms simulated annealing, especially when integrated with LINGO software for fleet allocation.[19]

According to (Yao Nan, Qin Jian-Hua, Zhu Xue-Qiong and Wang Hong-Chang,2022), this paper introduces a dynamic path planning algorithm for patrol robots that improves traditional particle filter optimization. The new algorithm handles unexpected obstacles and task changes in real time, ensuring efficient and secure path planning. Simulations show its effectiveness in dynamic scenarios.[20]

Corresponding to (Verma, Janardan Kumar, Ranga and Virender,2022), this research presents a decentralized coordination strategy for robots in the Internet of Robotic Things (IoRT) to perform patrolling and target tracking. Robots are divided into two groups for perimeter and area patrolling, using dynamic waypoints and a self-triggered communication system. The solution is validated through simulations, demonstrating good performance in patrolling and tracking.[21]

Uniform to (Yuan, Xiaojuana, Shi, and Chunsheng,2019), The paper explores optimizing tourism routes using an improved ant colony algorithm. Tested on travel simulation problems, the improved algorithm finds shorter and more efficient paths compared to the standard ant colony algorithm, with faster convergence and better solutions.[22] As reported by (Nafis Ahmed,Chaitali, J. PawaseORCID and

KyungHi Chang) This study proposes a particle swarm optimization (PSO) algorithm for distributed 3-D path planning of multiple UAVs to ensure full area surveillance. The algorithm allows UAVs to independently plan optimal paths while avoiding collisions. Simulations show that the method produces effective and efficient surveillance paths.[22]

2.8 LIST OF LITERATURE REVIEW WITH TABLE

Table 2.1:Literature Review Comparison

No	Title	Objective	Results	Limitations	Software Used	Dataset Used	Author(s)
1	Research on Route Optimization Based on Multiagent and Genetic Algorithm for Community Patrol	To solve the problem of route planning for security patrol in smart communities using a multi-agent model and genetic algorithm.	The designed multi-agent system can obtain optimal results, with intuitive and visible route planning that meets dynamic requirements.	Limited testing and real-world validation required.	AnyLogic	GIS map	Yanyun Fu, Yiping Zeng, Deyong Wang, Hui Zhang, Yang Gao, Yi Liu
2	Analysing the Police Patrol Routing Problem: A Review	To discuss solution methods for the police patrol routing problem (PPRP).	(Hybrid) GA, routing policies, and local search are most valuable for solving the PPRP.	No specific focus on PPRP in existing literature; knowledge gap remains.	Not specified	Not specified	Maite Dewinter, Christophe Vandeviver, Tom Vander Beken, Frank Witlox
3	A Path Planning Method Based on Multi-Objective Cauchy Mutation Cat Swarm Optimization Algorithm for	To propose a path planning algorithm for intelligent patrol cars using multi-objective optimization.	MOCMCSO algorithm balances shortest path and path smoothness, outperforming MOCSO and	Limited to simulations; real-world applicability not fully assessed.	Not specified	Not specified	Dongming Zhao, Huimin Yu, Xiang Fang, Lei Tian, Pengqian Han

	Navigation System of Intelligent Patrol Car		MOPSO in simulations.				
4	Street Patrol Routing Optimization in Smart City Management Based on Genetic Algorithm: A Case in Zhengzhou, China	To minimize average response time and number of inspectors using a genetic algorithm for patrol path optimization.	The proposed algorithm reduces average response time and number of inspectors, demonstrating efficiency in different time scenarios.	Limited to a specific city; applicability to other regions not assessed.	Not specified	Patrol data	Yirui Jiang, Hongwei Li, Binbin Feng, Zekang Wu, Shan Zhao, Zhaohui Wang
5	Determining Accurate Patrol Routes Using Genetic Algorithm and Ant Colony	To develop precise patrol routes using ant colony and genetic algorithms based on crime hotspots.	Accurate patrol routes are obtained, improving patrol effectiveness.	Limited to urban areas with crime hotspots; applicability to other contexts not assessed.	Not specified	Not specified	Qiu Mingyue, Zhang Xueying
6	UAV routing with genetic algorithm based matheuristic for border security missions	To ensure border security using UAVs with a genetic algorithm based matheuristic approach.	GABM approach decreases the number of UAVs and their flight distances significantly in various scenarios.	Limited to the Turkey-Syria border; applicability to other borders not assessed.	Not specified	Not specified	Omer Ozkan, Muhammed Kaya
7	A Multiobjective Hybrid Optimization Algorithm for Path	To improve path planning for coal mine patrol robots using a	The proposed algorithm shortens path length, reduces	Limited to simulations; real-world	ROS	Not specified	Yongxin Gao, Zhonglin

	Planning of Coal Mine Patrol Robot	hybrid algorithm.	time, and removes redundant points, showing effectiveness and superiority.	validation required.			Dai, Jing Yuan
8	Smart Patrolling Based on Spatial-Temporal Information Using Machine Learning	To improve security and reduce crime using an AI and ML algorithm for generating police patrol routes.	The algorithm effectively designs patrolling routes, improving citizen security and reducing police resource spending.	Limited to Quito City, Ecuador; scalability to other cities not assessed.	OpenStreetMap API	Crime data	Cesar Guevara, Matilde Santos
9	A hybrid spectral clustering simulated annealing algorithm for the street patrol districting problem	To optimize street patrol districting using a hybrid algorithm.	The proposed algorithm effectively solves the street patrol districting problem.	Limited testing on real instances from Zhengzhou, China.	Not specified	Not specified	Yirui Jiang, Shan Zhao, Hongwei Li, Yulu Qin, Xiaoyue Yang
10	Multi-UAV Cooperative Task Planning for Border Patrol based on Hierarchical Optimization	To improve the patrol efficiency of border patrol drones using a hierarchical optimization algorithm.	The proposed scheme can effectively plan multi-UAV coordinated missions for border patrol.	Not specified	Not specified	Not specified	Hai-shi Liu, Yu-xuan Sun, Nan Pan, Qi-yong Chen, Xiao-jue Guo, Di-lin Pan
11	Stochastic Multi-Robot Patrolling with Limited Visibility	To develop patrolling policies for multiple robots using limited	Proposed visibility-based non-deterministic patrolling method shows	May not account for all real-world factors influencing	Not specified	Not specified	Alam, Rahman, Carrillo, et al.

		visibility regions and non-deterministic patrolling paths.	effectiveness in simulations and physical implementation.	patrolling scenarios.			
12	Path Optimization Using Metaheuristic Techniques for a Surveillance Robot	To optimize trajectories of a robotic surveillance system using genetic algorithm, particle swarm optimization, and pattern search methods.	Pattern search method quickly obtains feasible solutions; PSO works better in complex environments.	Results may vary based on initial trajectory and environment complexity.	MATLAB	Not specified	Mario Peñacoba, Jesús Enrique Sierra-García, Matilde Santos, Ioannis Mariolis
13	Path Planning Optimization of Intelligent Vehicle Based on Improved Genetic and Ant Colony Hybrid Algorithm	To improve path planning for intelligent vehicles using an improved genetic and ant colony hybrid algorithm.	The proposed hybrid algorithm effectively reduces the number of turns in simple and complex grid maps.	Limited to simulation and physical experiments; may not account for all real-world factors.	MATLAB	Not specified	Kangjing Shi, Li Huang
14	Optimization based routing model for the dynamic path planning of emergency vehicles	To develop a dynamic path planning scheme for routing emergency vehicles using an Exponential Bird Swarm	Proposed Exp-BSA method achieves overall best performance compared to existing methods in simulations.	Limited to simulation setups; real-world applicability not fully assessed.	Not specified	Not specified	Jose, C, Vijula Grace, K.S.

		Optimization Algorithm.					
15	Optimization of the distance-constrained multi-based multi-UAV routing problem with simulated annealing and local search-based matheuristic to detect forest fires: The case of Turkey	To develop a matheuristic algorithm for routing UAVs to detect forest fires, using simulated annealing and local search techniques.	Proposed algorithm effectively mitigates forest fire risks in a real-life case study for Turkey.	Limited to Turkish case study; scalability to other regions not explored.	MATLAB, ILOG	Not specified	Omer Ozkan
16	Learning-Inspired Immune Algorithm for Multiobjective-Optimized Multirobot Maritime Patrolling	To develop an immune algorithm for multi-objective optimization of multirobot maritime patrolling.	Proposed algorithm generates multiple patrolling schemes for decision-makers and outperforms state-of-the-art methods.	Limited to simulations; real-world implementation not assessed.	Not specified	Not specified	Li Huang, MengChu Zhou, Hua Han, Shouguang Wang, Aiiad Albeshri
17	A MATLAB-Based Application to Solve Vehicle Routing Problem Using GA	To develop a MATLAB-based application for solving the vehicle routing problem using a genetic algorithm.	Proposed algorithm successfully finds potential locations for warehouse setup based on real-life scenarios.	Limited to generated data based on real-life scenarios; real-world applicability may vary.	MATLAB	Not specified	Rathore, Jain, Parida

18	Genetic Algorithm for Optimizing Routing Design and Fleet Allocation of Freeway Service Overlapping Patrol	To optimize routing design and fleet allocation for freeway service patrols using a genetic algorithm.	Overlapping patrol strategy outperforms non-overlapping strategy; GA performs better than SA.	Limited to numerical experiments; applicability to other regions not assessed.	Not specified	Sioux Falls	Xiuqiao Sun, SciProfilesScilitPreprints, Jian Wang, Weitiao Wu, Wenjia Liu
19	Dynamic Path Planning Based on Improved Particle Filter Optimization for Patrol Robots	To develop a dynamic path planning algorithm for patrol robots using an improved particle filter optimization technique.	Proposed algorithm effectively handles dynamic obstacles and task changes encountered by patrol robots.	Limited to simulations; real-world validation required.	Not specified	Not specified	Yao Nan, Qin Jian-Hua, Zhu Xue-Qiong, Wang Hong-Chang
20	A Decentralized Coordination Algorithm for Patrolling and Target Tracking in Internet of Robotic Things (IoRT) using Dynamic Waypoints and Self-triggered Communication	To propose a decentralized coordination strategy for patrolling and target tracking in IoRT using dynamic waypoints and self-triggered communication.	Proposed solution effectively coordinates multiple robots for patrolling and target tracking tasks.	Limited to simulations in ROS and Gazebo; real-world implementation not assessed.	Not specified	Not specified	Verma, Janardan Kumar, Ranga, Virender

21	Research on tourism individualized route management based on intelligent optimization algorithm	To optimize tourism routes using ant colony and improved ant colony algorithms.	Improved ant colony algorithm achieves smaller optimal path and average path length compared to ant colony algorithm.	Limited to simulations; real-world applicability not fully assessed.	MATLAB	Benchmark 27, Att48, kroA100	Yuan, Xiaojuana, Shi, Chunsheng
22	Distributed 3-D Path Planning for Multi-UAVs with Full Area Surveillance Based on Particle Swarm Optimization	To develop a trajectory planner for multi-UAVs with full area surveillance using Particle Swarm Optimization.	Proposed algorithm generates feasible optimal trajectories for UAVs to surveil entire areas of interest.	Limited to simulations; real-world validation required.	Not specified	Not specified	Nafis Ahmed,Chaitali, J. PawaseORCID and KyungHi Chang

2.9 Overview of studies

The discussion initiates with a global perspective on the critical importance of optimizing security patrol routes and checkpoints, emphasizing the shortcomings of traditional methods and the necessity for systematic planning to mitigate security threats effectively. Transitioning to methodologies, the discourse highlights the pivotal role of MATLAB in facilitating optimization through Genetic Algorithms (GA), Multi-Agent Systems (MAS), and Particle Swarm Optimization (PSO), elucidating their applications in dynamic security contexts. This section encapsulates the versatility of MATLAB in implementing sophisticated optimization strategies tailored to address evolving security dynamics.

Furthermore, a synthesis of pertinent literature encapsulates diverse studies and research endeavors, showcasing the breadth of applications and methodologies in security patrol route optimization. By meticulously examining each study's objectives, methodologies, and outcomes, this overview underscores the collective insights garnered from empirical investigations in the field. Concluding with a discussion on challenges and opportunities, the discourse underscores critical considerations such as data availability, computational resources, and ethical considerations, providing a comprehensive overview of the complexities inherent in deploying MATLAB-based optimization solutions for security management.

2.10 Summary

The literature review provides a comprehensive exploration of security patrol route optimization, addressing its global significance, methodological approaches, empirical studies, and practical considerations. It underscores the inadequacies of traditional patrol methods and emphasizes the need for systematic planning to effectively mitigate security threats. The review elucidates the role of MATLAB in implementing optimization techniques such as Genetic Algorithms (GA), Multi-Agent Systems (MAS), and Particle Swarm Optimization (PSO), showcasing its versatility in dynamic security contexts. Through a synthesis of relevant literature, diverse studies and research endeavors are examined, highlighting the breadth of applications and methodologies in security optimization. The review culminates with a discussion on challenges and opportunities, elucidating critical considerations such as data availability, computational resources, and ethical considerations, thus providing a holistic overview of the intricacies involved in deploying MATLAB-based solutions for security management.

