UNIVERSITÀ DEGLI STUDI DI CAGLIARI

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
PH.D. COURSE IN MATHEMATICS AND COMPUTER SCIENCE
CYCLE XXXIII

PH.D. THESIS

Path Planning for Robot and Pedestrian Simulations

S.S.D. INF/01

CANDIDATE
Raza Saeed

SUPERVISORS
Prof. Diego Reforgiato Recupero
Prof. Paolo Remagnino

PHD COORDINATOR
Prof. Michele Marchesi

Final examination academic year 2019/2020
April 2021

Abstract

The thesis is divided into two parts. The first part presents a new proposed method for solving the path planning problem to find an optimal collision-free path between the starting and the goal points in a static environment. Initially, the grid model of the robot's working environment is constructed. Next, each grid cell's potential value in the working environment is calculated based on the proposed potential function. This function guides the robot to move toward the desired goal location, it has the lowest value at the goal location, and the value increase as the robot moves further away. Next, a new method, called Boundary Node Method (BNM), is proposed to find the initial feasible path. In this method, the robot is simulated by a nine-node quadrilateral element, where the centroid node represents the robot's position. The robot moves in the working environment toward the goal point with eight-boundary nodes based on the boundary nodes' characteristics. In the BNM method, the initial feasible path is generated from the sequence of the waypoints that the robot has to traverse as it moves toward the goal point without colliding with obstacles. The BNM method can generate the path safely and efficiently. However, the path is not optimal in terms of the total path length. An additional method, called Path Enhancement Method (PEM), is proposed to construct an optimal or near-optimal collision-free path. The generated path obtained by BNM and PEM may contain sharp turns. Therefore, the cubic spline interpolation is used to create a continuous smooth path that connects the starting point to the goal point. The performance of the proposed method is compared with the other path planning methods in terms of path length and computational time. The obtained results revealed that the proposed method achieved better performance than other path planning methods.

Moreover, the multi-goal path planning problem is investigated to find the shortest collision-free path connecting a given set of goal points in the robot working environment. This problem combines two sub-problems: first, optimize the sequence of the goal points located in the free working space; second, compute the shortest collision-free path between the goal points. In this study, the genetic algorithm (GA) is implemented to optimize the sequence of the goal points. Once the goal points sequence is available, the BNM method is used to generate an initial collision-free path between every pair of the sequenced goal points. Afterwards, the PEM method is used to find an optimal or near-optimal path by reducing the waypoints and overall path length.

Furthermore, to verify the performance of the proposed method, several experimental tests have been performed on the $e-puck$ robot with different obstacle configurations and various positions of goal points. The experimental results showed that the proposed method could construct the shortest collision-free path and direct the real physical robot to the final destination point.

At the end of the first part of the thesis, we investigate the multi-goal path planning problem for the multi-robot system such that several robots reach each goal. The simulation results showed that the proposed method computes the shortest path effectively for multiple robots without colliding with obstacles and other robots in a static environment. However, in interactive virtual environments, the requirements for solving the path planning problem are different. In addition to being collision-free, the shortest paths for a large number of virtual pedestrians (robots, agents, etc.) through complex environments need to be planned simultaneously in real-time.

In the second part of this thesis, we proposed a new method for simulating pedestrian crowd movement in a virtual environment. The first part of this thesis concerning the generation of the shortest collision-free path is used. In this method, we assumed that the crowd consists of multiple groups with a different number and various types of pedestrians. In this scenario, each group's intention is different for visiting several goal points with varying sequences of the visit. The proposed method uses the multi-group $microscopic$ model to generate a real-time trajectory for each pedestrian navigating in the pedestrianized area of the virtual environment. Additionally, an $agent-based$ model is introduced to simulate pedestrian' behaviours. Based on the proposed method, every single pedestrian in each group can continuously adjust their attributes, such as $position$, $velocity$, etc. Moreover, pedestrians optimize their path independently toward the desired goal points while avoiding obstacles and other pedestrians in the scene. At the end of this part of the thesis, a statistical analysis is carried out to evaluate the performance of the proposed method for simulating the crowd movement in the virtual environment. The proposed method implemented for several simulation scenarios under a variety of conditions for a wide range of different parameters. The results showed that the proposed method is capable of describing pedestrian' behaviours in the virtual environment.

Acknowledgements

I would like to convey my heartfelt gratitude and sincere appreciation to all people who have helped and inspired me during my doctoral study. Especially, I would like to express my deepest gratitude to my supervisor Prof. Diego Reforgiato Recupero, for his advice and guidance. I also would like to thank my co-supervisor, Prof. Paolo Remagnino, for his support and help during my period as a visiting researcher at Kingston University and for giving me the opportunity to collaborate with the other researchers in the Robot Vision Team (RoViT). I would like to express my gratitude to the Erasmus Placedoc program for giving me the chance to spend a 6-month research period at Kingston University to gain experience and new skills abroad. Last but not least, I would like to extend my thanks to my family, especially my wife and my daughter, for their support and patience throughout my study.

Publications

The research reported in this thesis has contributed to the following publications:

- R. A. Saeed, Diego Reforgiato Recupero, and Paolo Remagnino, 2021. Simulating Crowd Behaviour Combining Both Microscopic and Macroscopic Rules. Information Sciences, submitted.
- R. A. Saeed, Diego Reforgiato Recupero, and Paolo Remagnino, 2021. Simulating People Dynamics. 17th International Conference on Intelligent Environments (IE2021), Dubai, United Arab Emirates, accepted.
- R. A. Saeed, Diego Reforgiato Recupero, and Paolo Remagnino, 2021. The Boundary Node Method for Multi-Robot Multi-Goal Path Planning Problems. Expert Systems, accepted.
- R. A. Saeed, Diego Reforgiato Recupero, and Paolo Remagnino, 2020. A Boundary Node Method for path planning of mobile robots. Robotics and Autonomous Systems 123: 103320.
- R. A. Saeed, and Diego Reforgiato Recupero, 2019. Path Planning of a Mobile Robot in Grid Space Using Boundary Node Method. 16th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Prague, Czech Republic.

Contents

Abstract **i**

Acknowledgements **iii**

Publications **v**

List of Figures **3**

List of Tables **9**

1 Introduction **13**
1.1 Overview and Motivation of the Thesis 13
1.1.1 Path Planning 13
1.1.2 Multi-Goal Path Planning 14
1.1.3 Multi-Robot Path Planning 15
1.1.4 Crowd Simulation and Path Planning 15
1.2 Research Gaps and Limitations 17
1.3 Objectives of the Thesis 18
1.4 Contributions of the Thesis 20
1.5 Structure of the Thesis 21

2 Related Work **23**
2.1 Path Planning 23
2.1.1 Multi-Goal Path Planning 25
2.1.2 Multi-Robot Multi-Goal Path Planning 26
2.2 Crowd Simulation and Path Planning 27

3 Proposed Method for Path Planning **31**
3.1 Problem Formulation 31
3.2 Proposed Method for Path Planning 32
3.2.1 Modelling of the Workspace 35
3.2.2 Boundary Node Method (BNM) 37
3.2.3 Path Enhancement Method (PEM) 46
3.2.4 Smooth Path Planning 47

3.2.5 Multi-Goal Path Planning . . . 49
3.2.6 Multi-Robot Multi-Goal Path Planning . . . 51

4 Proposed Method for Crowd Simulation 53
4.1 Problem Formulation . . . 53
4.2 Proposed Method for Crowd Simulation . . . 54
4.2.1 Environment Setting . . . 57
4.2.2 Motion Computation and Steering Behaviours . . . 66
4.2.3 Locomotion . . . 75

5 Path Planning Results 77
5.1 Path Planning Simulation Results . . . 77
5.1.1 Boundary Node Method (BNM) . . . 77
5.1.2 Path Enhancement Method (PEM) . . . 79
5.1.3 Path Smoothing Using Interpolation Technique . . . 80
5.1.4 Irregular-Obstacle Environment . . . 81
5.1.5 Three-Dimensional Environment . . . 82
5.1.6 Multiple Robot System . . . 83
5.1.7 Performance Evaluation . . . 83
5.1.8 Multi-Goal Path Planning . . . 88
5.2 Path Planning with Physical Robots . . . 97
5.2.1 Experimental Results of Path Planning . . . 97
5.2.2 Experimental Results for Multi-Goal Path Planning . . . 99

6 Crowd Simulation Results 103
6.1 Crowd Simulation . . . 103
6.1.1 Simulation of Simple Scenario . . . 104
6.1.2 Simulation of Real-Life Crowd Movements . . . 110
6.1.3 Comparison with Different Methods . . . 119
6.1.4 Statistical Analysis . . . 120

7 Conclusions and Future Work 123
7.1 Conclusions . . . 123
7.2 Future Works . . . 125

Bibliography 127

List of Figures

3.1 A simple example of a multi-goal path planning problem in a $2D$ workspace. (a) shows the sequence of the goal points, the path marked in red dashed line, and the goals are shown in grey square objects. (b) presents the optimal or near-optimal collision-free path for a mobile robot between the goal points, plotted in the solid blue line. 32

3.2 (a) a nine-node quadrilateral element along, (b) the robot motion directions, (c) the simulated robot in the exploration area. 33

3.3 Flow diagram of the proposed method for the path planning problem. . . 34

3.4 The potential value of the grid cells in the workspace in $3D$ view with contour plot. The size of the workspace is 50×50, and the the goal point C_g is located at $a)$ $(40, 45)$ and $b)$ $(25, 25)$. 36

3.5 $2D$ models of the robot working environment with obstacles. 37

3.6 Illustrates the collision avoidance in the static environment by using the BNM method, (a) the initial positions of the robot and boundary nodes, (b) the new updated positions, (c) obstacles avoidance and change the motion direction. 40

3.7 Demonstrate the mobile robot exploration in a $2D$ working environment by using BNM. 41

3.8 The workspace contains long horizontal obstacles that block the path of the robot. 43

3.9 An illustrative simulation example of the local minima problem in a 2D environment with U-shape obstacle. 43

3.10 An example of the path planning for the robot navigation in the C -space. (a) The shortest path is found using PEM, where the solid red line represents the shortest path. (b) The obtained solution of IFP by using BNM, where the sequence of the red circle objects represents the IFP. 47

3.11 Create the shortest path from "14" waypoints in the 2D workspace, where the red circle objects sign the waypoints. (a) construct the shortest path by using the PEM method. (b) generate the IFP by using the BNM method. (c) determine the shortest path using PEM and the smooth path by applying the spline method. 48

3.12 There are three different working environments with varying layouts of obstacles, the size of the workspace is 67×109, and the position of g_1 and g_2 located randomly in the free space C_{free}. The contour lines represent the potential value, and the pink circle objects represent the goal points, g_1 and g_2. 50
3.13 Collision avoidance between two robots, (a) the initial positions of the robots before the collision, (b) robot-robot collision avoidance and change motion direction, (c) the robots in their new updated positions. 52

4.1 Graphical representation of the given simple scenario. 55
4.2 Flowchart illustrating the steps of the new proposed method for crowd simulation. 56
4.3 The layout of the virtual environment with obstacles. 58
4.4 A nine-node quadrilateral element with a safety zone (a) along with its motion directions (b) and simulated pedestrian in a virtual environment (c). 70
4.5 Personal space requirements. 71
4.6 Sketch of the simulated pedestrian, as the current pedestrian move forward, it may interfere with the neighboring pedestrian: (a) the distance between the current pedestrian and the neighboring pedestrian is higher than a certain distance $(2r)$, (b) $pedestrian\,collision\,avoidance$ is used to prevent collision between pedestrians. 72
4.7 The three different types of steering behaviors (a) Separation, (b) Cohesion, and (c) Alignment. A similar figure is described by [1] 73

5.1 Simulation results of the generated initial feasible path (IFP) for all three workspace scenarios using BNM. 78
5.2 Simulation results of the generated path for all three workspace scenarios by using PEM. 79
5.3 Simulation results of the generated smooth path for all three workspace scenarios by using the cubic spline method. 80
5.4 Simulation results of the generated smooth path for all three workspace scenarios by using $PCHIP$. 81
5.5 Examples of different workspace with different obstacle shapes (a and e), and the simulation results of the generated path for two workspace scenarios by using BNM (b and f), PEM (c and g), and cubic spline method (d and h). 82
5.6 Simulation results for constructing a collision-free path by using BNM (a) and PEM (b) to solve the path planning problem in a three-dimensional $(3D)$ workspace. 83
5.7 Simulation results for solving the multi-robot path planning problem. . . . 84
5.8 Simulation results for solving the path planning problem in a $2D$ workspace by using BNM (a), PSO (b), $A\text{-}Star$ (c), and APF (d). . . . 85

5.9 Simulation results from the implementation of the PEM method for optimizing the generated paths by using BNM (a), PSO (b), $A\text{-}Star$ (c), and APF (d). 85

5.10 Simulation results from the implementation of the cubic spline method for smoothing the generated paths by using BNM (a), PSO (b), $A\text{-}Star$ (c), and APF (d). 86

5.11 Simulation results for generating the path by using BNM (a), PEM (b), and cubic spline (c) in the workspace that previously has been used in [2–5]. 86

5.12 Performance evaluation of BNM, PSO, GA, and $A\text{-}Star$ to find the collision-free path for "1000" independent runs, (a) presents the obtained results of the path length, and (b) presents the computational time results. 88

5.13 An illustrated example of the multi-goal path planning problem for three randomly-selected goal points. (a) the path is determined by using GA, plotted in a red dashed line. (b) the IFP is generated by using BNM, plotted in the blue line. (c) optimize the generated path by using PEM, plotted in the blue line. 89

5.14 Illustrates the steps of solving a multi-goal path planning problem. The BNM is applied to generate the IFP for the robot to move from g_1 to g_2 (a), from g_2 to g_3 (b), and from g_3 to g_1 (c). The PEM is implemented to generate the shortest path from g_1 to g_2 (d), from g_2 to g_3 (e), and from g_3 to g_1 (f). 90

5.15 The influence of the number of goal points on the total computational time required to solve MTP for each simulated working environment. 92

5.16 The first scenario of the multi-goal path planning problem with "20" randomly-selected goal points. (a) the path is formulated by connecting the sequenced goal points, plotted in a red dashed line. (b) the IFP is generated by using BNM, plotted in the blue line. (c) optimize the IFP by using PEM, plotted in the blue line. 93

5.17 The second scenario of the multi-goal path planning problem with "20" randomly-selected goal points. (a) the path is formulated by connecting the sequenced goal points, plotted in a red dashed line. (b) the IFP is generated by using BNM, plotted in the blue line. (c) optimize the IFP by using PEM, plotted in the blue line. 93

5.18 The third scenario of the multi-goal path planning problem with "20" randomly-selected goal points. (a) the path is formulated by connecting the sequenced goal points, plotted in a red dashed line. (b) the IFP is generated by using BNM, plotted in the blue line. (c) optimize the IFP by using PEM, plotted in the blue line. 94

5.19 The influence of the number of robots on the total computational time to solve MTP using the proposed method for each simulated scenario. . . . 95

5.20 Simulation results for solving the MTP: (a) the sequence of the goal points obtained from the implementation of GA, the red square objects represent the goal points. $(b \rightarrow f)$ multiple robots $(5\ robots)$ move to visit multiple-goal points $(4\ goals)$ in a simulated working environment with obstacles. 96

5.21 Simulation results for solving the MTP: the number of goal points is fixed $(4\ goals)$, and the number of robots is varied $(1 \rightarrow 6)$. 96

5.22 The $e\text{-}puck$ mobile robot was used for the experimental test (a). The experimental set-up (b) for testing the performance of $BNM\&PEM$ to solve the path planning problem in a static environment. 97

5.23 Simulation and experimental results: (a) simulation result to generate a shortest collision-free path by using $BNM\&PEM$, and $(b) \rightarrow (f)$ $e\text{-}puck$ robot positions at different locations in the robot's working environment. 98

5.24 The experimental set-up for testing the performance of $BNM\&PEM$ to solve multi-goals path planning problem in a static environment. 99

5.25 Simulation results of the proposed method for solving the MTP: (a) the sequence of the goal points obtained from the implementation of GA, (b) the initial feasible path (IFP) generated from the BNM method, (c) the shortest collision-free path is generated by using PEM. 100

5.26 Experimental results: (a) shows the initial locations of the $e\text{-}puck$ robot and the goal points in the working environment. The movement of the robot is shown in (b) from g_1 to g_2, (c) from g_2 to g_3, $(d\,\&\,e)$ from g_3 to g_4, and (f) from g_4 to g_1. The shortest path that the robot has to follow to reach the destination point is represented by blue and yellow dashed-line, and the starting and destination goal points are represented by the yellow and red dashed circles, respectively. 101

6.1 Goal points for each group of pedestrians: each $goal\,point$ represents by two-points, the first point is the entrance of the goal, and the second point represents the inside of the goal area. 105

6.2 keyframes for activating groups to move in the simulated environment. . . 106

6.3 Pedestrian' trajectory: showing the pedestrians' movement towards their goal points in a simulated environment. 107

6.4 Pedestrian' velocity: change the walking velocity of all pedestrians. . . . 107

6.5 The minimum distance between pedestrians. 108

6.6 Pedestrian' energy level: change of the pedestrians' energy level at each $keyframe$. 109

6.7 Pedestrian's trajectory: the graph shows pedestrians' movement in different directions in the simulated walking area at different $keyframes$: 65, 100, 145, 165, 200, 230, 240, 270, 287, and 310. The red circle objects represent the pedestrians' trace in the virtual environment at each $keyframe$. The left graph shows the generated planned pedestrian's trajectory, and the right graph shows the simulation result of the pedestrian's trajectory in the virtual environment after the simulation runs for 500 $keyframe$. 109
6.8 A large-scale $3D$ model of the commercial mall populated by virtual groups of pedestrians [6]. 111
6.9 Screen-shot of the shopping mall with obstacles: shows the walking area on the ground floor of the shopping mall with the $3D$ obstacles. 111
6.10 Shows different $3D$ characters animation obtained from $Mixamo$ [7] . . 112
6.11 Screen-shot of pedestrian flow: a) shows the initial configuration of the simulation, and b) the pedestrians' trajectories in the final stage of the simulation. 114
6.12 Screen-shot of crowd simulation: groups of pedestrians move in different directions to reach their desired goal points while avoiding the static obstacles and pedestrians in the scene. 115
6.13 Keyframes for activating groups of pedestrians to move in the virtual environment. 116
6.14 Screen-shot of the simulation: showing the pedestrian's movement towards the goal points in a scene. 116
6.15 Pedestrian' velocity: change the walking velocity of all pedestrians. . . . 116
6.16 Screen-shot of the simulation: shows a single pedestrian trying to move through the walking area toward the goal point, and he starts to change his motion direction near the safety area around the stairs at $keyframes$ a)118, b)122, and c) 124. 117
6.17 Screen-shot of the simulation: shows a group of pedestrians, consists of two pedestrians (pedestrian A and pedestrian B), that using the obstacle avoidance method to pass the obstacle. Afterwards, pedestrians keep moving toward their desired goal points: a) collision avoidance between pedestrians and obstacle at $keyfram = 94$, b) collision avoidance between pedestrian A and obstacle at $keyfram = 95$, and c) collision avoidance between pedestrian B and obstacle at $keyfram = 96$. 118
6.18 Screen-shot of the simulation: shows a group of pedestrians, consists of two pedestrians (pedestrian A and B), passing each other using the pedestrian avoidance method. a) collision avoidance takes place between pedestrians A and B at $keyframe$ 97. b) pedestrian B changes the motion direction to avoid pedestrian A without any collision at $keyframe$ 98, and then they keep moving to reach their goal point. 119

6.19 Simulation and performance evaluation: (a) the number of pedestrians, and (b) the computational time required to generate the pedestrians' trajectories in each group using the proposed method. 121

7.1 Different types of elements to simulate a mobile robot, such as a) nine-node circular element, b) seventeen-node quadrilateral element, and c) thirteen-node octagonal element. 126

List of Tables

3.1 Characteristics of three different workspace scenarios. 37

4.1 Simple scenario for formulating groups of pedestrians in the virtual environment. 54

5.1 The total computational time and path length of the initial feasible path (IFP) and the shortest path by using BNM and PEM. 78
5.2 Summary of the obtained results from the implementation of BNM, PSO, $A\text{-}Star$, and APF for solving the path planning problem. 84
5.3 Total computational time required to find the shortest path by using BNM and improved GA. 87
5.4 Mean and standard deviation (Std) of the computational time and the path length for "1000" independent runs to find the collision-free path using BNM, PSO, GA, and $A\text{-}Star$, ... 87
5.5 Performance evaluation results: The mean and standard deviation of the computational time (in $seconds$) required to find the feasible path for a single mobile robot in three different working environment. 91
5.6 Characteristics of three different example scenarios with different number of goal points. 94
5.7 Performance evaluation results: the mean and standard deviation of the computational time (CT) in $seconds$ to find a collision-free path for each instance in different simulated scenario. 95

6.1 Pedestrians' distribution with different group size: presents the number of groups in the simulation ($nGroups$ ="4"), number of pedestrians of each type in the same group ($nPedestrians_{Types}$), number of pedestrians in each group ($nPedestrians_{group} = [4, 4, 5, 3]$), and the total number of pedestrians in the crowd ($nPedestrians_{total}$ ="16") 104
6.2 Maximum and minimum energy level for each type of pedestrian 110
6.3 Pedestrians' group size distributions in the crowd, where pedestrian is denoted by p. 113
6.4 Pedestrians type contribution in the crowd. 113

6.5 Ability of simulating group dynamics with the proposed method for the crowd simulation (GA: Group Avoidance, GI: Group–pedestrian Interaction, InterE: Inter-group Emotion Contagion. LF: Leader–Follower, GF: Group Formation, GS: Group Structure, GC: Group Cohesion, GCo: Group Cooperation, PB: Peer Behavior, IntraE: Intra-group Emotion Contagion). 120
6.6 The mean and the standard deviation Std of the pedestrians' number (NOP), and the computational time (CT) (in $seconds$, $[S]$) required to find the trajectories for the pedestrians in each group. 120

List of Algorithms

1 Calculate the potential value of the grid cells in the workspace. 35
2 Calculate the values of Δx and Δy. 39
3 Illustrates the steps to bring the robot out of the local minima area. 42
4 Boundary Node Method (BNM) 45
5 Path Enhancement Method (PEM) 49
6 Generate workspace and obstacles 60
7 Generate pedestrians with their attributes. 61
8 Generate goal points 63
9 Setting the initial state of the pedestrians in the crowd $Pedestrians_{group}$. 65
10 Create a list of random keyframe for activating groups in the scene. 66
11 Motion computation and steering behaviors 69
12 Pedestrian collision avoidance 73
13 Goal-directed behavior 74

Abbreviations

$Notation$	Description
BNM	Boundary Node Method
PEM	Path Enhancement Method
IFP	Initial Feasible Path
C_{obs}	Space occupied by obstacles
C_{free}	Free Space, $C_{free} = C$–C_{obs}
$nGroups$	number of groups of people in the crowd
$nPedestrians_{group}$	number of pedestrians in each group
$nPedestrians_{Types}$	number of the each type of pedestrian in the same group
$Pedestrians_{crowd}$	all pedestrians in the crowd
$Pedestrians_{group}$	pedestrians in each group
$nPedestrians_{crowd}$	number of pedestrians of all groups in the crowd
$nPedestrians_{total}$	total number of pedestrians in the crowd
$n3DObstacles$	number of $3D$ obstacles in the scene
$nObstacles$	total number of the grid cells occupied by obstacles
$nGoals$	number of goal points on the ground floor
$GoalPoints^{list}$	list of all existing goal points
$Goals_{group}$	goal points for each group
$Goals_{crowd}$	all goal points of all groups in the crowd
$nGoals_{group}$	number of goal points for each group
$nGoals_{crowd}$	number of goal points of all groups in the crowd
$nRandom_{goals}$	random number of goal points for each group
$Goals_{index}$	goal index
$nRandom_{goalsIndex}$	random number for goal index
$nRandom_{Types}$	random number of pedestrians of each type
$nFrames$	number of frames
$Positions_{crowd}$	all pedestrian's position in the crowd
$Position_{p_e}$	pedestrian's position
$Velocity_{p_e}$	pedestrian's velocity
$Energy_{p_e}$	pedestrian's energy
$kFrame_{activate}$	list of random number for key-frame to activate groups
r, R	Radius of the safety zone around pedestrian and obstacles, $unit$

Chapter 1

Introduction

1.1 Overview and Motivation of the Thesis

1.1.1 Path Planning

In robotics, path planning aims to find a collision-free path for a mobile robot to move from a starting point to a goal point in a given working environment based on certain optimization criteria such as walking distance, walking time, energy consumption, etc [8–10]. It is expected that the robot reaches the final destination point safely through the shortest walking path within the minimum computational time. Path planning has been widely applied in many robotic applications to perform various tasks in many domains such as nuclear facilities [11], space exploration [12], rescue missions, landmines, and enemies in the war field [13]. Moreover, the path planning approaches are useful for repeatable tasks in static environments where optimality is essential (e.g., industrial applications) [14]. These factors make path-planning an interesting and challenging subject for researchers [13].

The path planning problem started around the sixties. However, the interest in the path planning problem for mobile robots grew after the authors' work in [15], and then many methods have been proposed [9, 16]. The existing methods are mainly categorized into classical and heuristic path planning [9, 13]. The classical methods include cell decomposition, potential field method, subgoal network, and road map [17]. These methods involve finding a set of defined steps to search for a path starting from an initial position to a goal position. In classical methods, only deterministic actions are considered [17, 18]. However, it has been found that the classical methods have some disadvantages such as the high computational cost, trapping into local minima, and high time complexity [8, 13, 16]. As classical search methods fail to find the exact solutions, many heuristic methods have been proposed, i.e., Genetic Algorithm (GA) [14, 19], Particle Swarm Optimization (PSO) [13], Artificial Neural Networks ($ANNs$), Ant Colony Optimization (ACO) [16, 20], and Fuzzy Logic (FL) [16]. Surveys work in [8, 20] showed that the heuristic path planning methods are computationally more

efficient in terms of path length, obstacle avoidance, and elapsed time [16]. Heuristic methods attempt to find a good solution to the path planning problem in a short amount of time. However, these methods are not guaranteed to provide an optimal solution [17, 18]. The combinatorial path planning methods can solve many path planning problems and construct optimal solutions efficiently [17, 21–24].

Many of the existing methods for robot path planning can find a path for the robot. However, in most cases, the quality of the generated path is not accurate enough, or their efficiency is not sufficient [4]. Researchers have always been seeking a better solution to improve the performance of the existing path planning methods. A list of the goals that researchers of several earlier works have pursued is the following: improve the accuracy [12, 25, 26], improve the efficiency [4, 27], increase safety [11, 12, 28], increase the capability [29], reduce the processing time [30, 31], overcome the non-reachable goal problem [32], pass through narrow passages [33], overcoming the local-trap problems, and improve the quality of planned paths [4].

1.1.2 Multi-Goal Path Planning

In robotic planning, the problem of finding the shortest collision-free path connecting a given set of goal points located in the robot working environment is called a multi-goal path planning problem (MTP) [34–36]. The MTP can be modelled by Traveling Salesman Problem (TSP), in which the paths between goal points have to be traversable by the robot. The requirement for collision-free path connecting goal points is the main reason why the problem is called the MTP rather than the TSP to emphasize its difficulty [37]. In the TSP, a salesman has to visit several cities (locations) with the constraint that the salesman should visit each city once. A salesman is interested in travelling on the shortest route linking a set of cities based on the given distance between them [38]. The shortest tour starts from a given city, passing through all the other cities and returning to the home city [35]. There are many methods to solve TSP, such as heuristic algorithms, genetic algorithms, simulated annealing algorithm, and ant colony optimization [39]. Furthermore, the TSP has been well studied, and efficient algorithms are available to solve the TSP problem [40].

In a simple case, optimizing the sequence of the goal points and the path between the goal points can be modelled by TSP [41, 42]. In the basic TSP, the probability of colliding obstacles along the path between goal points is not considered [43]. As the effects of obstacles are taken into account, the path planning problem can be solved in two steps. First, optimize the goal points sequence, then an optimal collision-free path between goals needs to be planned. The literature shows that GA is widely applied to perform a random search to solve optimization problems. By taking advantage of its strong optimization capability [35, 44, 45], GA has been used to solve the TSP problem (e.g., [35, 38, 46]). Once the sequence of the goal points is determined, the path planning method is used to construct a collision-free path between every pair of the sequenced

goal points [42]. The path that starts from the initial goal point, passing through all intermediate goal points, and returning to the initial goal point is called a multi-goal path. A multi-goal path is assembled by simply connecting goal-to-goal points, extract the path between goal points iteratively until the path (tour) is completed. The length of the multi-goal path is the sum of the length of the goal-to-goal path.

1.1.3 Multi-Robot Path Planning

Recently, using multiple mobile robots rather than a single mobile robot has attracted many researchers' attention [47], and it has been widely used in industrial plants and warehouses [48]. The multi-robot path planning problem has been studied since the 1980s [49–53], and several methods have been proposed during this period [51]. Multi-robot path planning methods have been used to determine the collision-free path for multiple robots from their given start positions to their goal positions. Two types of collisions have to consider to plan collision-free paths for multiple mobile robots: robot–obstacle and robot–robot. In general, the multi-robot path planning problem is considered as an optimization problem, the solution to the optimization problem provides the optimal path [51, 54].

In robotics, the path planning problem for multiple robots focuses on generating the shortest paths for multiple robots to move from their starting point to their final destination point. However, in a virtual environment with a crowd of pedestrians (robots, agents, etc.), the requirements for solving the path planning problem are different. In addition to being collision-free, paths for a large number of virtual pedestrians through complex environments need to be planned simultaneously in real-time. In both problems, multiple units navigate in a working environment without colliding with each other or static obstacles. The existing methods for solving the path planning problem for multiple robots can handle only a few robots, and these methods are not suited for real-time applications. In contrast, in a virtual environment, the concentrate is on simulating the movement of a large number of virtual pedestrians independently in real-time. In a computer program, simulating the movement of pedestrians is called crowd simulation. Pedestrians are referred to as "agents", there are other terms that can be used, such as robots, entities, etc., depending on the application. In this thesis, we will use the term "pedestrians" to avoid confusion.

1.1.4 Crowd Simulation and Path Planning

Simulation of crowd movement has attracted the interest of many scientists from different research fields. Over the last several years, many techniques have been studied on crowd movement behaviours, path planning, and navigation in various disciplines, including computer graphics, virtual reality, social science, statistical physics, robotics, pedestrian and evacuation dynamics, etc. These techniques can be grouped into two main categories: $macroscopic$ and $microscopic$. The $macroscopic$ techniques focus on the aggregate

behaviours of crowds. These techniques predict the motion of agents (pedestrians) and other characteristics such as $densities$, $speeds$, and emerging behaviours. However, $microscopic$ techniques have been used to find each agent's path that avoids the static or dynamic obstacles and other agents in the environment [55].

An important issue for crowd simulation concerning the agents' movement is whether the crowds being simulated is $homogeneous$ or $heterogeneous$ [55, 56]. $Homogeneous$ crowds correspond to instances where each agent has very similar behaviour or goal. Many models have been proposed for $homogeneous$ crowd simulation. For example, $continuum$ crowd model [57], allow a small, fixed number of goals, and aggregate dynamics for dense crowd simulation [55, 58]. The $continuum$ models involve treating the crowd as a whole, and their application in large crowds composed of hundred thousand agents [59]. Other examples include models based on $flow - based$ models [60, 61] which are governed by differential equations, and this model uniformly dictates the flow of crowds across space. In $heterogeneous$ crowds, each agent in the crowd maintains a distinct, observable identity (see [62]). This identity is observed in the agent's goals, desired speeds, cooperation, and many other factors that affect the motion of each agent [55, 56]. In the crowd simulation literature, many $heterogeneous$ techniques have been proposed for simulating $heterogeneous$ crowds [63, 64], the most widely used is the $agent - based$ models [65]. In $agent - based$ model, the motion of each agent in the crowd is computed separately, and it is possible to simulate crowds with a different motion for different agents [55, 56]. For large-scale analysis and prediction of crowd behaviour, $continuum$ models are more efficient than $agent - based$ models [59], where the computational time does not increase significantly with the number of agents. However, $continuum$ models suffer from different drawbacks, such as a reduced validity range [65]. In $agent - based$ models, agents follow some predetermined rules of behaviour, which allow agents in the model to behave naturally and autonomously. This unique characteristic makes $agent - based$ modelling particularly suitable for the study of agents' behaviour in complex environments [66].

In $agent - based$ models, agents (pedestrians) continuously adjust their behaviour in the simulated environment [67], and different simulation parameters can be defined for each crowd member [57]. These models often result in more realistic crowd movement behaviours and detailed simulations with each agent making independent decisions. However, the $agent - based$ models also have drawbacks, i.e., global path planning for each agent becomes computationally expensive, particularly in real-time contexts. Therefore, to simulate the agents' movement using $agent - based$ models, most existing path planners fail [68]. As a result, most $agent - based$ models address only local motion planners for collision avoidance. Furthermore, interactions of an agent with other agents or with the environment are often performed at a local level, and sometimes it can result in undesirable $macroscopic$ behaviours [69]. The $boids$ algorithm is considered one of the most popular $agent - based$ models [70]. The $boids$ algorithm generates steering behaviours that resemble flocking, herding, and school behaviours commonly observed

in animal motion [56]. This algorithm has been used for modelling and simulating crowds in a virtual environment [62]. Furthermore, it has been widely used in games and generating special effects in movies [55]. The flocking behaviours are simulated through the $boids$ algorithm, as it is well known for expressing the motion of the groups [62].