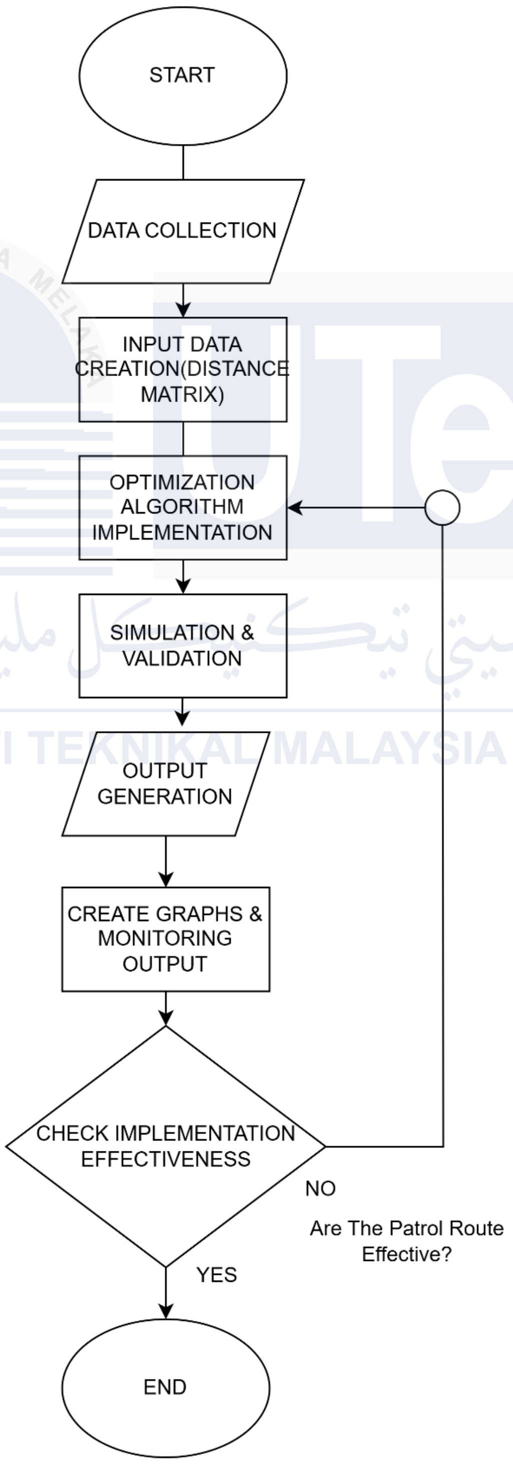
CHAPTER 3

METHODOLOGY

3.1 Introduction

The methodology for this project will involve the development of an efficient and systematic approach to optimize the security patrolling route at UTeM. This is meant to minimize travel distance and time to fully cover all checkpoints. The nearest neighbor algorithm, one of the well-known route optimization techniques, is adopted in this project to find the shortest path for patrolling. These distances and times were calculated between all checkpoints using Google Maps, tabulated in the form of well-structured matrices. The basis for these becomes the grounds upon which the computation of optimal patrol routes is done by considering prevailing conditions. This research methodology integrates math modeling with the design of an algorithm to ensure that practical usability and reliability are warranted when the methodology of improving security details' performances is enhanced.

3.2 Flowchart



START
DATA COLLECTION
INPUT DATA CREATION(DISTANCE MATRIX)
OPTIMIZATION ALGORITHM IMPLEMENTATION
SIMULATION & VALIDATION
OUTPUT GENERATION
CREATE GRAPHS & MONITORING OUTPUT
CHECK IMPLEMENTATION EFFECTIVENESS
NO
Are The Patrol Route Effective?
YES
END

Figure 3.1: Flowchart

3.3 Data Collection and Analysis

To begin my project, I used Android apps such as MyGPSCoordinate to gather location data at each checkpoint. This involved recording precise coordinates using GPS technology. Once I had the coordinates for each checkpoint, I used an NFC tool app to program NFC tags specifically for each checkpoint. These NFC tags were configured with the respective GPS coordinates previously recorded.

After setting up the NFC tags, I organized all the collected data into an Excel file. This file now serves as the central input for my project. It contains detailed location information for each checkpoint, which is crucial for developing and optimizing security patrol routes and checkpoint placements using MATLAB.

In summary, the initial steps involved using Android apps to collect GPS coordinates at checkpoints and then configuring NFC tags with this location data. By storing this information in an Excel file, I have established a solid foundation for the subsequent phases of my project, where I will utilize MATLAB to enhance security measures by optimizing patrol routes and strategically placing checkpoints to minimize blind spots.

3.4 Route Optimization and Algorithm Designing

- Matrix construction:

One of the most important parts of the Nearest Neighbor algorithm is to build the matrices. For this project, I have made two 29×29 matrices, one for distances and another one for time. Every element of this matrix will give the distance or time between any two checkpoints pairwise, while the diagonals will be zero since the distance or time from a checkpoint to itself is zero. Because these matrices are symmetric meaning the value from Checkpoint A to B is the same as from Checkpoint B to A. All data used to populate these matrices came from Google Maps, which assures that the information is valid in a real-world context. I prepared some Excel files, distance.xlsx for the distance matrix and time taken.xlsx for the time matrix. These were carefully created based on data retrieved from Google Maps about all the possible routes between the 29 checkpoints so that the algorithm can calculate the shortest path effectively. The distance matrix serves as an input to calculate the shortest route based on physical distance, while the time matrix considers travel duration, which may vary due to factors such as traffic. All this was necessary for the algorithm to provide accurate and efficient results for the security patrolling task at UTeM.

- Nearest Neighbor Algorithm:

The Nearest Neighbor is a greedy strategy for route optimization problems, it can be used for security patrolling checkpoints algorithm. It starts from an initial location, at each step chooses that unvisited location that is closest according to a distance matrix and finally returns to the originating location after visiting all the other locations. The algorithm prioritizes local optimization at each step, seeking the shortest immediate path without considering the overall route's global optimization. In practice, the Nearest Neighbor algorithm works in an efficient way to determine a near-optimal solution for visiting a set of locations. For this project at UTeM in security patrolling, this algorithm will select the best route for patrol starting from a given checkpoint where at every step one travels to the nearest unvisited checkpoint until all the checkpoints are visited. This makes the travel distance and time as little as possible, thus making the patrol efficient and orderly. This would complete the patrol cycle, as the campus would be fully covered by returning to the starting point.

- Simulation and Testing:

In the development process, the designs will be subjected to rigorous simulation and testing using MATLAB. This iterative approach will validate whether designed algorithms are effective in optimizing patrol routes or not. Scenarios considered for simulation include, varying patrol frequencies, changing weights to edges based on dynamic security priorities, and runtime adjustment of routes according to simulated security incidents. This will hopefully cut on the blind spots within patrol coverage through continued testing and refinement of the algorithms for enhanced efficiency in security patrol.

By integrating matrix construction, optimization techniques, and MATLAB's computational capability, the route optimization algorithm will be systematically tailored to meet the project's objectives. This will enhance security measures through optimized patrol route planning, which will cover all critical areas, maximize patrol efficiency, and improve responsiveness to dynamic security challenges.

3.5 BLOCK DIAGRAM

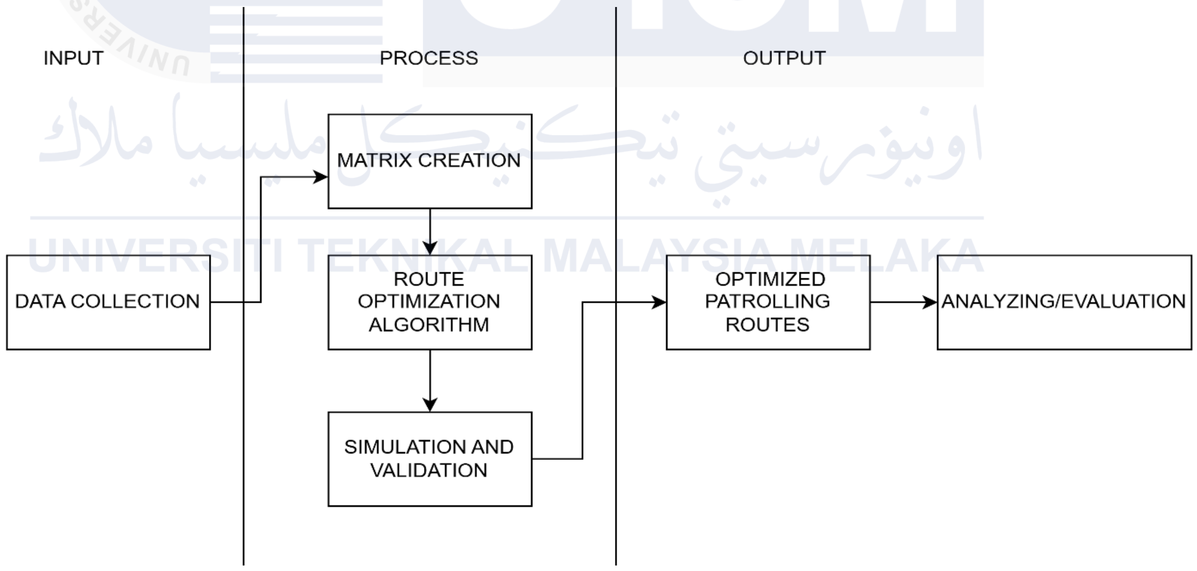

INPUT
PROCESS
OUTPUT
MATRIX CREATION
DATA COLLECTION
ROUTE OPTIMIZATION ALGORITHM
SIMULATION AND VALIDATION
OPTIMIZED PATROLLING ROUTES
ANALYZING/EVALUATION

Figure 3.2: Block Diagram

3.6 Summary

This project aimed to optimize path for patrolling routes at UTeM is based on the principle of minimum distance while traveling but involving all checkpoints. First and foremost, the GPS coordinates for 29 checkpoints have been measured using an Android application, MyGPSCoordinate. The coordination then was programmed into the NFC tags, which make for easy identification during patrolling. All this data collected was tabulated in an Excel file, the central input into route optimization and checkpoint placement. It ensures that all the succeeding steps of the project rely on accurate and valid data.

Two 29×29 symmetric matrices-one for distance and one for travel time to compute the optimal routes for patrolling. These matrices are filled with data from Google Maps, pairwise distances or times between checkpoints, while the diagonal values are set to zero to reflect no travel between the same checkpoints. The distance matrix is used for finding the shortest routes, while the time matrix considers factors like traffic conditions to calculate the fastest routes. This construction of the matrices is very important for the accuracy and effectiveness of the algorithm.

A greedy optimization algorithm known as the Nearest Neighbor Algorithm, was implemented to find the near-optimal patrol routes. From any given initial location, the algorithm selects the closest unvisited checkpoint at each step and returns to the starting point after covering all locations. In this approach, the algorithm does local optimization greedily, thus being efficient and quite suitable for the patrolling requirements of UTeM. This algorithm ensures the least travel distance and time and hence increases the overall efficiency and orderliness of the patrol cycle.

The simulation and validation in MATLAB can go through rigorous testing and refinement of the proposed routes. Further scenarios can include changes in patrol frequency, dynamic prioritization based on area criticality, and response to any simulated incident. It is a continuous testing to be able to implement comprehensive checkpoint coverage, reduce blind spots, and respond effectively to security challenges. Outputs also involve optimized patrol routes, including studies and visualizations by graphs for the purpose of monitoring and assessment.

The method developed here will incorporate advanced data collection, mathematical modeling, and the Nearest Neighbor Algorithm to realize optimally efficient route construction for security patrols. This project can provide practical usability and reliability through simulation of the process in MATLAB. In that sense, this would be a development of good security framework with efficient patrol planning in UTeM and full coverage in general, plus high adaptability against dynamic security needs.

CHAPTER 4

RESULTS AND DISCUSSIONS

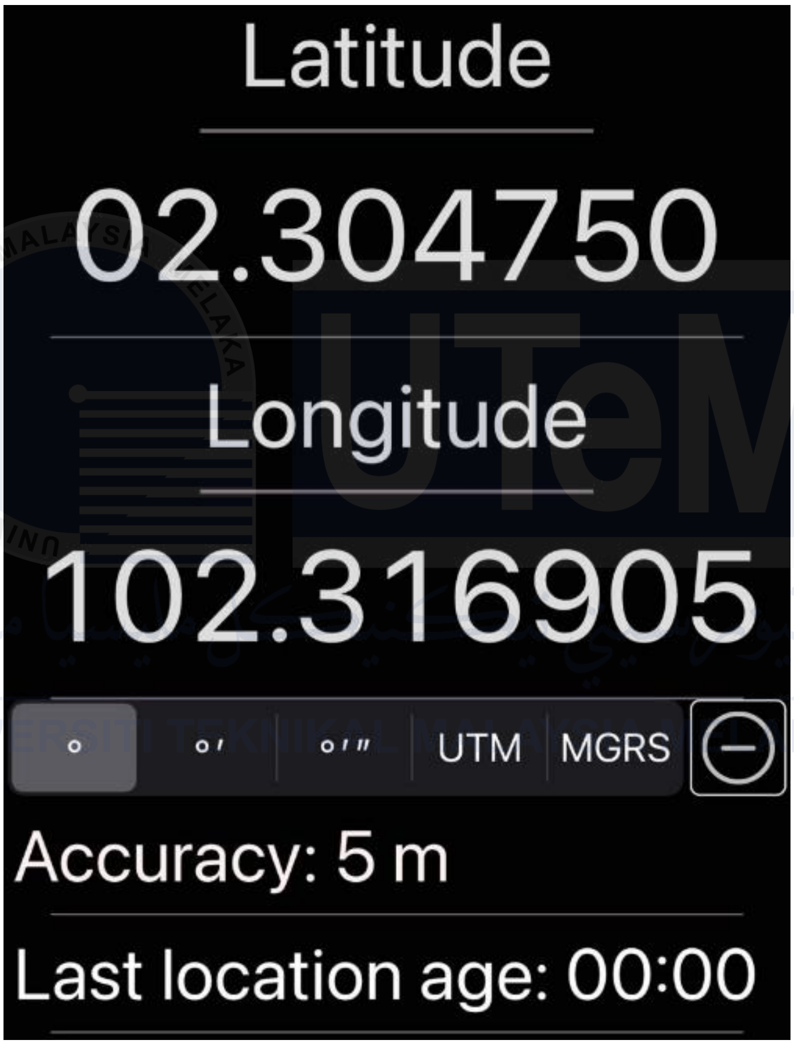
4.1 Introduction

This chapter presents the findings from the development and implementation of the MATLAB-based program designed to optimize security patrol routes and checkpoints. The results from various stages of the project, including data collection, algorithm development, simulation, and implementation, are thoroughly analyzed and discussed. During the data collection phase, latitude and longitude data were gathered and used to configure checkpoints. Key metrics such as patrol coverage efficiency, response times, and the effectiveness of minimizing blind spots are evaluated. The discussion interprets these results in the context of the project's objectives, highlighting the strengths and areas for improvement in the developed system. This chapter aims to provide a comprehensive understanding of how the proposed solutions perform in practice and their potential impact on enhancing security measures.