An agent's motion computation is split into two distinct tasks: $global$ and $local$ navigation [55, 56]. $Global$ navigation aims to compute a long-term collision-free path towards a goal position that only the static obstacles consider. In contrast, $local$ navigation techniques take into account the motion of dynamic obstacles and other agents in the environment and steer each agent towards its goal position [55].

In real-world applications, crowd movement simulation provides valuable tools to planners and designers for improving efficiency and safety in public places such as airport terminals [66], shopping malls, train stations [71], theatres, etc [72]. While planning the architecture of buildings interested in simulation of people moving around their intended design so that shops, entrances, corridors, emergency exits, and seating can be placed in suitable locations [72]. In new architectural or urban domains, the crowd simulation is used to predict crowd flows, crowd analysis, and architectural analysis benefits from exploring the environment as quickly as possible and as many options as possible [56]. Additionally, crowd movement simulations have been proposed and successfully applied to various scenarios and case studies [73], i.e., crowds associated with transport systems, sporting and general spectator occasions, holy sites, political demonstrations, fire escape [59], emergency evacuation [56, 66, 72, 74, 75], and trading port [62]. Moreover, crowd movement simulations have been used to develop a level-of-service concept, design elements of pedestrian facilities, planning guidelines [74], safety planning [56] and support transportation planners or managers to design timetables [72]. Furthermore, crowd modelling and simulations allow the safer and more efficient design of civil structures like big buildings, markets, and stadiums [76]. Realistic pedestrian behaviour simulation in crowded places has diverse applications in architecture design, emergency evacuation, urban planning, personnel training, education, and entertainment [67]. Additionally, realistic simulations based on pedestrian behaviour such as obstacle avoidance, stress response, and avoid other pedestrians have many applications such as computer game designers, movies, and virtual environments [56]. Considerable effort has been made on locomotion, realistic pedestrian movement behaviours, path planning, and navigation in large virtual environments [77]. The experimental studies of pedestrian movement behaviours, observations, and data recording by the most sophisticated techniques are nowadays extensively available, and also many human walking attitudes have been pointed out [78].

1.2 Research Gaps and Limitations

Despite remarkable contributions that have been made, there are several essential gaps and limitations that still need to be addressed, as outlined in the following points:

1. Solving the path planning problem in a complex environment with a good quality solution is still a challenging issue [79, 80]. Moreover, it becomes much more difficult to find an optimal path within a reasonable amount of time [9]. In computational complexity theory, the path planning problem is classified as a non - deterministic polynomial time problem (NP-complete). The required computational time grows exponentially as the size of the path-planning problem increases [16]. The computational time is still too high in several works, and the search for an optimal path might not succeed [9].

2. The computational time required for solving the multi-goal path planning problem for a single and multiple mobile robot systems is high [34, 37, 81]. Moreover, depending on the problem complexity, finding the shortest collision-free path between goal points and optimizing the sequence of goal points will be very difficult [35] and computationally challenging problems [82]. Furthermore, sequencing goal points before searching for the collision-free path may generate a near-optimal path because goal points are arranged according to a distance function that ignores the obstacles. Additionally, path planning between two sequenced goals may be challenging [40]. While significant progress has been made in this area, there is still no effective approaches [83].

3. Developing a new crowd simulation method by using an $agent - based$ model is challenging. In a large-scale $agent - based$ crowd simulation, using global path planning for each agent (pedestrian) becomes computationally expensive, particularly in real-time contexts [57, 67]. Therefore, the crowd simulation methods with a strong computational performance for predicting the crowd movement are desirable [56]. Apart from the pedestrian motion, a crowd model needs to address the dynamic interactions between pedestrians [57], and interaction between groups can significantly influence the crowd behaviour. In the past, limited research has been done previously in considering group dynamics [55, 66, 72].

1.3 Objectives of the Thesis

The main objectives of this thesis are stated below:

1. Develop a new path planning method to find an optimal collision-free path between starting and goal points for a mobile robot in a relatively short computational time.

2. Solve a multi-goal path planning problem for single and multiple mobile robots by generating a shortest collision-free path connecting a given set of goal points located in the robot working environment.

3. Perform an experimental study on a real robot to verify the performance and effectiveness of the proposed method for generating a collision-free path, and illustrate how the robot can navigate along the path.

4. Develop a new method for simulating the heterogeneous crowds' movement in a virtual environment. The crowd consists of multiple groups and different sizes with various types of pedestrians, where each type of pedestrian has its own characteristics, and each group's intention is different.

5. Investigates real-time path planning for each pedestrian in each group while avoiding obstacles and other pedestrians in the crowd.

This study proposes a novel path planning method for a mobile robot in a two-dimensional ($2D$) static environment. In the proposed method, called Boundary Node Method (BNM), the robot is simulated by a nine-node quadrilateral element, where the centroid node represents the robot's location. The robot is exploring the environment with the help of the node's potential value at each location, where the potential value is calculated based on the proposed potential function. The initial feasible path is generated from the sequence of the waypoints that the robot has to traverse as it moves toward the goal point without colliding with obstacles. The proposed method can efficiently and efficiently generate the path, but the path is not optimal in terms of the total path length. Therefore, an additional method, called Path Enhancement Method (PEM), is proposed to construct an optimal or near-optimal collision-free path. Moreover, a new version of the proposed method is introduced to handle the multi-goal path planning problem. This method is implemented to find an optimal goal-to-goal path between a set of goal points in the obstacle-filled environment. Additionally, the BNM method extended further to solve the multi-goal path planning problem for multi-robot systems by generating the shortest collision-free path connecting every pair of the sequenced goal points among obstacles. Finally, an experimental study has been carried out on the real robot to verify the performance of the proposed method for generating a collision-free path for single and multiple goal points.

In order to evaluate the contribution of the proposed method, the performance of the proposed method is compared with four well-recognized path planning methods, namely, PSO, GA, $A-Star$, and Artificial Potential Field (APF). The PSO algorithm is widely used in path planning problems [25, 29, 80] as it is fast and simple [80], easy to implement [84], and a powerful method [21] to solve mobile robot navigation problems [29]. Moreover, the $A-Star$ algorithm is an effective method to search for the short path [28], which is implemented for many path planning applications [85], and it is mainly employed on an environmental grid [26]. In addition, GA is a robust optimization method among the existing approaches for robot motion planning problem [4]. By taking advantage of its strong optimization ability, the GA has been widely used in a previous study to generate an optimal path [2]. The APF method is fast [20, 80, 85], simple [32, 85], easy to implement [85], and it has good results for solving path planning problems [20]. The disadvantage of the APF method is related to the local minima problem whereby the robot can get stuck [32, 80, 85].

Furthermore, in this study, a new method for simulating crowd movement in a complex virtual environment is proposed. In this method, many groups of different sizes and various types of pedestrians (agents, entities, etc.) are considered. Pedestrians in many groups moving through the walking area inside a virtual environment with different directions to reach their goal points. The proposed method uses the $multi-group\,microscopic$ model for generating a real-time trajectory for each pedestrian navigating in the virtual environment. Moreover, an $agent-based$ model is introduced into the proposed method for modelling each pedestrians' behaviours, where each type of pedestrian has its own attributes. In this method, several steering behaviours are introduced such as $separation$, $cohesion$, $alignment$, $obstacle\,avoidance$, $pedestrian\,collision\,avoidance$, $flocking$, $goal-directed\,steering\,behavior$, etc. In addition, several techniques have been used for combining steering behaviours to a single steering force that allows pedestrians to walk toward their destination points. Based on the proposed method, pedestrians constantly adjust their position, optimize their path toward the desired goal points, and avoid obstacles and pedestrians when they move closer.

1.4 Contributions of the Thesis

The main contributions of this thesis can be summarized in the following points:

1. The developed method, BNM, can generate the initial feasible path (IFP) for a mobile robot from the sequence of the waypoints that the robot has to traverse as it moves from a starting point to a goal point without colliding with obstacles. An additional method, PEM, was developed to find an optimal or near-optimal path from IFP by reducing the number of waypoints and the overall path length.

2. The BNM method uses an optimization technique to generate a collision-free path in a relatively short computational time. Moreover, the computational time required to solve the path planning problem does not significantly increase with increasing the environment's complexity. Additionally, this method does not work through random operations, and there is no uncertainty in generating points, which leads to finding the final solution for the problem without variation in the solution.

3. The comparison of the statistical performance between the developed method and other path planning methods revealed that the developed method solved the path planning efficiently in terms of computational time and path length.

4. The BNM method extended further to solve multi-goal path planning problem for single and multiple mobile robots by generating the shortest path connecting every pair of the goal points sequentially without colliding with any obstacle or other robots in the system.

5. The experimental tests on the real robot ($e-puck$) show the efficiency of the developed method for solving the path planning problem. The test results demonstrate

how the robot navigates in the working environment toward the goal point(s) safely and quickly in a reasonable time.

6. The proposed method for crowd simulation has been well applied to simulate the crowd movement, where the crowd consists of multiple groups with a different number and various types of pedestrians. In this method, pedestrians in each group continuously adjusted their paths and updated their attributes independently in real-time.

7. The proposed method introduced various attributes and different steering behaviours for each pedestrian in the crowd. Implementing this method is very efficient for simulating the crowd movement in a complex domain scattered with obstacles. The proposed method can be expanded to include other pedestrians' attributes in the crowd for more rich and complex simulations.

1.5 Structure of the Thesis

The thesis is organized as follows:

- **Chapter 2** provides a literature review on the path planning problem. Moreover, an overview of the previous works related to the multi-goal path planning problem MTP for single and multiple mobile robots are introduced. This chapter presents the related studies on crowd simulation, group dynamics, modelling approaches, and navigation algorithms.

- **Chapter 3** introduces the problem formulation for robot path planning, and the details of the developed method for generating the shortest collision-free path is described with several illustrative examples. This chapter also presents the extended method to solve the multi-goal path planning problem MTP for single and multiple mobile robot systems.

- **Chapter 4** provides the problem formulation for simulating crowd movement in a virtual environment. Additionally, this chapter presents the description and configuration of the proposed crowd simulation method for simulating the crowd movement in a virtual environment.

- **Chapter 5** presents the implementation of the proposed method for solving the path planning problem and discuss the simulation results for several working environments with different obstacle layouts. Additionally, the performance of the proposed method in comparison with the other path planning methods is presented in terms of path length and computational time. This chapter also presents the implementation of the proposed method for solving the multi-goal path planning problems for single and multiple mobile robot systems. Moreover, the validation of the developed method for solving the path planning problem and multi-goal path

planning problem has been examined by performing experimental tests on the real robot. Moreover, this chapter presents and discusses the experimental results obtained from implementing the developed method introduced in Chapter 3 on the $e-puck$ robot.

- **Chapter 6** presents the implementation of the proposed crowd simulation method for investigating the crowd movement. Different scenarios are examined, and the results of the simulation study related to crowd behaviour and movement are presented and discussed.
- Finally, the main conclusions and directions for future research are provided in **Chapter 7**.

Chapter 2

Related Work

2.1 Path Planning

The path planning problem has attracted many researchers' attention due to its uncertainties, complexities, and real-time nature [79]. The path planning problem for mobile robots has been widely discussed in the literature, and various approaches have been proposed for solving the problem. For example, the authors in [9] proposed a new methodology to solve the path planning problem in two steps. First, they generate the initial path based on the surrounding point-set (SPS), which refers to a set of points surrounding the obstacles. Then, they applied the path improvement algorithm to get the optimal path by using the outcome of the first step. As stated in [9], this method has a low-level of randomness that reduces the variation of solutions, and also, this method can generate points in narrow or small spaces in the map. Another technique is the Bacterial Potential Field (BPF) developed by [22] to compute an optimal path for a mobile robot in a real-world scenario with static and dynamic obstacles. As reported in [22], the path planning with the BPF allows a robot to navigate autonomously without being trapped in local minima.

Several researchers combined algorithms to improve path planning performance. For example, the authors in [21] presented a hybrid meta-heuristic $GA - PSO$ algorithm for mobile robot navigation to find an optimal path between starting and ending points in a grid environment. The authors stated that the proposed algorithm avoids time complexity and premature convergence in conventional GA and PSO algorithms. The hybrid $GA - PSO$ is used to generate the initial feasible path. Afterwards, a cubic B-spline technique is applied to construct a near-optimal collision-free path. To reduce the complexity of robot path planning, authors in [86] proposed a hierarchical path planning method by integrating fuzzy theory and genetic algorithm. Researchers in [87] suggested another method named $SACOdm$ based on Simple Ant Colony Optimization Meta-Heuristic ($SACO - MH$) to solve the path planning problem. One of the main contributions of $SACOdm$ is the inclusion of memory capabilities to artificial ants to prevent stagnation. An additional methodology has been proposed in [23] by integrating

Artificial Bee Colony (ABC) algorithm with an evolutionary programming algorithm. In this method, the ABC algorithm is applied to generate a feasible path. Then the feasible path is enhanced by using an evolutionary programming algorithm.

Many researchers built on top of the existing methods to improve their performance and to overcome their limitations. Authors in [28] proposed an improved version of the $A-Star$ algorithm to overcome the inherent drawbacks of the original $A-Star$. The main improvement of the proposed method is that the local path is planned before the next search in the current node's neighbourhood. The $A-Star$ algorithm calculates the heuristic function's value at each node on the working area. Then it checks adjacent nodes to find the optimal solution with zero probability of collision, but the time complexity is too high. To overcome this problem, the authors in [30, 31] introduced several improvements to $A-Star$ for reducing the computational time and increase overall performance. Another improvement is the minimization of the path length by reducing the number of local paths. Several researchers investigated the capabilities of GA for solving the path planning problem for mobile robots in static and dynamic environments. For example, [19] proposed a new fitness function for GAs [27] proposed the Knowledge-based GA, [4] proposed an adaptive GA, and [88] suggested a new variant of GA using binary codes through the matrix. The authors in [32] presented a new repulsive potential function to solve non-reachable goals with obstacles nearby ($GNRON$) in the potential field method. Calculation of artificial potential values is another solution to obtain a collision-free path. The potential field method was first used by [89] for solving the robot motion planning problem. This method works based on attraction and repulsive values, these two fields are produced by target and obstacles, and the robot is considered a moving object in these fields. The robot moves toward the target based on the negative slope of the potential function. The problem with this approach is that the robot can get stuck in local minima [4]. Consequently, various techniques have been proposed to avoid the local minima, i.e., authors in [90] tried to solve the problem using harmonic potential functions around obstacles. In order to overcome the local minima and heavy computational time, the rapidly-exploring random tree (RRT) and the probabilistic roadmap (PRM) algorithm are introduced due to their remarkable practical performance and strong theoretical properties [91, 92]. These algorithms work by computing multiple distributed random points in the free workspace and connect them to construct a tree or graph; after that, a search method is used to find a path [28]. In RRT, the most important factor that affects the overall efficiency of path planning is how to select a tree to extend or connect. The RRT algorithm has been widely used in the literature, i.e., authors in [93] proposed a novel learning-based multi-$RRTs$ (LM-RRT) approach for robot path planning in complex environments with narrow passages. As stated in [93], this approach can guarantee global path planning efficiency and enhance each tree's local space exploration ability. Investigation of the shortest path in a minimum computational time for the global path planning carried out in [29] by using modified PSO. Authors in [94] have compared PSO and Q-learning for multi-robot obstacle avoidance. The single robot case showed that the final performance

obtained with the Q-learning approach is very similar to the one obtained with PSO. The authors in [33] presented several path planning algorithms for navigating mobile robots among obstacles and weighted regions. These algorithms can search for an optimal path and intelligently rotate the robot to pass through the narrow passages. To reduce the chances of collisions between robots and obstacles, researchers in [26] presented a new path planning technique. They assumed that the virtual obstacle's size increased approximately $(2n + 1)$ times the cell size in the workspace.

Researchers provided great effort to solve the path planning problem for mobile robots in practical application, and they proposed many new methods. For example, the authors in [14] presented preliminary results of applying a two-Kinect cameras system on a two-wheeled indoor mobile robot for off-line optimal path planning. To solve the path planning problem for rovers, authors in [12] presented a new algorithmic improvement. This study proposed OUM-BD over the Ordered Upwind Method (OUM) to include a bi-directional search. They stated that the proposed method OUM-BD is faster than the existing OUM. Authors in [13] proposed a multi-objective path planning algorithm based on improved PSO for robot navigation. For emergency evacuation simulation, authors in [25] proposed a new path planning approach. In this approach, the Extended Social Force Model ($ESFM$) is combined with the improved ABC algorithm to improve crowd evacuation efficiency. Another approach called Grid-Based Random Tree Star (GB-RRT) was developed by [11] to provide a minimum dose path for occupational workers in nuclear facilities in complex environments. The probabilistic roadmap (PRM) method has been applied by [95] to optimize the walking path and reduce the staff's radiation exposure in a radioactive environment of nuclear facilities. They showed that the proposed method has a good effect on path-planning, and it can generate a path in a short computational time.

2.1.1 Multi-Goal Path Planning

Multi-goal path planning has been widely used in many robotics applications, such as surveillance, manufacturing, autonomous inspection, and assembly. The existing methods solve the MTP in a two-step process [81]: first, constructing a path by optimizing the sequence of the goal points in a robot working environment without considering obstacles; then, constructing a path (tour) when obstacles are introduced into the environment, in which the robot must reach all sequenced goal points, and it visits each goal once. The MTP in the presence of obstacles has been discussed in the literature. Many solutions have been found in this area. For example, self-organizing map (SOM) [37, 96], branch-detected algorithm and heuristic algorithm [39], boundary iterative-deepening depth-first search ($BIDDFS$) algorithm [97], Lin-Kernighan Heuristics (LKH) algorithm [98], generalized travelling salesman problem with neighbourhoods [41], probabilistic roadmap (PRM) planner [40], ant colony optimization (ACO) combined with a sampling-based point-to-point planning algorithmhm [81], branch-detected algorithm, and heuristic algorithm [39], adaptive neural network [99],

Partially Observable Markov Decision Process ($POMDP$) framework [43], hierarchical distance computation based on the A^* search algorithm [100].

The MTP has been previously investigated for many practical problems, and several approaches have been proposed. For example, planning for a robotic arm [34, 40], hexapod walking robot [96], industrial manipulators [101], spot welding task [82,100], inspection planning [102], search and rescue mission [103], planetary exploration [39], inspection and surveillance applications [81], coordinate measuring machines ($CMMs$) [40], finding cracks in large structures [104], office-like environments to perform common tasks of picking up and delivering things such as mail, goods, trash recycled paper, etc [98]. The author in [34] considered the practical application of the robot motion planning problem, e.g., in spot-welding, car-painting, inspection, and measurement tasks [34] to compute a near-optimal path of a mobile robot.

2.1.2 Multi-Robot Multi-Goal Path Planning

In previous research work, different methods were used to solve the multi-robot multi-goal path planning problem, i.e., the authors in [47] considered the cooperative path planning problem of multiple mobile robots in an unknown indoor environment. They proposed a novel obstacle avoidance and real-time navigation algorithm. This algorithm consists of global path planning and local path planning via a hybrid artificial fish swarm algorithm ($HAFSA$) and an expansion logic strategy. The application of adaptive Charged System Search (CSS) algorithms has been used [105] to find an optimal path for multiple mobile robots . They examined these algorithms on holonomic wheeled platforms in static environments. A new planning algorithm for multi-goal path planning, called Space-Filling Forest (SFF), is proposed by [42]. The single robot ceiling vision $SLAM$ has been extended by [53] to multi-robot formations to address global localization problems in unknown environments. The author in [106] proposed improved classical Q-learning and improved (PSO) with perturbed velocity ($QIPSO$-DV) algorithm to construct an optimal collision-free path multi-robots path planning from predefined starting and goal positions for each robot in the robot working environment. The authors in [107] addressed the multi-robot multi-goal motion planning to solve the vehicle routing problem for mobile robots by using Monte-Carlo search. Additionally, new path planning and collision-avoidance methods were introduced by [49] to solve multi-robot multi-goal path planning problems. The motion planning problem for multi-robot spot-welding cells in the construction of car doors studied by [50]. The authors in [52] present a solution to an overall mission wherein a team of robots visit several mission sights and carry some operation. The multiple mobile robot solutions are extremely useful in spacecraft, rescue, transportation, etc [53].

2.2 Crowd Simulation and Path Planning

Simulation of crowd movement and the interaction between pedestrians (agents, entities, etc.) have been discussed in the literature. Many techniques have been studied for crowd simulation and modelling behaviours [78]. These techniques considered the $microscopic$ model to simulate the path for every single agent in the working environment while avoiding the obstacles and other agents [55]. The authors in [74] provided a $microscopic$ model for simulating crowd flow based on the generalized force model. In the crowd simulation literature, many methods are proposed to avoid obstacles and navigate through the environment, where the motion of dynamic obstacles and other agents in the environment is taken into account [55]. Examples of these methods including potential-based methods [108], $boid$-like methods [1, 70, 70], geometric and velocity methods [109–111], field-based methods [69, 112–114], Data-driven methods [113], Least effort crowds [76, 115–118].

For simulating large dense crowds, several techniques have been introduced in the literature. For example, [58] presents an approach for simulating the crowds using discrete agents as a single continuous system. Also, a crowd model based on continuum dynamics provided by [57]; in this model, the dynamic potential field is integrated with global navigation. The crowd flows numerically investigated by [119] using the particle methods and continuous pedestrian model based on a generalized force model reported in [74]. The authors in [59] derived the governing equations of motion for simulating different types of pedestrians in the $2D$ environment. In literature, many techniques are studied for modelling agents' behaviours. For example, the authors in [66, 75, 108] introduced a social force model, and also [75, 76] provided a cellular automaton model to simulate pedestrian behaviour in a different environment. There are many other approaches to simulate crowd behaviour based on cognitive modelling and behavior [71], psychological or sociological factors [77], personality models [64], and stress modelling [55, 120].

Moreover, the $multi-agent$ systems ($MASs$) has been used for studying human and social behaviours [69]. A $MASs$ is a system composed of a set of autonomous and heterogeneous agents distributed in an environment. The authors in [73] refer to the MAS for studying the crowd behaviour based on individuals' local behaviour and the interactions with their surrounding environment. Most of the $multi-agent$ systems perform local collision avoidance, where the global path planning techniques are needed to provide goal-directed capability [67].

The effect of groups on crowd movement was investigated in [76] using the least effort cellular automata algorithm. They show that the formation of the groups has an essential effect on the crowd movement, in which groups move slower than the individual pedestrians, and they act as a barrier that slows down the crowd. To evaluate the impact of group dynamics on the crowd movement, an $agent-based$ model is proposed

by [66, 73]. The authors in [72] suggested a multi-group $microscopic$ model based on the interacting particle system coupled with the eikonal equation for describing groups' behaviour. Crowd simulation of different groups with $heterogeneous$ behaviours in digital cultural heritage investigated by [62]. In order to generate $heterogeneous$ crowd behaviours, the pedestrians' characteristics and interactions among pedestrians need to be simulated [64]. The problem of generating $heterogeneous$ crowd behaviours by adjusting the stimulation parameters is investigated in [64]. Furthermore, [72] discussed a multi-behaviours $microscopic$ model combining with the social force model and optimal path computation. Moreover, simulating a crowd with a wide variety of pedestrians' type is essential, where the characteristics of pedestrian and surrounding environment are governor the crowd behaviours.

The authors in [121] provided a literature review from traditional models to recent models to introduce the researchers' crowd simulation models (e.g., group simulation, emotion contagion). Traditional models can simulate general crowd dynamics of both $microscopic$ and $macroscopic$ models, and the traditional models can simulate most of the usual crowds. However, the results are not as realistic as real-world human behaviour. The recent crowd simulation studies from group simulation to social psychology crowds are possible to simulate realistic crowds. Future research directions suggested aiming to develop new applications focused on more realistic, natural, and efficient crowd simulation [121].

In a real crowd simulation application, the authors in [111] proposed an algorithm to reconstruct and visualize continuous traffic flows. First, at two locations on a highway, they record each car's positions and the corresponding time instances. Then they proposed an algorithm to reconstruct the traffic flows. Another crowd simulation method is presented by [56] for generating the human-like trajectories in the simulated environment based on geometric techniques. They analyzed the $macroscopic$ flows for simulating the crowd movement, and they compared the results with the pedestrian flows seen in real human crowds. In animation and games, the authors in [1] combined steering behaviours to make the character have the ability to navigate their world in a life-like manner. Lack of data-sets of the crowd behaviours (i.e., individual tracks, personal characteristics such as $gender$, $height$, etc.) makes it difficult to verify and validate crowd modelling techniques [55].

In crowd animation, the authors in [1] combined steering behaviours to make the virtual characters have the ability to navigate their environment. Moreover, the problem of directing and controlling virtual crowds addressed by [69], they presented a method to guide virtual agents toward a desired goal position. To perform multiple virtual agents' path planning, the authors in [67] presents a $Multi - agent$ navigation graph for each agent in real-time. Having a keyframe-based interface is the right solution for tracking the movement of virtual agents (characters, pedestrians, entities, etc.) in a virtual environment in real-time. In literature, several techniques are proposed to animate crowds. Most of

these techniques use a simple representation for each agent, i.e., circular shape in $2D$ plane or cylindrical object in $3D$ space [55, 71].

Chapter 3

Proposed Method for Path Planning

3.1 Problem Formulation

The path planning problem formulation involves searching the two-dimensional environment for an optimal collision-free path for a mobile robot that connects the starting position to the goal position. Let us consider a two-dimensional ($2D$ workspace $C = R^2$ for a mobile robot, the region of space occupied by obstacles is denoted by C_{obs}, and the obstacle-free region is represented by $C_{free} = C - C_{obs}$. The continuous workspace is divided into square grid cells. The grid cells have integer coordinates in the form $C_{(x,y)} \in C$, where $1 \leq x \leq n$ and $1 \leq y \leq m$. A given cell can either correspond to the navigable area $C_{(x,y)} \in C_{free}$ or to the space occupied by obstacles $C_{(x,y)} \in C_{obs}$. Each grid cell $C_{(x,y)}$ in C_{free} has potential value $E_{(x,y)} \in E$, which is calculated according to the proposed potential function. The workspace's boundary grid cells are considered obstacles, and these grid cells are represented by $C_{BGC} \subset C_{obs}$. The robot position in the workspace is denoted by $C_{r(x_r,y_r)}$. The starting and the goal points are denoted by $C_{s(x_s,y_s)}$ and $C_{g(x_g,y_g)}$, respectively. We assume that all information related to the workspace known in advance, and the obstacles are assumed fixed, meaning that they do not change their position during the simulation.

In this study, the problem formulation extended further to solve the multi-goal path planning problem MTP for single and multiple mobile robots. A set of goal points $g_i, (i = 1...n)$ are created randomly inside the free space C_{free}, where the sequence of the goal points is unknown. Each robot is requested to visit all goal points and find the shortest path starting from the first goal g_1 and tracking it through all intermediate goal points $g_i, (i = 1...n)$, then return to the first goal g_1. The robot needs to optimize the path between goal points, and the robot is required to avoid obstacles. In this study, obstacles are assumed to be static, and the robot is treated as a dynamic obstacle for other robots. It desired to solve the path planning problem and find an optimal or near-optimal path for each robot, in which each robot reaches all goal points.

Figure 3.1 illustrates a simple example of MTP in a $2D$ working environment with obstacles. As illustrated in Figure 3.1a, the space occupied by obstacles is presented in black, and the navigable area is presented in white. The obstacles are assumed to be fixed in their positions. The goal points ($4\ goal$) are located randomly in the robot working environment, where a single point represents each goal. The robot has to reach all goal points to fulfil a given task. However, the sequence of the goal points in which they should reach is not defined. In this scenario, we can determine an optimal or near-optimal collision-free path between each pair of the goal point in two steps. First, determine the optimal sequence of the goal points over the free working space (see Figure 3.1a) to minimize the path's length. As shown in the figure, the arrows indicate the optimal sequence of the goal points. The path starts from the $1^{st}\ goal$, passing through the $2^{nd}\ goal$, $3^{rd}\ goal$, and $4^{th}\ goal$ and then return to the $1^{st}\ goal$. Once the sequence of the goal points is available, the path planning method used to construct a collision-free path between every pair of the sequenced goal points, as illustrated in Figure 3.1b. The path that starts from the initial goal point, and passing through all intermediate goal points, then returning to the initial goal point is called a multi-goal path. A multi-goal path is assembled by simply connecting goal-to-goal points. The length of the multi-goal path is the sum of the length of all goal-to-goal paths.

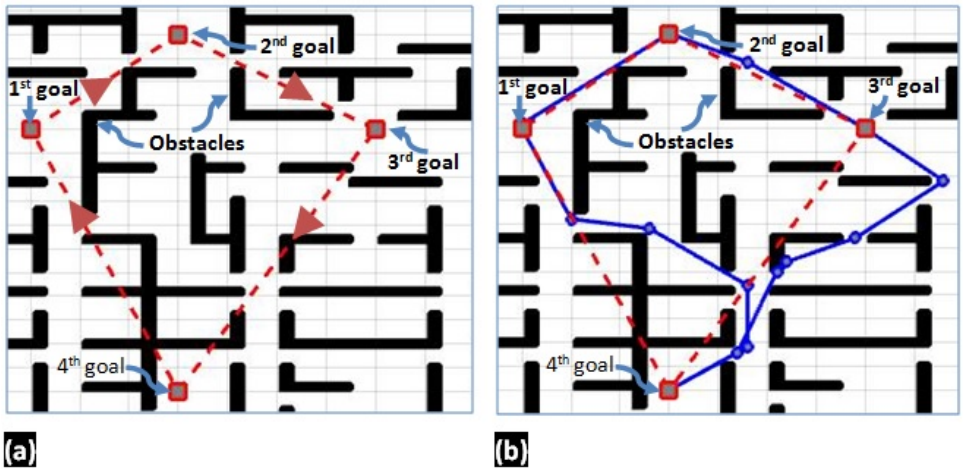


Figure 3.1: A simple example of a multi-goal path planning problem in a $2D$ workspace. (a) shows the sequence of the goal points, the path marked in red dashed line, and the goals are shown in grey square objects. (b) presents the optimal or near-optimal collision-free path for a mobile robot between the goal points, plotted in the solid blue line.

3.2 Proposed Method for Path Planning

This section describes the proposed path planning method to find an optimal or near-optimal collision-free path for a mobile robot in a $2D$ robot working environment. In the

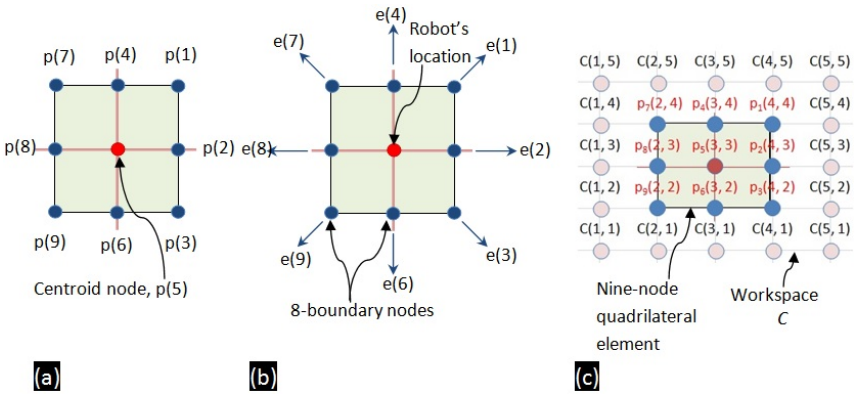


Figure 3.2: (a) a nine-node quadrilateral element along, (b) the robot motion directions, (c) the simulated robot in the exploration area.

proposed method, called Boundary Node Method (BNM), the robot is simulated by a nine-node quadrilateral element $p_{(q)}, (q = 1...9)$. The centroid node $p_{(5)}$ is considered the robot's position, and it moves with eight-boundary nodes in the working environment. The boundary nodes $p_{(1\rightarrow4)}$ and $p_{(6\rightarrow9)}$ are distributed uniformly around the robot's position, as shown in Figure 3.2a. The robot moves forward and changes its motion direction (see Figure 3.2b) based on the potential value and features of the boundary nodes. Potential value $E_{(q)}, (q = 1...9)$ for the robot and boundary nodes are equivalent to the potential value of the corresponding generated points in the workspace. The potential value is calculated based on the proposed potential function. We considered only eight-generated grid points overlapping with the eight-boundary nodes (see Figure 3.2c) rather than considering all of the generated points, which is leads to less computational time. All the visited waypoints w that the robot visits represent the obtained initial feasible path (IFP) that connects the starting point C_s to the goal point C_g.

The proposed method consists of four main steps (see Figure 3.3):

1. Construct the $2D$ grid model for the robot working environment, and then calculate the potential value of the grid cells based on the new proposed potential function.

2. Generate the initial feasible path (IFP) for the mobile robot by using Boundary Node Method (BNM).

3. Construct an optimal or near-optimal path from the IFP by using Path Enhancement Method (PEM).

4. Generate a continuous smooth path by using the cubic spline method.

Figure 3.3 shows an overview of the four steps mentioned above. The detail of each step explained in the following subsections.

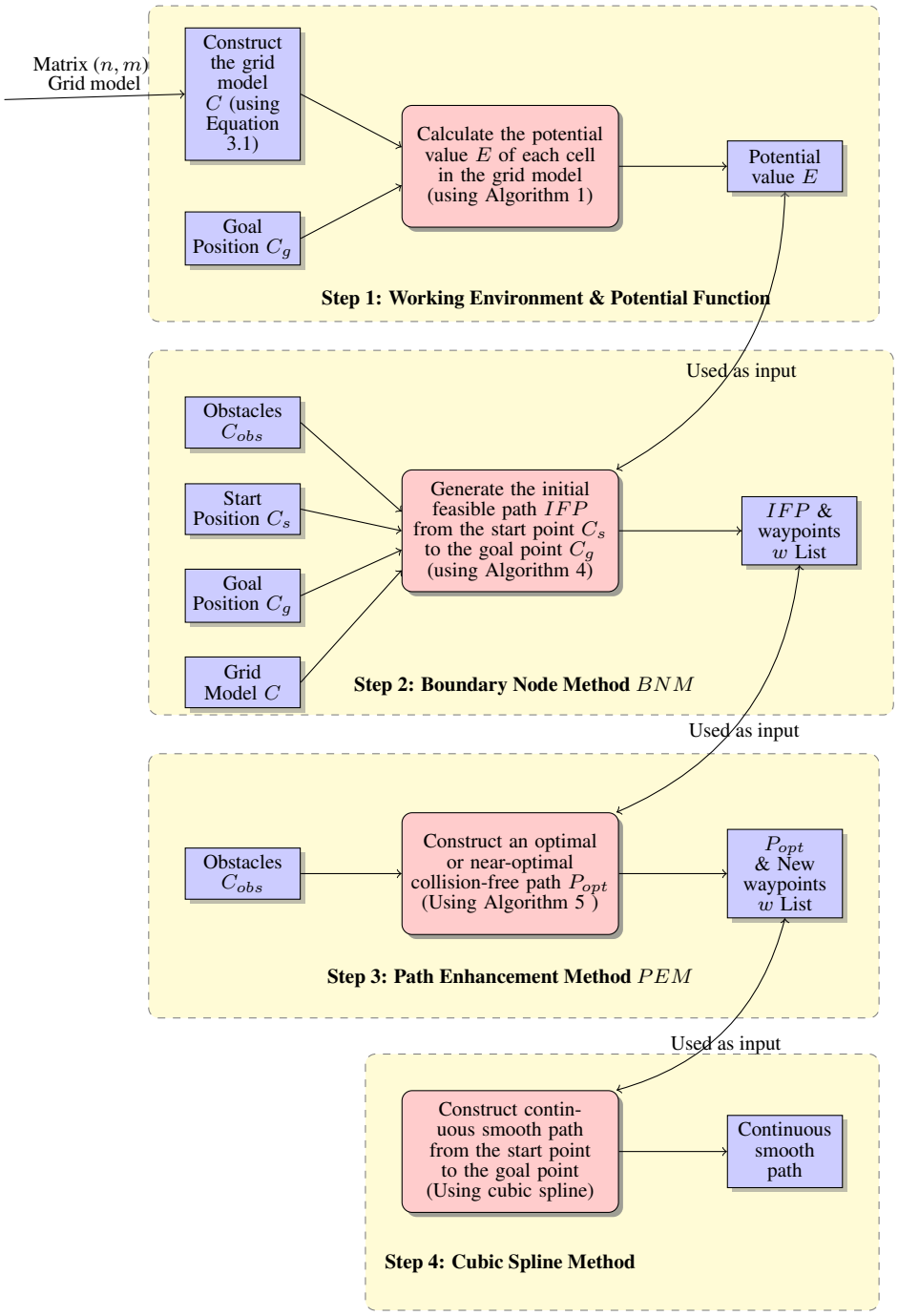


Figure 3.3: Flow diagram of the proposed method for the path planning problem.

3.2.1 Modelling of the Workspace

In the proposed method, all the grid cells of the given workspace meet the following equation:

$$C = \sum_{x=1}^{n} \sum_{y=1}^{m} C_{(x,y)} \tag{3.1}$$

n and m represent the workspace's width and height, respectively, and $C_{(x,y)}$ represents the grid cells in the workspace. After constructing the workspace model, each grid cell's potential value is calculated based on the new proposed potential function, as explained in the next subsection.

Potential Function PF

This section presents a new proposed potential function to calculate the potential value $E_{(k)}$, $(1 \leq k \leq N)$ for N grid cells in the workspace C. The procedure for calculating the potential value $E_{(k)}$ based on the proposed potential function illustrated in Algorithm 1. Figure 3.4 shows two examples of the proposed function. In this figure, the cell's colour represents the potential value. For example, the blue cell corresponds to the cell with the lowest potential value. The yellow cell corresponds to the cell with the highest potential value. As shown in the figure, the shape of the potential function is conic. This function has the lowest potential value at the goal point. The potential value increases as the robot move further away. Because the goal point has the lowest potential value, it attracts the robot to move toward that point.

Algorithm 1 Calculate the potential value of the grid cells in the workspace.

Inputs:
  C_g and $C(h,k)$, $(h = 1...2)$, and $(k = 1...N)$
Initialize:
  $E(k) \leftarrow 0, (k = 1...N)$
$D = sqrt((x_s - x_g)^2 + (y_s - y_g)^2)$
$m = ((y_s - y_g)/(x_s - x_g))$
$c = (y_s - m * x_s)$
$ll = sqrt(m^2 + b^2), \quad (b = -1)$
for k = 1 to N **do**
  $d_p(1,k) = sqrt((C(1,k) - x_g)^2 + (C(2,k) - y_g)^2)$
  $L(1,k) = m \times C(1,k) + b \times C(2,k) + c$
  $d_l(1,k) = |L(1,k)|/ll$
  $E(k) = sqrt(d_l(1,k)^2 - d_p(1,k)^2)$
end for

In Algorithm 1, the computed $E_{(k)}$ represents the potential value of each grid cell $C_{(h,k)}$, $(h = 1...2)$ and $(k = 1...N)$ in the workspace C. The minimum potential value

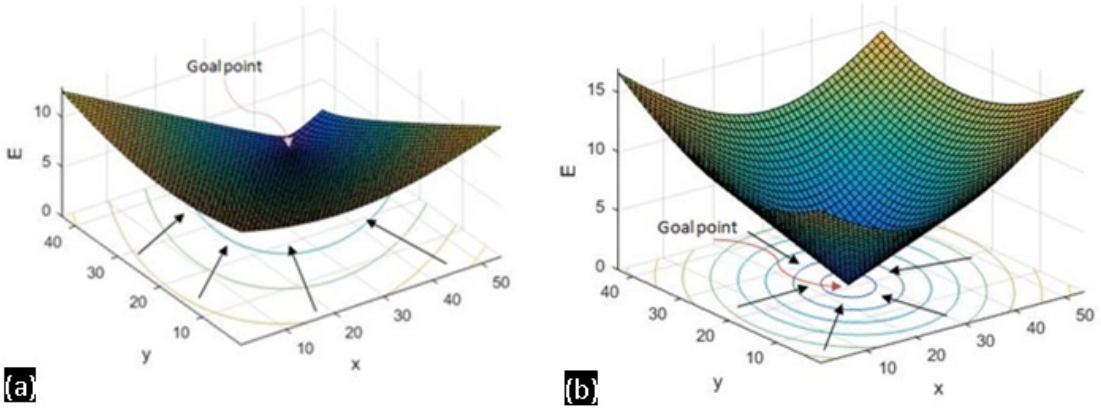


Figure 3.4: The potential value of the grid cells in the workspace in $3D$ view with contour plot. The size of the workspace is 50×50, and the the goal point C_g is located at a) $(40, 45)$ and b) $(25, 25)$.

is located at the goal point $C_{g(x_g,y_g)}$. The distance between the starting point $C_{s(x_s,y_s)}$ and the goal point $C_{g(x_g,y_g)}$ is represented by D, and the slope of a straight line D is denoted by m. The distance between the goal point $C_{g(x_g,y_g)}$ and surrounding point $C_{(h,k)}$ in the workspace is represent by $d_{p(1,k)}$.

Obstacles Representation

After constructing the workspace and calculating the potential value for each grid cells, several static obstacles are distributed at different locations in the workspace C. To reduce the proposed method's complexity, we assume that the obstacles form a set of square cells $(1 \times 1\ unit)$. The centre of the obstacle's cells are defined by a matrix $C_{obs(h,l)}$, $(h = 1, 2)$ and $(l = 1...O)$, where O represents the number of obstacles. The distance d between the centre and the edge of the obstacle is constant, $d = 0.5\ unit$. As the robot moves close to the obstacle, they should keep a certain margin for safety. In this study, to avoid the possibility of overlapping the paths traced by the robot and obstacle boundary, a safety zone around the obstacles is created.

An example of three different workspace scenarios with varying layouts of obstacles shown in Figure 3.5. The characteristics of these three workspace scenarios are illustrated in Table 3.1. The workspaces are shown in Figure 3.5 divided into square grid cells, where each cell is considered either an obstacle C_{obs} or non-obstacle C_{free}. The potential value of the grid cells in the C_{free} is calculated based on the proposed potential function, as illustrated in Algorithm 1. The workspace's grid cells use different colours to differentiate between C_{free} and C_{obs}. The black cells represent C_{obs}, and the coloured cells represent the potential value in the C_{free}. The safety zone defined by several grey square grid cells of the size $(1 \times 1\ square\ unit)$ around the obstacles.

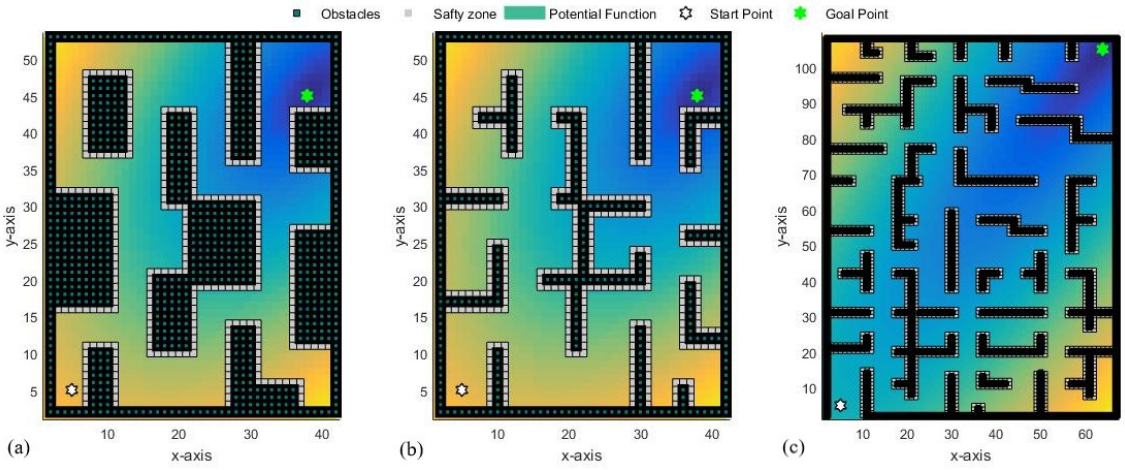


Figure 3.5: $2D$ models of the robot working environment with obstacles.

Table 3.1: Characteristics of three different workspace scenarios.

Workspace *No.*	$\boldsymbol{C}_{s(x,y)}$	$\boldsymbol{C}_{g(x,y)}$	**Workspace [*cells*]**	**Obstacles [*cells*]**
1	(5,5)	(38,45)	2226	770
2	(5,5)	(38,45)	2226	345
3	(5,5)	(65,105)	7303	904

For the first (see Figure 3.5a) and second (see Figure 3.5b) designed scenarios, obstacles represent about 34.6% and 15.5% of the workspace, respectively. In the third scenario (see Figure 3.5c), a more complex environment with a high number of obstacles is considered, and obstacles represent about 12.4% of the workspace. After constructing the workspaces with obstacles and calculating the potential value of the grid cells, the BNM method is used to determine the collision-free path, the detail of the developed method is described in the following subsection.

3.2.2 Boundary Node Method (BNM)

The BNM method consists of three steps:

1. Simulate the robot,

2. Movement of the robot in the workspace, and

3. Obstacle avoidance

Simulate the robot

In the simulated model, the nodes are denoted by $p_{(q)}, (q = 1...9)$ and their locations are formulated by Equation 3.2. At any iteration t, the current location of nodes denoted by $p_1(t)$. The x and $y-$ coordinates of the nodes' location represented by two vectors $x_1(t) = (x_{11}, x_{12}, ..., x_{19})$ and $y_1(t) = (y_{11}, y_{12}, ..., y_{19})$, respectively. The current location of nodes $p_1(t)$ is formed by vertically concatenating $x_1(t)$ and $y_1(t)$, $p_1(t) = [x_1(t);\ y_1(t)]$.

$$p(q) = \begin{cases} x, y & q\text{=5} \\ (x + v_x, y), (x, y + v_y), (x - v_x, y), (x, y - v_y) & q = 2, 4, 6, \text{and } 8 \\ (x + v_x, y + v_y), (x - v_x, y + v_y), (x - v_x, y - v_y), (x + v_x, y - v_y) & q = 1, 3, 7, \text{and } 9 \end{cases} \tag{3.2}$$

where x and y represent the coordinate of the robot location p_r, and v_x and v_y represent the horizontal and vertical distances between p_r and boundary nodes, $v_x = v_y = 1$ $unit$. The nodes $p_1(t)$ can move only in eight-possible directions $e_{(u)}, (u = 1...8)$ (see Figure 3.2*b*), as explained in the next subsection.

Movement of the robot in the workspace

At each iteration t, the current location of the robot and boundary nodes $p_1(t)$ move in one particular direction. The new updated nodes' locations $p_2(t)$ are calculated according to the following equations:

$$x_2(t) = x_1(t) + \Delta x \tag{3.3}$$
$$y_2(t) = y_1(t) + \Delta y \tag{3.4}$$
$$p_2(t) = [x_2(t);\ y_2(t)] \tag{3.5}$$

where $x_2(t)$ and $y_2(t)$ represent the coordinates of the new updated nodes' locations.

The values of the new updated nodes' locations $p_2(t)$ depend on the current nodes' locations $p_1(t)$ and the values of Δx and Δy, where Δx and Δy are computed by using Algorithm 2. In this method, the values of g_x and g_y represent the distance between the current location of the robot $p_{r(x_r,y_r)}$ and the goal point C_g in x and y directions, respectively. The variables s_x and s_y represent the variation of the potential value between $p(2)\&p(8)$ and between $p(4)\&p(6)$, respectively. The values of s_x and s_y are calculated by using Equation 3.6 and 3.7.

$$s_x(t) = E(p_1(1,8), p_1(2,8)) - E(p_1(1,2), p_1(2,2)) \tag{3.6}$$
$$s_y(t) = E(p_1(1,6), p_1(2,6)) - E(p_1(1,4), p_1(2,4)) \tag{3.7}$$

The values of Δx and Δy have the same sign as the variation of the potential value (both positive or both negative). The coefficients α and β influence the convergence

Algorithm 2 Calculate the values of Δx and Δy.

Inputs:
    C_g, $p_r(t)$
$E(q), (q = 1...9) \leftarrow E$
$s_x, s_y \leftarrow$ Equation 3.6 and 3.7
$g_x = x_r(t) - x_g$
$g_y = y_r(t) - y_g$
if $s_x < 0$ **then**
    compute $g_x = -1 * g_x$
end if
if $s_y < 0$ **then**
    compute $g_y = -1 * g_y$
end if
if $g_x = 0$ **then**
    compute $\Delta x = 0$ and $\Delta y = \beta * g_y$
else if $g_y = 0$ **then**
    compute $\Delta x = \alpha * g_x$ and $\Delta y = 0$
else
    compute $\Delta x = \alpha * g_x$ and $\Delta y = \beta * g_y$
end if

behaviour. The distance between $p_r(t)$ and C_g is decreasing step by step until the robot reaches the global minimum at the goal location.

The BNM method uses an optimization technique based on the lowest potential value to accelerate the robot to find the path and yield to fast convergence. Among all boundary nodes, the node with the lowest potential value is chosen as the best position and denoted by p_{best}. At each iteration t, the robot update its position to the best position p_{best}. The boundary nodes help the robot move toward the goal location and guide the robot to avoid obstacles, which we will discuss in the next subsection.

Obstacle Avoidance

In the workspace with no obstacles, the robot will reach the goal point along a straight line from any starting point. As obstacles exist, the robot interferes with obstacles when the distance between the robot and the obstacles is less than the distance d. Therefore, the robot and boundary nodes must avoid obstacles and change their moving direction by selecting a new position in the C_{free}.

To explain the obstacle avoidance, consider a simple example shown in Figure 3.6. Initially the boundary nodes $p(1 \rightarrow 4)$ and $p(6 \rightarrow 9)$ are generated around the robot position $p(5)$ by using Equation 3.2. As shown in Figure 3.6a, the red object represents

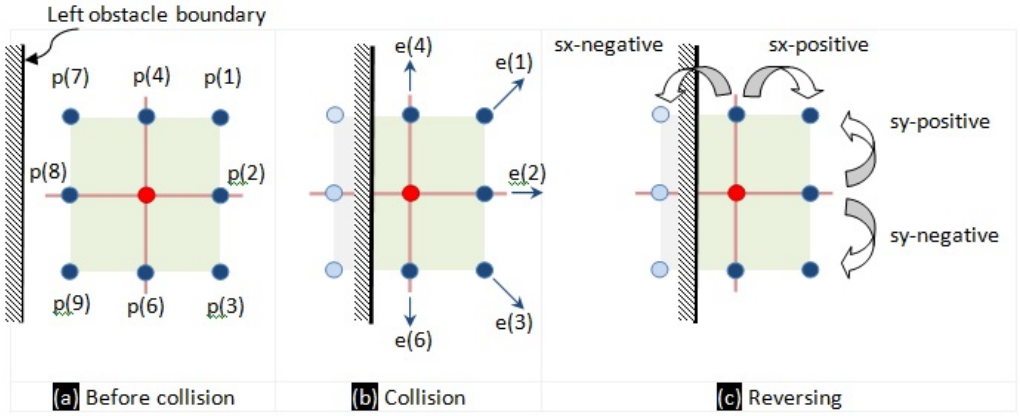


Figure 3.6: Illustrates the collision avoidance in the static environment by using the BNM method, (a) the initial positions of the robot and boundary nodes, (b) the new updated positions, (c) obstacles avoidance and change the motion direction.

the robot's position, and the blue objects represent the boundary nodes. At iteration t, the robot and boundary nodes are changing their positions from the current position (see Figure 3.6a) to the newly updated position (see Figure 3.6b) by using Equations 3.3, 3.4, and 3.5. As a result, the nodes $p(7)$, $p(8)$, and $p(9)$ interfere with the obstacles (see Figure 3.6b). Therefore, the robot needs to investigate the workspace to find the next position without colliding obstacles. In this case, the robot will move in the y-direction either to the upward or to the downward direction. The motion direction depends on the value of s_y (see Figure 3.6c), the robot moves backwards if $s_y(t)$ is negative, and the robot moves forward if $s_y(t)$ is positive.

To demonstrate how the robot avoids the obstacles and changes its motion direction with the help of boundary nodes, let us consider an illustrative example shown in Figure 3.7. As demonstrated in Figure 3.7a and 3.7b, in iterations $t = 1 \rightarrow 4$, the robot starts to move from $p_1(t)$ to $p_2(t)$ toward the goal point by using Equations 3.3, 3.4, and 3.5. At each iteration, all obstacles in the working environment are examined for possible collision with the direct path from $p_1(t)$ to $p_2(t)$. As the robot moves toward the goal point C_g in the iteration $t = 5 \rightarrow 6$, nodes $p(1)$, $p(2)$, and $p(3)$ interfere with obstacles (see Figure 3.7c). This problem implies that the robot can only move in y-direction, either upward (if s_y is positive) or downward (if s_y is negative). The next position of the robot must be in an upward direction because the value of s_y is positive ($E(6) > E(4)$). The same procedure is repeated for iterations $t = 7 \rightarrow 10$ by shifting the robot upward until the robot passes the block of obstacles, as shown in Figures 3.7d and 3.7e. For the iterations, $t = 11 \rightarrow 16$, the BNM method directs the robot to move forward (see Figures 3.7f and 3.7g) until the robot reaches the final destination point at C_g (see Figure 3.7h).

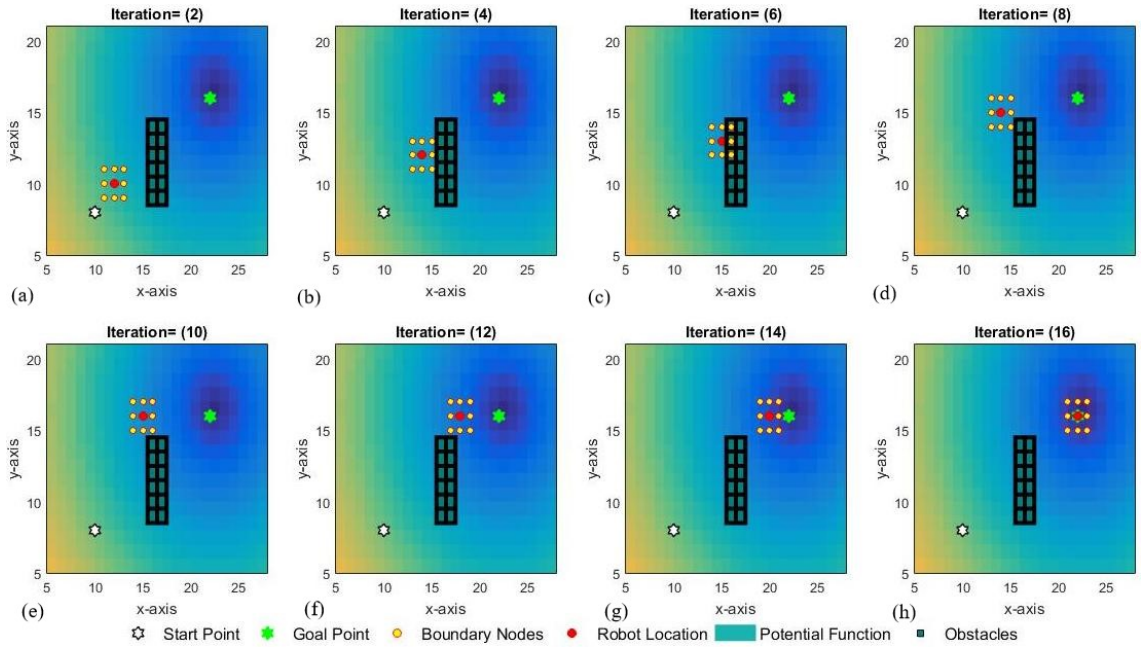


Figure 3.7: Demonstrate the mobile robot exploration in a $2D$ working environment by using BNM.

Suppose that the long horizontal set of obstacles block the robot path as illustrated in Figure 3.8a. While the robot moves toward the goal point, nodes $p(1)$, $p(4)$, and $p(7)$ interfere with obstacles. Therefore, the robot needs to change its motion direction along x-direction to avoid the obstacles. The motion direction depends on the value of s_x. If $s_x(t)$ is positive, the robot moves to the right, and if $s_x(t)$ is negative, the robot moves to the left. In this case, the nodes at $p(8)$ and $p(2)$ have the same level of the potential value $E(8) = E(2)$. This implies that the variation of the potential value between $p(2)$ and $p(8)$ is equal to zero ($s_x = 0$). Therefore, the robot moves in both directions (see Figure 3.8b). As shown in the figure, nodes $p(7)$, $p(8)$, and $p(9)$ move one step to the left and nodes $p(1)$, $p(2)$, and $p(3)$ move one step to the right at each iteration. Two temporary sets, described as a "waiting list", of visited grid cells on the left and right-side are stored. As the robot reaches the end of obstacles on the left-side earlier (see Figure 3.8c), the BNM method chooses the stored data-set on the left-side and disregards the stored data on the right-side.

The BNM method introduces a new technique to solve the local minima problem, as illustrated in Algorithm 3. This algorithm executes a sequence of steps that pulls the robot out of the local minima area. For instance, consider the workspace with a U-shaped obstacle shown in Figure 3.9, the robot starts to move at position $(3, 15)$ (see Figure 3.9a) toward the goal point at $(23, 3)$. Similarly, in Figure 3.9b, the robot moves from $(23, 15)$ to $(3, 3)$. The robot uses two different modes while moving in the

Algorithm 3 Illustrates the steps to bring the robot out of the local minima area.

Inputs: $C_{obs}, s_x, s_y, p_1(t), p_2(t)$
Check line segments between $p_{1(q),(q=1...9)}(t)$ and $p_{2(q),(q=1...9)}(t)$ for feasibility
If line between $p_{1(q)}(t)$ and $p_{2(q)}(t)$ interfered C_{obs} then $chk_{(q)} = 1$ otherwise $chk_{(q)} = 0$
Construct matrix $chk_{(q),(q=1...9)}$
while $sum(chk) > 0$ **do**
    if $chk_{(1)}, chk_{(2)},$ and $chk_{(3)} = 1$ **then**
        $p_{2x}(t)$=$p_{2x}(t)$-c_1, $c_1\, is\, constant$
        if $s_y > 0$ then $p_{2y}(t)$=$p_{2y}(t)$+c_2 otherwise $p_{2y}(t)$=$p_{2y}(t)$-c_2
        repeat $steps\, 2, 3,\, and\, 4$
        update $p_{1y}(t) \leftarrow p_{2y}(t)$
        store p_5 in a waypoints w list
    end if
    if $chk_{(1)}, chk_{(4)},$ and $chk_{(7)} = 1$ **then**
        $p_{2y}(t)$=$p_{2y}(t)$-c_2, $c_2\, is\, constant$
        if $s_x > 0$ then $p_{2x}(t)$=$p_{2x}(t)$+c_1 otherwise $p_{2x}(t)$=$p_{2x}(t)$-c_1
        repeat $steps\, 2, 3,\, and\, 4$
        update $p_{1x}(t) \leftarrow p_{2x}(t)$
        store p_5 in a waypoints w list
    end if
    if $chk_{(7)}, chk_{(8)},$ and $chk_{(9)} = 1$ **then**
        $p_{2x}(t)$=$p_{2x}(t)$+c_1
        if $s_y > 0$ then $p_{2y}(t)$=$p_{2y}(t)$+c_2 otherwise $p_{2y}(t)$=$p_{2y}(t)$-c_2
        repeat $steps\, 2, 3,\, and\, 4$
        update $p_{1y}(t) \leftarrow p_{2y}(t)$
        store p_5 in a waypoints w list
    end if
    if $chk_{(3)}, chk_{(6)},$ and $chk_{(9)} = 1$ **then**
        $p_{2y}(t)$=$p_{2y}(t)$+c_2
        if $s_x > 0$ then $p_{2x}(t)$=$p_{2x}(t)$+c_1 otherwise $p_{2x}(t)$=$p_{2x}(t)$-c_1
        repeat $steps\, 2, 3,\, and\, 4$
        update $p_{1x}(t) \leftarrow p_{2x}(t)$
        store p_5 in a waypoints w list
    end if
end while
return $p_1(t), p_2(t), w$

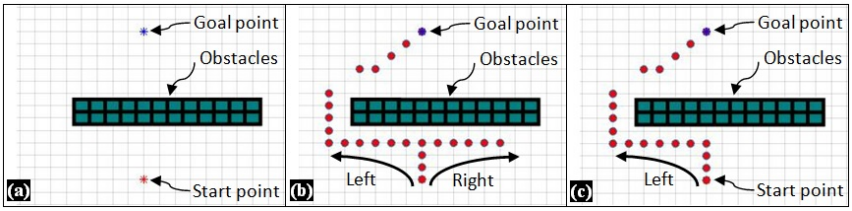


Figure 3.8: The workspace contains long horizontal obstacles that block the path of the robot.

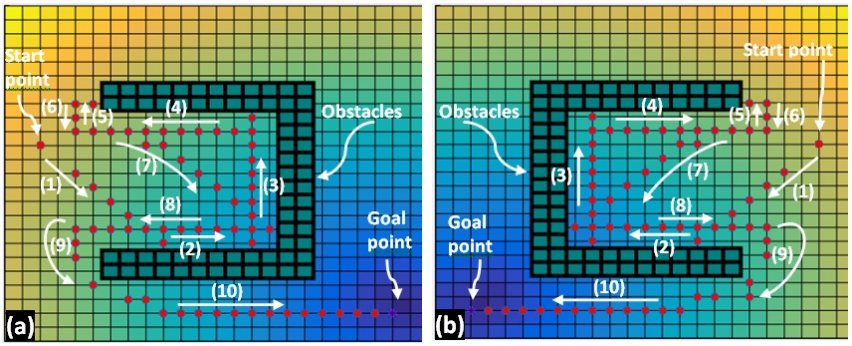


Figure 3.9: An illustrative simulation example of the local minima problem in a 2D environment with U-shape obstacle.

simulated environment, namely the "normal mode" and the "local minimum recovery mode". In the normal mode, for iteration $(t)_{t=1,..,6}$, as the robot moves from the point $p_1(t)$ toward the point $p_2(t)$, the line between $p_1(t)$ and $p_2(t)$ does not intersect with the obstacles (see $step\,(1)$ Figure 3.9). In order to check the feasibility of the path represented by each line segment between corresponding points in $p_1(t)$ and $p_2(t)$, we create a new row matrix $chk_{(q)}, (q = 1...9)$. The value of each element of the row matrix is equal to "0" or "1". At iteration t, for the $1^{st}\,element\,(q = 1)$, if the line between the first node of the simulated model in $p_1(t)$ and $p_2(t)$ intersects with the obstacles, then the value $chk_{(q)} = 1$, otherwise $chk_{(q)} = 0$. The same procedure is repeated for the $2^{nd}\,element\,(q = 2)$, $3^{rd}\,element\,(q = 3)$ until the last $element\ (q = 9)$. In the normal mode, all the values of $chk_{(q)}, (q = 1...9)$ are equal to "0", and the robot travels with the help of Algorithm 4. In the recovery mode, the robot switches to Algorithm 3, as shown in $steps\,(2 \rightarrow 9)$ in Figure 3.9. In $t = 7$, as the robot moves forward from $p_1(t)$ to $p_2(t)$, the line segment connecting corresponding nodes $(3, 6, and\,9)$ intersects the obstacles (see Figure 3.9). In this case the values of $chk_{(3)}$,$chk_{(6)}$, and $chk_{(9)}$ are equal to "1", then the robot moves to the right (see Figure 3.9a) or to the left (see Figure 3.9b) with the help of the Algorithm 3. Once the robot has successfully come out of the local minima, it can move smoothly again by using Algorithm 4.

The proposed potential function is similar to the attractive potential field in the sense

that both guide the robot to move toward the desired goal location, but the process for calculating the potential value E is different (see Algorithm 1). In this study, the potential value $E_{(1,k)}$ calculated using Equation 3.8.

$$E_{(1,k)} = f(C_g, C_{(h,k)}) \tag{3.8}$$

As illustrated in Figure 3.9, the robot starts to move in $step\,(1)$ toward the goal until it collides with obstacles. When the simulated robot detects a collision, the BNM method defines the interfered points' position in the boundary nodes by using Equation 3.9.

$$p_{(t,h)} = [x_2(t); y_2(t)], p_{(t,h)} \in R^2 : p_{(t,h)} = (p_r \cap C_{obs}) \tag{3.9}$$

To avoid obstacles, the new updated location of nodes $p_2(t)$ is calculated according to the Equations 3.10 and 3.11, as follow:

$$x_2(t) = x_1(t) + f(E_{(1,k)}, p_{(t,h)}, p_1, p_2, C_{obs}) \tag{3.10}$$
$$y_2(t) = y_1(t) + f(E_{(1,k)}, p_{(t,h)}, p_1, p_2, C_{obs}) \tag{3.11}$$

For the $step\,(2)$ to $step\,(9)$ (see Figure 3.9), the BNM method gives the highest priority to the obstacle avoidance processes and the lowest priority to the potential value. Afterwards, in the $step(10)$, the proposed method gives the highest priority to the potential value until the robot reaches the goal point. As shown in Figure 3.9, the robot did not block the U- shape obstacles; it always finds the path (if it exists) to reach the final destination point.

In this study, the BNM method is used to generate the IFP for the mobile robot to move from the starting C_s to the goal C_g points without colliding with any obstacles. The IFP is generated from a set of waypoints w that the robot visits before reaching the final destination point at C_g. For better clarity, the waypoints are connected into a continuous path. The line segment connects two waypoints in sequence represented by $P_{l,l+1}$. The length of all line segments that connect all waypoints sequentially represents the length of IFP. A complete path IFP is formed by concatenation of all inter-line segments $P_{l,l+1}, 1 \leq l \leq w-1$ as follows: $IFP = [P_{1,2}, P_{2,3} \ldots, P_{w-1,w}]$. The main steps to find IFP for the mobile robot using the BNM method are summarised in Algorithm 4.