4.2 Data Collection

Figure 4.1: Collecting Checkpoint latitude and longitude coordinates

During my journey, I visited 29 different checkpoints to capture their latitude and longitude coordinates and ensured all necessary configurations were completed on-site.


Latitude

02.304750

Longitude

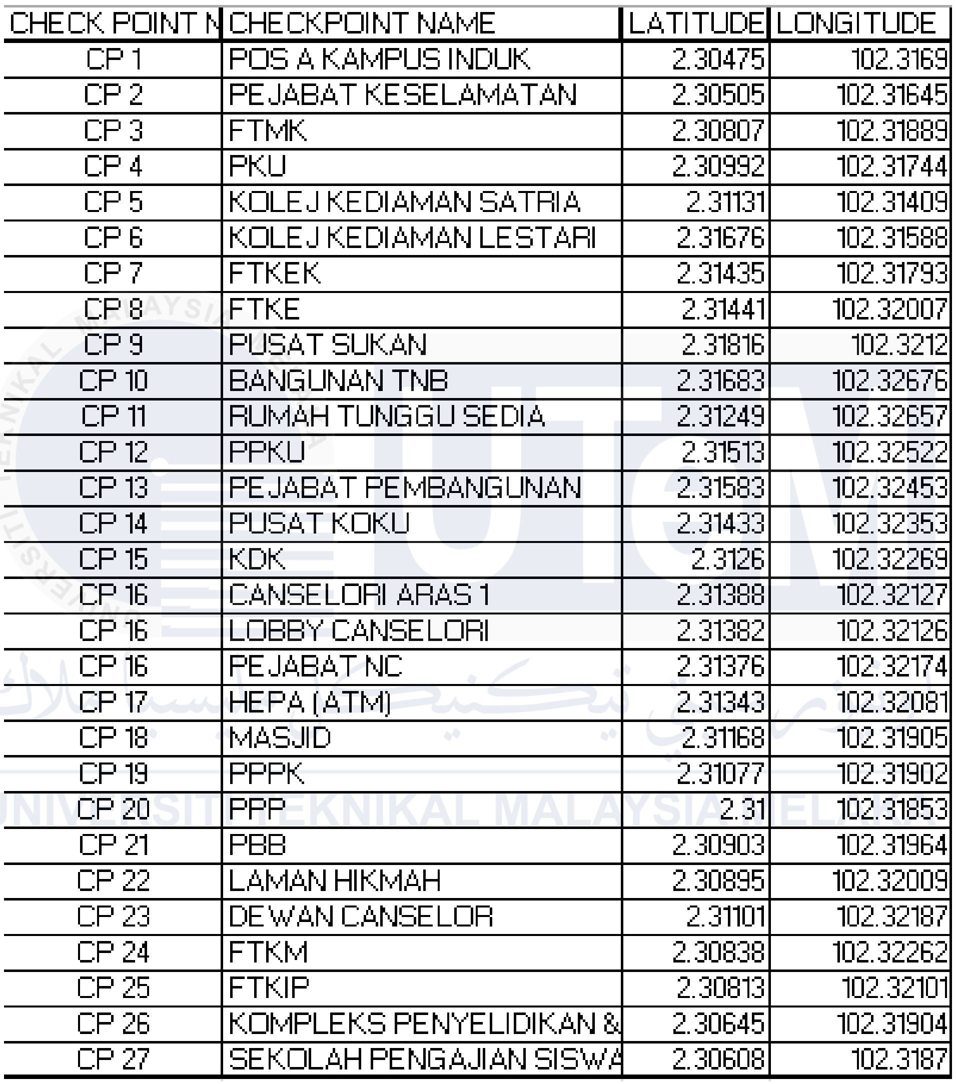
102.316905

Accuracy: 5 m

Last location age: 00:00

Figure 4.2: Example latitude and longitude coordinates

This is an example of data that appeared when I collected the latitude and longitude coordinates. I took a screenshot and saved it for reference.



CHECK POINT N	CHECKPOINT NAME	LATITUDE	LONGITUDE
CP 1	POS A KAMPUS INDUK	2.30475	102.3169
CP 2	PEJABAT KESELAMATAN	2.30505	102.31645
CP 3	FTMK	2.30807	102.31889
CP 4	PKU	2.30992	102.31744
CP 5	KOLEJ KEDIAMAN SATRIA	2.31131	102.31409
CP 6	KOLEJ KEDIAMAN LESTARI	2.31676	102.31588
CP 7	FTKEK	2.31435	102.31793
CP 8	FTKE	2.31441	102.32007
CP 9	PUSAT SUKAN	2.31816	102.3212
CP 10	BANGUNAN TNB	2.31683	102.32676
CP 11	RUMAH TUNGGU SEDIA	2.31249	102.32657
CP 12	PPKU	2.31513	102.32522
CP 13	PEJABAT PEMBANGUNAN	2.31583	102.32453
CP 14	PUSAT KOKU	2.31433	102.32353
CP 15	KDK	2.3126	102.32269
CP 16	CANSELORI ARAS 1	2.31388	102.32127
CP 16	LOBBY CANSELORI	2.31382	102.32126
CP 16	PEJABAT NC	2.31376	102.32174
CP 17	HEPA (ATM)	2.31343	102.32081
CP 18	MASJID	2.31168	102.31905
CP 19	PPPK	2.31077	102.31902
CP 20	PPP	2.31	102.31853
CP 21	PBB	2.30903	102.31964
CP 22	LAMAN HIKMAH	2.30895	102.32009
CP 23	DEWAN CANSELOR	2.31101	102.32187
CP 24	FTKM	2.30838	102.32262
CP 25	FTKIP	2.30813	102.32101
CP 26	KOMPLEKS PENYELIDIKAN &	2.30645	102.31904
CP 27	SEKOLAH PENGAJIAN SISWA	2.30608	102.3187

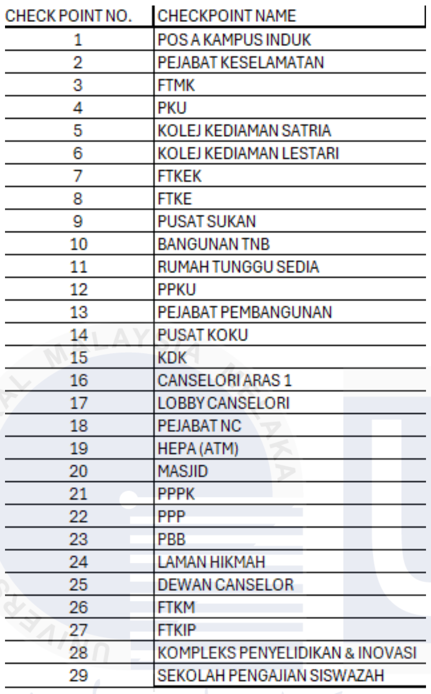
Figure 4.3: All Checkpoints Data

After collecting each set of data and configuring it in NFC, I organized everything neatly in an Excel file, like the table above.

4.3 Data Structuring

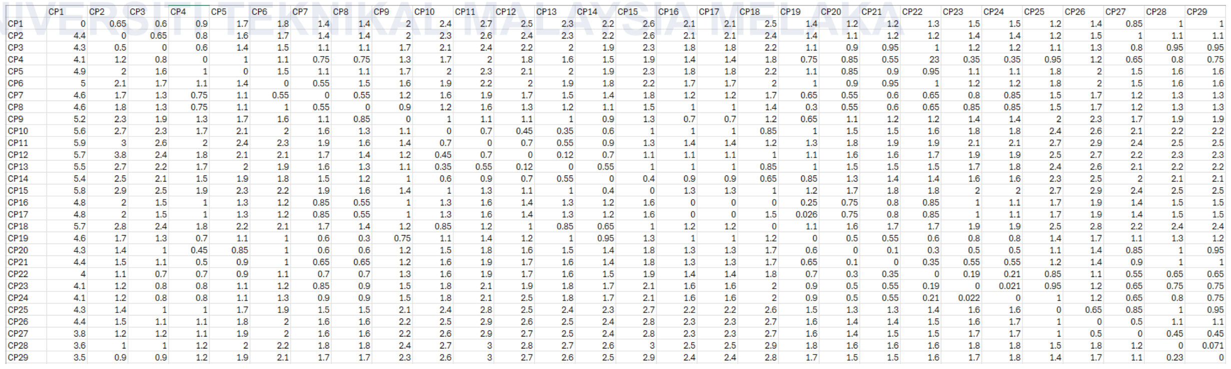
After collecting the latitude and longitude data, I started organizing it to make it more useful. Using the coordinates, I looked up the checkpoints on Google Maps to find the distances and travel times between them. I carefully recorded this information in Excel files, making sure everything was accurate and clear.

Next, I created a 29×29 matrix in Excel. Each box in the matrix shows the distance or time between two checkpoints, while the diagonal boxes are set to zero because there's no distance between a checkpoint itself. This matrix is a key input for the Nearest Neighbor Algorithm, which I used to calculate the shortest and most efficient patrol routes. With this approach, the algorithm can help create a security patrolling plan that covers all checkpoints in the least amount of time and distance.



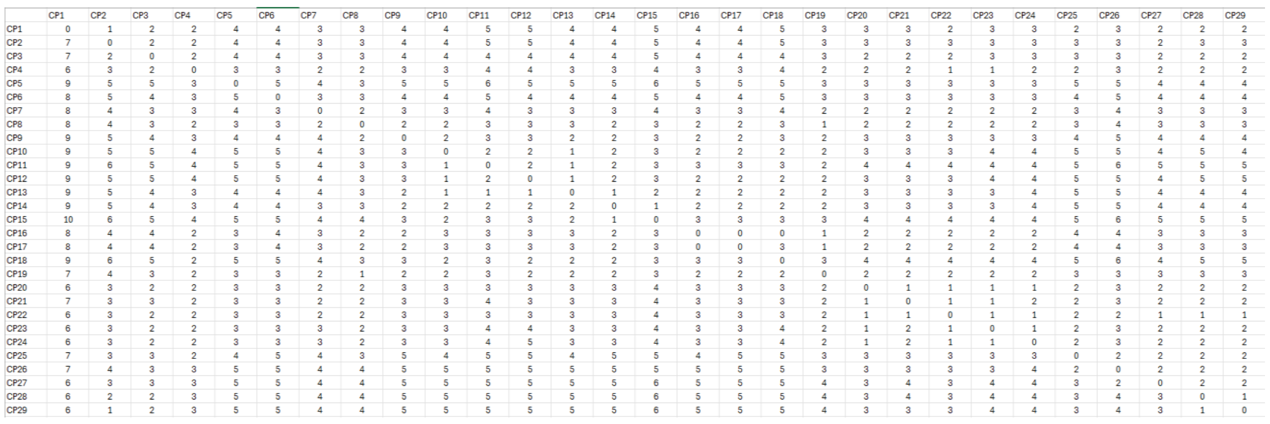
CHECK POINT NO.	CHECKPOINT NAME
1	POS A KAMPUS INDUK
2	PEJABAT KESELAMATAN
3	FTMK
4	PKU
5	KOLEJ KEDIAMAN SATRIA
6	KOLEJ KEDIAMAN LESTARI
7	FTKEK
8	FTKE
9	PUSAT SUKAN
10	BANGUNAN TNB
11	RUMAH TUNGGU SEDIA
12	PPKU
13	PEJABAT PEMBANGUNAN
14	PUSAT KOKU
15	KDK
16	CANSELORI ARAS 1
17	LOBBY CANSELORI
18	PEJABAT NC
19	HEPA (ATM)
20	MASJID
21	PPPK
22	PPP
23	PBB
24	LAMAN HIKMAH
25	DEWAN CANSELOR
26	FTKM
27	FTKIP
28	KOMPLEKS PENYELIDIKAN & INOVASI
29	SEKOLAH PENGAJIAN SISWAZAH