According to Algorithm 4, the robot starts to move at the point $C_{s(x_s,y_s)}$ toward the goal point $C_{g(x_g,y_g)}$. The current nodes' location $p_1(t)$ of all nodes $p_{(q)}, (q = 1...9)$ at iteration t is formulated by Equation 3.2, where the x and $y-$ coordinates of the robot location p_r at the first iteration coincide with the x_s and y_s of the start point $C_{s(x_s,y_s)}$. The node with the lowest potential value among all boundary nodes is chosen as the best position and it is denoted by p_{best}, where the potential values $E_{(q)}, (q = 1...9)$ of

nodes computed by using Algorithm 1. For iteration t, the new updated location of nodes $p_2(t) = [x_2(t); y_2(t)]$ calculated by Equations 3.3, 3.4 and 3.5. The variation of the potential value s_x and s_y calculated by using the Equations 3.6 and 3.7. Afterwards the line segments between $p_1(t)$ and $p_2(t)$ check for feasibility. If the collision is not found, then a new set of $E_{(q)}, (q = 1...9)$ and p_{best} need to be calculated, as previously explained. Subsequently, the current location $p_1(t)$ updates to the newly calculated location $p_2(t)$, and the robot $p_r(t)$ updates its position to the best position at p_{best}. The BNM method stores the robot's location $p_r(t)$ in a waypoints w list. On the other hand, if the line segments between $p_1(t)$ and $p_2(t)$ collides with obstacles, another updated location for $p_2(t)$ needs to be found, as explained in subsection above. This procedure will continue until the mobile robot reaches the final destination point at $C_{g(x_g,y_g)}$ or the maximum number of iterations is reached.

Algorithm 4 Boundary Node Method (BNM)

Inputs:
C_s, C_g, C_{obs}, and $C(x, y), (x = 1...n, y = 1...m)$, maximum iteration number M
Initialize:
$p_1 \leftarrow Equation\ 3.2,\ x = x_s,\ y = y_s$
$E(k), (k = 1...N) \leftarrow$ Algorithm 1
$E(q), (q = 1...9) \leftarrow E$
$p_{best} \leftarrow$ minimum $E(q)$
$s_x, s_y \leftarrow$ Equation 3.6 and 3.7
while $(x_r \neq x_g \quad or \quad y_r \neq y_g \quad$ within a $M)$ **do**
    $p_2(t) \leftarrow$ Equation 3.3, 3.4, and 3.5
    $s_x, s_y \leftarrow$ Equation 3.6 and 3.7
    Check the line segment between $p_1(t)$ and $p_2(t)$ for feasibility
    if $p(t)$ interfered with C_{obs} **then**
        $p_2(t) \leftarrow Obsticle\ Avoidance$
    end if
    $E(q), (q = 1...9) \leftarrow E$
    $p_{best} \leftarrow$ minimum $E(q)$
    $p_1(t) \leftarrow p_2(t)$
    $p_r(t) \leftarrow p_{best}$
    $p_r(t)$ in a waypoints w list
end while
$IFP \leftarrow$ waypoints w list
$P_{opt} \leftarrow$ Algorithm 5
$U \leftarrow$ Equation 3.12
End

Time complexity is the computational complexity that estimates the run-time of an algorithm. In the developed method, the computational time required for generating the

waypoints w of the IFP is calculated based on the computational time needs for the line "2" to "18" in Algorithm 4. To analyze the time complexity of the developed method, we assumed that the size of the simulated model is defined by q ($q = 9$), the number of iterations is defined by M, the problem size is denoted by N ($N = n \times m$), and the number of iterations needed by the robot to pass the block of obstacles is denoted by M_1.

1. In step 2, the time complexity of computing $p_1(q), (q = 1...9)$ is $T_1 = O(q)$.
2. In step 3, the time complexity of computing $E(k), (k = 1...N)$ is $T_2 = O(N)$.
3. In steps 4-6, the time complexity of calculating $E(q)$, p_{best}, s_x, and s_y, is $T_3 = O(q)$.
4. In steps 4-6, 8-9, the time complexity of calculating $p_2(t)$ is $T_4 = O(M * q)$.
5. In step 10, the time complexity of collision test between a line segment (between $p_1(t)$ and $p_2(t)$) and obstacles is $T_5 = O(M * N * q)$.
6. In steps 11-13, in case the line segment between $p_1(t)$ and $p_2(t)$ collides with obstacles, the time complexity of these steps is $T_6 = O(M * N * M_1 * q)$.
7. In steps 14-18, the time complexity of determining the new set of $E(q), (q = 1...9)$ and p_{best}, together with updating p_1 and p_r, and store p_r in a waypoints w list is $T_7 = O(q)$.

The total time complexity of the developed method is: $T = T_1 + T_2 + T_3 = T_4 + T_5 + T_6 + T_7\ T = O(q) + O(N) + O(q) + O(M*q) + O(M*q*N) + O(M*N*M_1*q) + O(q) = O(N * M * M_1)$

The obtained IFP is a safe path for a mobile robot between C_s and C_g; however, it is not the shortest path. A new method, called Path Enhancement Method PEM, is developed to reduce the overall path length, as explained in the next subsection.

3.2.3 Path Enhancement Method (PEM)

This section introduces the PEM method to generate the shortest path (see Figure 3.10a) from IFP (see Figure 3.10b). The PEM method is used to reduce the number of waypoints of the IFP to obtain an optimal or close-to-optimal path. As shown in Figure 3.10, the waypoints of the IFP are represented by red circle objects, and the thick red line represents the obtained shortest path. To explain the basic idea of PEM, consider an illustrative example shown in Figure 3.11.

As illustrated in Figure 3.11a, the IFP consists of "14" waypoints w, and they are connected by line-segments. In this example, the robot starts to move from the starting point and passes through all the intermediate waypoints until it reaches to the goal point. As shown in the Figure 3.11b, the line segment U has two end points (let's say u_1 and

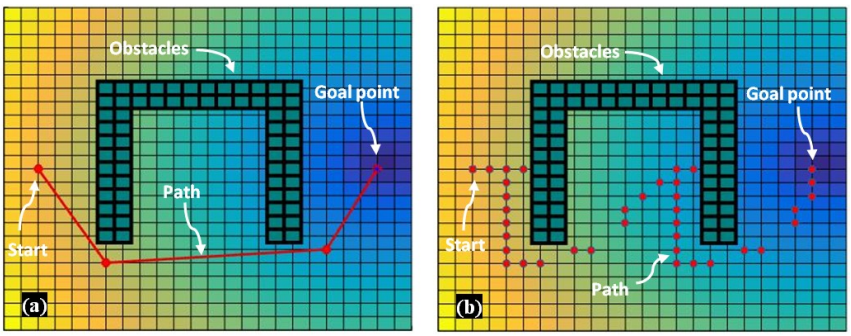


Figure 3.10: An example of the path planning for the robot navigation in the C -space. (a) The shortest path is found using PEM, where the solid red line represents the shortest path. (b) The obtained solution of IFP by using BNM, where the sequence of the red circle objects represents the IFP.

u_2). For the first line segments U_1, the starting position of u_1 coincides at the C_s. In order to determine the position of u_2, the PEM method connects u_1 with $w_{(j)}, (j = 1...J)$, $J = 14$, iteratively. First, u_1 is connected with the first waypoint $w_{(1)}$, then the line between these two points is checked for feasibility. If a collision is not found, then u_1 is connected to $w_{(2)}$. Afterward the line between u_1 and $w_{(2)}$ is checked for feasibility. If the line does not collide with any obstacles, then u_1 is connected to $w_{(3)}$, and this procedure continues in the same way until $j = 12$. When $j = 12$, u_1 is connected to $w_{(12)}$; in this case the line between these two points collides with obstacles, as shown in Figure 3.11a. Therefore, u_2 of the first line segment is placed in $w_{(11)}$. For the second line segment U_2, the left-hand end u_1 is coincides at u_2 of the first line segment. In order to find u_2 of the second line segment, the PEM connect u_1 with $w(j), (j = 12...14)$, iteratively. Therefore, u_1 is connected with $w_{(12)}$, $w_{(13)}$, and $w(14)$ one after another, and the lines between u_1 and these points are check for feasibility. As shown in Figure 3.11a, these line segments did not collide with obstacles. Therefore, u_2 of the second line segment is placed in C_g. The total length of the shortest path U is calculated by summing the length of all the line segments $U(i)$ in the path between C_s and C_g, as follows:

$$U = \sum_{i=1}^{I} (sqrt(u_{1x}(i) - u_{2x}(i))^2 + (u_{1y}(i) - u_{2y}(i))^2) \tag{3.12}$$

where I represents the number of the line segments, which is equal to "2" in this example. $u_{1x}(i)$, $u_{2x}(i)$, $u_{1y}(i)$, $u_{2y}(i)$ represent the coordinates of the endpoints of the line segment $U(i)$. The general procedure of PEM is illustrated in Algorithm 5.

3.2.4 Smooth Path Planning

The generated path obtained by BNM and PEM may contain sharp turns; this goes against many real-world applications where smooth paths are preferred [122]. Moreover,

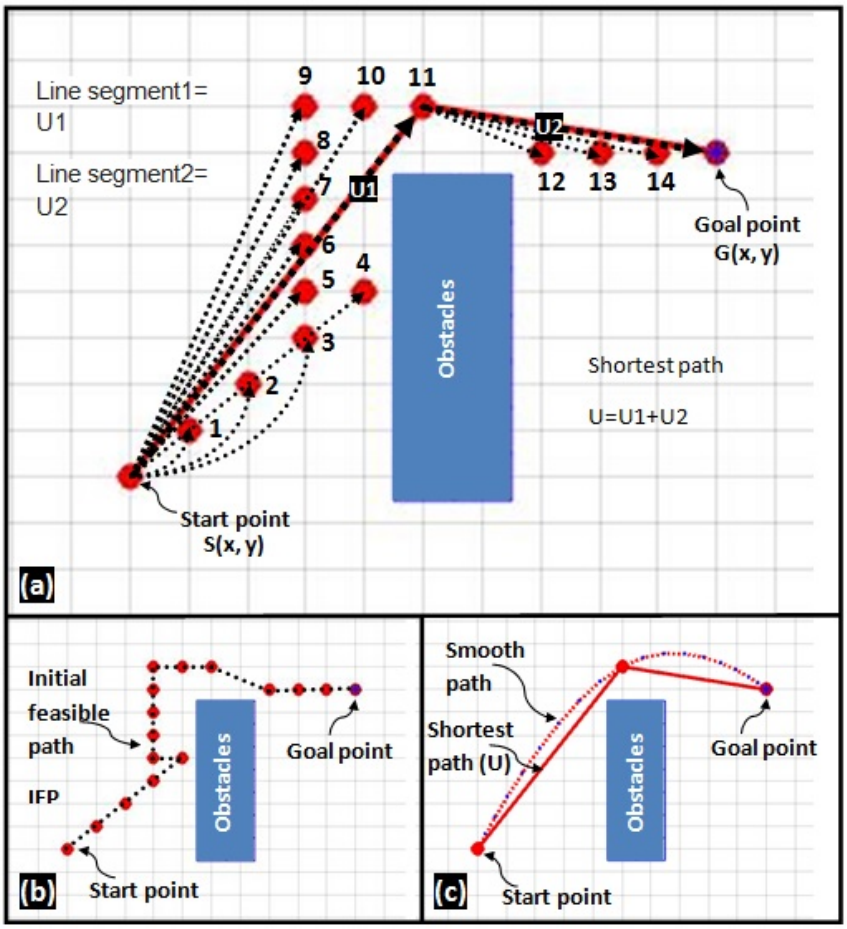


Figure 3.11: Create the shortest path from "14" waypoints in the 2D workspace, where the red circle objects sign the waypoints. (a) construct the shortest path by using the PEM method. (b) generate the IFP by using the BNM method. (c) determine the shortest path using PEM and the smooth path by applying the spline method.

Algorithm 5 Path Enhancement Method (PEM)

Inputs:
    C_{obs}, and $w(j), (j = 1...J)$
$j \leftarrow 1$
while $j \leq J$ **do**
    $u_2 = w(j)$
    check the line U_i for feasibility between u_1 and u_2
    if $U_{(i)}$ collide with C_{obs} **then**
        store u_1, $u_2 = w(j-1)$
        $u_1 \leftarrow w(j-1)$
    end if
    $j \leftarrow j + 1$
end while
inserts C_s and C_g to the beginning and the end of the new waypoints list.

the robot may not be able to make a sharp turn due to its momentum [17, 21]. Therefore, the cubic spline interpolation is adopted to generate a continuous smooth path that connects the starting point to the goal point. The spline method is one of the most efficient curve interpolating methods, which has many applications in robotics, signal processing, and computer graphics [4, 21].

Consider the generated path by using PEM as shown in Figure 3.11c, the path consists of two line segments U_1 and U_2 between C_s and C_g in the form of X and Y vectors, where X=[x_1 x_2 x_3] and Y=[y_1 y_2 y_3]. The cubic spline interpolation is used to calculate the spline for three waypoints ($w = 3$). Therefore, a new vector h is generated for "200" points between the starting point (x_1, y_1) and the goal point (x_3, y_3). The vectors of interpolated values x_{sp} and y_{sp} calculated based on the equations x_{sp}=$Spline(h_n, x, h)$ and y_{sp}=$Spline(h_n, y, h)$, where h_n = [1 2 3]. Then, we draw the line that passes through these points. The constructed path passes smoothly through the waypoints thus eliminating the sharp turn, as illustrated in the Figure 3.11c.

3.2.5 Multi-Goal Path Planning

This section describes using the proposed method to construct the shortest path between several goals points $g_i, (i = 1...n)$. As explained in subsection 3.2, the first step of the developed method is constructing a $2D$ grid model for the robot working environment. The potential value of each grid cell is calculated based on the new proposed potential function. In the second step, the initial feasible path (IFP) between each pair of goal points is determined by using BNM. The IFP is generated from the waypoints w that the robot has to traverse as it moves toward the destination point without colliding obstacles. In the third step, the PEM method is used to construct an optimal or near-optimal path by reducing the waypoints w and the overall path length. The detail of

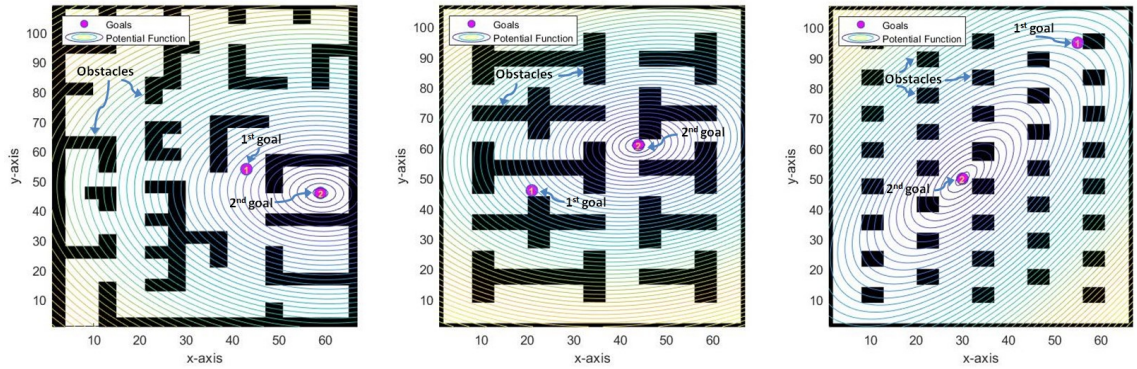


Figure 3.12: There are three different working environments with varying layouts of obstacles, the size of the workspace is 67×109, and the position of g_1 and g_2 located randomly in the free space C_{free}. The contour lines represent the potential value, and the pink circle objects represent the goal points, g_1 and g_2.

these steps explained in the previous section. An example of three different workspace scenarios with varying layouts of obstacles shown in Figure 3.12. In these scenarios, the workspace consists of (67×109) grid cells. As shown in Figure 3.12, the number of static obstacles $(1 \times 1\ unit^2)$ are distributed at different locations in the workspace, where the number of grid cells occupied by obstacles in these workspaces equal to "1682", "1060", and "956" grid cells. The white region represents the C_{free}, and the black objects represent the C_{obs}. The goal points are positioned randomly in the free workspace C_{free}.

After constructing the workspaces' models, the potential value E is calculated for each grid cell in the workspace C based on the proposed potential function. This function is used to direct the robot from the starting goal point $g_1(x_{g1}, y_{g1})$ toward the destination goal point $g_2(x_{g2}, y_{g2})$. The potential function has the lowest potential value at g_2, and the potential value increases as the robot move further away. As shown in Figure 3.12, the line's colour represents the potential value, i.e., the blue line corresponds to the lowest potential value, and the yellow line corresponds to the highest potential value. Subsequently, the BNM method is used to generate the IFP for the robot to move between goal points in the working space without colliding obstacles. As the robot moves close to the obstacle, the robot and the boundary nodes must avoid obstacles and change their moving direction by selecting a new position in the C_{free}.

As the robot moves from g_i to g_{i+1}, $(i = 1...n)$, where n is the number of the goal points, it generates the path from the sets of waypoints $w_{(j)}, (j = 1...J)$ that the robot visits sequentially. For better clarity, the waypoints are connected into a continuous path. The path between each pair of goals consists of a finite number of straight-line segments joining the waypoints. The extracted path between each pair of goals is assembled sequentially until the path is completed into a single connected component. The path length

formed by concatenation of all line segments that connect all goal points is representing the length of the IFP, as follows: $IFP = [P_{1,2}, P_{2,3} \ldots, P_{n-1,n}]$. The complete path starts in g_1, and passes through all the goal points $g_i, (i = 2, ..., n)$, and then returns to g_1 is called the obtained multi-goal path. The obtained path between goal points is safe; however, it is not the shortest path. Therefore, the PEM is used to generate the shortest path from IFP by reducing waypoints w.

3.2.6 Multi-Robot Multi-Goal Path Planning

This section presents the implementation of the proposed method for solving the multi-robot multi-goal path planning problem by generating the shortest collision-free path connecting goal points sequentially among obstacles. This study considered several mobile robots that visiting several goals points $g_i, (i = 1...n)$. Each robot finds its path independently to visit all goal points without collision with static obstacles or other robots. The robot working environment and the obstacles defined in advance, and the goal points are scattered randomly in the free workspace.

The problem formulation for multi-robot path planning determines each robot's next position from their existing positions in the workspace by avoiding collision with other robots and obstacles. To illustrate the multi-robot path-planning problem, we consider a group of mobile robots that plan their path in the same simulated working environment (48×44 square grid cells). At any instant, a groups of robots (m robots) at positions $R_{(t)} = r_1, r_2, ..., r_m$ start to move to visit a set of goal points $G_{(t)} = g_1, g_2, ..., g_n$ scattered randomly in the $2D$ workspace with static obstacles. Each robot moves from the starting position, through all of the intermediate goal points, then return to the starting position. Such that all the robots reach each goal g, and each goal $g \in G_{(t)}$ visited by each robot $r \in R_{(t)}$ once. In this study, each robot uses the BNM method to find IFP from any goal point to the next goal point in the workspace without colliding with any obstacle or other robot. The IFP for each robot is generated from a set of waypoints w that the robot visits.

To demonstrate the collision avoidance between two robots and changes in their motion direction, we consider two simple cases shown in Figure 3.13. As illustrated in Figure 3.13a, both robots start to move with boundary nodes in different directions. While the robot moves forward in the first case, nodes $p_{(2)}$ and $p_{(3)}$ of the first simulated robot interfere with the second robot. Nodes $p_{(7)}$ and $p_{(8)}$ of the second simulated robot interfere with the first robot. In the same way, in the second case, node $p_{(3)}$ of the first simulated robot interfere with the second robot. Node $p_{(7)}$ of the second simulated robot interfere with the first robot (see Figure 3.13b). The robot needs to change its motion direction to solve this situation along the y-axis. The motion direction depends on the characteristics of the boundary nodes. Based on the position of the boundary nodes and their potential values, the robot moves forward. It changes its motion direction by shifting one step backwards and then one step in the downward direction. This process continues until the robot passes the other robot, as shown in Figure 3.13c.

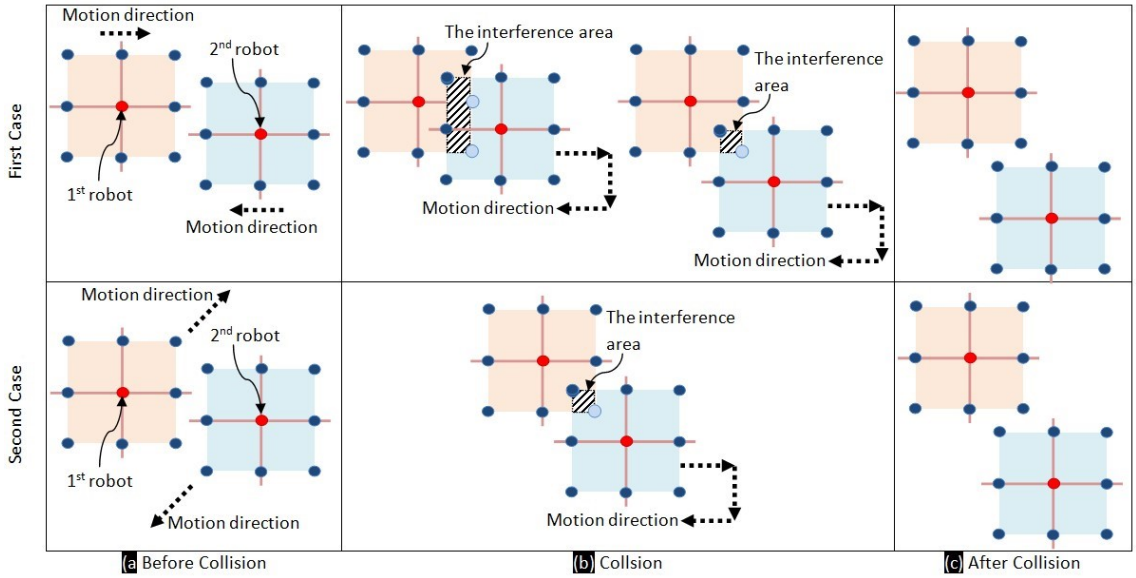


Figure 3.13: Collision avoidance between two robots, (a) the initial positions of the robots before the collision, (b) robot-robot collision avoidance and change motion direction, (c) the robots in their new updated positions.

Chapter 4

Proposed Method for Crowd Simulation

4.1 Problem Formulation

This section presents the problem formulation for predicting the crowd movement in the virtual environment. In this study, a crowd of pedestrians with different group sizes and various types of pedestrians moving through the virtual environment for a limited time. Each group of pedestrians visiting several goal points with varying sequences of the visit. Moreover, the number of each type of pedestrian in each group is different, and each type of pedestrian has its own attributes, such as $position$, $velocity$, $energy$, etc. In the virtual environment, there are several obstacles at different locations that prevent pedestrians from moving through, and pedestrians need to keep a certain distance for safety reasons. For simulating the crowd movement, it is required to formulate groups of pedestrians with a wide variety of characteristics in the front of the virtual environment around the starting point. Based on the different schedule models, groups of pedestrians are appropriately introducing into the virtual environment. After that, they will navigate inside the virtual environment to visit several goal points, and then they will leave the virtual environment. Sometimes, pedestrians may consider undertaking other activities; it can happen between two sequential visits. In this study, pedestrians have to adjust their attributes and optimize their paths continuously in the virtual environment. Moreover, pedestrians need to avoid stationary obstacles and other pedestrians when they move closer.

To demonstrate the groups' formulation in the crowd, let us consider a very simple scenario, as shown in Figure 4.1 and Table 4.1. For simplicity, we consider only two groups for simulating pedestrians' movement. The first group consists of "3" pedestrians, and the second group consists of "5" pedestrians. We assumed only "2" types of pedestrians, namely $agent1$ and $agent2$, in each group. The first group has "3" pedestrians of $agent1$ and "0" pedestrians of $agent2$, and in the second group, "2" pedestrians of $agent1$ and "3" pedestrians of $agent2$ are considered. We assume that the first group ($Group-1$) visit $goal_2$, $goal_6$, $goal_4$, and $goal_7$, and the intention of the second group ($Group-2$) is

Table 4.1: Simple scenario for formulating groups of pedestrians in the virtual environment.

Variables	**Values**
Number of groups	$nGroups$=2 ($Group-1$, $Group-2$)
Number of pedestrians in $Group-1$	3 $pedestrians$
Number of pedestrians in $Group-2$	5 $pedestrians$
Number of pedestrian's type	$nTypes$ = 2 ($agent1_1, agent1_2$)
$nTypes$ in $Group-1$	$nPedestrians_{Types}[1]$=[3, 0]
$nTypes$ in $Group-2$	$nPedestrians_{Types}[2]$=[2, 3]
$Pedestrians_{group}[1]$	$[agent1_1, agent1_2, agent1_3]$
$Pedestrians_{group}$[2]	$[agent1_1, agent1_2, agent2_1, agent2_2, agent2_3]$
Number of the goal points for $Group-1$	$Goals_{group}$[1]= [$Start\_point$, $Entrance$, $Goal_2$, $Goal_6$, $Goal_4$, $Goal_7$, $Exit$, $End-point$]
Number of the goal points for $Group-2$	$Goals_{group}$ [2]= [$Start\_point$, $Entrance$, $Goal_8$, $Goal_5$, $Goal_1$, $Exit$, $End-point$]

different, in which they visit $goal_8$, $goal_5$, and $goal_1$(see Table 4.1). Initially, pedestrians ($Group-1$ and $Group-2$) are distributed randomly in front of the virtual environment around the starting point (see Figure 4.1). After that, all created groups in the virtual environment need to be appropriately introduced into the working environment based on the specific schedule. In this example, groups ($Group-1$ and $Group-2$) start to move from their starting points. Then, the groups of pedestrians enter the virtual environment through the $Entrance$, and they navigate inside the virtual environment to visit several goal points ($Goals_{group}[1]$ and $Goals_{group}[2]$). Afterward, they move toward the entrance/exit to leave the virtual environment, and they continue to move until they reach the last destination point located at the $End-point$. In this study, the number of groups, pedestrians in each group, types of pedestrians in each group, and the goal points are created randomly. The variables are illustrated in Table 4.1, and the given scenario graphically is illustrated in Figure 4.1.

4.2 Proposed Method for Crowd Simulation

This section introduces a new developed method to simulate the movement of crowds in the virtual environment. For simulating the crowd movement in the virtual environment, we consider multiple groups of different sizes and various types of pedestrians. In this study, pedestrians in many groups moving through the navigable area inside a virtual environment with different directions to reach their destination points. Pedestrians should stay in the group, and they need to keep a certain distance between themselves.

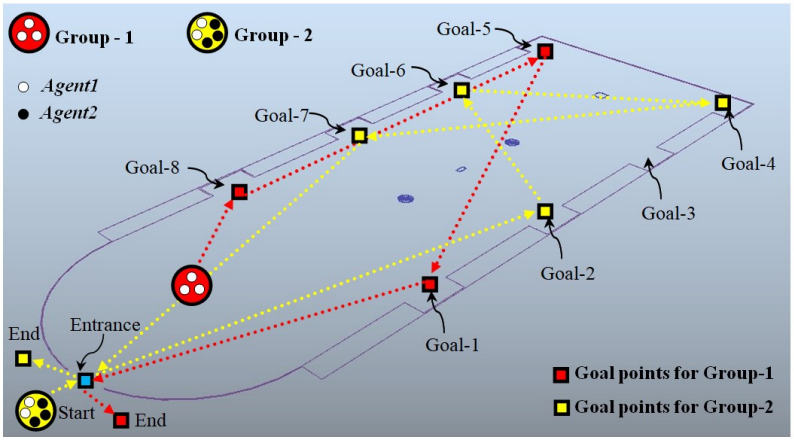


Figure 4.1: Graphical representation of the given simple scenario.

Pedestrians in the same group have the same goal points to visit. Each group is assumed to have a different list of goal points, where each goal point corresponds to a specific region in the virtual environment. The virtual environment consists of walls, obstacles, or other areas that may not be accessible to pedestrians in addition to the goal points.

The proposed method uses the $multi - group$ microscopic model for generating a real-time trajectory for each pedestrian while navigating in the walking area. Moreover, an $agent - based$ model is introduced into the proposed method for modeling pedestrian's behaviors, where each type of pedestrian has its own attributes. This method introduced several steering behaviors to help pedestrians to behave and interact with the other pedestrian in the virtual environment. The steering behaviours include $separation$, $cohesion$, $alignment$, $obstacle\,avoidance$, $pedestrian\,collision\,avoidance$, $flocking$, $goal-directed\,steering\,behavior$, etc. Additionally, some techniques combine the steering behaviors to a single steering force that allows pedestrians to move toward their destination points independently in real-time. The combinations of steering behaviors are used to generate $heterogeneous$ crowd behaviors. During the simulation, pedestrians moving in groups toward their destination point in the simulated working environment, and also pedestrians avoid obstacles and other pedestrians when they move closer. Moreover, the goal point for each group of pedestrians is changing dynamically, and the desired goal point is defined at each step from the list of goal points. In the proposed method, pedestrians carry out various crowd activities, such as adjusting their attributes, avoiding collisions, optimizing their paths, changing their behavior, etc. In this study, pedestrians use the local knowledge for local collision avoidance and for interacting with other pedestrians. Also, pedestrians use global knowledge for long-term planning and to provide goal directing capability. The proposed method is illustrated by the flowchart shown in Figure 4.2, and the steps followed in the flowchart is explained in detail in the following subsections.

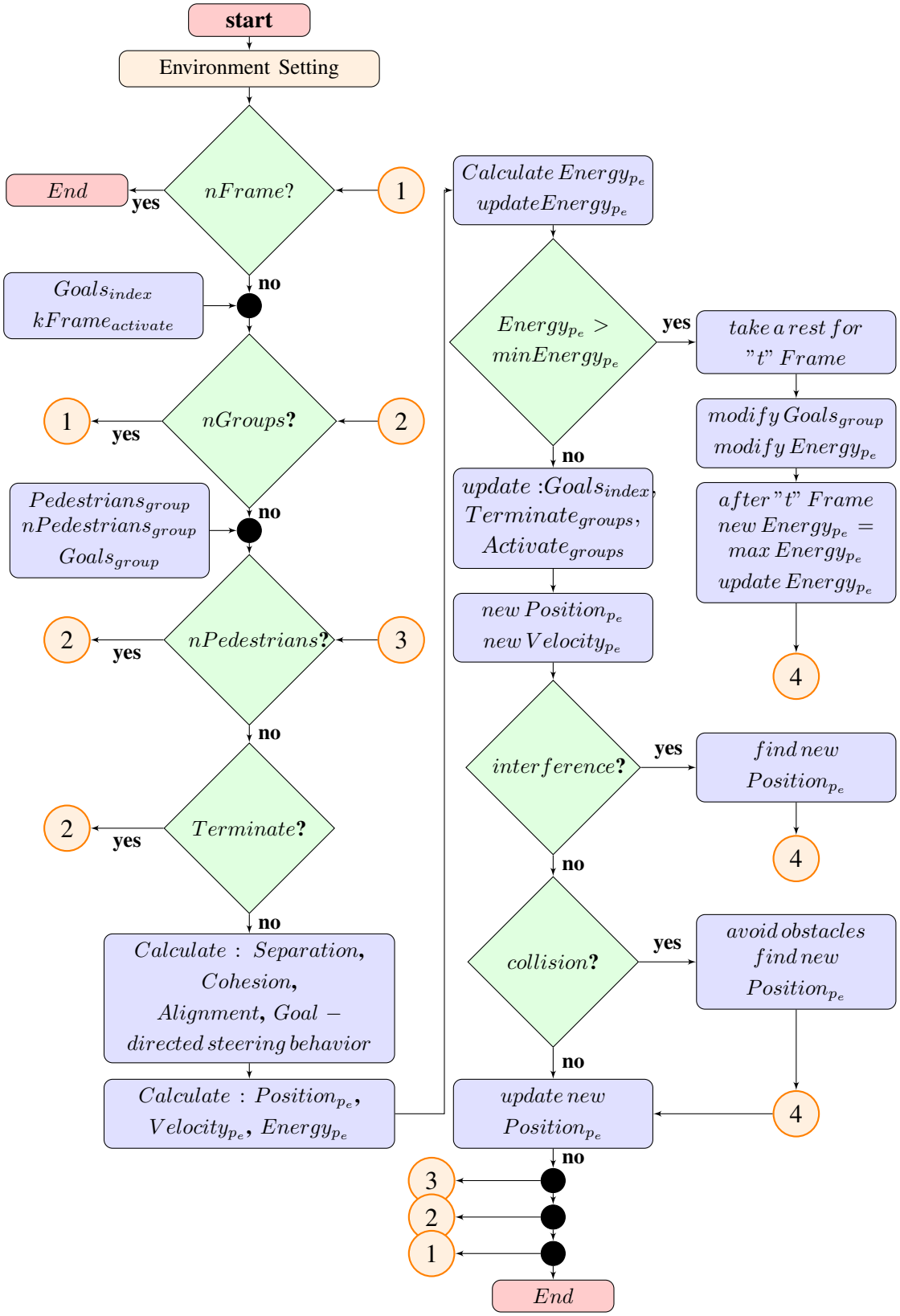


Figure 4.2: Flowchart illustrating the steps of the new proposed method for crowd simulation.

In this study, pedestrians continuously adjust their position based on the proposed method in real-time. Therefore, visualizing the $3D$ motion of pedestrians as they track their trajectories is needed. For that reason, the virtual environment and the virtual pedestrians are imported into the $3D$ animation software package $Autodesk\,Maya$ [123] via a custom $Python$ script. Then the $Maya$'s keyframe system is used to visualize the simulated scenario. In the proposed method, we created $Python$ script models and embedded them into the $Autodesk\,Maya$ for generating real-time trajectories. The code is written step-by-step from scratch in $Pycharm$ [124]. Then $Pycharm$ [124] is connected to $Autodesk\,Maya$ [123] by using $MayaCharm$ Plugin.