Figure 4.4: Excel data of Checkpoint name and number



	CP1	CP2	CP3	CP4	CP5	CP6	CP7	CP8	CP9	CP10	CP11	CP12	CP13	CP14	CP15	CP16	CP17	CP18	CP19	CP20	CP21	CP22	CP23	CP24	CP25	CP26	CP27	CP28	CP29
CP1	0	0.65	0.6	0.9	1.7	1.8	1.4	1.4	2	2.4	2.7	2.5	2.3	2.2	2.6	2.1	2.1	2.5	1.4	1.2	1.2	1.3	1.5	1.5	1.2	1.4	0.85	1	1
CP2	4.4	0	0.65	0.8	1.6	1.7	1.4	1.4	2	2.3	2.6	2.4	2.3	2.2	2.6	2.1	2.1	2.4	1.4	1.1	1.2	1.2	1.4	1.4	1.2	1.5	1	1.1	1.1
CP3	4.3	0.5	0	0.6	1.4	1.5	1.1	1.1	1.7	2.1	2.4	2.2	2	1.9	2.3	1.8	1.8	2.2	1.1	0.9	0.95	1	1.2	1.2	1.1	1.3	0.8	0.95	0.95
CP4	4.1	1.2	0.8	0	1	1.1	0.75	0.75	1.3	1.7	2	1.8	1.6	1.5	1.9	1.4	1.4	1.8	0.75	0.85	0.55	23	0.35	0.35	0.95	1.2	0.65	0.8	0.75
CP5	4.9	2	1.6	1	0	1.5	1.1	1.1	1.7	2	2.3	2.1	2	1.9	2.3	1.8	1.8	2.2	1.1	0.85	0.9	0.95	1.1	1.1	1.8	2	1.5	1.6	1.6
CP6	5	2.1	1.7	1.1	1.4	0	0.55	1.5	1.6	1.9	2.2	2	1.9	1.8	2.2	1.7	1.7	2	1	0.9	0.95	1	1.2	1.2	1.8	2	1.5	1.6	1.6
CP7	4.6	1.7	1.3	0.75	1.1	0.55	0	0.55	1.2	1.6	1.9	1.7	1.5	1.4	1.8	1.2	1.2	1.7	0.65	0.55	0.6	0.65	0.8	0.85	1.5	1.7	1.2	1.3	1.3
CP8	4.6	1.8	1.3	0.75	1.1	1	0.55	0	0.9	1.2	1.6	1.3	1.2	1.1	1.5	1	1	1.4	0.3	0.55	0.6	0.65	0.85	0.85	1.5	1.7	1.2	1.3	1.3
CP9	5.2	2.3	1.9	1.3	1.7	1.6	1.1	0.85	0	1	1.1	1.1	1	0.9	1.3	0.7	0.7	1.2	0.65	1.1	1.2	1.2	1.4	1.4	2	2.3	1.7	1.9	1.9
CP10	5.6	2.7	2.3	1.7	2.1	2	1.6	1.3	1.1	0	0.7	0.45	0.35	0.6	1	1	1	0.85	1	1.5	1.5	1.6	1.8	1.8	2.4	2.6	2.1	2.2	2.2
CP11	5.9	3	2.6	2	2.4	2.3	1.9	1.6	1.4	0.7	0	0.7	0.55	0.9	1.3	1.4	1.4	1.2	1.3	1.8	1.9	1.9	2.1	2.1	2.7	2.9	2.4	2.5	2.5
CP12	5.7	3.8	2.4	1.8	2.1	2.1	1.7	1.4	1.2	0.45	0.7	0	0.12	0.7	1.1	1.1	1.1	1	1.1	1.6	1.6	1.7	1.9	1.9	2.5	2.7	2.2	2.3	2.3
CP13	5.5	2.7	2.2	1.7	2	1.9	1.6	1.3	1.1	0.35	0.55	0.12	0	0.55	1	1	1	0.85	1	1.5	1.5	1.5	1.7	1.8	2.4	2.6	2.1	2.2	2.2
CP14	5.4	2.5	2.1	1.5	1.9	1.8	1.5	1.2	1	0.6	0.9	0.7	0.55	0	0.4	0.9	0.9	0.65	0.85	1.3	1.4	1.4	1.6	1.6	2.3	2.5	2	2.1	2.1
CP15	5.8	2.9	2.5	1.9	2.3	2.2	1.9	1.6	1.4	1	1.3	1.1	1	0.4	0	1.3	1.3	1	1.2	1.7	1.8	1.8	2	2	2.7	2.9	2.4	2.5	2.5
CP16	4.8	2	1.5	1	1.3	1.2	0.85	0.55	1	1.3	1.6	1.4	1.3	1.2	1.6	0	0	0	0.25	0.75	0.8	0.85	1	1.1	1.7	1.9	1.4	1.5	1.5
CP17	4.8	2	1.5	1	1.3	1.2	0.85	0.55	1	1.3	1.6	1.4	1.3	1.2	1.6	0	0	1.5	0.026	0.75	0.8	0.85	1	1.1	1.7	1.9	1.4	1.5	1.5
CP18	5.7	2.8	2.4	1.8	2.2	2.1	1.7	1.4	1.2	0.85	1.2	1	0.85	0.65	1	1.2	1.2	0	1.1	1.6	1.7	1.7	1.9	1.9	2.5	2.8	2.2	2.4	2.4
CP19	4.6	1.7	1.3	0.7	1.1	1	0.6	0.3	0.75	1.1	1.4	1.2	1	0.95	1.3	1	1	1.2	0	0.5	0.55	0.6	0.8	0.8	1.4	1.7	1.1	1.3	1.2
CP20	4.3	1.4	1	0.45	0.85	1	0.6	0.6	1.2	1.5	1.8	1.6	1.5	1.4	1.8	1.3	1.3	1.7	0.6	0	0.1	0.3	0.5	0.5	1.1	1.4	0.85	1	0.95
CP21	4.4	1.5	1.1	0.5	0.9	1	0.65	0.65	1.2	1.6	1.9	1.7	1.6	1.4	1.8	1.3	1.3	1.7	0.65	0.1	0	0.35	0.55	0.55	1.2	1.4	0.9	1	1
CP22	4	1.1	0.7	0.7	0.9	1.1	0.7	0.7	1.3	1.6	1.9	1.7	1.6	1.5	1.9	1.4	1.4	1.8	0.7	0.3	0.35	0	0.19	0.21	0.85	1.1	0.55	0.65	0.65
CP23	4.1	1.2	0.8	0.8	1.1	1.2	0.85	0.9	1.5	1.8	2.1	1.9	1.8	1.7	2.1	1.6	1.6	2	0.9	0.5	0.55	0.19	0	0.021	0.95	1.2	0.65	0.75	0.75
CP24	4.1	1.2	0.8	0.8	1.1	1.3	0.9	0.9	1.5	1.8	2.1	2.5	1.8	1.7	2.1	1.6	1.6	2	0.9	0.5	0.55	0.21	0.022	0	1	1.2	0.65	0.8	0.75
CP25	4.3	1.4	1	1	1.7	1.9	1.5	1.5	2.1	2.4	2.8	2.5	2.4	2.3	2.7	2.2	2.2	2.6	1.5	1.3	1.3	1.4	1.6	1.6	0	0.65	0.85	1	0.95
CP26	4.4	1.5	1.1	1.1	1.8	2	1.6	1.6	2.2	2.5	2.9	2.6	2.5	2.4	2.8	2.3	2.3	2.7	1.6	1.4	1.4	1.5	1.6	1.7	1	0	0.5	1.1	1.1
CP27	3.8	1.2	1.2	1.1	1.9	2	1.6	1.6	2.2	2.6	2.9	2.7	2.5	2.4	2.8	2.3	2.3	2.7	1.6	1.4	1.5	1.5	1.7	1.7	1	0.5	0	0.45	0.45
CP28	3.6	1	1	1.2	2	2.2	1.8	1.8	2.4	2.7	3	2.8	2.7	2.6	3	2.5	2.5	2.9	1.8	1.6	1.6	1.6	1.8	1.8	1.5	1.8	1.2	0	0.071
CP29	3.5	0.9	0.9	1.2	1.9	2.1	1.7	1.7	2.3	2.6	3	2.7	2.6	2.5	2.9	2.4	2.4	2.8	1.7	1.5	1.5	1.6	1.7	1.8	1.4	1.7	1.1	0.23	0

Figure 4.5: Excel Data of Distance Matrix

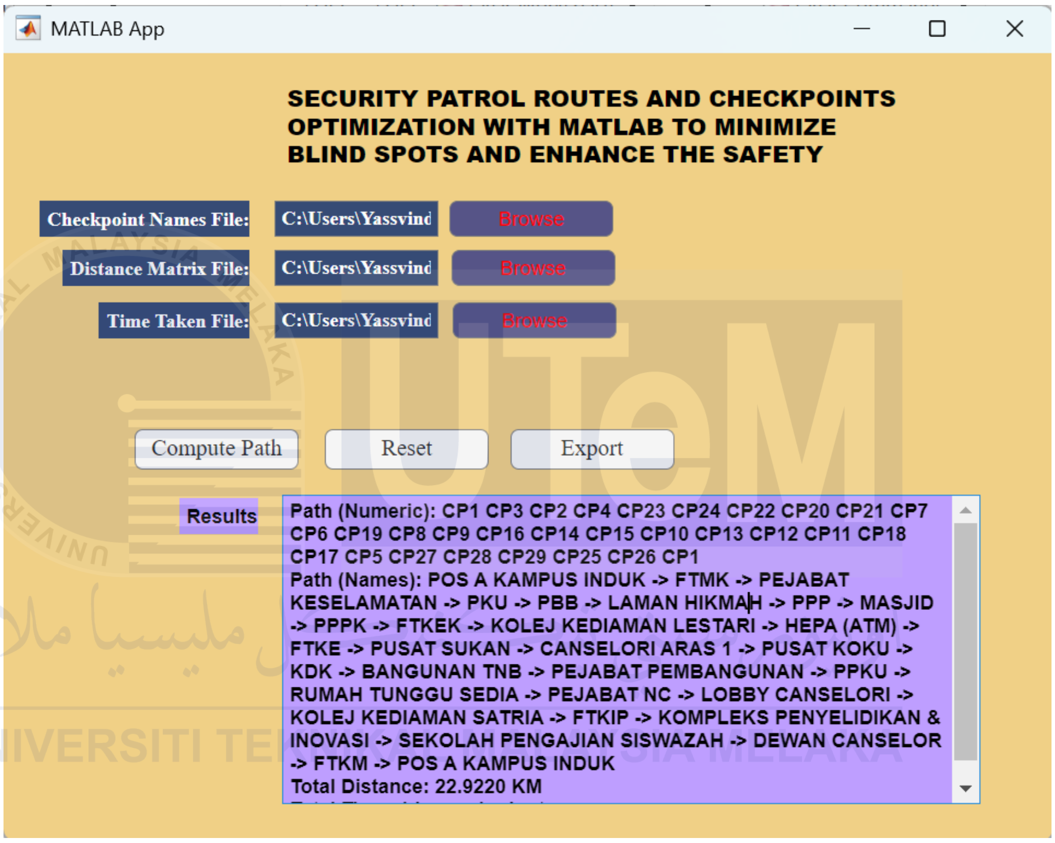


	CP1	CP2	CP3	CP4	CP5	CP6	CP7	CP8	CP9	CP10	CP11	CP12	CP13	CP14	CP15	CP16	CP17	CP18	CP19	CP20	CP21	CP22	CP23	CP24	CP25	CP26	CP27	CP28	CP29
CP1	0	1	2	2	4	4	3	3	4	4	5	5	4	4	5	4	4	5	3	3	3	2	3	3	2	3	2	2	2
CP2	7	0	2	2	4	4	3	3	4	4	5	5	4	4	5	4	4	5	3	3	3	3	3	3	3	3	2	3	3
CP3	7	2	0	2	4	4	3	3	4	4	4	4	4	4	5	4	4	4	3	2	2	2	3	3	3	3	2	2	2
CP4	6	3	2	0	3	3	2	2	3	3	4	4	3	3	4	3	3	4	2	2	2	1	1	2	2	3	2	2	2
CP5	9	5	5	3	0	5	4	3	5	5	6	5	5	5	6	5	5	5	3	3	3	3	3	3	5	5	4	4	4
CP6	8	5	4	3	5	0	3	3	4	4	5	4	4	4	5	4	4	5	3	3	3	3	3	3	4	5	4	4	4
CP7	8	4	3	3	4	3	0	2	3	3	4	3	3	3	4	3	3	4	2	2	2	2	2	2	3	4	3	3	3
CP8	8	4	3	2	3	3	2	0	2	2	3	3	3	2	3	2	2	3	1	2	2	2	2	2	3	4	3	3	3
CP9	9	5	4	3	4	4	4	2	0	2	3	3	2	2	3	2	2	3	2	3	3	3	3	3	4	5	4	4	4
CP10	9	5	5	4	5	5	4	3	3	0	2	2	1	2	3	2	2	2	2	3	3	3	4	4	5	5	4	5	4
CP11	9	6	5	4	5	5	4	3	3	1	0	2	1	2	3	3	3	3	2	4	4	4	4	4	5	6	5	5	5
CP12	9	5	5	4	5	5	4	3	3	1	2	0	1	2	3	2	2	2	2	3	3	3	4	4	5	5	4	5	5
CP13	9	5	4	3	4	4	4	3	2	1	1	1	0	1	2	2	2	2	2	3	3	3	3	4	5	5	4	4	4
CP14	9	5	4	3	4	4	3	3	2	2	2	2	2	0	1	2	2	2	2	3	3	3	3	4	5	5	4	4	4
CP15	10	6	5	4	5	5	4	4	3	2	3	3	2	1	0	3	3	3	3	4	4	4	4	4	5	6	5	5	5
CP16	8	4	4	2	3	4	3	2	2	3	3	3	3	2	3	0	0	0	1	2	2	2	2	2	4	4	3	3	3
CP17	8	4	4	2	3	4	3	2	2	3	3	3	3	2	3	0	0	3	1	2	2	2	2	2	4	4	3	3	3
CP18	9	6	5	2	5	5	4	3	3	2	3	2	2	2	3	3	3	0	3	4	4	4	4	4	5	6	4	5	5
CP19	7	4	3	2	3	3	2	1	2	2	3	2	2	2	3	2	2	2	0	2	2	2	2	2	3	3	3	3	3
CP20	6	3	2	2	3	3	2	2	3	3	3	3	3	3	4	3	3	3	2	0	1	1	1	1	2	3	2	2	2
CP21	7	3	3	2	3	3	2	2	3	3	4	3	3	3	4	3	3	3	2	1	0	1	1	2	2	3	2	2	2
CP22	6	3	2	2	3	3	2	2	3	3	3	3	3	3	4	3	3	3	2	1	1	0	1	1	2	2	1	1	1
CP23	6	3	2	2	3	3	3	2	3	3	4	4	3	3	4	3	3	4	2	1	2	1	0	1	2	3	2	2	2
CP24	6	3	2	2	3	3	3	2	3	3	4	5	3	3	4	3	3	4	2	1	2	1	1	0	2	3	2	2	2
CP25	7	3	3	2	4	5	4	3	5	4	5	5	4	5	5	4	5	5	3	3	3	3	3	3	0	2	2	2	2
CP26	7	4	3	3	5	5	4	4	5	5	5	5	5	5	5	5	5	5	3	3	3	3	3	4	2	0	2	2	2
CP27	6	3	3	3	5	5	4	4	5	5	5	5	5	5	6	5	5	5	4	3	4	3	4	4	3	2	0	2	2
CP28	6	2	2	3	5	5	4	4	5	5	5	5	5	5	6	5	5	5	4	3	4	3	4	4	3	4	3	0	1
CP29	6	1	2	3	5	5	4	4	5	5	5	5	5	5	6	5	5	5	4	3	3	3	4	4	3	4	3	1	0