The proposed method consists of three main steps: environment setting, motion computation, and steering behaviors, and locomotion, as follow:

1. Environment setting: this step explains the construction of the virtual environment, obstacles, pedestrians, goal points, and the initial state of the simulated scenario (see Section 4.2.1).

2. Motion computation and steering behaviors: this step introduces the proposed method for simulating the crowd and computes the pedestrians' movement. Several steering behaviors are described, and several techniques are used for combining steering behaviors to a single steering force to direct pedestrians toward their destination points. Furthermore, this step explains the obstacles and pedestrians' collision avoidance (see Section 4.2.2).

3. locomotion: this step concentrates on the virtual pedestrian's animation in the virtual environment. The locomotion converts the steering force to a control signal motion, which guides the pedestrians to move in the virtual environment toward their destination point (see Section 4.2.3).

4.2.1 Environment Setting

The first step of the proposed method (see Figure 4.2) introduces the working environment, obstacles, pedestrians, and goal points. Moreover, this step identifies the initial state of the pedestrian's attributes: $position$, $velocity$, etc. Also, interactions between pedestrians with the environment have to be described. The detail of all these steps is provided in the following subsections.

Remark. *During this simulation, the virtual pedestrian's Position is x, y, and z axes. We assumed that the pedestrian is restricted to move in a $2D$ space in the proposed method. Therefore, the pedestrian's Position is defined by two independent coordinates, x and z, and the value of the $y-$ coordinate is set to zero.*

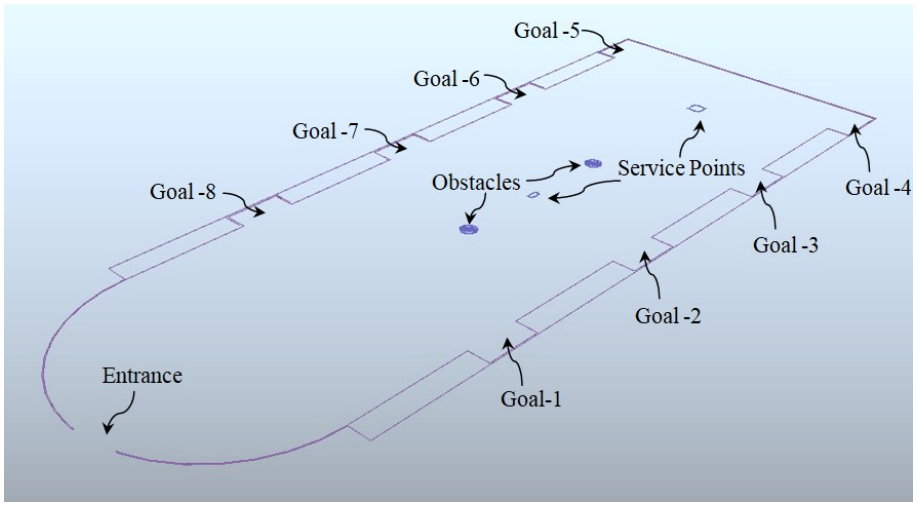


Figure 4.3: The layout of the virtual environment with obstacles.

Virtual environment and obstacles

This section describes the generation of the virtual environment with obstacles. First, the virtual environment model is imported into the scene, as shown in Figure 4.3. The walking area of the virtual environment is considered a two-dimensional ($2D$) workspace (C) for all simulations. The workspace center is located on the original point $(0, 0, 0)$, and the continuous workspace is divided into square grid cells $(x = 1, z = 1)\, unit$. After generating the workspace, the obstacle models import into the scene to prevent the pedestrians from walking through the space occupied by obstacles. The region of space occupied by obstacles is denoted by C_{obs}, and the obstacle-free region is represented by C_{free}. The grid cells have integer coordinates in the form $C_{(x,z)} \in C$, with $1 \leq x \leq w$, and $1 \leq z \leq l$, where w and l are represent the width and length of the simulated workspace. A given cell can either correspond to a navigable area $C_{(x,z)}$ or to a space occupied by obstacles $C_{(x,z)} \in C_{obs}$. For each group during the simulation, each grid cell $C_{(x,z)}$ in C_{free} has a potential value $E_{(x,y)} \in E$, which is calculated according to the proposed potential function (see Subsection 3.2.1).

Pedestrians need to pay attention to the obstacles, and they have to keep a certain distance from the obstacles. In this study, obstacles are distributed at different locations in the free space. A safety zone is created around the obstacles to avoid the possibility of overlapping the paths traced by pedestrians with the obstacle boundaries. The safety zone is represented by a circle that the pedestrians can not enter during their motion. The safety zone radius is constant, R, where the value of R depends on the size of obstacles. Steps for constructing the workspace and obstacles are illustrated in Algorithm 6, and a brief description of each step is provided as follow:

Step 1: Define number of obstacles $n3DObstacles$. Algorithm 6-Line(1)

Step 2: Import the models of the workspace and obstacles into the scene. Algorithm 6-Lines(2,10)

Step 3: Determine the values of width w and length l of the simulated workspace from the model in the scene. Algorithm 6-Line(3)

Step 4: Divide the continuous workspace into several square grid cells, and determine the coordinates of each grid cell in the form $C_{(i,j)} \in C$, where $-w/2 \leq i \leq w/2$, and $-l/2 \leq j \leq l/2$. Algorithm 6-Lines(5-9)

Step 5: Set the value of each grid cell in the free space C_{free} inside the working environment C to "0". Algorithm 6-Line(7)

Step 6: Set the value of each grid cell of the space occupied by obstacles C_{obs} inside the working environment C to "1". Algorithm 6-Line(17)

Step 7: Create $2D$ array from the grid cells values, where the grid cells $C_{(x,z)}$ can either correspond to the navigable area $C_{(x,z)} \in C_{free}$ or to a space occupied by obstacles $C_{(x,z)} \in C_{obs}$. Algorithm 6-Lines(24,25)

Step 8: Create a safety zone around each obstacle with a radius of R, and there are $n3DObstacles$ obstacles in the scene. Each safety zone consists of a set of square grid cells; the total number of the grid cells occupied by obstacles is equal to $nObstacles$. The centre of the grid cells occupied by obstacles are denoted by the matrix $C_{obs(h,l)}$, $(h = 1, 2)$, $(l = 1...nObstacles)$.

Pedestrians

This section explains the creating of the virtual pedestrians and setting attributes associated with each type of pedestrian in the crowd; these attributes include $gender$, age, $position$, $velocity$, $energy$, etc. Creating pedestrians in the scene and setting attributes of each pedestrian is illustrated in Algorithm 7, and a brief description of each step is provided as follow:

Step 1: Define the number of groups ($nGroups$), number of pedestrian's type $nTypes$, and constant c_{p1}. Algorithm 7-Line(1)

Step 2: Import the models of the pedestrians into the scene. Algorithm 7-Line(5)

Step 3: Simulate the crowd with several groups ($nGroups$), where each group consists of a different number of pedestrians ($nPedestrians_{group}$).

Step 4: Create pedestrians with their attributes for each type ($nTypes$) of pedestrians. This study classified pedestrians into seven different types ($nTypes$=7) to establish the range of variation; each type of pedestrian has an associated characteristic. Algorithm 7-Lines(14-18)

Algorithm 6 Generate workspace and obstacles

Inputs:
    $n3DObstacles$
$import\,the\,workspace\,model\,into\,the\,scene$
$width(w), length(l) \leftarrow workspace\,in\,the\,scene$
$C_{obs} = [\,], C_{free} = [\,],$
for i = $-w/2$ to $w/2$ **do**
    for j = $-l/2$ to $l/2$ **do**
        $C(i,j) \leftarrow 0$
    end for
end for
$import\,obstacles\,models\,into\,the\,scene$
$move\,obstacles\,to\,different\,locations$
$create\,a\,safety\,zone\,around\,obstacles\,with\,the\,radius\,of\,(r)$
for obs = 1 to $n3DObstacles$ **do**
    $x_1, z_1 \leftarrow obstacles\,model$
    for k = $x_1 - r$ to $x_1 + r$ **do**
        for l = $z_1 - r$ to $z_1 + r$ **do**
            if $distance\,between\,(x_1, z_1)\,and\,(k,l) < R$ **then** $C(k,l) \leftarrow 1$
            end if
        end for
    end for
end for
for k = $-w/2$ to $w/2$ **do**
    for l = $-l/2$ to $l/2$ **do**
        if $C(i,j) = 1$ **then** $assign\,C(k,l)\,as\,C_{obs}$
        else $assign\,C(k,l)\,as\,C_{free}$
        end if
    end for
end for

Step 5: Formulate all pedestrians in the crowd ($Pedestrians_{crowd}$) with all their attributes by concatenation of all pedestrians ($nPedestrians_{group}$ and $Pedestrians_{group}$) of all groups. Algorithm 7-Line(21-22)

Step 6: Determine the total number of pedestrians in the crowd ($nPedestrians_{total}$) by summing all pedestrians in all groups. Algorithm 7-Line(23)

Algorithm 7 Generate pedestrians with their attributes.

1: **Inputs:**
   $nGroups$, $nTypes$, c_{p1}
2: $nPedestrians_{crowd} \leftarrow [\,]$
3: $Pedestrians_{crowd} \leftarrow [\,]$
4: $nPedestrians_{total} \leftarrow 0$
5: $import\, pedestrians\, models\, into\, the\, scene$
6: **for** g_r = 1 to $nGroups$ **do**
7: $nPedestrians_{group} \leftarrow 0$
8: $Pedestrians_{group} \leftarrow [\,]$
9: **for** c_l= 1 to $nTypes$ **do**
10: $nRandom_{Types} \leftarrow random\, number\, between\, [1, c_{p1}]$
11: $nPedestrians_{Types} \leftarrow nRandom_{Types}$
12: $nPedestrians_{group} \leftarrow nPedestrians_{group} + nPedestrians_{Types}$
13: **for** p_e = 1 to $nPedestrians_{Types}$ **do**
14: $create\, pedestrian\, with\, attributes\, such\, as\, (position,\, velocity,\, scale,\, energy\, level)$
15: $define\, function\, to\, get/set\, position\, of\, pedestrians$
16: $define\, function\, to\, get/set\, velocity\, and\, energy\, level\, of\, pedestrians$
17: $define\, function\, to\, constrains\, an\, object's\, orientation$
18: $Pedestrians_{group}.append(Pedestrian)$
19: **end for**
20: **end for**
21: $Pedestrians_{crowd}.append(Pedestrians_{group})$
22: $nPedestrians_{crowd}.append(nPedestrians_{group})$
23: $nPedestrians_{total} \leftarrow nPedestrians_{total} + nPedestrians_{group}$
24: **end for**

Goal points

In the simulated environment, each group is assumed to have different goal points to visit in terms of the number and sequence of the goal points. As shown in Figure 4.3, the goal points ($nGoals$) in the simulated environment corresponding to a specific region in the environment. Steps for formulating the list of goal points for each group in the simulated environment is illustrated in Algorithm 8, and a brief description of each step is provided as follow:

Step 1: Define the number of the goal points $nGoals$ and their locations ($Goals_{group}$) in the crowd. Algorithm 8-Line(1)

Step 2: Denoted the list of all goal points existed in the scene by $GoalPoints^{list}$, where we obtained the values of $x-coordinates$ ($Goals_x$) and $z-coordinate$ ($Goals_z$) of each goal in the scene (see Figure 4.3), the $y-coordinate$ ($Goals_y$) is set to zero vector. Algorithm 8-Line(3)

Step 3: Create the number of goal points randomly ($nRandom_{goals}$) between [1, $nGoals$] for each group, and also the sequence of the goal points is generated randomly. Algorithm 8-Line(6)

Step 4: Create the starting and the final destination points for each group, which are located outside the working environment. Algorithm 8-Lines(16-23)

Step 5: Generate the $x-coordinate$ for both the start and the end points randomly between $[(-w/2), (w/2)]$, the $y-coordinate$ is set to "0", and the value of $z-coordinate$ is equal to $l/2 + c_{s1}$. The value of constant c_{s1} represents how far the goal point from the outside of the working environment in z direction. Algorithm 8-Lines(17-20)

Step 6: Formulate the overall goal points' array of the crowd ($Goals_{crowd}$) by concatenating of goal points' array of all groups ($Goals_{group}$). Algorithm 8-Line(27)

The initial pedestrians' state

This section demonstrates the steps involved in setting the initial state of each pedestrian in each group $Pedestrians_{group}$ in terms of the pedestrian's position $Position_{p_e}$ and velocity $Velocity_{p_e}$. Pedestrians $Pedestrians_{crowd}$ in each created group $Pedestrians_{group}$ are distributed randomly outside the working environment. Then they start to move to reach the goal point in the list of the goal points ($Goals_{group}$). Steps to compute the initial value of $Position_{p_e}$ and $Velocity_{p_e}$ for each pedestrian are illustrated in Algorithm 9, and a brief description of each step is provided as follow:

Step 1: First, this algorithm starts with the previously created input data: number of groups $nGroups$, number goal points $nGoals$, all pedestrians in the crowd $Pedestrians_{crowd}$, number of pedestrians in the crowd $nPedestrians_{crowd}$, the list of goal points of all groups in the crowd $Goals_{crowd}$, and define constants c_{f1}, c_{f2} and c_{f3}. Algorithm 9-Line(1)

Step 2: Create random position for each pedestrian $Position_{p_e}$ ($p_e = 1, .., nPedestrians_{group}[g_r]$) in each group g_r, ($g_r = 1, ..., nGroups$) around the starting point $Goals_{group}[1]$. The distance between the randomly created point $_{(x,z)}$ and the starting point depend on the value of constant c_{f1}. The starting point is represent the first goal point in the goal points list $Goals_{group}$ that belong to the group g_r. Algorithm 9-Lines(6-11)

Algorithm 8 Generate goal points

Inputs:
$nGroups$, $nGoals$
$Goals_{crowd} \leftarrow [\,]$, $Goals_{group} \leftarrow [\,]$, $nGoals_{crowd} \leftarrow [\,]$
$GoalPoints^{list} \leftarrow 3D\,model\,in\,the\,scene$
for g_r = 1 to $nGroups$ **do**
  $nGoals_{group} \leftarrow 0$
  $nRandom_{goals} \leftarrow between\,[1, nGoals]$
  for g_o = 1 to $nRandom_{goals}$ **do**
    $nRandom_{goalsIndex} \leftarrow between\,[1, nGoals]$
    $x - coordinate \leftarrow GoalPoints^{list}[nRandom_{goalsIndex}]$
    $y - coordinaten \leftarrow 0$
    $z - coordinate \leftarrow GoalPoints^{list}[nRandom_{goalsIndex}]$
    $goal_{position} \leftarrow [x - coordinate, y - coordinate, z - coordinate]$
    $Goals_{group}.append(goal_{position})$
  end for
  $create\,the\,starting\,and\,final\,destination\,points$
  for g_p = 1 to 2 **do**
    $x - coordinate \leftarrow random\,number\,between[(-w/2), (w/2)]$
    $y - coordinate \leftarrow 0$
    $z - coordinate \leftarrow l/2 + c_{s1}$
    $goal_{position} \leftarrow [x - coordinate, y - coordinate, z - coordinate]$
    if $g_p = 1$ **then** $Goals_{group}.append(first, goal_{position})$
    else if $g_p = 2$ **then** $Goals_{group}.append(last, goal_{position})$
    end if
  end for
  $nGoals_{group} \leftarrow nRandom_{goals} + 2$
  $nGoals_{crowd} \leftarrow nGoals_{crowd}.append(nGoals_{group})$
  $Goals_{crowd}.append(Goals_{group})$
end for

Step 3: Check the distance between the generated position $Position_{p_e}$ and the position of all other created pedestrians $Position_{nPede}$, the distance should not reside within a certain value (c_{f2}). Algorithm 9-Lines(12-20)

Step 4: Formulate the list of all pedestrian's Position in the crowd ($Pedestrians_{crowd}$) by concatenation of all randomly created Position of pedestrians $Position_{p_e}$ of all groups. Algorithm 9-Line(11)

Step 5: Create random velocity value $Velocity_{p_e}$ ($p_e = 1, .., nPedestrians_{group}[g_r]$) for each pedestrian in each group g_r, ($g_r = 1, ..., nGroups$), where the value of the constant c_{f3} creates diversity in walking velocity $Velocity_{p_e}$ in a specific range. Algorithm 9-Lines(25-30)

Step 6: Update pedestrians' positions $Position_{p_e}$ (see Algorithm 9-Line(24)) and velocity $Velocity_{p_e}$ (see Algorithm 9-Line(31)) in the scene.

In addition to setting of the pedestrian's attributes include position $Position_{p_e}$ and velocity $Velocity_{p_e}$, the other pedestrians' attributes such as $gender$, age, $energy$, etc., are need to initialised as well.

In this study, all groups ($Pedestrians_{crowd}[g_r]$, $g_r = 1, ..., nGroups$) do not enter to the simulated environment all together at the same time. Groups in the crowd need to decide whether and when (keyframe $kFrame$) they start to move from the starting point. Based on the specific schedule, all groups of pedestrians ($Pedestrians_{crowd}$) which are created outside the simulated environment are appropriately introduced into the simulated environment, as described in Algorithm 10. Based on this algorithm, each group g_r, ($g_r = 1, ..., nGroups$) activate to move at randomly created keyframe ($kFrame_{activate}[g_r]$) between (1, $nFrames$*c_{a1}), where $nFrames$ represent the total number of frames and c_{a1} is a constant. At the first keyframe, the value of $Activate_{groups}[g_r]$ is set to "off" for each group (g_r), this value will switch to "on" when the keyframe reach to $kFrame_{activate}[g_r]$, then the group $Pedestrians_{group}$ starts to move. In the same way, at the first keyframe, the value of $Terminate_{groups}[g_r]$ is set to "off" for each group (g_r). The value of $Terminate_{groups}[g_r]$ will switch to "on" when the group $Pedestrians_{group}$ reaches the final destination point $Goals_{group}[last]$ outside the simulated environment.

The groups' initial configurations in terms of the number of the pedestrian, type of pedestrian, number of goal points and their sequence, pedestrian's attributes, and most of the other variables are generated randomly. Additionally, it is essential to simulate each pedestrian independently and introduce interactions between pedestrians and the simulated environment. In this study, we create a set of functions to get and set the pedestrian's attributes in the scene in real-time.

Algorithm 9 Setting the initial state of the pedestrians in the crowd $Pedestrians_{group}$.

Inputs:
$nGroups$, $nGoals$, $Pedestrians_{crowd}$, $nPedestrians_{crowd}$, $Goals_{crowd}$, c_{f1}, c_{f2}, c_{f3}
$Positions_{crowd} \leftarrow [\,]$
for g_r = 1 to $nGroups$ **do**
    $Pedestrians_{group} \leftarrow Pedestrians_{crowd}[g_r]$, $nPedestrians_{group} \leftarrow nPedestrians_{crowd}[g_r]$
    $Goals_{group} = Goals_{crowd}[g_r]$
    for p_e = 1 to $nPedestrians_{group}$ **do**
        $Position_{p_e} \leftarrow [\,]$
        $x - coordinate \leftarrow (Goals_{group}[1][0] + random.random() * cf1)$
        $y - coordinate \leftarrow 0$
        $z - coordinate \leftarrow (Goals_{group}[1][2] + random.random() * cf1)$
        $Position_{p_e} \leftarrow [x, y, z]$, $nPede \leftarrow 1$
        while $nPede < P_e$ **do**
            $interfere \leftarrow$ "$True$"
            while $interfere \leftarrow$ "$True$" **do**
                if distance between $Position_{p_e}$ and $Positions_{crowd}[nPede]$<c_{f2} **then**
                    $create\ another\ random\ position\ for\ Position_{p_e}$
                else
                    $interfere \leftarrow$ "$False$"
                end if
            end while
            $nPede \leftarrow nPede + 1$
        end while
        $Positions_{crowd}.append(Position_{p_e})$
        $update\ position\ in\ Pedestrians_{group}[p_e]\ with\ Position_{p_e}$
        $create\ random\ velocity\ for\ Pedestrians_{group}[p_e]$
        $Velocity_{p_e} \leftarrow [\,]$
        $v_x - coordinate \leftarrow random\ number\ *\ c_{f3}$
        $v_y - coordinate \leftarrow 0$
        $v_z - coordinate \leftarrow random\ number\ *\ c_{f3}$
        $Velocity_{p_e} \leftarrow [v_x, v_y, v_z]$
        $update\ velocity\ in\ Pedestrians_{group}[p_e]\ with\ Velocity_{p_e}$
    end for
end for

Algorithm 10 Create a list of random keyframe for activating groups in the scene.

1: **Inputs:**
   $nGroups$, $Pedestrians_{crowd}$, c_{a1}
2: $kFrame_{activate} = [\,]$
3: $Activate_{groups} = [\,]$
4: $Terminate_{groups} = [\,]$
5: **for** g_r = 1 to $nGroups$ **do**
6: $random_{keyframe} \leftarrow random\,integer\,between[1, nFrames * c_{a1}])$
7: $kFrame_{activate}.append(random_{keyframe})$
8: $Activate_{groups}.append("off")$
9: $Terminate_{groups}.append("off")$
10: **end for**
11: $sort\,Pedestrians_{crowd}\,according\,to\,kFrame_{activate}$

4.2.2 Motion Computation and Steering Behaviours

This section presents the technique to compute each pedestrian's movement in the simulated environment and directs the groups through the walking area until they terminate the simulation. This section also describes various types of steering behaviors that help pedestrians in each group to move in the virtual environment. The pedestrians' behaviours include activities such as flocking with neighbouring pedestrians, goal-directing, obstacle avoidance, and pedestrian collision avoidance. The initial state of the $pedestrians$, $goals$, $obstacles$, and $interaction\,between\,pedestrians\,with\,environment$ of the simulated crowd are defined previously. At each key-frame ($keyframe$) pedestrians are moving forward from the current position to the new updated position ($Position_{p_e}$) with the walking velocity $Velocity_{p_e}$. The new updated $Position_{p_e}$ and $Velocity_{p_e}$ is calculated based on the steering behaviours with the current pedestrian's position and velocity. Algorithm 11 explains pedestrians' movement in the virtual environment from the starting point to the ending point (see Figure 4.2), and this process is illustrated in the following steps.

Step 1: First, this algorithm starts with the previously created input data: number of groups $nGroups$, groups' activate list $Activate_{groups}$, groups' terminate list $Terminate_{groups}$, pedestrians in the crowd $Pedestrians_{crowd}$, number of the pedestrians in the crowd $nPedestrians_{crowd}$, list of the goal points for each group in the crowd $Goals_{crowd}$, list of the pedestrian's Position in the crowd $Positions_{crowd}$, and number of frames $nFrames$. Algorithm 11-Line(1)

Step 2: At each $keyframe$ ($keyframe = 1, ..., nFrames$), the activate group's value $Activate_{groups}[g_r]$ set to "on" when $keyframe$ reach to $kFrame_{activate}[g_r]$ (see Algorithm 11-Line(3)). Then, for each activate group we will define:

1. Index of the goal point $Goals_{crowd}[g_r]$ (goal index, $Goal_i$). Algorithm 11-Line(7)

2. Pedestrians in the group $Pedestrians_{crowd}[g_r]$. Algorithm 11-Line(9)
3. Number of pedestrians in the group $nPedestrians_{crowd}[g_r]$. Algorithm 11-Line(10)
4. The goal points $Goals_{crowd}[g_r]$ for the group $nPedestrians_{crowd}[g_r]$. Algorithm 11-Line(11)

Step 3: For each pedestrian ($p_e = 1, .., nPedestrians_{group}[g_r]$), the current pedestrian's position ($Position_{p_e}$) is obtained from the scene in real-time, and the value of pedestrian's velocity $Velocity_{p_e}$ and pedestrian's energy level $Energy_{p_e}$ is obtained from the attributes of pedestrian $Pedestrians_{group}[p_e]$. Algorithm 11-Line(14)

Step 4: At each $keyframe$, ($keyframe = 1, ..., nFrames$), the new $Velocity_{p_e}$ is calculated for each pedestrian $Pedestrians_{group}[p_e]$ in activated groups. In this study, pedestrian' velocity is limited to $mSpeed$ ($unite/timestep$), and the value of constant c_{s1} affect the speed of the pedestrians in the scene. Algorithm 11-Line(23)

Step 5: For each pedestrian $Pedestrians_{group}[p_e]$ in activated group $Pedestrians_{crowd}[g_r]$, a new position $Position_{p_e}$ inside the simulated environment is computed based on the steering behaviors that we will explain in the following subsections. Algorithm 11-Lines(23)

Step 6: During this simulation, pedestrians need to keep a certain distance ($2r$) between themselves. Therefore, the new updated $Position_{p_e}$ of each pedestrian has to check for collision with neighboring pedestrians, as illustrated in Figure 4.6. Algorithm 11-Lines(24)

Step 7: The proposed method for crowd simulation uses the BNM method to check the new updated $Position_{p_e}$ for collision with obstacles. Suppose a collision happens between pedestrians and obstacles. In that case, the BNM method computes another new position $Position_{p_e}$ for the pedestrian to avoid collision with obstacles. Algorithm 11-Lines(26)

Step 8: At each $keyframe$ ($keyframe = 1, ..., nFrames$) the energy level of every single pedestrian is decrease as they walk through the environment as follow: $Energy_{p_e} = Energy_{p_e}$ - c_{s1} (see Algorithm 11-Line(15)), where the constant c_{s1} represent the reduced value at each $keyframe$. Suppose the value of $Energy_{p_e}$ reaches below the minimum $Energy$ level. In that case, all pedestrians in the group will visit the service point in the walking area after finishing the current task (see Algorithm 11-lines(29-30)). Meanwhile, the group stays in the service point, their energy will increase again, and they stay in the service point until their energy reaches the maximum level. Then they will start again to move to visit the rest of the goal points. Algorithm 11-lines(31-36).

Step 9: At each $key - frame$, if the distance between any pedestrian $Position_{p_e}$ in the group $Pedestrians_{group}$ and $Goals_{group}[Goal_i]$ is reached below c_{s2}, it indicates that this group is arrived to the current goal point. Therefore, this group has to get a new goal point from the goal points list and update the goal's $index$ ($Goals_{index}$). Algorithm 11-lines(17-22)

Step 10: At each $keyframe$, pedestrian changes its position to the new position at $Position_{p_e}$ in real time, afterward, pedestrians update their attributes include $Velocity_{p_e}$ and $energy_{p_e}$ in $Pedestrians_{group}[p_e]$ with the new calculated attributes. Algorithm 11-lines(38,39))

Step 11: When the group $Pedestrians_{crowd}[g_r]$ reach to last goal point $Goals_{group}[last]$, the value of $Terminate_{groups}[g_r]$ is switch to "on" and also the value $Activate_{groups}[g_r]$ is switch to "off" as well. Algorithm 11-Line(19-20)

In this study, different steering behavior are considered to direct the pedestrians toward their goal points in the simulated environment. The steering behaviours include $alignment$, $cohesion$, $separation$, $obstacle\,avoidance$, $pedestrian\,collision\,avoidance$ which are discussed in the literature (see [1]). Additionally, other steering behaviors such as $goal - directed$ behavior has been included in the steering behaviors to control pedestrians' motion and direct them to the goal points. In this study, the current implementation of the $obstacle\,avoidance$ and the $collision\,avoidance$ steering behaviour is different from the local approximation strategy discussed in Ref [1]. In the local approximation strategy, a bounding box is created around the vehicle, and the box moves with the vehicle forward in the scene. Afterwards, depending on where the box intersects with the obstacles a steering force is added, this type of avoidance strategy produces smooth steering behaviour [125]. In order to direct pedestrians to avoid the collision in this study, the BNM method is used to generate a collision-free path for each pedestrian when they move close to the obstacles in real-time. Using the global path planning in real-time contexts is difficult for modelling pedestrians' behaviors by using the $agent - based$ model because it becomes computationally expensive. Therefore, the $agent - based$ models used in this study separate local collision avoidance from global path planning. Based on the extended BNM method, each pedestrian in the crowd is simulated by a nine-node quadrilateral element (see Figure 4.4a). The nodes are denoted by $p(q), (q = 1...9)$, the centroid node $p(5)$ represents the pedestrian's location, as illustrated in Figure 4.4a&c, the nodes $p(1 \rightarrow 4)$ with $p(6 \rightarrow 9)$ represent the eight-boundary nodes that help pedestrians to move forward and avoid obstacles. As shown in Figure 4.4b, pedestrian and boundary nodes $p(q), (q = 1...9)$ are restricted to move in eight-possible directions $e(u), (u = 1...8)$ in the workspace. More details on the BNM method for path planning and its applicability to various problem area is provided in Section 3.2.

Algorithm 11 Motion computation and steering behaviors

Inputs:
$nFrames$, $nGroups$, $Activate_{groups}$, $Terminate_{groups}$, $Pedestrians_{crowd}$, $nPedestrians_{crowd}$, $Goals_{crowd}$, $Positions_{crowd}$, c_{s1}
for $keyframe$ = 1 to $nFrames$ **do**
  if $keyframe = kFrame_{activate}$ **then** $Activate_{groups}[g_r] \leftarrow$ "on"
  end if
  $p_c \leftarrow 0$, $Goals_{index}([1] * nGroups)$
  for g_r = 1 to $nGroups$ **do**
    $p_c \leftarrow p_c + 1$, $Goal_i \leftarrow Goals_{index}[g_r]$
    if ($Terminate_{groups}[g_r] \leftarrow$ "off" and $Activate_{groups}[g_r] \leftarrow$ "on") **then**
      $Pedestrians_{group} \leftarrow Pedestrians_{crowd}[g_r]$
      $nPedestrians_{group} \leftarrow nPedestrians_{crowd}[g_r]$
      $Goals_{group} \leftarrow Goals_{crowd}[g_r]$
      for p_e = 1 to $nPedestrians_{group}$ **do**
        $Compute\ Separation$, $Cohesion$, $Alignment$
        $(Position_{p_e}, Velocity_{p_e}, Energy_{p_e}) \leftarrow Pedestrians_{group}[p_e]$
        $Energy_{p_e} \leftarrow Energy_{p_e} - c_{s1}$, $Pedestrians_{group}[p_e] \leftarrow Energy_{p_e}$
        if $Energy_{p_e} > minimum\ Energy_{p_e}$ **then**
          if $dist((Position_{p_e}, Goals_{group}[Goal_i]) < c_{s2}$ **then**
            $Goal_i \leftarrow Goal_i + 1$, $update\ Goals_{index}[g_r] \leftarrow Goal_i$
            if $goal_i = len(Goals_{group})$ **then**
              $Terminate_{groups}[g_r], Activate_{groups}[g_r] \leftarrow$ "on","off"
            end if
          end if
          $compute\ new\ Velocity_{p_e}\ and\ Position_{p_e}$
          if $ind\ collision =$ "$True$" **then** $Position_{p_e} \leftarrow Algorithm\ 12$
          end if
          if $Obstacle\ collision =$ "$True$" **then** $Position_{p_e} \leftarrow BNM$
          end if
        else
          $stay\ in\ service\ point$, $modify\ Goals_{group}\ and\ Energy_{p_e}$
          $update\ Pedestrians_{group}[p_e]\ with\ new\ Energy_{p_e}$
          if $staying\ time\ is\ finished$ **then**
            for p_e = 1 to $nPedestrians_{group}$ **do**
              $new\ Energy_{p_e} \leftarrow maximum\ Energy_{p_e}$
              $update\ Pedestrians_{group}[p_e] \leftarrow new\ Energy_{p_e}$
            end for
          end if
        end if
        $update\ Pedestrians_{group}[p_e] \leftarrow new\ (Position_{p_e}, Velocity_{p_e})$
        $Positions_{crowd}[p_c] \leftarrow Position_{p_e}$, $update\ keyframe\ and\ position_{p_e}$
      end for
    end if
  end for
end for

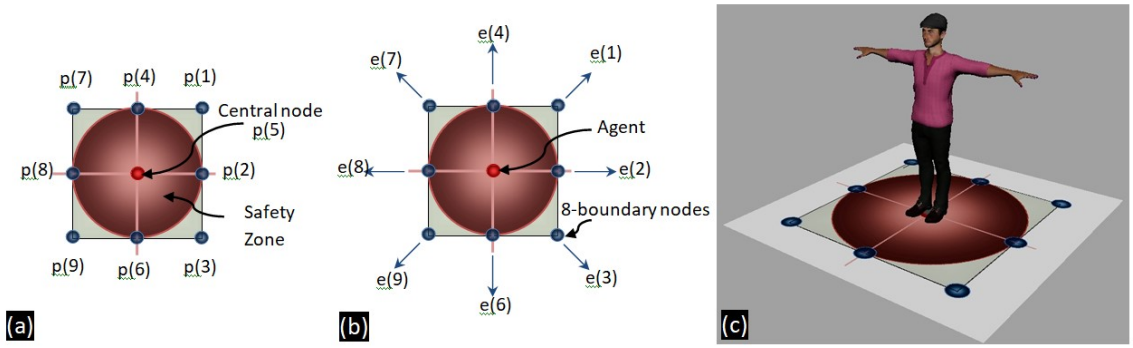


Figure 4.4: A nine-node quadrilateral element with a safety zone (a) along with its motion directions (b) and simulated pedestrian in a virtual environment (c).