Figure 4.6: Excel Data of Time Taken Between Checkpoints

Based on the data I collected from Universiti Teknikal Malaysia Melaka (UTeM), I compiled three comprehensive files to serve as input for my GUI application. The first file is a detailed list of checkpoint names paired with their respective numbers, allowing for precise identification and organization of locations. The second file documents the distances between each checkpoint, providing a clear understanding of the spatial relationships and travel requirements. The third file includes the time taken to travel between these checkpoints, which adds a practical dimension for planning and analysis. To ensure accuracy and reliability, the distance and time data were sourced directly from Google Maps. These files form a structured and well-organized dataset, enabling the GUI to function effectively for route optimization.

4.4 Results and Analysis


MATLAB App

SECURITY PATROL ROUTES AND CHECKPOINTS OPTIMIZATION WITH MATLAB TO MINIMIZE BLIND SPOTS AND ENHANCE THE SAFETY

Checkpoint Names File: C:\Users\Yassvind Browse

Distance Matrix File: C:\Users\Yassvind Browse

Time Taken File: C:\Users\Yassvind Browse

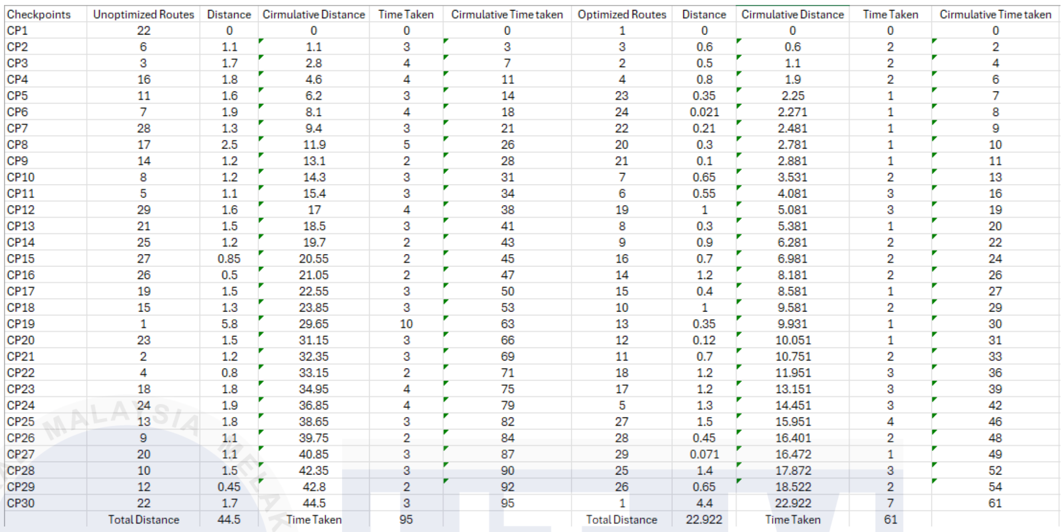
Compute Path | Reset | Export

Results

```
Path (Numeric): CP1 CP3 CP2 CP4 CP23 CP24 CP22 CP20 CP21 CP7
CP6 CP19 CP8 CP9 CP16 CP14 CP15 CP10 CP13 CP12 CP11 CP18
CP17 CP5 CP27 CP28 CP29 CP25 CP26 CP1
Path (Names): POS A KAMPUS INDUK -> FTMK -> PEJABAT
KESELAMATAN -> PKU -> PBB -> LAMAN HIKMAH -> PPP -> MASJID
-> PPPK -> FTKEK -> KOLEJ KEDIAMAN LESTARI -> HEPA (ATM) ->
FTKE -> PUSAT SUKAN -> CANSELORI ARAS 1 -> PUSAT KOKU ->
KDK -> BANGUNAN TNB -> PEJABAT PEMBANGUNAN -> PPKU ->
RUMAH TUNGGU SEDIA -> PEJABAT NC -> LOBBY CANSELORI ->
KOLEJ KEDIAMAN SATRIA -> FTKIP -> KOMPLEKS PENYELIDIKAN &
INOVASI -> SEKOLAH PENGAJIAN SISWAZAH -> DEWAN CANSELOR
-> FTKM -> POS A KAMPUS INDUK
Total Distance: 22.9220 KM
```

Figure 4.7: MATLAB Graphical User Interface

The results of the MATLAB optimization for security patrol routes and checkpoints demonstrate an efficient path for minimizing total distance and time while ensuring comprehensive coverage of the checkpoints. The computed path starts and ends at the same checkpoint, ensuring a closed-loop route suitable for patrol purposes. The total distance covered is 22.9220 KM, and the patrol completes within approximately 1 hour and 1 minute, indicating that the algorithm successfully optimized the route to enhance safety and minimize blind spots. This optimization can improve patrol efficiency and reliability in practical applications.



Checkpoints	Unoptimized Routes	Distance	Cirmulative Distance	Time Taken	Cirmulative Time taken	Optimized Routes	Distance	Cirmulative Distance	Time Taken	Cirmulative Time taken
CP1	22	0	0	0	0	1	0	0	0	0
CP2	6	1.1	1.1	3	3	3	0.6	0.6	2	2
CP3	3	1.7	2.8	4	7	2	0.5	1.1	2	4
CP4	16	1.8	4.6	4	11	4	0.8	1.9	2	6
CP5	11	1.6	6.2	3	14	23	0.35	2.25	1	7
CP6	7	1.9	8.1	4	18	24	0.021	2.271	1	8
CP7	28	1.3	9.4	3	21	22	0.21	2.481	1	9
CP8	17	2.5	11.9	5	26	20	0.3	2.781	1	10
CP9	14	1.2	13.1	2	28	21	0.1	2.881	1	11
CP10	8	1.2	14.3	3	31	7	0.65	3.531	2	13
CP11	5	1.1	15.4	3	34	6	0.55	4.081	3	16
CP12	29	1.6	17	4	38	19	1	5.081	3	19
CP13	21	1.5	18.5	3	41	8	0.3	5.381	1	20
CP14	25	1.2	19.7	2	43	9	0.9	6.281	2	22
CP15	27	0.85	20.55	2	45	16	0.7	6.981	2	24
CP16	26	0.5	21.05	2	47	14	1.2	8.181	2	26
CP17	19	1.5	22.55	3	50	15	0.4	8.581	1	27
CP18	15	1.3	23.85	3	53	10	1	9.581	2	29
CP19	1	5.8	29.65	10	63	13	0.35	9.931	1	30
CP20	23	1.5	31.15	3	66	12	0.12	10.051	1	31
CP21	2	1.2	32.35	3	69	11	0.7	10.751	2	33
CP22	4	0.8	33.15	2	71	18	1.2	11.951	3	36
CP23	18	1.8	34.95	4	75	17	1.2	13.151	3	39
CP24	24	1.9	36.85	4	79	5	1.3	14.451	3	42
CP25	13	1.8	38.65	3	82	27	1.5	15.951	4	46
CP26	9	1.1	39.75	2	84	28	0.45	16.401	2	48
CP27	20	1.1	40.85	3	87	29	0.071	16.472	1	49
CP28	10	1.5	42.35	3	90	25	1.4	17.872	3	52
CP29	12	0.45	42.8	2	92	26	0.65	18.522	2	54
CP30	22	1.7	44.5	3	95	1	4.4	22.922	7	61
	Total Distance	44.5	Time Taken	95		Total Distance	22.922	Time Taken	61	

Figure 4.8: Result Excel Data

Then I took those results and organized them neatly in an Excel sheet, comparing the unoptimized and optimized routes according to distance and time. Further, I created a cumulative distance-time for both routes and plotted graphs depicting the difference between them. From the graph, the difference will evidently show just how much more efficient the routes get after the process of optimization, it just depicts the process reducing the distance traveled hence time taken in the simplest yet understandable way possible.

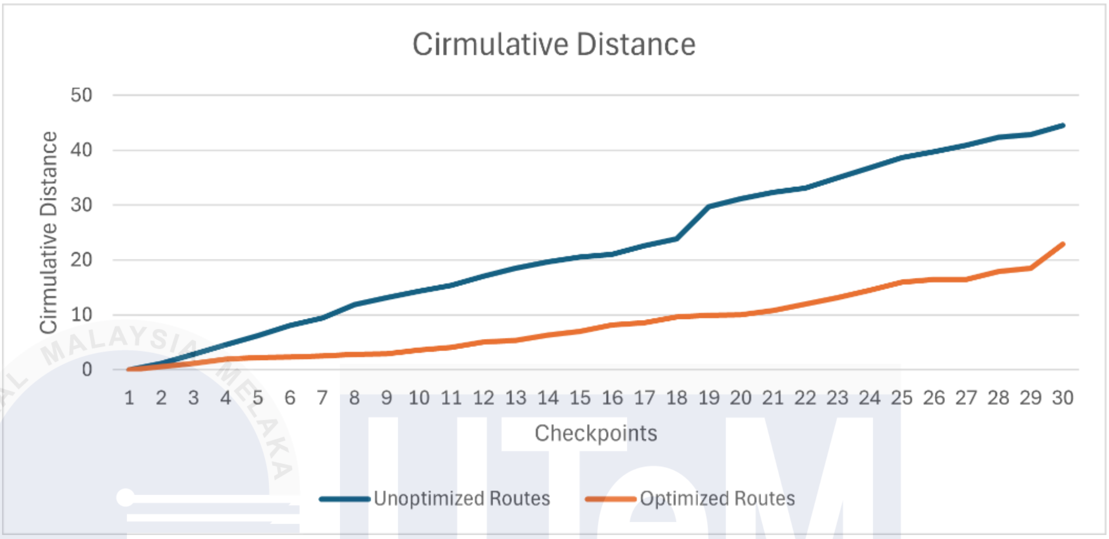


Figure 4.9: Optimized VS Unoptimized Cumulative Distance

This graph shows how the cumulative distance increases as checkpoints are visited. The optimized route (orange line) grows much slower compared to the unoptimized route (blue line). This clearly demonstrates that the optimized route significantly reduces the total distance covered, making the patrol much more efficient.

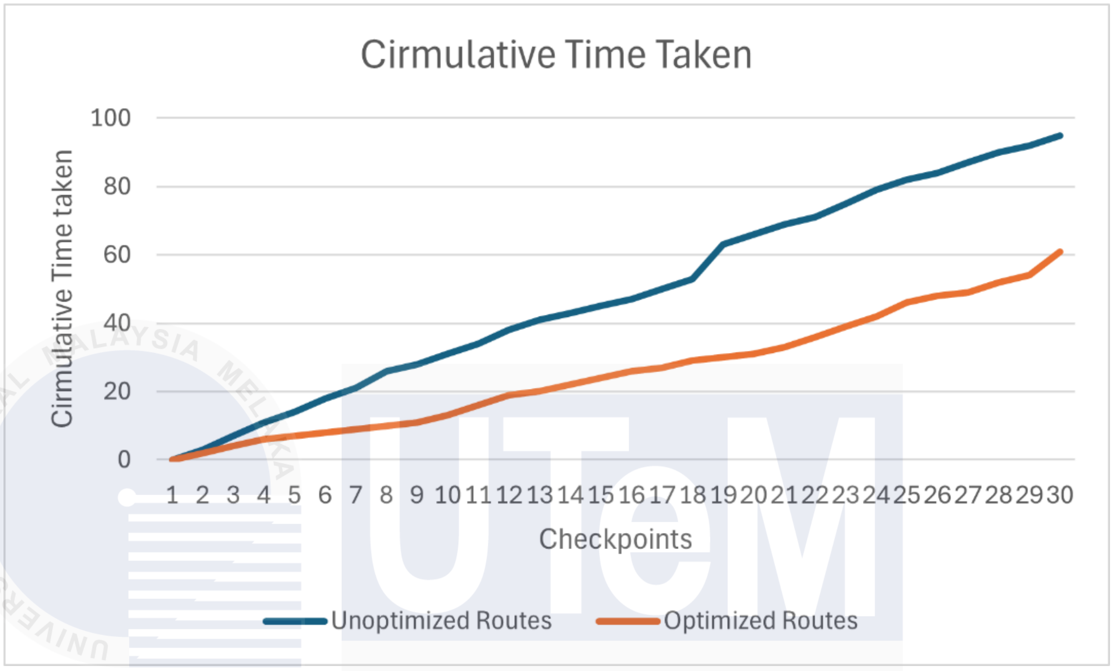


Figure 4.10: Optimized Vs Unoptimized Cumulative Time Taken

This graph tracks the cumulative time spent on both routes. It’s easy to see that the optimized route (orange) consistently takes less time than the unoptimized one (blue). This shows how optimization not only saves distance but also makes the patrol quicker.

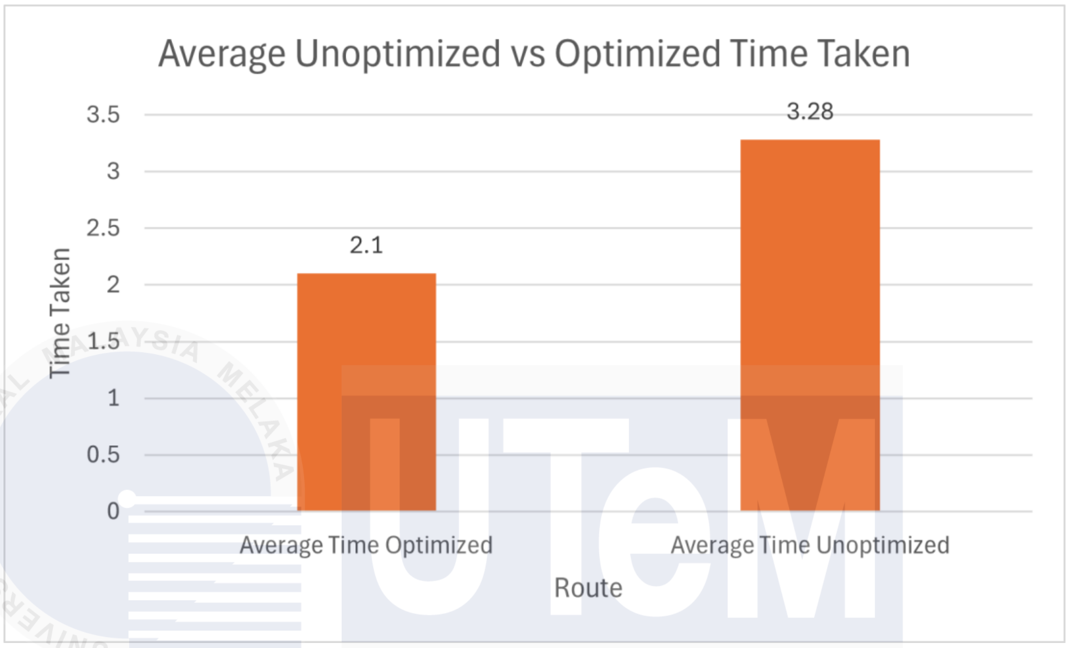


Figure 4.11: Optimized Vs Unoptimized Average Time Taken

This bar chart highlights the average time spent per checkpoint. The optimized route only takes 2.1 minutes, while the unoptimized route takes 3.28 minutes. That's about a 36% reduction in time, which is a significant improvement for any security patrol operation.

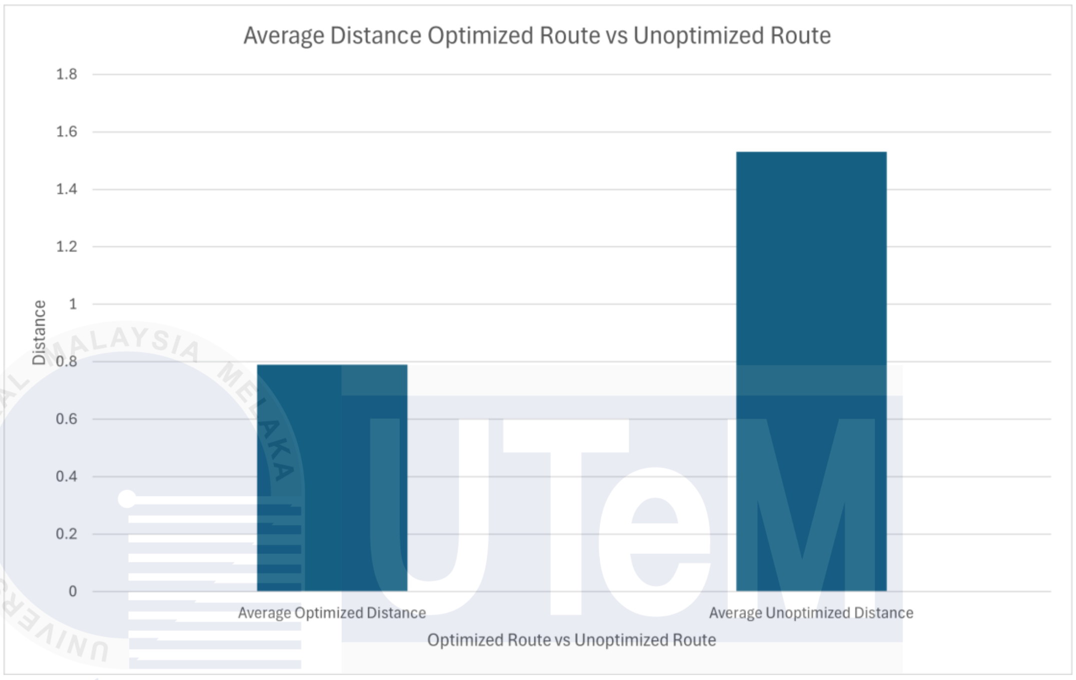


Figure 4.12: Optimized Vs Unoptimized Average Distance Per Checkpoint

This graph focuses on the average distance per checkpoint. The optimized route averages 0.8KM, while the unoptimized route is much higher at 1.6KM. This means the optimized route cuts the distance by half, saving resources and effort.

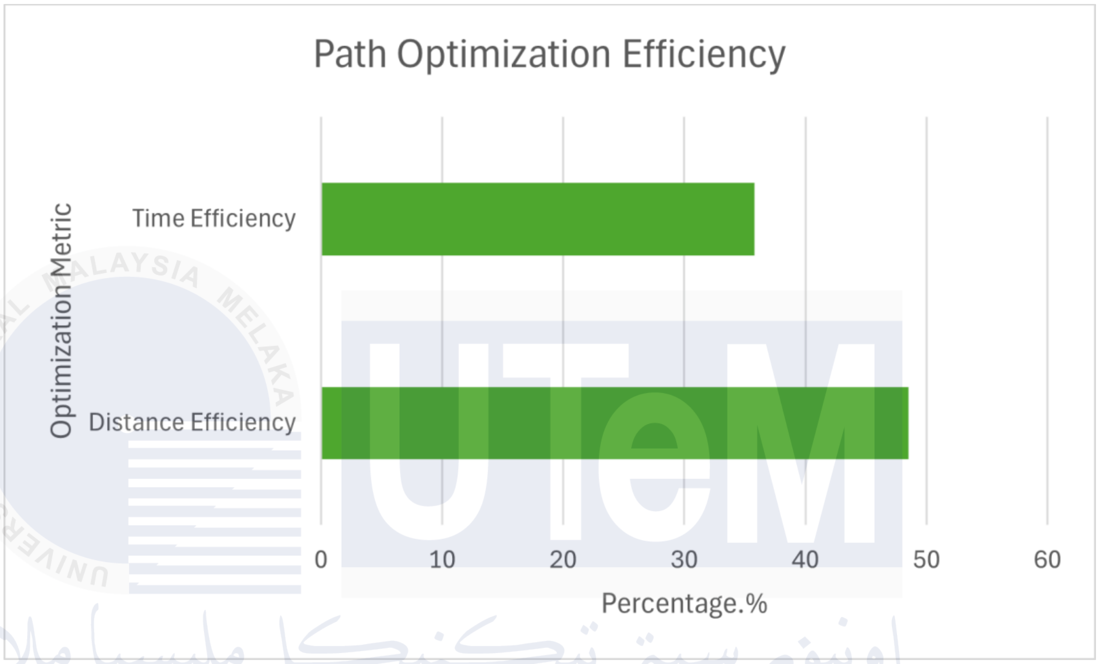


Figure 4.13: Path Optimization Efficiency

This chart gives a clear summary of how much better the optimized route is. It achieves 45% distance efficiency and 38%-time efficiency compared to the unoptimized route. These numbers show the practical impact of using the optimized path.

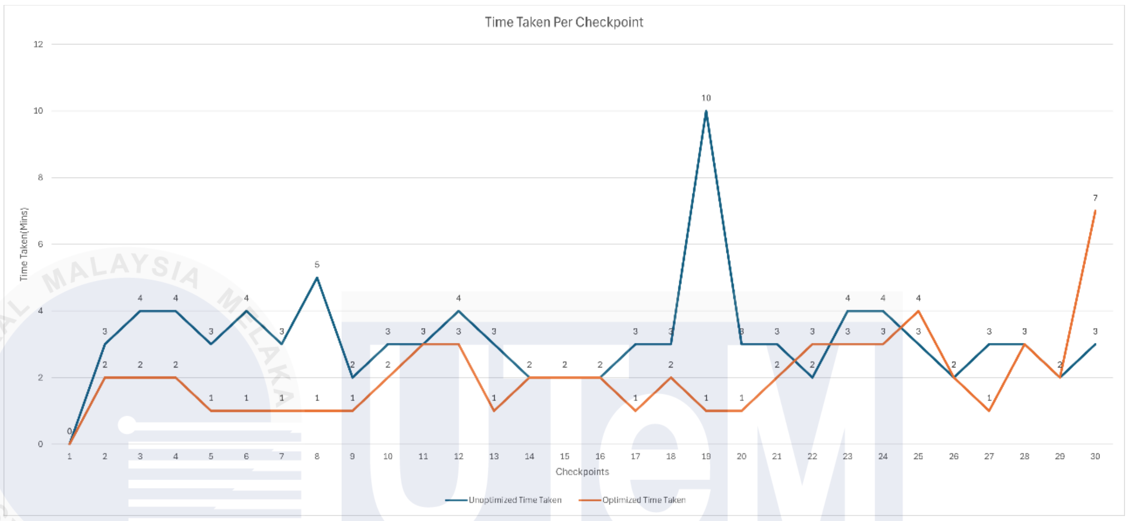


Figure 4.14: Time Taken Per Checkpoints

This graph looks at how much time is taken at each checkpoint individually. The optimized route (orange) is almost always faster than the unoptimized one (blue).

The spikes in the blue line highlight inefficiencies in the unoptimized route that the optimization process avoids.

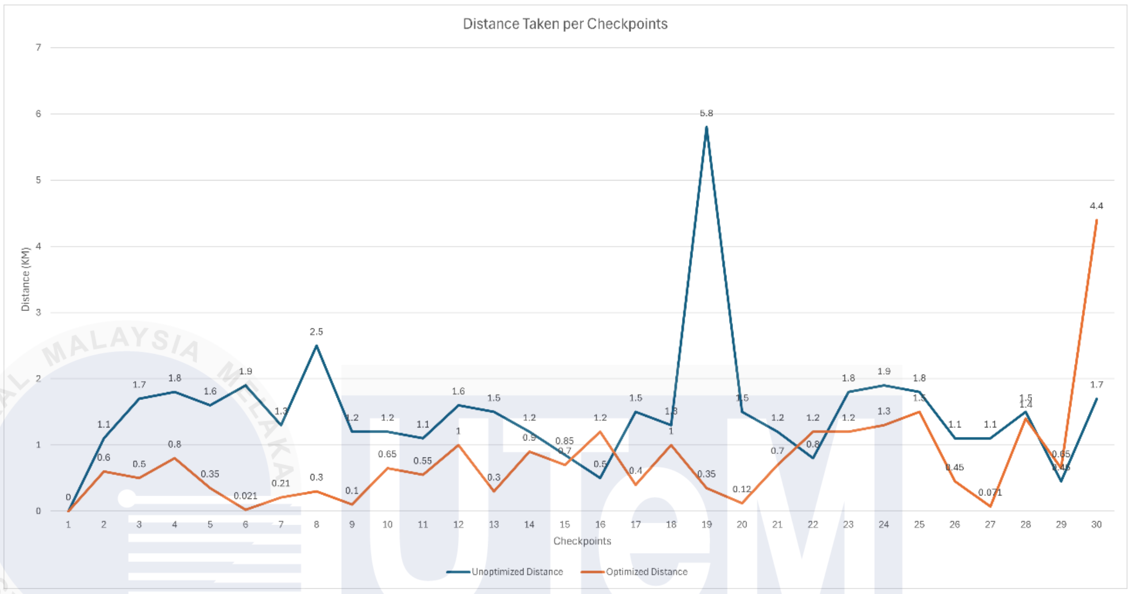


Figure 4.15: Distance Taken Per Checkpoints

This graph shows the distance covered for each checkpoint. Just like the time graph, the optimized route (orange) consistently stays below the unoptimized route (blue). The peaks in the blue line emphasize how much extra travel the unoptimized route requires, which is eliminated in the optimized one.