Obstacle Avoidance: based on the $obstacles\,avoidance$ procedure using the extended BNM method, the pedestrian and boundary nodes are steering to avoid obstacles and changing their motion direction by selecting a new position in the free space C_{free}. Obstacle avoidance gives a pedestrian the ability to move in a complex environment cluttered with obstacles. At each $keyframe$, the $obstacles\,avoidance$ method considers all obstacles in the walking area when a pedestrian moves near the safety zone around obstacles. While the $2D$ distance between the pedestrian's position and center of the obstacle is less than (r+R), the pedestrian interferes with the safety zone around obstacles, where R and r represent the radius of the safety zone around the obstacle and pedestrian, respectively.

Pedestrian Collision Avoidance: in this study, groups of pedestrians navigating in the virtual environment, which full of pedestrians walking in the same area from different directions. Each moving pedestrian represents a dynamic obstacle for the remaining pedestrians in the scene. Other pedestrians influence the pedestrian's motion in the crowd, which introduces interactions among pedestrians. Pedestrians should not reside within the personal space requirement, the so-called private sphere, of the other pedestrians. Usually, each pedestrian has a safety zone with a $radius, r$ (see Figure 4.5), pedestrians should keep a certain distance from the other neighboring pedestrians located too close.

For instance, as the current pedestrian tends to move forward, it should keep a certain distance from other neighboring pedestrians in the walking area, as demonstrated in Figure 4.6a. This study assumes that the distance between pedestrians should not be less than ($2r$). Suppose that the distance between current and neighboring pedestrian drops below a specific value ($2r$). In that case, the current pedestrian changes its motion direction slightly in surrounding areas to prevent the collision. For this situation, the authors in [74] mentioned that pedestrians usually choose the fastest route to reach their next destination, but not the shortest one. In general, pedestrians consider detours and the comfort of walking, thereby minimizing the effort [74]. We have proposed an appropriate reactive

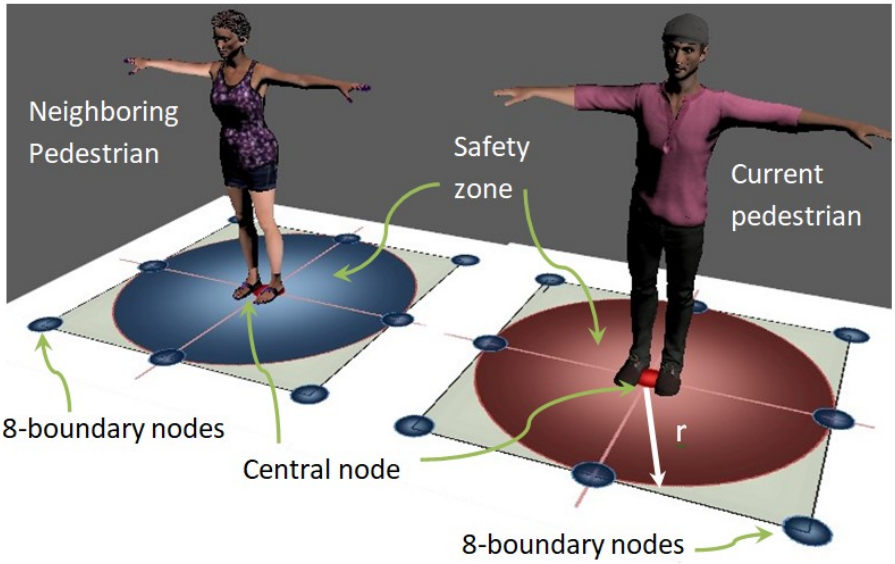


Figure 4.5: Personal space requirements.

behavior, called $pedestrian\ collision\ avoidance$, for preventing collision between pedestrians walking closer, and the pedestrian collision avoidance is illustrated in Figure 4.6b. Suppose the current and neighboring pedestrian move closer, and the distance between them drops below $2r$. In that case, the current pedestrian steers to either left or right for a newly calculated position $Position_{p_e}$, which is located on the third vertex of an equilateral triangle (see Figure 4.6b). The overall process of pedestrian collision avoidance illustrated in Algorithm 12, and the concept of collision avoidance is explained in the following steps:

Step 1: First, this algorithm starts with the previously created input data: position of all pedestrians in the crowd $Positions_{crowd}$, new calculated pedestrian's position $Position_{p_e}$, current pedestrian's position $Position_{p_e}$, number of all pedestrians in the crowd $nPedestrians_{total}$s. Algorithm 12-Line(1)

Step 2: Check for interfere between current new pedestrian $Position_{p_e}$ and all pedestrians $Position_{p_e}$ ($p_e = 1, ..., nPedestrians_{total}$) in the crowd, if the distance between new pedestrian $Position_{p_e}$ and other pedestrians in the crowd is less than $2r$, then the interfere will occur and a new position $Position_{p_e}$ need to be calculated. Algorithm 12-Line(3)

Step 3: Suppose a collision happens between pedestrians, the proposed method implements $pedestrian\ collision\ avoidance$ to avoid collision between pedestrians, as shown in Figure 4.6b. There are two possibilities for the new pedestrian's position ($right-hand\ side$ or $left-hand\ side$) located on the third vertex of equilateral triangles. Algorithm 12-lines(4-13). The pedestrian will choose $right-hand\ side$ or $left-$

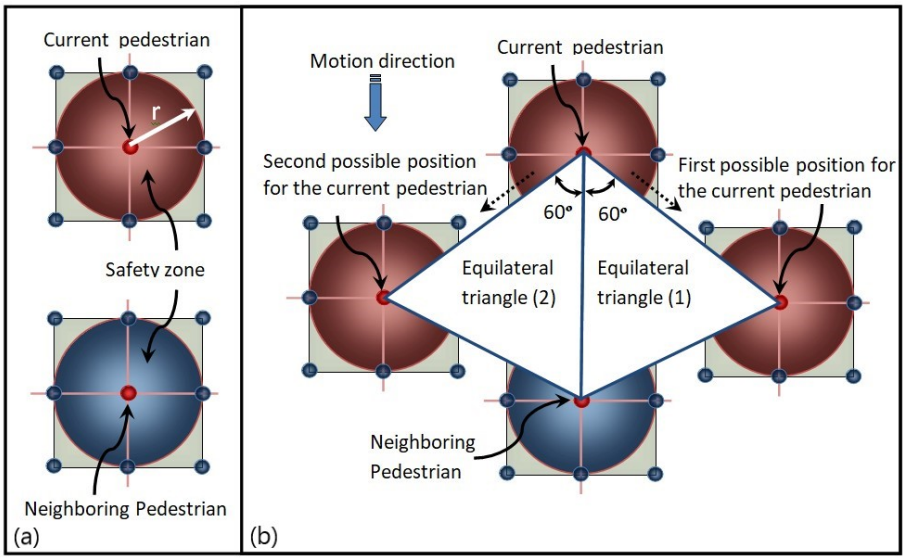


Figure 4.6: Sketch of the simulated pedestrian, as the current pedestrian move forward, it may interfere with the neighboring pedestrian: (a) the distance between the current pedestrian and the neighboring pedestrian is higher than a certain distance $(2r)$, (b) $pedestrian\ collision\ avoidance$ is used to prevent collision between pedestrians.

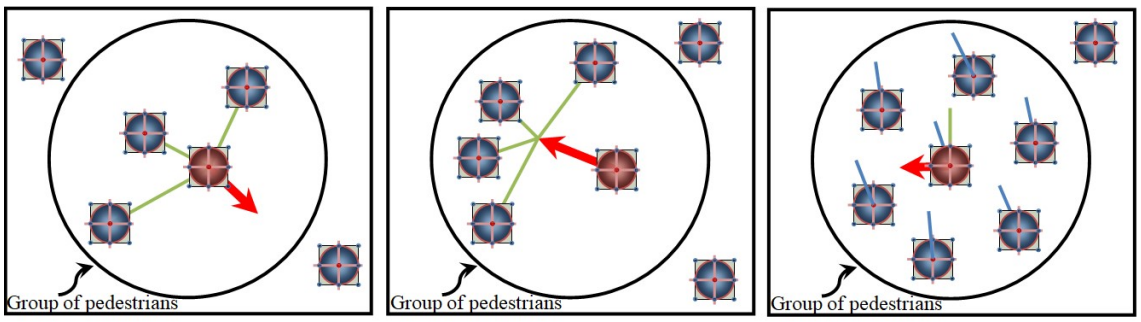
$hand\ side$ to avoid collision problem based on the random value. Algorithm 12-lines(6-8)

Step 4: As long as the current pedestrian $Position_{p_e}$ and the other pedestrians in the crowd are not interfering, the pedestrian will move from the current position $Position_{p_e}$ to the new calculated position $Position_{p_e}$. Then, pedestrians update their positions $Position_{p_e}$ in the scene and in $Positions_{crowd}$.

Flocking Behaviour: in this study, pedestrians are always moving with the same group, and there is a cooperation behavior among pedestrians in the same group ($Pedestrians_{group}$). The pedestrians' steering behaviors in the same group demonstrate how pedestrians react with each other, and pedestrians outside of the group are ignored. In this study, steering behaviors such as $alignment$, $cohesion$, and $separation$ relate to the groups of pedestrians are considered. These behaviors influence on controlling of the pedestrians' motion, more precisely, on the $position$ and $velocity$ of the pedestrians. The $separation$ steering behavior gives a pedestrian the ability to maintain a certain separation distance from other pedestrians in the same group, as shown in Figure 4.7a. The $separation$ steering behavior is used to prevent pedestrians from crowding together. Moreover, the $cohesion$ steering behavior gives a pedestrian the ability to cohere with the other pedestrians, as shown in Figure 4.7b. At the same time, the $alignment$ steering behavior tends to turn pedestrian, so it aligned with its neighboring pedestrians, as demonstrated in Figure 4.7c. For details on the steering behaviors, see Refs. [1, 70].

Algorithm 12 Pedestrian collision avoidance

Inputs:
$Positions_{crowd}$,
$newPosition_{p_e}$,$currentPosition_{p_e}$,$nPedestrians_{total}$
for p_e = 1 to $nPedestrians_{total}$ **do**
  if $(distance\,between\,new\,Position_{p_e}\,and\,Positions_{crowd}[p_e]) < (2 \times r)$ **then**
    $\Delta x = newPosition_{p_e}[0] - Positions_{crowd}[p_e][0]$
    $\Delta z = newPosition_{p_e}[2] - Positions_{crowd}[p_e][2]$
    if $random.random() < 0.5$ **then** $rotate_{angle} \leftarrow 60.0$
    else $rotate_{angle} \leftarrow -60.0$
    end if
    $\alpha \leftarrow rotate_{angle}/180 * math.pi$
    $new_x \leftarrow newPosition_{p_e}[0] + math.cos(\alpha) * \Delta x + math.sin(\alpha) * \Delta x$
    $new_y \leftarrow 0$
    $new_z \leftarrow newPosition_{p_e}[2] + math.sin(-\alpha) * \Delta x + math.cos(\alpha) * \Delta z$
    $newPosition_{p_e} \leftarrow [new_x, new_y, new_z]$
  else if $(distance\,between\,new\,Position_{p_e}\,and\,Positions_{crowd}[p_e])) = 0)$ **then**
    $newPosition_{p_e} \leftarrow currentPosition_{p_e}$
  end if
end for



(a) Separation **(b) Cohesion** **(c) Alignment**

Figure 4.7: The three different types of steering behaviors (a) Separation, (b) Cohesion, and (c) Alignment. A similar figure is described by [1]

Goal-directed Behaviour : pedestrians' motivation in each group walking towards a well-defined goal at a certain point in the virtual environment modeled by a $goal - directed\ behaviour$. The computation of the goal-directed steering behavior ($goal_{vector}$) is explained in the Algorithm 13. As shown in this algorithm, the distance between the current position of pedestrian ($current\ Position_{p_e}$) and the current active goal ($current\ Goals_{group}$) is calculated. On condition that the distance between these two points is less than a certain value (C_{g1}), then a new goal point is assigned for the current group. Suppose the current goal point $Goals_{group}$ is the last destination point. In that case, the group will terminate the simulation, and this group is not active anymore. Otherwise, the $goal\_directed$ steering behavior calculates the $goal_{vector}$ and normalizing the $goal_{vector}$ by applying a weighting $goal_{weight}$, as illustrated in Algorithm 13.

Algorithm 13 Goal-directed behavior

Inputs:
$current\ Goals_{group}$, $current\ Position_{p_e}(p_e \leftarrow index)$, $goal_{weight}$, C_{g1}
$Goal_i \leftarrow Goals_{index}[g_r]$
if $(distance\ between\ current\ Position_{p_e}\ and\ current\ Goals_{group}) < C_{g1}$ **then**
    $Goals_{index}[g_r] \leftarrow Goals_{index}[g_r] + 1$
    if $Goals_{index}[g_r] \leftarrow length(Goals_{group})$ **then**
        $terminate[g_r] \leftarrow$ "on", $active_{group}[g_r] \leftarrow$ "off"
    end if
end if
$goal_{vector} =\leftarrow subtracting(current\ Goals_{group}, current\ Position_{p_e})$
$goal_{vector} \leftarrow normalizing(goal_{vector}, goal_{weight})$
....
$Combining\ with\ other\ behaviors,\ if\ there\ is$
....

Combining Behaviours: combining behaviors can happen in two ways: (1) switching (2) blending [1]. As the walking circumstance changes in the simulated environment, the pedestrian may "switch" between behavioral modes. Alternatively, these behaviors, which are acting in parallel, are commonly "blended" together. For example, in normal situations, as the pedestrians moving around through the simulated environment toward their destination, they blend both flocking and goal-directed behavior to a single steering force vector to allow the group to walk toward their goal. Pedestrians in the group always must stay moving with the group toward their destination, and they cannot afford to ignore either component behavior. Furthermore, suppose that a group of pedestrians move in a given environment, and they approach an obstacle or other pedestrians in the same or different groups. This situation leads to a behavioral switch from moving to collision avoidance. All pedestrians are trying to avoid collisions with obstacles and pedestrians,

and pedestrians will not prefer to remain with the group while avoiding obstacles. Multiple steering behaviors previously discussed guarantee the shortest collision-free paths for pedestrians, and pedestrians never get stuck behind obstacles. The proposed method uses $Python$ scripted models to evaluate pedestrians' behaviors and computed a new state of pedestrians, and also generates $3D$ scenes and viewed in the animation software package $Autodesk\,Maya$ [123], as illustrated in the following subsection.

4.2.3 Locomotion

In this study, the proposed method computes the pedestrians' movement taking into account the steering behaviors, and generates locomotion for each pedestrian in the virtual environment in real-time. In this study, the $Maya$'s keyframe is used for $3D$ animating of a large crowd of multi-groups of pedestrians walking around through the virtual environment toward their destination points. As an implementation of the pedestrian flow, we used seven different pedestrians for this simulation. Then, we imported into the $Maya$ via $Python$ scripted models. Subsequently, $Maya$ allows the scripted models to control the pedestrians. Based on the proposed method, the position ($Position_{p_e}$) and the velocity ($Velocity_{p_e}$) of each virtual pedestrian in the scene are updated in real-time.

Chapter 5

Path Planning Results

5.1 Path Planning Simulation Results

In this study, the proposed method is implemented in the $Python$ and $Matlab$ programming language on a laptop computer with Intel(TM) Core(TM) i5-8300H CPU, 2.3GHz, and 8GB RAM. The performances of the developed method have been tested on many different workspace scenarios with different obstacle layouts. We have changed the workspace's size and the number of obstacles scattered in the workspace in all tested scenarios. Additionally, the starting C_s and the goal C_g points are positioned in different locations in the free space C_{free}. For the workspace scenarios shown in Figure 3.5, the proposed method is used to find an optimal or near-optimal path from C_s to C_g. The simulation results to find the shortest path for the mobile robot is presented in Subsections 5.1.1 to 5.1.5. The implementation of the proposed method for generating the shortest path between C_s and C_g in multi-robot systems is discussed in Subsection 5.1.6. Additionally, the performance evaluation of the proposed method compared with the other path planning methods is presented in Subsection 5.1.7. Then, the application of the proposed method for solving the multi-goal path planning problems for single and multiple mobile robot systems is presented in Subsection 5.1.8.

5.1.1 Boundary Node Method (BNM)

This section presents the simulation results of the generated IFP between C_s and C_g for the workspaces shown in Figure 3.5 by using BNM. The obtained IFP is represented by a set of waypoints $w_{(j)}, (j = 1 \rightarrow J)$. Each new position of waypoint $w_{(j+1)}$ allocated after the position of the current waypoint $w_{(j)}$, where J represents the time in which the robot reaches the goal point.

The simulation results for all tested scenarios are presented in Figure 5.1, and the summary of the obtained results is provided in Table 5.1. The figure shows that the obtained IFP allows the robot to move from C_s to C_g without colliding obstacles. Red

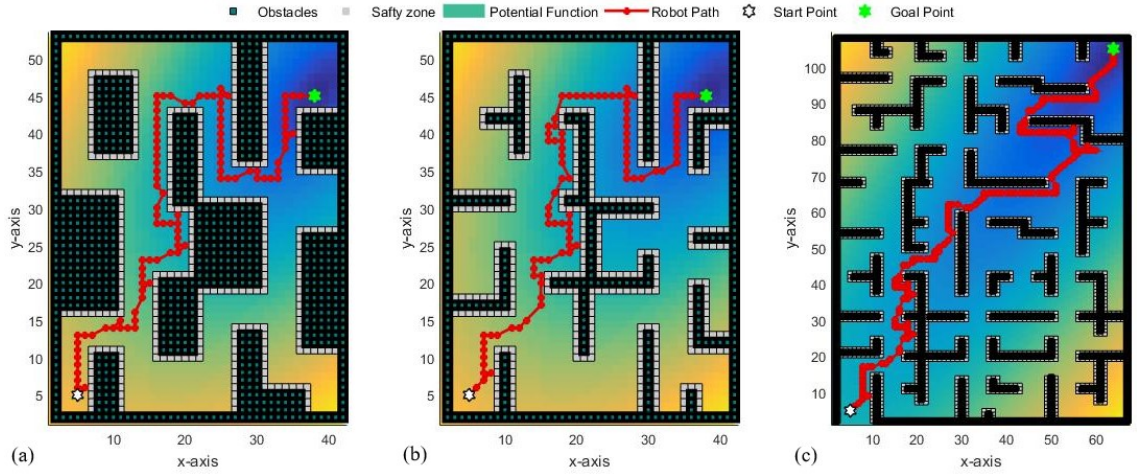


Figure 5.1: Simulation results of the generated initial feasible path (IFP) for all three workspace scenarios using BNM.

Table 5.1: The total computational time and path length of the initial feasible path (IFP) and the shortest path by using BNM and PEM.

Workspace No.	**Total Computational Time [s]**		**Total Path Length [unit]**	
	IFP	**Shortest Path**	**IFP**	**Shortest Path**
1	0.955601	1.043110	112.9662	83.4301
2	0.896196	1.004691	110.1285	81.0895
3	1.100025	1.138494	201.3783	146.3850

circle objects represent the path's waypoints, and for better clarity, these waypoints are connected into a continuous path. It can be seen that the BNM method can overcome the local minima problem. From Table 5.1, we can see that the developed method provides the collision-free path for the robot in a short time, in particular for the highly complex environment shown in Figure 5.1c. As presented in the third scenario, the total computational time to find a IFP is less than 1.1 $second$.

The results show that the BNM method has been well applied for generating IFP for a mobile robot, and this method has achieved good results in terms of safety and short computational time. However, the generated path is not optimal in terms of the total path length. To reduce the path length, a new developed method, called PEM, is used as explained in Subsection 3.2.3, and the obtained results are presented in the following subsection.

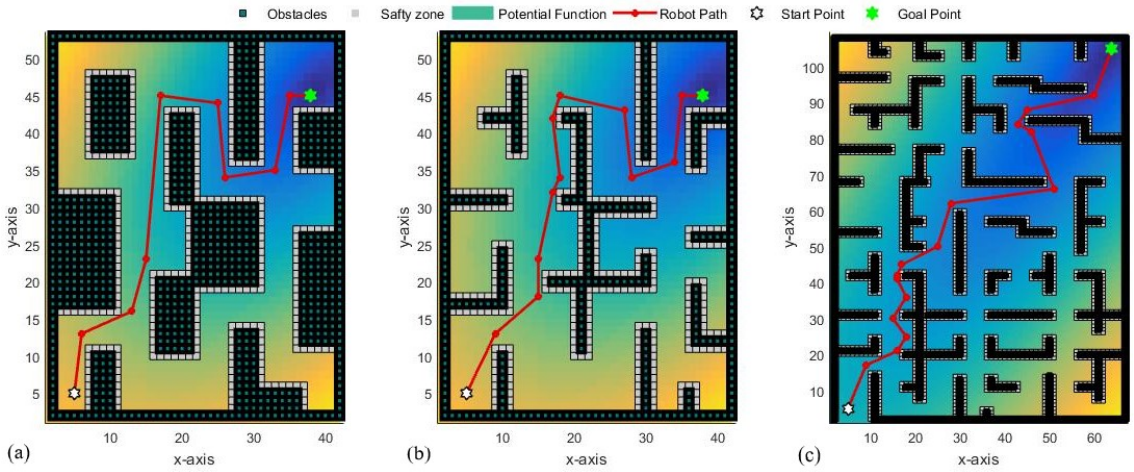


Figure 5.2: Simulation results of the generated path for all three workspace scenarios by using PEM.

5.1.2 Path Enhancement Method (PEM)

This section presents the obtained results from implementing the PEM method to find an optimal or near-optimal collision-free path for the three workspace scenarios. The simulation results are shown in Figure 5.2. Moreover, the obtained computational time with the path length for all simulated scenarios is provided in Table 5.1. As shown in Figure 5.2, the PEM method can find the collision-free path covering the least number of waypoints, where the solid red lines represent the best solution. According to the obtained results presented in Table 5.1, it can be stated that the total path length for all three designed workspaces was significantly reduced, and the percentage of enhancement for all three scenarios are 26.2%, 26.4%, and 27.3%, respectively.

The geometrical complexity of the working environment is the main factor affecting the computational time. However, the simulation results show that the computational time required to obtain the IFP and the shortest path by using $BNM\&PEM$ is not increased significantly with the growing complexity of the workspace. For example, the workspace's size and the number of obstacles are increased 3.2 and 2.6 times, respectively, from the second scenario (see Figure 5.2b) to the third scenario (see Figure 5.2c). Accordingly, the total computational time to find the IFP and the shortest path is increased only around 1.5 and 1.4 times, respectively (see Table 5.1). In the second scenario (see Figure 5.2b) and the first scenario (see Figure 5.2a), the required computational time to determine the IFP and the shortest path is increased by 6.6% and 3.8%, respectively, as the number of obstacles is increased by 123.2% for the same workspace. For each iteration t during the search process, all obstacles in the workspace are examined for possible collisions with the direct path from the current $p_1(t)$ to updated $p_2(t)$ nodes' location.

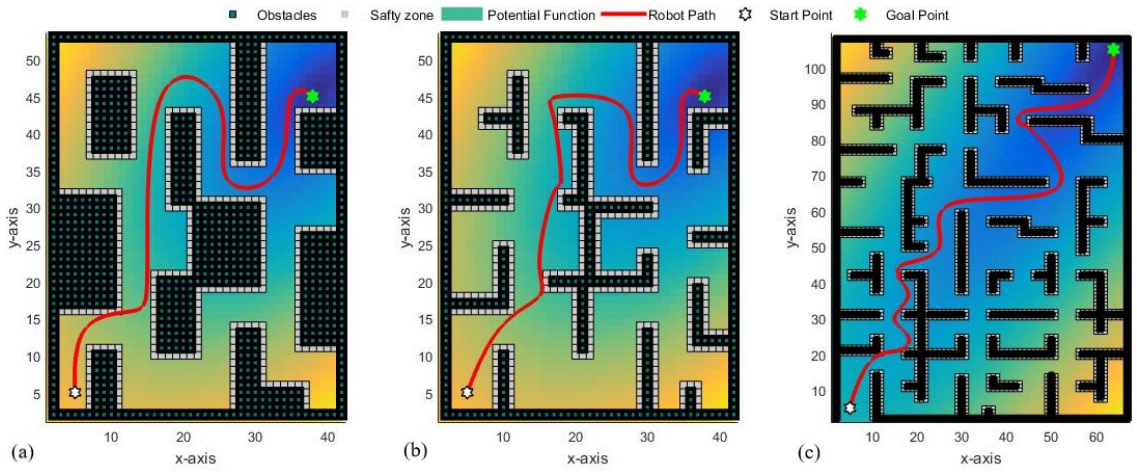


Figure 5.3: Simulation results of the generated smooth path for all three workspace scenarios by using the cubic spline method.

5.1.3 Path Smoothing Using Interpolation Technique

In the obtained results shown in Figure 5.2, it can be observed that the proposed method generates the path that consists of straight lines between waypoints with sharp turns. In real applications, when the robot follows a path in the workspace, it may not be able to make a sharp turn, and also, it is not the safest path for the robot. In order to improve the path concerning the robot dynamics, the cubic spline method is applied to construct a continuous smooth path that connects the starting point to the endpoint for all three designed workspaces, and the results are presented in Figure 5.3.

The results demonstrate that the spline method can generate a continuous smooth path to eliminate sharp turns. At the same time, the cumulative length of the smooth path shown in Figure 5.3 is longer than the cumulative length of the line segment path presented in Figure 5.2. The path length is increased by 7%, 4.4%, and 4.5% for all three scenarios, respectively.

The cubic spline method aims to generate a smooth path for the shortest path that connects the starting point to the goal point. However, in some cases, the constructed smooth path can bring the robot close to the safety zone around the obstacles, or the robot collides with the safety zone, which is undesirable in practice. In order to avoid the possibility of overlapping the path traced by the robot with the safety zone, additional waypoints can be inserted between the original waypoints until no safety zone or obstacles were found along the resulting path, as explained [122]. Alternatively, Piecewise Cubic Hermite Interpolating Polynomial ($PCHIP$) can be used to construct a continuous smooth path, as illustrated in [14, 126]. The $PCHIP$ is similar to the cubic spline interpolation, but

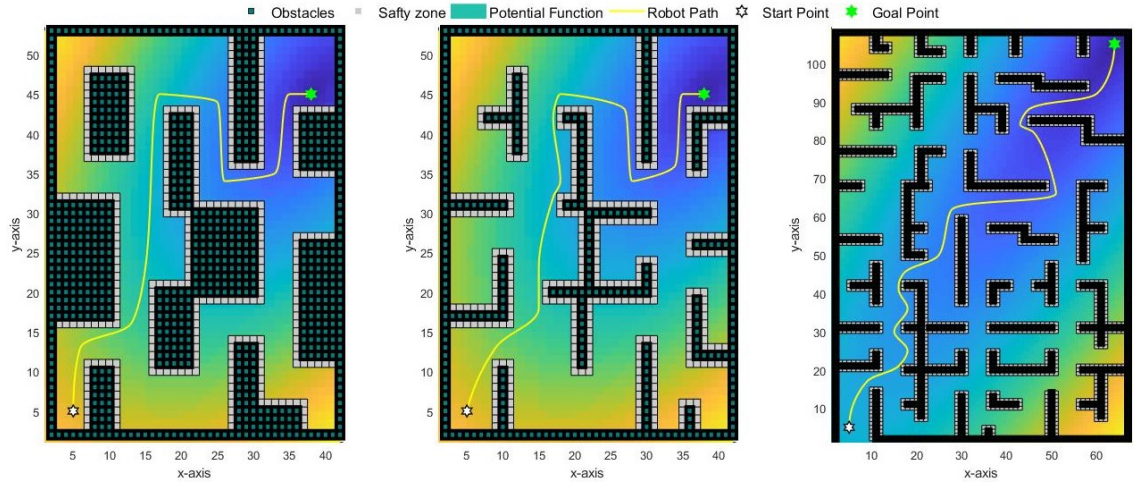


Figure 5.4: Simulation results of the generated smooth path for all three workspace scenarios by using $PCHIP$.

the $PCHIP$ ensures a shape-preserving and avoiding the overshoots and oscillations that could arise from spline interpolation. The generated path from the X and Y vectors of the waypoints w is a zigzag line; we generate a new vector x_i of about "1000" points from the start point to the goal point. $y_i = PCHIP(X, Y, x_i)$ returns a vector of interpolated values y_i containing elements corresponding to the x_i. The resulting x_i versus y_i gives the smoothed path. Figure 5.4 shows the obtained results of the smoothed paths generated by using $PCHIP$. The generated paths' lengths increased by 4.4%, 2.4%, and 2.6% for all three scenarios, respectively. We can see the difference between the interpolation results produced by $PCHIP$ and cubic spline in Figures 5.3 and 5.4.

5.1.4 Irregular-Obstacle Environment

The proposed method uses the grid-based method to create a workspace environment. The workspace environment is divided into many small square grid cells of the same size (1×1 $unit$). Each grid cell can either correspond to a navigable area or a space occupied by obstacles. Different obstacle shapes can be generated, such as circular or non-convex obstacles, by approximating the obstacles' shape and dividing it into square grid cells. The completeness of the obstacles' shape depends on the resolution of the grid environment. Figures 5.5a and 5.5e show two examples of different workspace scenarios. In these scenarios, the workspace consists of (50×50) grid cells, and the number of the obstacles in the workspace is "312" and "316" grid cells, respectively. The starting C_s and goal C_g points are positioned in the free space C_{free} at $(5, 5)$ and $(45, 45)$, respectively. The proposed method is used to determine an optimal or near-optimal path from C_s to C_g. The simulation results for generating the IFP between C_s and C_g are presented in Figures 5.5b and 5.5f. The figures show that the obtained IFP could successfully drive

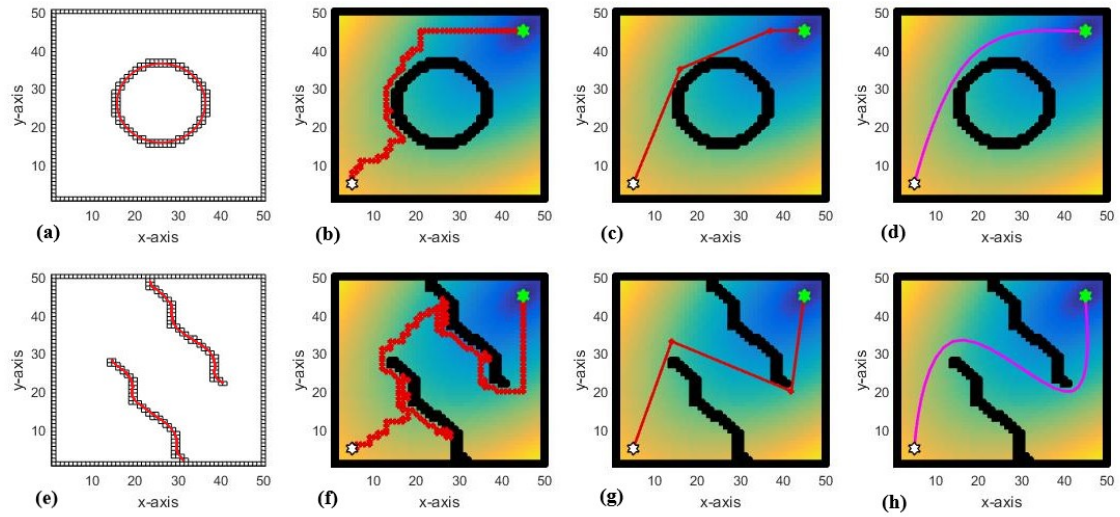


Figure 5.5: Examples of different workspace with different obstacle shapes (a and e), and the simulation results of the generated path for two workspace scenarios by using BNM (b and f), PEM (c and g), and cubic spline method (d and h).

the robot toward the goal while avoiding obstacles in the highly complex environment. The robot's location is represented by a red circle object at each iteration. Figures 5.5c and 5.5g present the obtained results of the PEM method to find an optimal or near-optimal path. As shown in the figures, the PEM method can generate a short path, where the solid red lines between C_s and C_g represent the best solution. Finally, the cubic spline interpolation is used to construct a continuous smooth path that connects the starting point to the goal point, and the results are presented in Figures 5.5d and 5.5h.

5.1.5 Three-Dimensional Environment

The proposed method extended further to include altitude as a third coordinate to solve the path planning problems in a three-dimensional ($3D$) workspace. The workspace discretized into uniform cubic grid cells ($1 \times 1 \times 1\ unit$), and the generated path is consists of a sequence of cubic cells in a $3D$ grid model. The proposed method was implemented for several $3D$ scenarios with satisfactory results. An example of the workspace scenario is presented in Figure 5.6. The left-hand side of the figure represents the simulation results of the generated initial feasible path (IFP) for the $3D$ workspace scenarios using BNM. The figure shows that the obtained IFP allows the robot to move from C_s to C_g without colliding obstacles. Green circle objects represent the path's waypoints, and for better clarity, these waypoints are connected into a continuous path. The right-hand side represents the obtained results from implementing the PEM method to find an optimal or near-optimal collision-free path. As shown in the figure the PEM method can

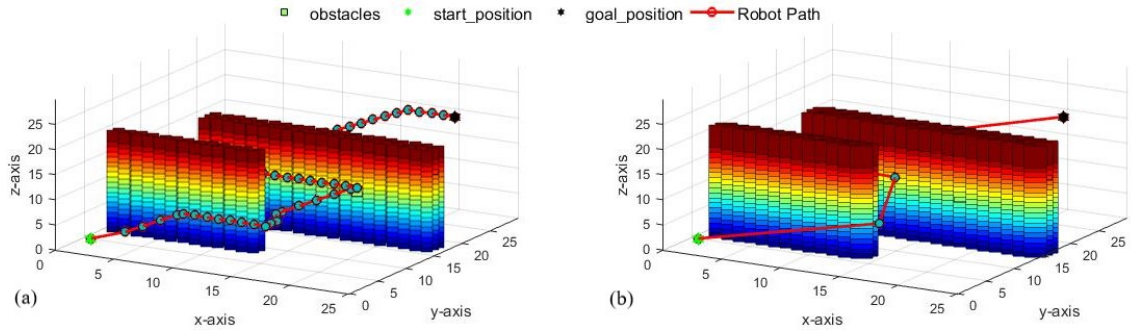


Figure 5.6: Simulation results for constructing a collision-free path by using BNM (a) and PEM (b) to solve the path planning problem in a three-dimensional $(3D)$ workspace.

find the collision-free path covering the least number of waypoints, where the solid red lines represent the best solution. The results demonstrate that the proposed method was successfully used to create the path from the initial point to the goal point in a given $3D$ workspace.