4.5 Summary

Chapter 4 presents the results and analysis of the MATLAB-based program developed for route and checkpoint optimization in security patrols. It begins with data collection on latitude and longitude coordinates from 29 checkpoints, which were gathered and configured. Using the collected data, distances and travel times between all checkpoints were calculated and organized into a 29×29 matrix as input for the Nearest Neighbor Algorithm.

It is the shortest, most efficient route computed by the algorithm, which includes a total distance of 22.9220 KM and can be traveled in about 1 hour and 1 minute. The best route that could provide complete coverage, allowing the least number of blind spots. Further analysis was made by comparing unoptimized and optimized routes regarding distance in time, presenting the efficiency enhancement on graphs. This chapter shows the practicality of the optimization process, which allows the possibility to help improve security by reducing travel time and distance of patrol routes.

CHAPTER 5
CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

A program in MATLAB will showcase the development and implementation of an optimal route of security patrols at UTeM together with security patrol checkpoints. The aim would be to make surveillance efficient while the blind spots will be minimal for maximum area coverage, mainly critical ones. Advanced data gathering, mathematical modeling, and optimization by algorithms together created results; thus, meeting the requirements on presenting a working framework that offers improvements in the security aspects.

Precise data collection with GPS technology and configuration of NFC tags for easy identification began the project, naming 29 checkpoints. This list was organized in Excel files for easy handling and was used as input for the optimization. To perform route calculations with precision, it was necessary to create two symmetric matrices-one of distances and one of travel times-both accurate and valid.

By implementing the Nearest Neighbor Algorithm, the project had successfully calculated the shortest and most efficient patrol route that covers the total distance of 22.9220 km in approximately 1 hour and 1 minute. The optimized route hugely reduced travel time and avoided many detours yet still be able to cover all checkpoints. Comparisons between unoptimized and optimized routes were shown through graphs for better visualization of how much more efficient the algorithm made the routes.

Simulation and testing with MATLAB proved effective with the proposed solution. Dynamic prioritization, variable patrol frequencies, and runtime adjustment are just some of the scenarios that prove the system can adapt to changing security needs. It was apparent from the visual output and analysis that the optimized routes save time and energy and improve safety due to previously unidentified blind spots.

In general, this work provides a system approach to the problem of security patrol planning that can be easily adapted for different environments. It integrates computational geometry, graph theory, and optimization techniques in order to obtain an efficient and scalable practical solution. The results show the possibilities of using MATLAB and data-driven algorithms to achieve the greatest possible patrol efficiency with minimum vulnerability while responding dynamically to security challenges. This work forms a very strong foundation for further advances in security management and optimization.

5.2 Commercialize

This project has immense potential for practical applications across various domains. For example, it can be deployed to enhance condominium security patrolling, ensuring that guards follow optimised routes to cover all areas efficiently, reducing the risk of blind spots. It is equally beneficial for police car patrolling, where dynamic route adjustments can be utilised to respond to real-time incidents or high-crime areas, improving public safety. In school security patrolling, the system can help security teams monitor critical zones such as entrances, playgrounds, and parking lots to ensure the safety of students and staff. Additionally, the project can support company operational line patrolling, where routine checks of equipment and facilities can be scheduled and optimized to minimize downtime and enhance operational efficiency. These diverse use cases highlight the versatility of the system in addressing safety, security, and operational challenges across different industries.

5.3 Future Works

The next development in this project can focus on the improvement of its functionality and adaptability. Dynamic route adjustment is one major improvement, updating patrol routes in real-time based on factors such as traffic, weather conditions, or security alerts. This ensures that routes are efficient and responsive to changing conditions. Another advancement is the integration with Geographic Information Systems (GIS), which would enable the visualization of patrol routes and checkpoints on a map, improving situational awareness and decision-making. Additionally, optimization for multiple patrol units can be explored, ensuring that teams operate simultaneously with coordinated routes to maximize coverage and minimize overlaps. Finally, the system can be enhanced by incorporating algorithms that identify and prioritize areas with limited patrol coverage so that no critical zones are overlooked. All these enhancements would collectively improve the overall effectiveness and adaptability of the security patrol system.

REFERENCES

[1] Y. Fu, Y. Zeng, D. Wang, H. Zhang, Y. Gao, and Y. Liu, "Research on Route Optimization Based on Multiagent and Genetic Algorithm for Community Patrol," in *2020 International Conference on Urban Engineering and Management Science (ICUEMS)*, IEEE, Apr. 2020, pp. 112–116. doi: 10.1109/ICUEMS50872.2020.00034.

[2] M. Dewinter, C. Vandeviver, T. Vander Beken, and F. Witlox, "Analysing the police patrol routing problem: A review," *ISPRS Int J Geoinf*, vol. 9, no. 3, 2020, doi: 10.3390/ijgi9030157.

[3] D. Zhao, H. Yu, X. Fang, L. Tian, and P. Han, "A Path Planning Method Based on Multi- Objective Cauchy Mutation Cat Swarm Optimization Algorithm for Navigation System of Intelligent Patrol Car," *IEEE Access*, vol. 8, pp. 151788–151803, 2020, doi: 10.1109/ACCESS.2020.3016565.

[4] Y. Jiang, H. Li, B. Feng, Z. Wu, S. Zhao, and Z. Wang, "Street Patrol Routing Optimization in Smart City Management Based on Genetic Algorithm: A Case in Zhengzhou, China," *ISPRS Int J Geoinf*, vol. 11, no. 3, p. 171, Mar. 2022, doi: 10.3390/ijgi11030171.

[5] Qiu Mingyue and Zhang Xueying, "Determining Accurate Patrol Routes Using Genetic Algorithm and Ant Colony," *Automatic Control and Computer Sciences*, vol. 57, no. 4, pp. 337–347, Aug. 2023, doi: 10.3103/S0146411623040065.

[6] O. Ozkan and M. Kaya, "UAV routing with genetic algorithm based matheuristic for border security missions," *An International Journal of Optimization and Control: Theories & Applications (IJOCTA)*, vol. 11, no. 2, pp. 128–138, Apr. 2021, doi: 10.11121/ijocta.01.2021.001023.

[7] C. Guevara, J. Jadán, C. Zapata, L. Martínez, J. Pozo, and E. Manjarres, "Model of Dynamic Routes for Intelligent Police Patrolling," MDPI AG, Oct. 2018, p. 1214. doi: 10.3390/proceedings2191214.

[8] Y. Gao, Z. Dai, and J. Yuan, "A Multiobjective Hybrid Optimization Algorithm for Path Planning of Coal Mine Patrol Robot," *Comput Intell Neurosci*, vol. 2022, pp. 1–10, Jun. 2022, doi: 10.1155/2022/9094572.

[9] C. Guevara and M. Santos, "Smart Patrolling Based on Spatial-Temporal Information Using Machine Learning," *Mathematics*, vol. 10, no. 22, p. 4368, Nov. 2022, doi: 10.3390/math10224368.

[10] Y. Jiang, S. Zhao, H. Li, Y. Qin, and X. Yang, "A hybrid spectral clustering simulated annealing algorithm for the street patrol districting problem," *Complex & Intelligent Systems*, vol. 9, no. 2, pp. 1791–1807, Apr. 2023, doi: 10.1007/s40747-022-00880-w.

[11] H. Liu, Y. Sun, N. Pan, Q. Chen, X. Guo, and D. Pan, "Multi-UAV Cooperative Task Planning for Border Patrol based on Hierarchical Optimization," *Journal of Imaging Science and Technology*, vol. 65, no. 4, pp. 040402-1-040402–8, Jul. 2021, doi: 10.2352/J.ImagingSci.Technol.2021.65.4.040402.

[12] T. Alam, Md. M. Rahman, P. Carrillo, L. Bobadilla, and B. Rapp, "Stochastic Multi-Robot Patrolling with Limited Visibility," *J Intell Robot Syst*, vol. 97, no. 2, pp. 411–429, Feb. 2020, doi: 10.1007/s10846-019-01039-5.

[13] M. Peñacoba, J. E. Sierra-García, M. Santos, and I. Mariolis, "Path Optimization Using Metaheuristic Techniques for a Surveillance Robot," *Applied Sciences*, vol. 13, no. 20, p. 11182, Oct. 2023, doi: 10.3390/app132011182.

[14] K. Shi *et al.*, “Path Planning Optimization of Intelligent Vehicle Based on Improved Genetic and Ant Colony Hybrid Algorithm,” *Front Bioeng Biotechnol*, vol. 10, Jul. 2022, doi: 10.3389/fbioe.2022.905983.
[15] C. Jose and K. S. Vijula Grace, “Optimization based routing model for the dynamic path planning of emergency vehicles,” *Evol Intell*, vol. 15, no. 2, pp. 1425–1439, Jun. 2022, doi: 10.1007/s12065-020-00448-y.
[16] O. Ozkan, “Optimization of the distance-constrained multi-based multi-UAV routing problem with simulated annealing and local search-based matheuristic to detect forest fires: The case of Turkey,” *Appl Soft Comput*, vol. 113, p. 108015, Dec. 2021, doi: 10.1016/j.asoc.2021.108015.
[17] L. Huang, M. Zhou, H. Han, S. Wang, and A. Albeshri, “Learning-Inspired Immune Algorithm for Multiobjective-Optimized Multirobot Maritime Patrolling,” *IEEE Internet Things J*, vol. 11, no. 6, pp. 9870–9881, Mar. 2024, doi: 10.1109/JIOT.2023.3326567.
[18] N. Rathore, P. K. Jain, and M. Parida, “A MATLAB-Based Application to Solve Vehicle Routing Problem Using GA,” 2020, pp. 285–298. doi: 10.1007/978-981-32-9487-5_22.
[19] X. Sun, J. Wang, W. Wu, and W. Liu, “Genetic Algorithm for Optimizing Routing Design and Fleet Allocation of Freeway Service Overlapping Patrol,” *Sustainability*, vol. 10, no. 11, p. 4120, Nov. 2018, doi: 10.3390/su10114120.
[20] Y. Nan, Q. Jian-Hua, Z. Xue-Qiong, and W. Hong-Chang, “Dynamic Path Planning Based on Improved Particle Filter Optimisation for Patrol Robots,” in *2022 7th Asia Conference on Power and Electrical Engineering (ACPEE)*, IEEE, Apr. 2022, pp. 1898–1903. doi: 10.1109/ACPEE53904.2022.9784034.
[21] J. Verma and V. Ranga, “A Decentralized Coordination Algorithm for Patrolling and Target Tracking in Internet of Robotic Things using Dynamic Waypoints and Self-triggered Communication,” *International Journal of Computing and Digital Systems*, vol. 12, no. 4, pp. 992–1003, Oct. 2022, doi: 10.12785/ijcds/120180.
[22] X. Yuan and C. Shi, “Research on tourism individualized route management based on intelligent optimization algorithm,” *Journal of Computational Methods in Sciences and Engineering*, vol. 19, no. 4, pp. 1065–1072, Nov. 2019, doi: 10.3233/JCM-193821.