5.1.6 Multiple Robot System

This section presents the implementation of the proposed method for collision avoidance in multi-robot systems. Figure 5.7 shows an example of the simulation result for a multi-robot system. In this example, there are four robots with four different starting C_s and goal C_g points corresponding to each robot, and there are "304" static obstacles in the workspace. The problem formulation determines each robot's path in the simulated environment by avoiding collision with static obstacles and other moving robots in the system. Each robot moves from a starting position C_s, through all the intermediate waypoints w until it reaches the goal position C_g. Each robot uses the BNM to find IFP between C_s and C_g in the workspace without colliding with any obstacles (see Figures 5.7a and 5.7d). The IFP is generated from a set of waypoints w that the robot visits before reaching the final destination point. Then, the PEM method used to reduce the number of waypoints of the IFP to find an optimal or close-to-optimal path (see Figures 5.7b and 5.7e). Finally, the cubic spline interpolation is applied to construct a continuous smooth path that connects the starting point to the goal point (see Figures 5.7c and f). The simulation results show that all the robots reached the final destination positions successfully without any collision with static obstacles or other robots.

5.1.7 Performance Evaluation

This section presents the performance evaluation of the proposed method compared to PSO, $A\text{-}Star$, and APF. For this purpose, a simple example of a workspace is created, as shown in Figure 5.8. The workspace's size is set to (43×68), where the space

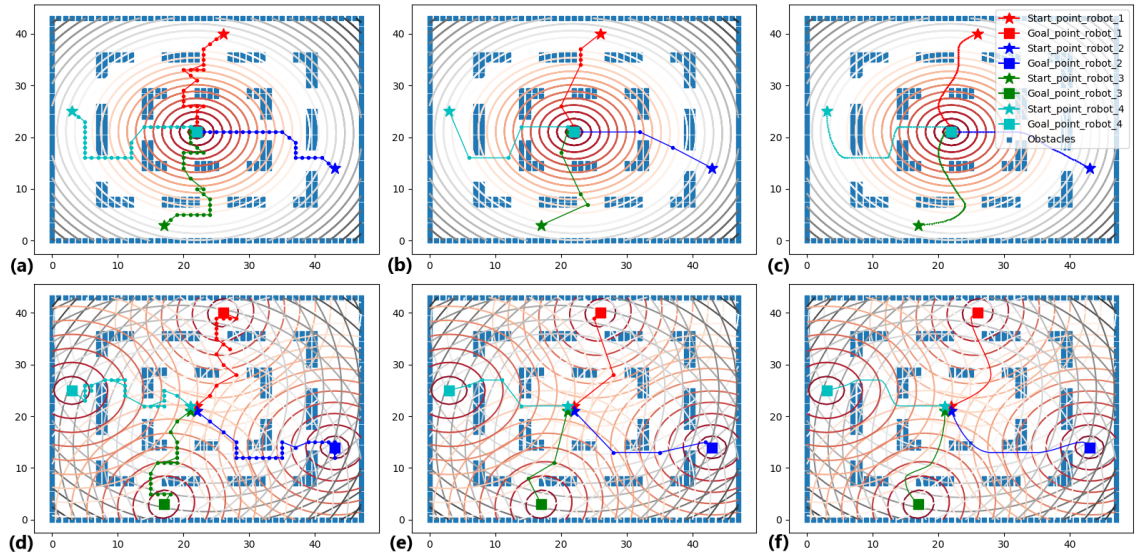


Figure 5.7: Simulation results for solving the multi-robot path planning problem.

Table 5.2: Summary of the obtained results from the implementation of BNM, PSO, $A\text{-}Star$, and APF for solving the path planning problem.

Method	Total Computational Time [s]	Total Path Length [unit]
BNM	0.82	53.49
PSO	1.51	57.70
$A\text{-}Star$	2.57	57.11
APF	0.66	61.00

occupied by obstacles C_{obs} consists of "1078" grid cells, and the obstacle-free space C_{free} consists of "1846" grid cells. After constructing the workspace with obstacles, all methods namely BNM, PSO, $A\text{-}Star$, and APF are used to find the shortest path between C_s at $(8, 10)$ and C_g at $(32, 56)$. The simulation results are shown in Figure 5.8, and the summary of the obtained results is provided in Table 5.2. By comparing the obtained results, which are presented in Table 5.2 for the four methods, it can be seen that the BNM method can find the shortest path within less than one second. The BNM method requires less than 55% and 32% of the computational time to find the shortest collision-free path using PSO and $A\text{-}Star$, respectively. In terms of the total path length, the shortest path achieved by the BNM method is about 7.2% and 6.3% shorter than the path length generated by PSO and $A\text{-}Star$, respectively. In this workspace, the computational time required to find the shortest path by using APF is lower by 20% compared to BNM. In contrast, the shortest path achieved by BNM is 12% shorter than the path length generated by APF.

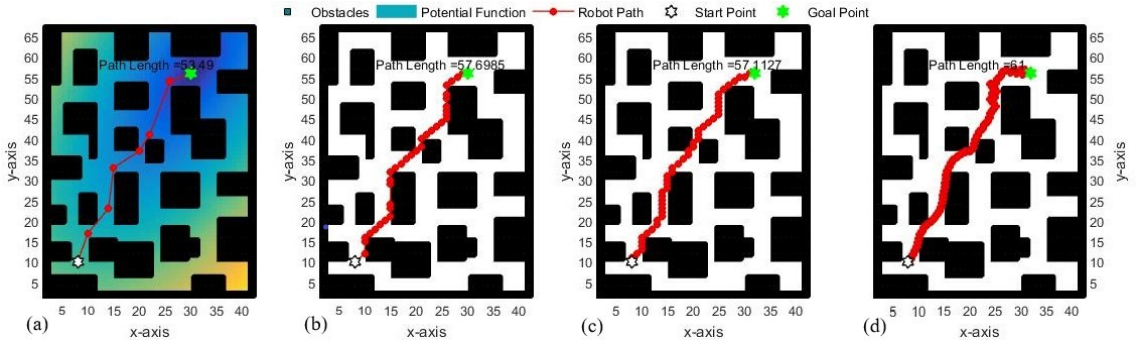


Figure 5.8: Simulation results for solving the path planning problem in a $2D$ workspace by using BNM (a), PSO (b), A-$Star$ (c), and APF (d).

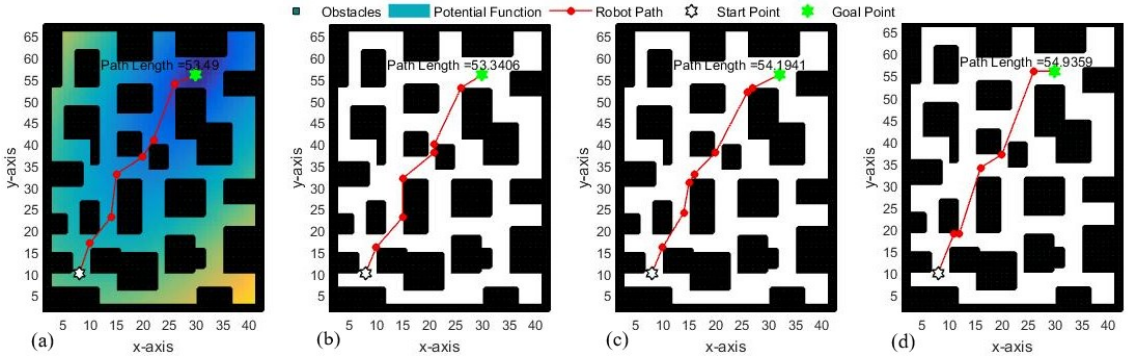


Figure 5.9: Simulation results from the implementation of the PEM method for optimizing the generated paths by using BNM (a), PSO (b), A-$Star$ (c), and APF (d).

The PEM method can also be used to optimize the generated paths obtained by using BNM, PSO, A-$Star$, and APF. The obtained results from the implementation of the PEM are presented in Figure 5.9. As shown in the figure, the PEM method can find the collision-free path covering the least number of waypoints, and the solid red lines represent the best solution. The results revealed that the length of the paths obtained from PSO, A-$Star$, and APF reduced by 7.6%, 5.1%, and 9.9%, respectively. Also, the cubic spline interpolation is used to generate a continuous smooth path that connects the starting point to the endpoint, and the results are presented in Figure 5.10.

To demonstrate the performance of the BNM method for solving the robot path planning problem in the workspace that has been used previously in [2–5], a $2D$ workspace is created as shown in Figure 5.11. The size of the workspace is set to (67×67), the space occupied by obstacles C_{obs} consists of "1520 "grid cells, and the obstacle-free space C_{free} consists of "2969" grid cells. After constructing the workspace with obstacles,

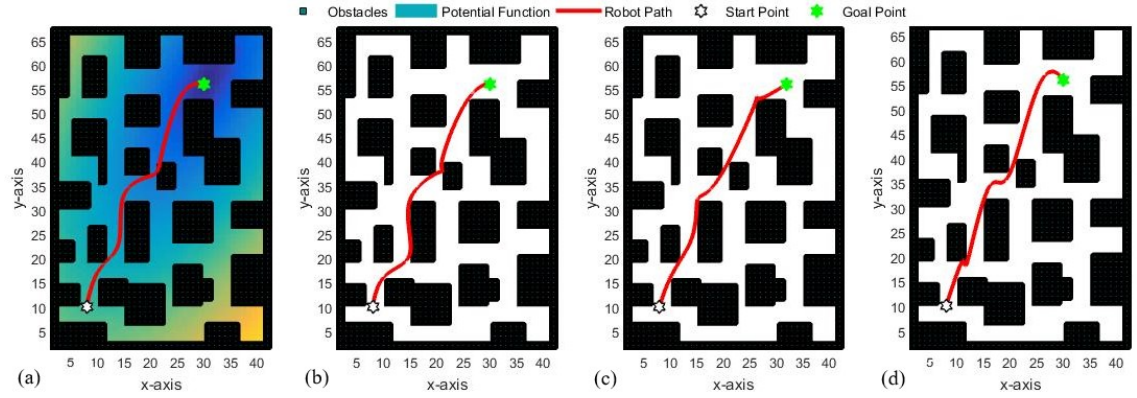


Figure 5.10: Simulation results from the implementation of the cubic spline method for smoothing the generated paths by using BNM (a), PSO (b), $A\text{-}Star$ (c), and APF (d).

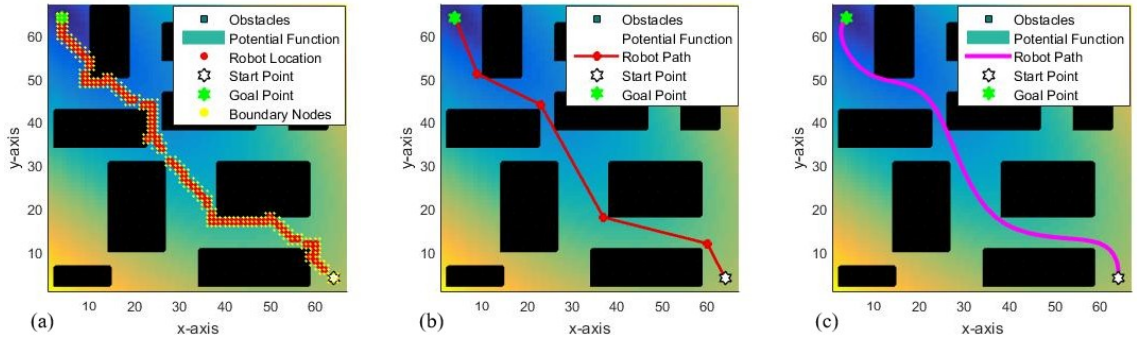


Figure 5.11: Simulation results for generating the path by using BNM (a), PEM (b), and cubic spline (c) in the workspace that previously has been used in [2–5].

the BNM method is used to generate IFP (Figure 5.11a) from the C_s at $(64, 4)$ to the C_g at $(4, 64)$. Then, the PEM method is implemented to find the shortest path (see Figure 5.11b). After that, the cubic spline method is applied to generate a smooth path (see Figure 5.11c). The simulation results compared with the results obtained by using an improved GA that have been used in the previous studies (see Table 5.3) to solve the path planning problem in the same workspace [2–5]. By comparing the obtained results from the proposed method and an improved GA in terms of the computational time, it is observed that the computational time of the proposed method was remarkably reduced.

The comparison results demonstrate the performance of the proposed method for solving the robot path planning problem. A simple workspace scenario has been selected for the comparison because the computational time to find the shortest collision-free path by other path planning methods is growing exponentially as the complexity of the

Table 5.3: Total computational time required to find the shortest path by using BNM and improved GA.

Method	**Total Computational Time** [s]
Improved GA Ref [3]	1.03
Improved GA Ref [4]	4.07
Improved GA Ref [2]	1.68
Improved GA Ref [5]	0.85
BNM	0.964

Table 5.4: Mean and standard deviation (Std) of the computational time and the path length for "1000" independent runs to find the collision-free path using BNM, PSO, GA, and A-$Star$,

Methods	**Computational Time,** CT [s]		**Path Length,** PL [$unit$]	
	$Mean_{CT}$	Std_{CT}	$Mean_{PL}$	Std_{PL}
BNM	0.0142	0.0072	30.7710	15.9340
A-$Star$	0.0489	0.0640	30.0907	14.6124
PSO	0.0217	0.0052	34.3788	15.2495
GA	0.1188	0.2122	33.0144	11.1463

path-planning problem increases. Even though in some circumstances, the path planning methods cannot find a feasible path [32, 80, 85]. In contrast, the proposed method BNM solves these problems.

To validate the proposed method and compare its performance with the A-$Star$, PSO, and GA, a $2D$ workspace is created. The workspace's size set to (60×60) and the space occupied by obstacles C_{obs} consists of "136" grid cells. Afterwards, the BNM, A-$Star$, PSO, and GA are implemented simultaneously to find the collision-free path for "1000" independent runs. At each time in the test, the starting point C_s, and the goal point C_g are created randomly. Moreover, the obstacles are distributed randomly in the working environment. Each random placement of the obstacles led to a different workspace layout. Two parameters are used to compare the path planning methods: the computational time and the path length. The mean and standard deviation (Std) of the calculated computational time and the path length are presented in Table 5.4. The obtained results have shown that the proposed method achieved the best solution within a reasonable computational time. Moreover, the computational time to find the collision-free path is decreased significantly compared with other path planning methods. Compared with PSO and GA, the BNM method showed noticeable improvement in the path length. The mean value of the path length obtained by the proposed method is smaller than that obtained from PSO and GA by 11.73% and 7.3%, respectively. However, the mean value of the path length generated by BNM is slightly greater than that obtained from A-$Star$ by 2.21%. The

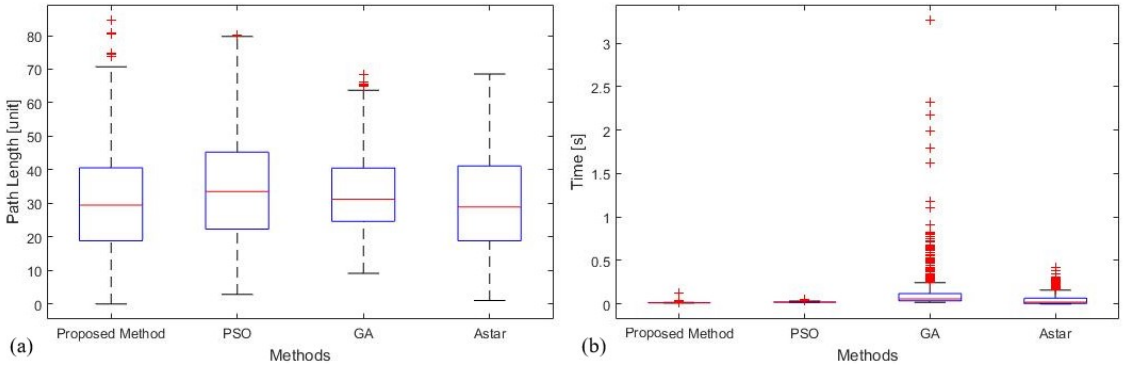


Figure 5.12: Performance evaluation of BNM, PSO, GA, and $A\text{-}Star$ to find the collision-free path for "1000" independent runs, (a) presents the obtained results of the path length, and (b) presents the computational time results.

graphical representation of the obtained results is illustrated in Figure 5.12. As shown in the figure, the PSO method has the least variance of computational time, and GA better than the other method in terms of variance of the path length. The comparative study shows that heuristic algorithms did not yield optimal results, and the results agree with [127].

5.1.8 Multi-Goal Path Planning

This section presents the implementation of the proposed method for solving the multi-goal path planning problem MTP for single and multiple mobile robot systems. The MTP is formulated to find a collision-free path for the robot by avoiding collision with obstacles in the simulated environment. The robot starts to move from the first goal point, and the robot moves through all the intermediate goal points, then it returns to the first goal position. The robot uses the proposed method to find the collision-free path to visit all the goal points in the simulated environment. In order to validate the performance of the developed method, different simulated configurations with different parameters have been conducted.

Simulation Results of Single-Robot System

This section presents the obtained simulation results of the multiple-goals path planning problem MTP for a single mobile robot whose task requires visiting multiple-goal points. In this study, GA and $BNM\&PEM$ have been implemented for solving the MTP in several simulated scenarios with different obstacles layouts and different goal points. To demonstrate the multiple-goals path planning problem in a given $2D$ workspace with obstacles, let us consider an illustrative example shown in Figure 5.13. The problem is solved as follow: first, the optimal sequence of the goal points (3 $goals$)

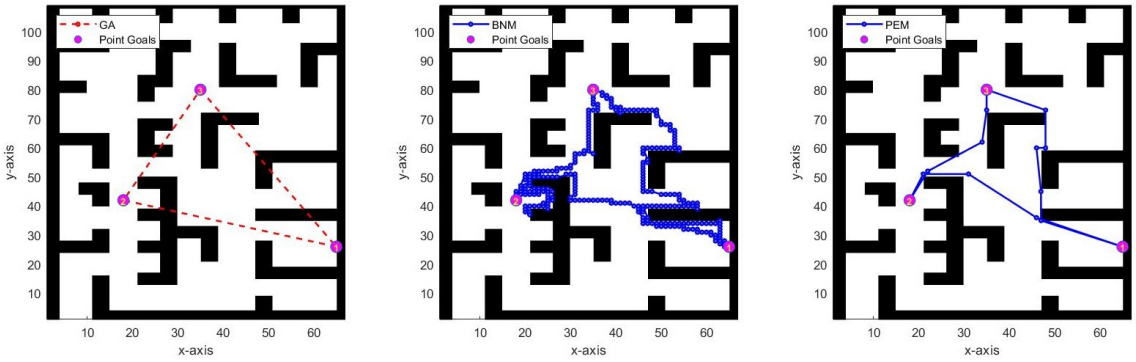


Figure 5.13: An illustrated example of the multi-goal path planning problem for three randomly-selected goal points. (a) the path is determined by using GA, plotted in a red dashed line. (b) the IFP is generated by using BNM, plotted in the blue line. (c) optimize the generated path by using PEM, plotted in the blue line.

located in a C_{free} is determined, regardless of the obstacles, by using GA (see Figure 5.13a). As shown in the figure, the path (tour) starts from g_1, passing through g_2 and g_3 and returns to g_1, where the goals marked in the pink circle object, and the red dashed lines represent the shortest path between sequenced goal points. Next, the BNM method is used to find the initial feasible path (IFP) connecting every pair of the sequenced goal points, as illustrated in Figure 5.13b. It can be seen clearly that the BNM successfully generated the IFP for the robot to move from g_1 to g_2 (see Figure 5.14a), from g_2 to g_3 (see Figure 5.14b), and from g_3 to g_1 (see Figure 5.14c). As the robot moves from g_i to g_{i+1}, $(i = 1...n)$ in the workspace, the IFP is constructed from the waypoints $w_{(j)}, (j = 1 \rightarrow J)$ that visited by the robot, where J represents the time required by the robot to reach the destination point. As shown in Figure 5.14$a \rightarrow c$, the waypoints w are represented by blue circles objects, each new waypoint position $w_{(j+1)}$ is allocated after the current waypoint position $w_{(j)}$. The obtained IFP is the collision-free path, and the waypoints do not fall on any obstacle, and also, the line segments that connect the waypoints do not intersect with obstacles.

The BNM method generated the IFP safely and efficiently, but the path is not optimal in terms of the total path length. Therefore, the PEM method is implemented to find an optimal or close-to-optimal path for the robot by reducing waypoints and the path length. Figure 5.13c presents the obtained results, where the thick blue line object represents the shortest path. A complete multi-goal path can be constructed by joining all line segments obtained from g_1 to g_2 (Figure 5.14d), from g_2 to g_3 (see Figure 5.14e), and from g_3 to g_1 (see Figure 5.14f). The exact path length L is the sum of path length between goal points sequentially $(l_1, \ldots, l_n)$.

In this study, the performance of the proposed method has been examined to solve

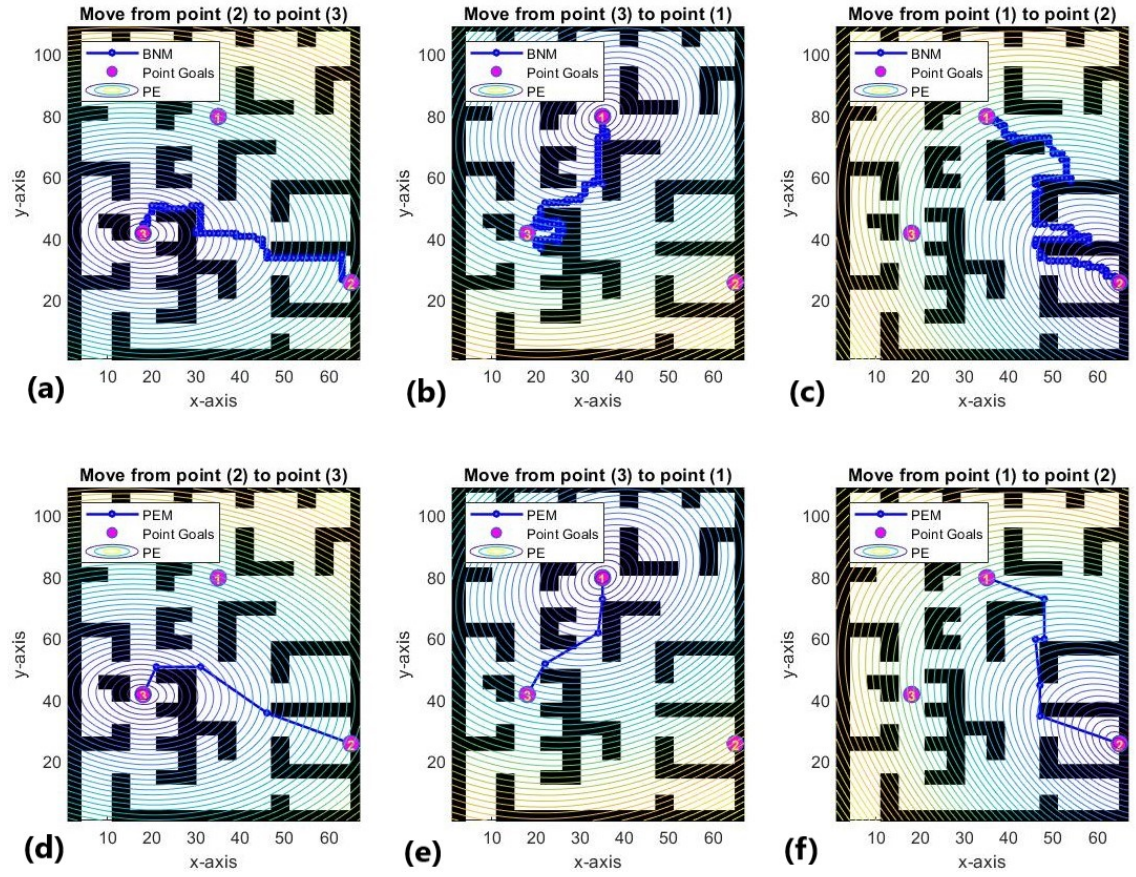


Figure 5.14: Illustrates the steps of solving a multi-goal path planning problem. The BNM is applied to generate the IFP for the robot to move from g_1 to g_2 (a), from g_2 to g_3 (b), and from g_3 to g_1 (c). The PEM is implemented to generate the shortest path from g_1 to g_2 (d), from g_2 to g_3 (e), and from g_3 to g_1 (f).

Table 5.5: Performance evaluation results: The mean and standard deviation of the computational time (in $seconds$) required to find the feasible path for a single mobile robot in three different working environment.

No. of Goals	**First Environment**		**Second Environment**		**Third Environment**	
	Mean$_{CT}$	**Std**$_{CT}$	**Mean**$_{PL}$	**Std**$_{PL}$	**Mean**$_{PL}$	**Std**$_{PL}$
2	4.9125	0.1966	4.8895	0.1249	4.5685	0.1641
4	5.1960	0.2057	5.1235	0.0953	4.8500	0.1126
6	5.3210	0.1501	5.2915	0.1200	5.0000	0.09531
8	5.4410	0.1210	5.4935	0.3305	5.0970	0.1425
10	5.6395	0.1245	5.7085	0.4026	5.1665	0.1111
12	5.6585	0.1608	5.8520	0.1461	5.2430	0.1246
14	5.7020	0.2240	5.9955	0.2518	5.3095	0.1236
16	5.7240	0.2368	6.0505	0.1719	5.3790	0.1587
18	5.7670	0.2157	6.1275	0.1831	5.4735	0.2103
20	5.8170	0.1787	6.1575	0.3323	5.5105	0.1694

the MTP for a single mobile robot in three obstacle-filled environment scenarios with complex obstacles layout (see Figure 3.12). In these scenarios, the given working environment is known in advance, and the obstacles are assumed to be static in their position. Several goal points (n) are positioned randomly in the free space of the working environment. In implementing and testing the developed methods, we compute the shortest collision-free path for the robot to reach all goal points and visit each goal once. Throughout this section, we consider different numbers of goal points (2, 4, 6, 8, 10, 12, 14, 16, 18, and 20 goal points). The proposed method has been tested for "200" randomly generated scenarios to find the initial feasible path. At each instance, the goal points are located randomly in the working environment, where each random placement of the goal points led to different results.

The performance of the proposed method has been examined in terms of the total execution time required to find the feasible path. The mean and standard deviation (Std) of the computational time for each instance is presented in Table 5.5. The graphical representation of the simulation results is given in Figure 5.15. The obtained results reveal that the proposed method can determine the shortest path between each instance's goal points within a reasonable computational time. In particular, for the high number of goal points in a highly complex environment, the total computational time to find the feasible path is around "6" $second$. The simulation results show that the mean value of the computational time is not increased significantly with increasing the number of goal points and the complexity of the workspace (see Table 5.5). The computational time needs to find the collision-free path depends on the number of goals and their ordering and the obstacles' geometric complexity.

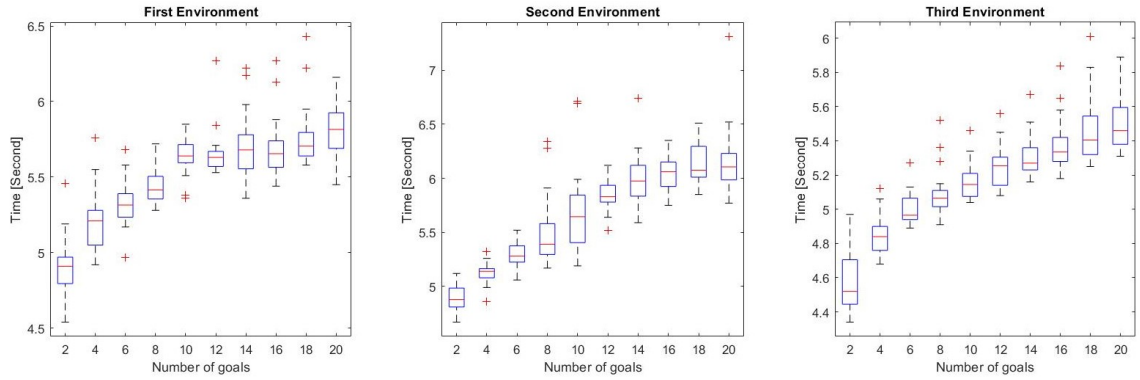


Figure 5.15: The influence of the number of goal points on the total computational time required to solve MTP for each simulated working environment.

An example of the obtained results for each simulated scenario with "20" goal points located randomly in the working environment are presented in Figures 5.16, 5.17, and 5.18. For each examined scenario, the GA is used to optimize the goal points sequence with the absence of obstacles. The obtained results are shown in Figures 5.16a, 5.17a and 5.18a, where the goal points marked in pink circle objects and the red dashed lines represent the shortest path between sequenced goal points. As shown in the figures, the robot visits all given goal positions located in the working environment with minimizes the path length and the total execution time.

Subsequently, the BNM method is used for generating the IFP between every pair of the sequenced goal points, and the simulation results are presented in Figures 5.16b, 5.17b and 5.18b. The obtained results of the IFP represented by a set of waypoints w, and the blue circle objects indicate the waypoints. For better clarity, these waypoints are connected into a continuous path. As observed from the figures, the final path allows the robot to move from goal-to-goal sequentially and avoid obstacles successfully.

From the obtained results, it can be seen that the BNM has been well applied to generate IFP and reached important achievements in terms of safety and short computational time. However, the path is not optimal in terms of the total path length. Therefore, the PEM method is used to reduce the overall length of the IFP. The obtained results for all tested scenarios are presented in Figures 5.16c and 5.17c and 5.18c, where the solid blue lines between goal points represent the final solution. As shown in the figures, the PEM method finds the collision-free path that covers the lowest number of waypoints, and the total path length for each tested scenario is reduced significantly. Moreover, the obtained results clearly show that the $BNM\&PEM$ methods provide a short and safe path for the robot to visit a given set of goal points (see Figures 5.16, 5.17 and 5.18).

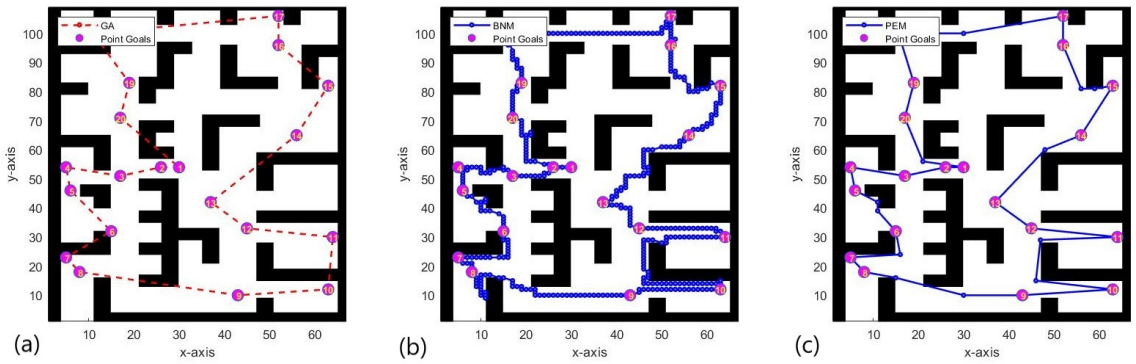


Figure 5.16: The first scenario of the multi-goal path planning problem with "20" randomly-selected goal points. (a) the path is formulated by connecting the sequenced goal points, plotted in a red dashed line. (b) the IFP is generated by using BNM, plotted in the blue line. (c) optimize the IFP by using PEM, plotted in the blue line.

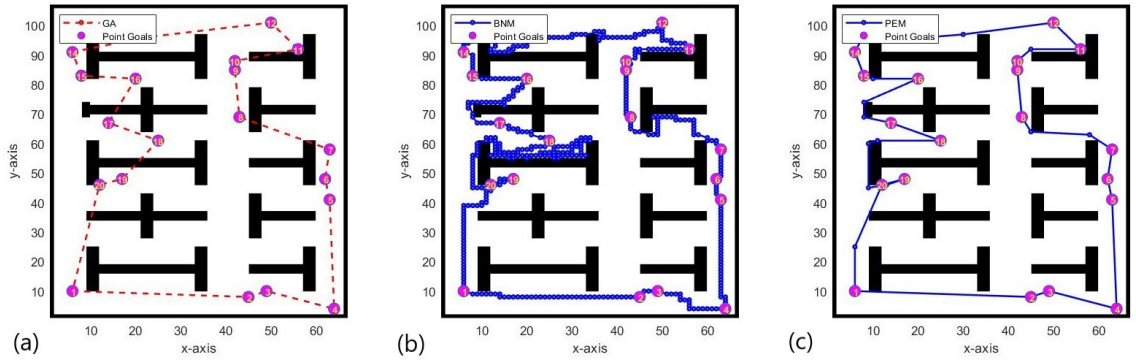


Figure 5.17: The second scenario of the multi-goal path planning problem with "20" randomly-selected goal points. (a) the path is formulated by connecting the sequenced goal points, plotted in a red dashed line. (b) the IFP is generated by using BNM, plotted in the blue line. (c) optimize the IFP by using PEM, plotted in the blue line.