APPENDICES

Coding

```
classdef PSM_BDP < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        Label                           matlab.ui.control.Label
        ResultsTextArea                 matlab.ui.control.TextArea
        ResultsTextAreaLabel            matlab.ui.control.Label
        BrowseButton_3                  matlab.ui.control.Button
        BrowseButton_2                  matlab.ui.control.Button
        BrowseButton                    matlab.ui.control.Button
        TimeTakenFileEditField          matlab.ui.control.EditField
        TimeTakenFileEditFieldLabel     matlab.ui.control.Label
        DistanceMatrixFileEditField     matlab.ui.control.EditField
        DistanceMatrixFileEditFieldLabel  matlab.ui.control.Label
        CheckpointNamesFileEditField    matlab.ui.control.EditField
        CheckpointNamesFileEditFieldLabel  matlab.ui.control.Label
        ExportButton                    matlab.ui.control.Button
        ResetButton                     matlab.ui.control.Button
        ComputePathButton               matlab.ui.control.Button
    end


    properties (Access = private)
       distance_matrix    % Stores the distance matrix
       time_matrix        % Stores the time taken matrix
       checkpoint_data    % Stores the checkpoint names and numbers
       total_path         % Stores the computed path
       total_distance     % Stores the total distance
       total_time         % Stores the total time
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: BrowseButton
        function BrowseButtonPushed(app, event)

            [file, path] = uigetfile('*.xlsx', 'Select Checkpoint Names File');
            if file
                app.CheckpointNamesFileEditField.Value = fullfile(path, file);
            end
        end

        % Button pushed function: ComputePathButton
        function ComputePathButtonPushed(app, event)
             try
                 % Load data from input files
                 app.distance_matrix =
xlsread(app.DistanceMatrixFileEditField.Value);
```

```
                 app.time_matrix = xlsread(app.TimeTakenFileEditField.Value);
                 app.checkpoint_data = 
readtable(app.CheckpointNamesFileEditField.Value, 'VariableNamingRule', 
'preserve');

                 % Validate loaded data
                 if isempty(app.distance_matrix) || isempty(app.time_matrix) || 
isempty(app.checkpoint_data)
                     error('One or more input files are empty or invalid.');
                 end

                 % Extract checkpoint numbers and names
                 checkpoint_numbers = app.checkpoint_data{:, 1};
                 checkpoint_names = app.checkpoint_data{:, 2};

                 % Set the starting checkpoint to the first checkpoint
                 start_checkpoint = 1;  % Assuming the first checkpoint in the 
list

                 % Initialize variables
                 n = size(app.distance_matrix, 1);
                 visited_checkpoints = zeros(1, n);
                 app.total_path = [];
                 app.total_distance = 0;
                 app.total_time = 0;
                 current_checkpoint = start_checkpoint;

                 % Loop to compute the shortest path
                 while sum(visited_checkpoints) < n
                     visited_checkpoints(current_checkpoint) = 1;
                     app.total_path = [app.total_path, current_checkpoint];
                     min_distance = inf;
                     next_checkpoint = -1;

                     for i = 1:n
                         if ~visited_checkpoints(i) && 
app.distance_matrix(current_checkpoint, i) > 0 ...
                                 && app.distance_matrix(current_checkpoint, i) 
< min_distance
                             min_distance = 
app.distance_matrix(current_checkpoint, i);
                             next_checkpoint = i;
                         end
                     end

                     if next_checkpoint == -1
                         break;
                     end

                     app.total_distance = app.total_distance + min_distance;
                     app.total_time = app.total_time + 
app.time_matrix(current_checkpoint, next_checkpoint);
                     current_checkpoint = next_checkpoint;
                 end

                 % Add return to the starting checkpoint
                 if current_checkpoint ~= start_checkpoint
                     app.total_path = [app.total_path, start_checkpoint];
```

```
                    app.total_distance = app.total_distance +
app.distance_matrix(current_checkpoint, start_checkpoint);
                    app.total_time = app.total_time +
app.time_matrix(current_checkpoint, start_checkpoint);
                end


                % Convert checkpoint numbers to numeric if necessary
                if iscell(checkpoint_numbers)
                    numeric_path = cellfun(@str2double,
checkpoint_numbers(app.total_path)); % Convert to numeric
                else
                    numeric_path = checkpoint_numbers(app.total_path); % Use
directly if already numeric
                end

                % Convert checkpoint names to strings if necessary
                if iscell(checkpoint_names)
                    path_names = checkpoint_names(app.total_path); % Extract
as cell array of strings
                else
                    path_names = cellstr(checkpoint_names(app.total_path)); %
Convert to cell array of strings
                end

                % Display results
                % Format numeric path as "CP1, CP2, ..."
                    formatted_numeric_path = sprintf('CP%d ', numeric_path);
                    formatted_numeric_path = strtrim(formatted_numeric_path);
% Remove trailing space

                    % Display results
                    % Convert total time to hours and minutes
                    hours = floor(app.total_time / 60);  % Extract hours
                    minutes = mod(app.total_time, 60);  % Extract remaining
minutes


                    % Display results
                    app.ResultsTextArea.Value = sprintf(['Path (Numeric):
%s\n' ...
                        'Path (Names): %s\n' ...
                        'Total Distance: %.4f KM\n' ...
                        'Total Time: %d hours %d minutes'], ...
                        formatted_numeric_path, strjoin(path_names, ' -> '),
app.total_distance, hours, minutes);


            catch ME

                % Handle errors and display a message
                uialert(app.UIFigure, sprintf('Error: %s', ME.message),
'Error');
            end
        end

        % Button pushed function: ResetButton
        function ResetButtonPushed(app, event)

            % Clear all inputs
            app.DistanceMatrixFileEditField.Value = '';
            app.TimeTakenFileEditField.Value = '';
            app.CheckpointNamesFileEditField.Value = '';

            % Clear results
            app.ResultsTextArea.Value = '';

           % Reset stored properties
           app.distance_matrix = [];
           app.time_matrix = [];
           app.checkpoint_data = [];
           app.total_path = [];
           app.total_distance = 0;
           app.total_time = 0;

        end

        % Button pushed function: ExportButton
        function ExportButtonPushed(app, event)

            [file, path] = uiputfile('*.txt', 'Save Results As');
            if file
                % Open the file for writing
                fileID = fopen(fullfile(path, file), 'w');

                % Write results to the file
                fprintf(fileID, 'Path (Numeric): %s\n',
num2str(app.total_path));
                fprintf(fileID, 'Path (Names): %s\n',
strjoin(app.checkpoint_data{app.total_path, 2}, ' -> '));
                fprintf(fileID, 'Total Distance: %.4f KM\n',
app.total_distance);
                fprintf(fileID, 'Total Time: %d minutes\n', app.total_time);
                % Close the file
                fclose(fileID);

                % Notify user of success
                uialert(app.UIFigure, 'Results exported successfully!',
'Success');
            end
        end

        % Button pushed function: BrowseButton_2
        function BrowseButton_2Pushed(app, event)

            [file, path] = uigetfile('*.xlsx', 'Select Distance Matrix File');
            if file
                app.DistanceMatrixFileEditField.Value = fullfile(path, file);
            end
        end

        % Button pushed function: BrowseButton_3
        function BrowseButton_3Pushed(app, event)

                [file, path] = uigetfile('*.xlsx', 'Select Time Taken File');
                if file
                    app.TimeTakenFileEditField.Value = fullfile(path, file);
                end
        end
```

```
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Color = [0.9412 0.8157 0.5255];
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'MATLAB App';

            % Create ComputePathButton
            app.ComputePathButton = uibutton(app.UIFigure, 'push');
            app.ComputePathButton.ButtonPushedFcn = createCallbackFcn(app,
@ComputePathButtonPushed, true);
            app.ComputePathButton.FontName = 'Times New Roman';
            app.ComputePathButton.FontSize = 14;
            app.ComputePathButton.FontColor = [0.149 0.149 0.149];
            app.ComputePathButton.Position = [81 227 100 25];
            app.ComputePathButton.Text = 'Compute Path';

            % Create ResetButton
            app.ResetButton = uibutton(app.UIFigure, 'push');
            app.ResetButton.ButtonPushedFcn = createCallbackFcn(app,
@ResetButtonPushed, true);
            app.ResetButton.FontName = 'Times New Roman';
            app.ResetButton.FontSize = 14;
            app.ResetButton.FontColor = [0.149 0.149 0.149];
            app.ResetButton.Position = [197 227 100 25];
            app.ResetButton.Text = 'Reset';

            % Create ExportButton
            app.ExportButton = uibutton(app.UIFigure, 'push');
            app.ExportButton.ButtonPushedFcn = createCallbackFcn(app,
@ExportButtonPushed, true);
            app.ExportButton.FontName = 'Times New Roman';
            app.ExportButton.FontSize = 14;
            app.ExportButton.FontColor = [0.149 0.149 0.149];
            app.ExportButton.Position = [310 227 100 25];
            app.ExportButton.Text = 'Export';

            % Create CheckpointNamesFileEditFieldLabel
            app.CheckpointNamesFileEditFieldLabel = uilabel(app.UIFigure);
            app.CheckpointNamesFileEditFieldLabel.BackgroundColor = [0.2118
0.2941 0.4588];
            app.CheckpointNamesFileEditFieldLabel.HorizontalAlignment =
'right';
            app.CheckpointNamesFileEditFieldLabel.FontName = 'Times New Roman';
            app.CheckpointNamesFileEditFieldLabel.FontWeight = 'bold';
            app.CheckpointNamesFileEditFieldLabel.FontColor = [1 1 1];
            app.CheckpointNamesFileEditFieldLabel.Position = [23 369 128 22];
            app.CheckpointNamesFileEditFieldLabel.Text = 'Checkpoint Names
File:';

            % Create CheckpointNamesFileEditField
            app.CheckpointNamesFileEditField = uieditfield(app.UIFigure,
'text');
            app.CheckpointNamesFileEditField.FontName = 'Times New Roman';
            app.CheckpointNamesFileEditField.FontWeight = 'bold';
            app.CheckpointNamesFileEditField.FontColor = [1 1 1];
            app.CheckpointNamesFileEditField.BackgroundColor = [0.2118 0.2941
0.4588];
            app.CheckpointNamesFileEditField.Position = [166 369 100 22];

            % Create DistanceMatrixFileEditFieldLabel
            app.DistanceMatrixFileEditFieldLabel = uilabel(app.UIFigure);
            app.DistanceMatrixFileEditFieldLabel.BackgroundColor = [0.2118
0.2941 0.4588];
            app.DistanceMatrixFileEditFieldLabel.HorizontalAlignment = 'right';
            app.DistanceMatrixFileEditFieldLabel.FontName = 'Times New Roman';
            app.DistanceMatrixFileEditFieldLabel.FontWeight = 'bold';
            app.DistanceMatrixFileEditFieldLabel.FontColor = [1 1 1];
            app.DistanceMatrixFileEditFieldLabel.Position = [37 339 114 22];
            app.DistanceMatrixFileEditFieldLabel.Text = 'Distance Matrix
File:';

            % Create DistanceMatrixFileEditField
            app.DistanceMatrixFileEditField = uieditfield(app.UIFigure,
'text');
            app.DistanceMatrixFileEditField.FontName = 'Times New Roman';
            app.DistanceMatrixFileEditField.FontWeight = 'bold';
            app.DistanceMatrixFileEditField.FontColor = [1 1 1];
            app.DistanceMatrixFileEditField.BackgroundColor = [0.2118 0.2941
0.4588];
            app.DistanceMatrixFileEditField.Position = [166 339 100 22];

            % Create TimeTakenFileEditFieldLabel
            app.TimeTakenFileEditFieldLabel = uilabel(app.UIFigure);
            app.TimeTakenFileEditFieldLabel.BackgroundColor = [0.2118 0.2941
0.4588];
            app.TimeTakenFileEditFieldLabel.HorizontalAlignment = 'right';
            app.TimeTakenFileEditFieldLabel.FontName = 'Times New Roman';
            app.TimeTakenFileEditFieldLabel.FontWeight = 'bold';
            app.TimeTakenFileEditFieldLabel.FontColor = [1 1 1];
            app.TimeTakenFileEditFieldLabel.Position = [59 307 92 22];
            app.TimeTakenFileEditFieldLabel.Text = 'Time Taken File:';

            % Create TimeTakenFileEditField
            app.TimeTakenFileEditField = uieditfield(app.UIFigure, 'text');
            app.TimeTakenFileEditField.FontName = 'Times New Roman';
            app.TimeTakenFileEditField.FontWeight = 'bold';
            app.TimeTakenFileEditField.FontColor = [1 1 1];
            app.TimeTakenFileEditField.BackgroundColor = [0.2118 0.2941
0.4588];
            app.TimeTakenFileEditField.Position = [166 307 100 22];

            % Create BrowseButton
            app.BrowseButton = uibutton(app.UIFigure, 'push');
            app.BrowseButton.ButtonPushedFcn = createCallbackFcn(app,
@BrowseButtonPushed, true);
            app.BrowseButton.BackgroundColor = [0.3373 0.3294 0.5294];
            app.BrowseButton.FontColor = [1 0 0];
            app.BrowseButton.Position = [273 369 100 22];
            app.BrowseButton.Text = 'Browse';
            % Create BrowseButton_2
            app.BrowseButton_2 = uibutton(app.UIFigure, 'push');
            app.BrowseButton_2.ButtonPushedFcn = createCallbackFcn(app,
@BrowseButton_2Pushed, true);
            app.BrowseButton_2.BackgroundColor = [0.3373 0.3294 0.5294];
            app.BrowseButton_2.FontColor = [1 0 0];
            app.BrowseButton_2.Position = [274 339 100 22];
            app.BrowseButton_2.Text = 'Browse';

            % Create BrowseButton_3
            app.BrowseButton_3 = uibutton(app.UIFigure, 'push');
            app.BrowseButton_3.ButtonPushedFcn = createCallbackFcn(app,
@BrowseButton_3Pushed, true);
            app.BrowseButton_3.BackgroundColor = [0.3373 0.3294 0.5294];
            app.BrowseButton_3.FontColor = [1 0 0];
            app.BrowseButton_3.Position = [275 307 100 22];
            app.BrowseButton_3.Text = 'Browse';

            % Create ResultsTextAreaLabel
            app.ResultsTextAreaLabel = uilabel(app.UIFigure);
            app.ResultsTextAreaLabel.BackgroundColor = [0.7451 0.6196 1];
            app.ResultsTextAreaLabel.HorizontalAlignment = 'right';
            app.ResultsTextAreaLabel.FontWeight = 'bold';
            app.ResultsTextAreaLabel.Position = [108 188 48 22];
            app.ResultsTextAreaLabel.Text = 'Results';

            % Create ResultsTextArea
            app.ResultsTextArea = uitextarea(app.UIFigure);
            app.ResultsTextArea.FontWeight = 'bold';
            app.ResultsTextArea.BackgroundColor = [0.7451 0.6196 1];
            app.ResultsTextArea.Position = [171 23 426 189];

            % Create Label
            app.Label = uilabel(app.UIFigure);
            app.Label.FontName = 'Arial Black';
            app.Label.FontSize = 14;
            app.Label.FontWeight = 'bold';
            app.Label.Position = [174 410 387 52];
            app.Label.Text = {'SECURITY PATROL ROUTES AND CHECKPOINTS ';
'OPTIMIZATION WITH MATLAB TO MINIMIZE '; 'BLIND SPOTS AND ENHANCE THE SAFETY'};

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = PSM_BDP

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```