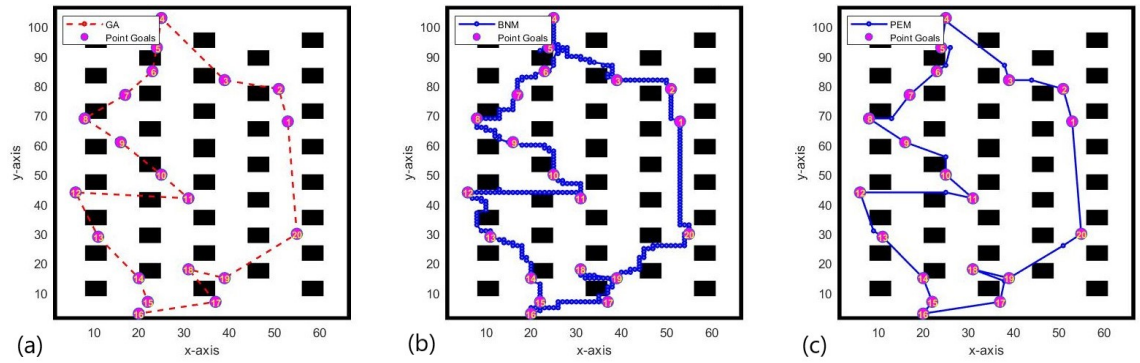


Figure 5.18: The third scenario of the multi-goal path planning problem with "20" randomly-selected goal points. (a) the path is formulated by connecting the sequenced goal points, plotted in a red dashed line. (b) the IFP is generated by using BNM, plotted in the blue line. (c) optimize the IFP by using PEM, plotted in the blue line.

Table 5.6: Characteristics of three different example scenarios with different number of goal points.

Scenarios	**Goal points**
First Scenario (four goal points)	[[14,5],[37,20],[29,41],[19,23]]
Second Scenario (three goal points)	[[14,5],[29,41],[19,23]]
Third Scenario (two goal points)	[[14,5] ,[19,23]]

Simulation Results of Multi-Robot System

In this section, We conducted different simulated scenarios with a different number of goal points and robots. A statistical analysis has been carried out to examine the performance of the proposed method by calculating the execution time to find the collision-free path for each simulated scenario. As shown in Table 5.6, a different number of goal points in different locations are considered in all tested scenarios. The problem is formulated to find a path for each robot in the simulated environment by avoiding collision with static obstacles and other moving robots in the simulated working environment. Each robot uses the BNM method to find the collision-free path to move from the first goal point through all the intermediate goal points until it returns to the first goal position.

The total computational time (CT) required to find the collision-free path for each scenario has been computed for "240" independent runs. The mean and standard deviation (Std) of the computational time is calculated, and the results are provided in Table 5.7. The graphical representation of the simulation results is presented in Figure 5.19. The simulation results reveal that all robots reached their final destination goal within a reasonable computational time without collision with either static obstacles

Table 5.7: Performance evaluation results: the mean and standard deviation of the computational time (CT) in $seconds$ to find a collision-free path for each instance in different simulated scenario.

No. of Robots	**First Scenario**		**Second Scenario**		**Third Scenario**	
	Mean$_{CT}$	**Std**$_{CT}$	**Mean**$_{CT}$	**Std**$_{CT}$	**Mean**$_{CT}$	**Std**$_{CT}$
1	12.6479	0.6727	11.8793	0.7119	6.0778	0.4230
2	25.3430	1.1184	24.2406	1.3893	12.1266	0.7627
3	38.3126	1.1113	36.4111	1.1663	18.5977	1.0296
4	52.5791	2.1483	48.9389	1.6548	25.3420	1.5899
5	66.2990	2.0072	61.7195	1.9908	31.8260	1.4962
6	80.8070	2.1736	76.3014	2.7425	38.7869	1.5057

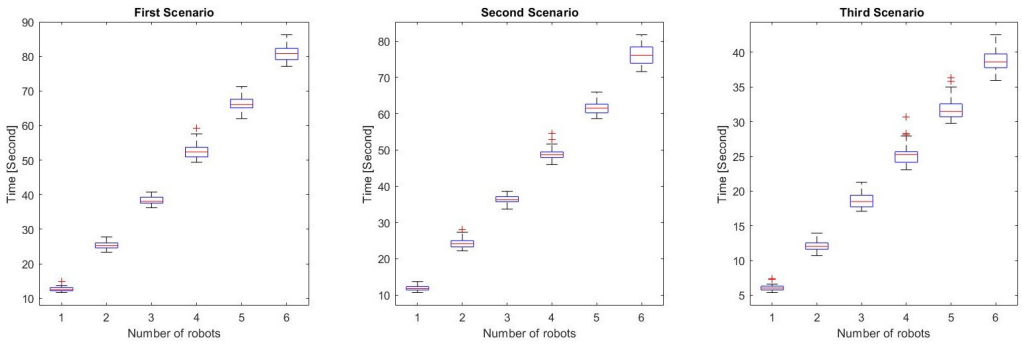


Figure 5.19: The influence of the number of robots on the total computational time to solve MTP using the proposed method for each simulated scenario.

or other robots. Moreover, it is observed that the mean value of the computational time to find the collision-free path increases linearly with an increase in the number of robots, as shown in Figure 5.19, and Table 5.7.

An example of the simulation results for the multi-robot multi-goal path planning problem is shown in Figure 5.20, with five-robots ($m = 5$), four-goal points ($n = 4$), and "304" static obstacles. In the first step, the GA is used to optimize the sequence of the goal points, scattered randomly in the simulated working environment with the absence of the obstacles (see Figure 5.20a). A group of robots ($5\ robots$) moves along a line (denoted by the solid blue line) to visit a group of goal points ($4\ goals$), marked in the red square objects. The BNM method is used to generate a goal-to-goal path to direct each robot from the starting goal point toward the next goal point while avoiding collision with static obstacles and other robots, as illustrated sequentially in Figures 5.20$b \rightarrow f$. Different colours of the small coloured circles represent the position of the robots.

Additionally, an example of the simulation results for different numbers of the robot is presented in Figure 5.21, the number of goal points fixed ("4" $goal\ points$) and the number

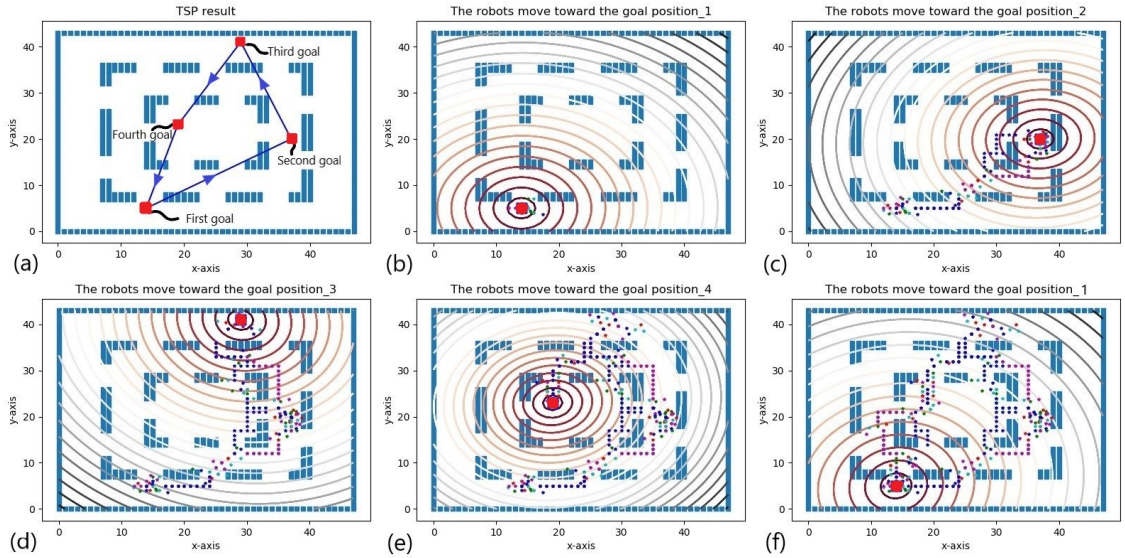


Figure 5.20: Simulation results for solving the MTP: (a) the sequence of the goal points obtained from the implementation of GA, the red square objects represent the goal points. $(b \rightarrow f)$ multiple robots $(5\ robots)$ move to visit multiple-goal points $(4\ goals)$ in a simulated working environment with obstacles.

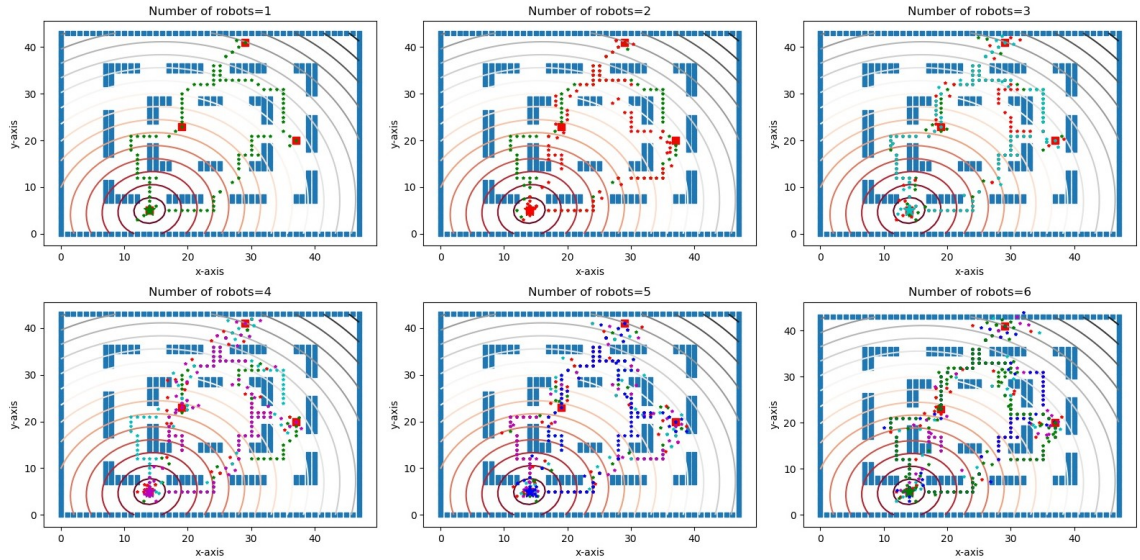


Figure 5.21: Simulation results for solving the MTP: the number of goal points is fixed $(4\ goals)$, and the number of robots is varied $(1 \rightarrow 6)$.

of robots changed from 1 to 6. The simulation results show that each robot generates a collision-free path independently. Moreover, all robots reach the final destination point successfully without collision with either static obstacles or other moving robots.

5.2 Path Planning with Physical Robots

5.2.1 Experimental Results of Path Planning

This section presents the implementation of the developed method, introduced in Chapter 3, on the real robot. The *e-puck* mobile robot, shown in Figure 5.22*a*, is used for the experimental test. The *e-puck* robot has a diameter of 75 mm, and it has two actuators that control the movement speed and direction of the robot. We have chosen an *e-puck* robot because the *e-puck* robot is very compact, small, and flexible [128]. There is also a library extension to $MATLAB$ to program the robot and integrate it with the developed method. The *e-puck* robot uses Bluetooth to connect to the computer, which allows the control programs to be remotely uploaded to the robot. Figure 5.22*b* shows the experimental set-up to demonstrate how the robot navigates along the collision-free path.

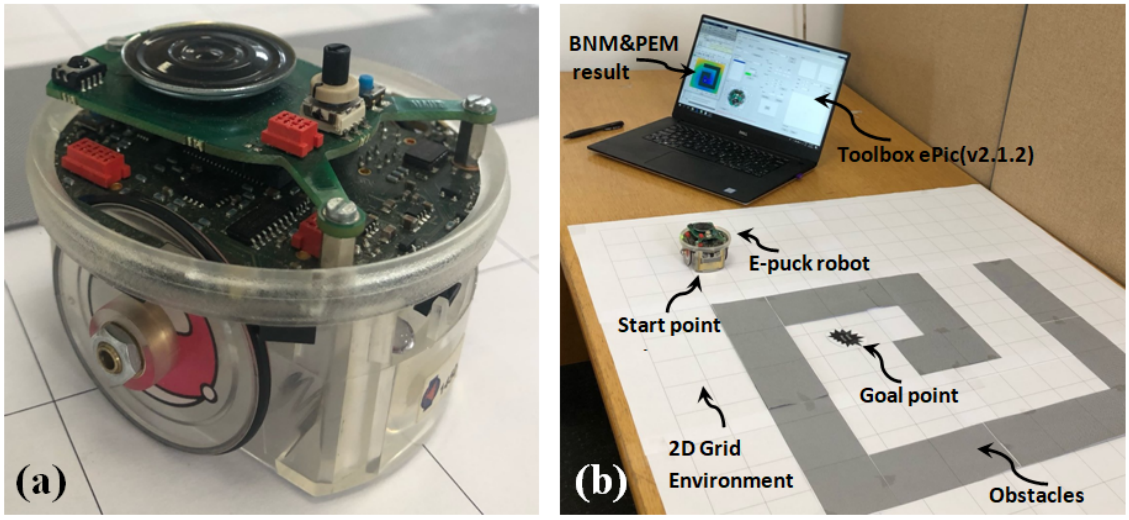


Figure 5.22: The *e-puck* mobile robot was used for the experimental test (a). The experimental set-up (b) for testing the performance of $BNM\&PEM$ to solve the path planning problem in a static environment.

First, the developed method is used to generate the shortest collision-free path to direct the robot to move from the starting point (C_s) to the goal point (C_g) while avoiding the

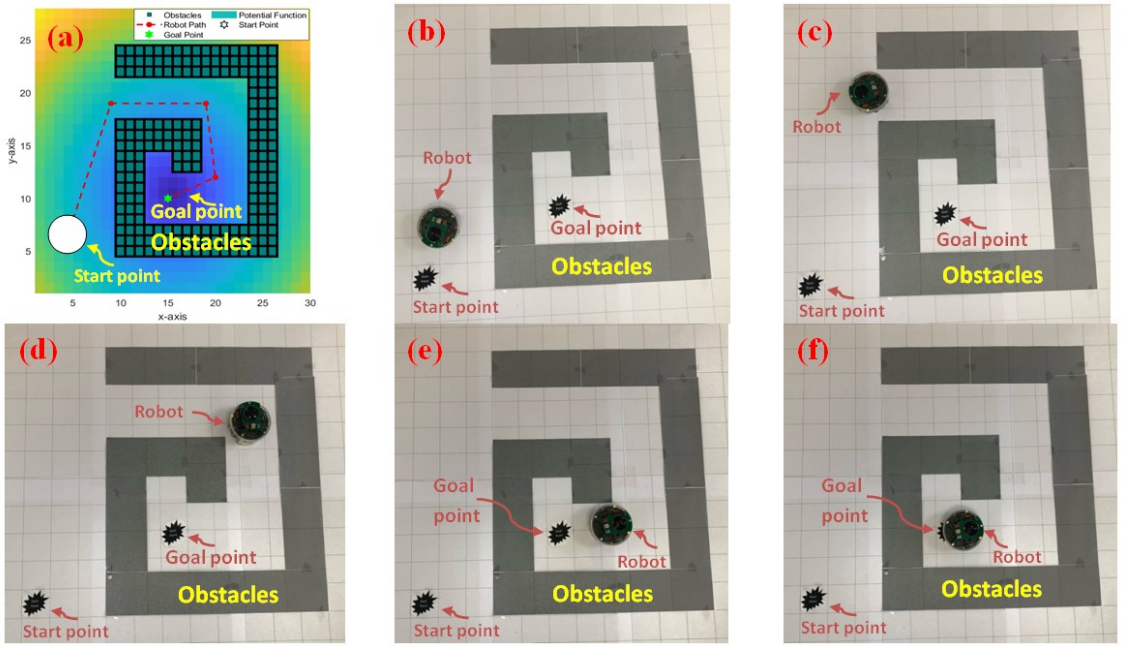


Figure 5.23: Simulation and experimental results: (a) simulation result to generate a shortest collision-free path by using $BNM\&PEM$, and $(b) \rightarrow (f)$ $e\text{-}puck$ robot positions at different locations in the robot's working environment.

obstacles, as illustrated in Figure 5.23a. The obtained shortest path from $BNM\&PEM$ consists of several waypoints $w_{(j),(j=1...J,J=5)}$. As shown in Figure 5.23a, the red circle objects represent the waypoints w, and the red dashed line represents the obtained shortest path. The motion data of the $e\text{-}puck$ robot is calculated based on the obtained results from the generated shortest collision-free path. Next, the $e\text{-}puck$ robot is connected to the computer via Bluetooth and the generated motion data are transmitted to the robot via a toolbox $ePic(v2.1.2)$, where $ePic(v2.1.2)$ is used to control $e\text{-}puck$ in $MATLAB$. Let $w_{1(x,y)}$ be the centroid of the first waypoint w_1 of the generated path, and $w_{2(x,y)}$ represents the centroid of the second waypoint w_2. Then, the orientation of the robot is calculated in $MATLAB$ by using $atan2(w_{2y}\text{-}w_{1y}, w_{2x}\text{-}w_{1x})$. Subsequently, to move the $e\text{-}puck$ robot towards the second waypoint w_2, the angle of the w_2 for the robot is calculated. Then, the $e\text{-}puck$ robot starts to move from w_1 to w_2, and this procedure is continuing until the robot reaches the goal point. Figures 5.23$(b \rightarrow f)$ shows the robot's position at different locations in the robot's working environment during the experimental test. The test results demonstrate that the proposed method can generate the shortest path to direct the $e\text{-}puck$ robot toward the goal point.

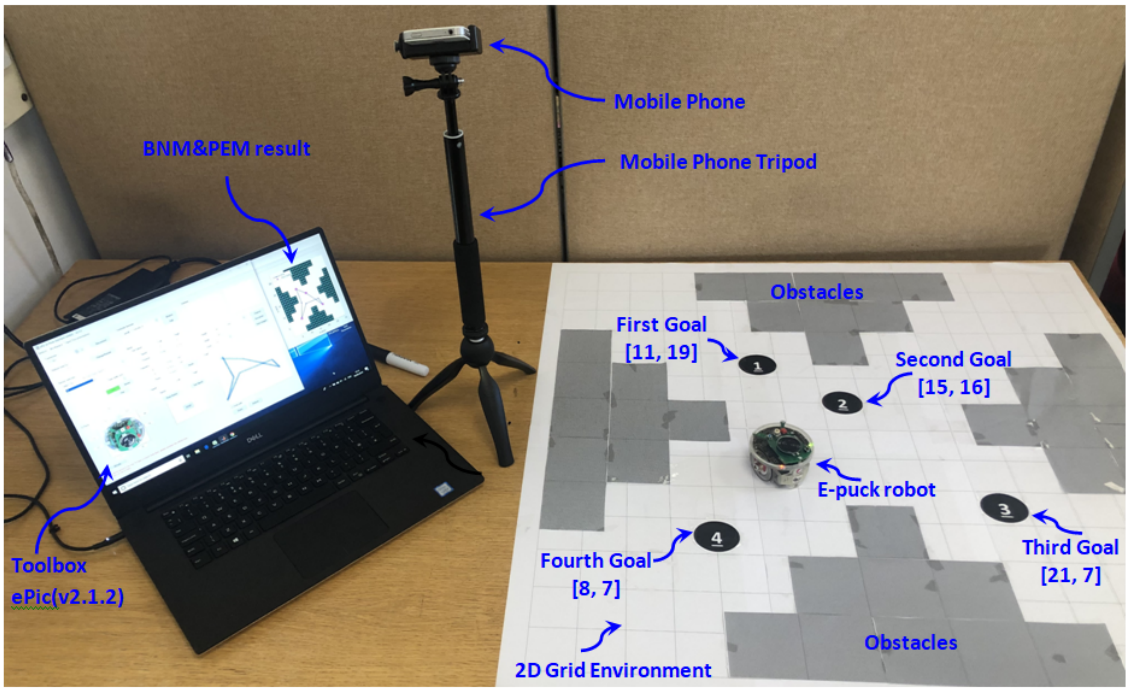


Figure 5.24: The experimental set-up for testing the performance of $BNM\&PEM$ to solve multi-goals path planning problem in a static environment.

5.2.2 Experimental Results for Multi-Goal Path Planning

In this section, the performance of the developed method for solving the multi-goals path-planning problem is examined by carrying out a set of experimental tests with the $e\text{-}puck$ mobile robot. Several experimental scenarios are tested with different positions of the goal points and various obstacles configuration. Figure 5.24 presents an example experimental scenario involving a single mobile robot that visits four randomly-selected goal points. This figure illustrates the experimental set-up and the robot working environment with obstacles for conducting the experimental test.

As a first step for solving this problem, the GA is used to optimize the sequence of the goal points scattered randomly in the $2D$ simulated working environment with the absence of obstacles, as shown in Figure 5.25a. The shortest path is constructed by linking all sequenced goal points, plotted in the red dashed line in Figure a. Then, the BNM method is used to generate IFP to direct the robot from its current goal position toward the second goal position while avoiding obstacles, as shown in Figure 5.25b. Finally, the PEM method is used to find an optimal or near-optimal collision-free path from the IFP by reducing the number of waypoints and the overall path length (see Figure 5.25c). The obtained shortest collision-free path is used to direct the $e\text{-}puck$ robot to move from the first goal point (g_1), passing through all intermediate goal points (g_2, g_3

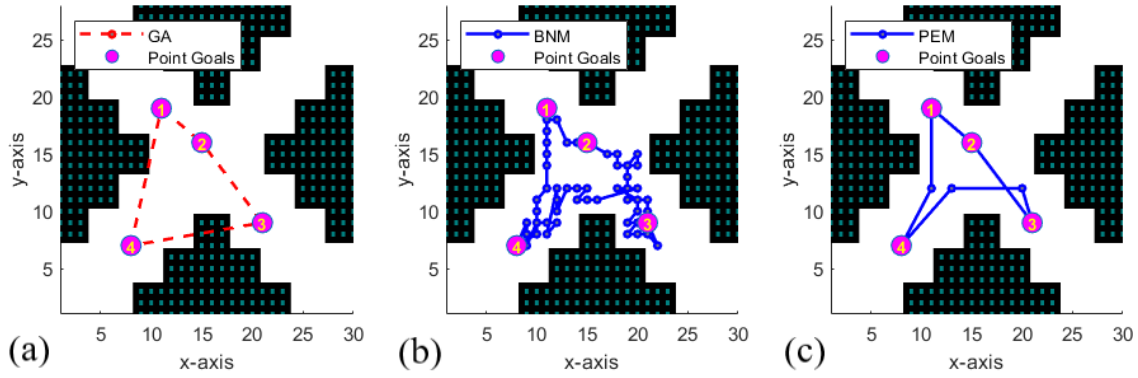


Figure 5.25: Simulation results of the proposed method for solving the MTP: (a) the sequence of the goal points obtained from the implementation of GA, (b) the initial feasible path (IFP) generated from the BNM method, (c) the shortest collision-free path is generated by using PEM.

and g_4), and return to the first goal point (g_1).

From the simulation results, as shown in the Figure 5.25c, the shortest path consists of the sequence of waypoints $w_{(j),(j=1...J),(J=7)}$. The e-$puck$ robot's motion data is calculated based on the generated data from the simulation results, as explained in the previous subsection. The e-$puck$ robot is connected to the computer via Bluetooth, and the generated motion data transmitted to the robot via a toolbox $ePic(v2.1.2)$. Afterwards, the e-$puck$ robot started to move from the first goal point g_1 towards the second goal point g_2 as illustrated in Figures 5.26. The same procedure is repeated for all intermediate goal points (g_3 and g_4) until the robot return to the first goal point (g_1). Figures 5.26($a \rightarrow f$) show the robot's positions at different locations in the robot working environment during the experimental test. The test results show that the developed method provides the goal-to-goal path to direct the e-$puck$ robot to move from g_1, passing through all intermediate goal points $g_i, (i = 2...n), (n = 4)$, and return to g_1. The proposed method generates the optimal path in the high-resolution grid environment. However, the path is not optimal in the low-resolution grid environment. The quality of the constructed path depends on the resolution of the grid map. Besides, there are uncertainties such as friction and slippery surfaces in a real environment, leading to non-optimal results.

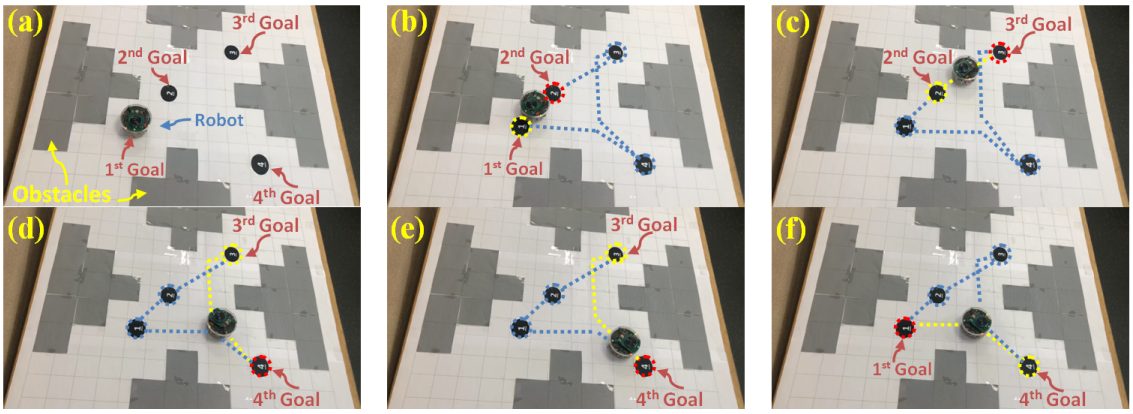


Figure 5.26: Experimental results: (a) shows the initial locations of the $e\text{-}puck$ robot and the goal points in the working environment. The movement of the robot is shown in (b) from g_1 to g_2, (c) from g_2 to g_3, $(d \,\&\, e)$ from g_3 to g_4, and (f) from g_4 to g_1. The shortest path that the robot has to follow to reach the destination point is represented by blue and yellow dashed-line, and the starting and destination goal points are represented by the yellow and red dashed circles, respectively.

Chapter 6

Crowd Simulation Results

6.1 Crowd Simulation

This chapter summarizes the implementation results of the proposed method for simulating the crowd movement in the virtual environment shown in Figure 4.3. The proposed method, described in Section 4.2, is used for generating real-time trajectories for groups of pedestrians navigating in the virtual environment. In this study, we present the pedestrians' trajectories for two different scenarios. The first scenario involves a problem with only four groups of pedestrians navigating in the walking area (see Subsections 6.1.1). In the second scenario, the real-life crowd situation is considered (see Subsections 6.1.2). Additionally, the performance of the proposed method for simulating the crowd movement is compared with the related techniques (see Subsections 6.1.3). Moreover, the effect of the crowd's size on the computational time is investigated for different simulations with various configurations, and statistical analysis for the obtained data was carried out (see Subsections 6.1.4).

In this simulation, pedestrians of different groups' sizes entering the virtual environment through the main entrance (see Figure 4.3). The entrance's width is "15" $units$, assuming that the distance in this study is measured in $unit$ of length. The width w and the length l of the simulated environment are set to "105" and "330" $unit$, respectively. Pedestrians continuously move forward in the navigable area to reach their desired goal points, and then they leave the simulated environment at the same entrance they came in. Inside the simulated environment, there are "8" regions ($goal\,points$) that the groups of pedestrians want to visit ("4" goals on both the right and left-hand side). Each goal point is corresponding to a specific region in the simulated environment. Moreover, there are "2" general service points. These points are considered a temporary goal point. We add them to the goal points when the group needs them (therefore, $nGoals$="10"). The x and $z-coordinates$ of the goal points are set to [37.5, 37.5, 37.5, -37.5, -37.5, -37.5, -37.5, 37.5] and [69, 10.5, -51, 10.5, -51, 69, -109.5, -109.5], respectively, and the services points are located at (0,0,0) and (0,0,-135). Additionally, the values of the parame-

Table 6.1: Pedestrians' distribution with different group size: presents the number of groups in the simulation ($nGroups$ ="4"), number of pedestrians of each type in the same group ($nPedestrians_{Types}$), number of pedestrians in each group ($nPedestrians_{group} = [4, 4, 5, 3]$), and the total number of pedestrians in the crowd ($nPedestrians_{total}$ ="16")

Groups	***Type***$_1$	***Type***$_2$	***Type***$_3$	***Type***$_4$	***Type***$_5$	***Type***$_6$	***Type***$_7$	$nPedestrians_{group}$
$group_1$	1	0	1	0	1	0	1	4
$group_2$	1	0	1	1	0	1	0	4
$group_3$	1	1	1	1	1	0	0	5
$group_4$	0	1	0	1	0	1	0	3
$total$	3	2	3	3	2	2	1	16

ters related to the steering behaviours such as [$cohesion\,radius$, $separation\,radius$, $alignment\,radius$, $cohesion\,weight$, $separation\,weight$, $alignment\,weight$] are set to [6, e, 1.8, 0.75, 0.55, 0.02].

During the simulation, each pedestrian was considered as a dynamic obstacle for other pedestrians. Every single pedestrian has a personal space requirement with a $radius, r$, where other pedestrians should not reside within the space. In this simulation, the radius of the safety space around pedestrians r is set to $0.9\ units$. The minimum distance between pedestrians is set to $2 \times r\ units$, such that pedestrians do not interfere with the other neighbouring pedestrians. We assume that there are four static obstacles ($n3DObstacles$="4") located $[(-20, 0, 80), (20, 0, 120), (-20, 0, 120),$ and $(20, 0, 80)]$ and a circular safety zone is created around each obstacle with a radius of $4\,units$. Moreover, there are other two static obstacles located at $[(-30, 0, -35), (30, 0, -35)]$, where a circular safety zone is created around these obstacles as well with a radius of $7\,units$. While pedestrians move close to the obstacles and other pedestrians, they should keep a certain space for safety. The detail of the simulation results is presented in the following subsections.

6.1.1 Simulation of Simple Scenario

In the simple crowd simulation scenario, we consider only four groups of pedestrians ($nGroups$ ="4"). Each group contains a different number of pedestrians with "7" different types ($nTypes$="7"). The number of each type of pedestrian in each group is generated randomly between $[0, N_c]$, it is assumed that $N_c = 1$. Pedestrian's type and group size distributions illustrated in Table 6.1. As shown in the table, the total number of pedestrians in each group is determined simply by summation of all type of pedestrians in the same group $nPedestrians_{group}$, which are $[4, 4, 5, 3]$. Therefore, the total number of pedestrians in the crowd is the sum of all pedestrians in the scene, which is equal to "16" pedestrians ($nPedestrians_{total}$="16").

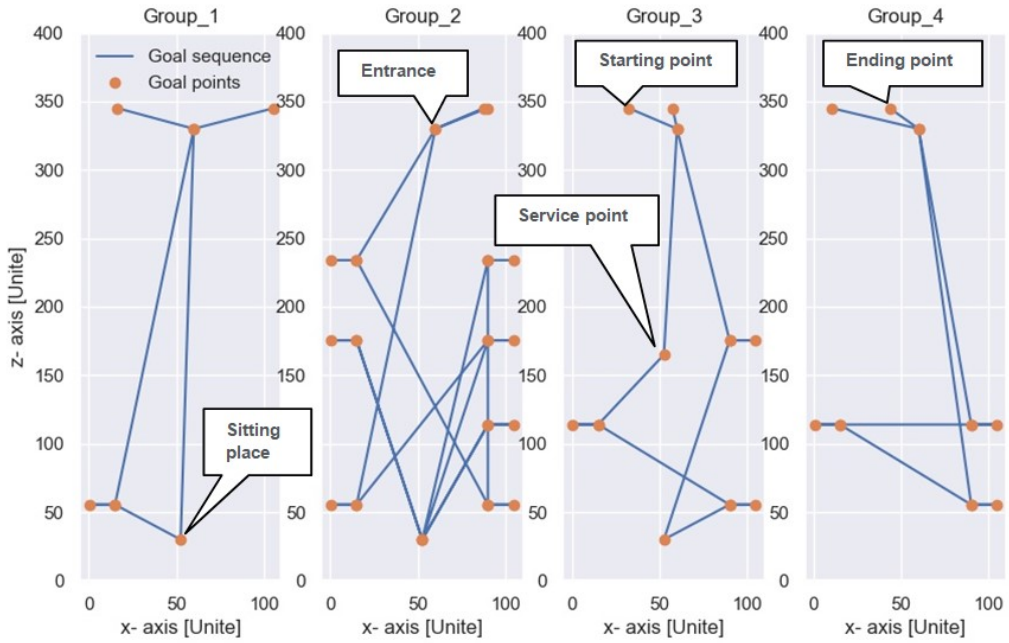


Figure 6.1: Goal points for each group of pedestrians: each $goal\,point$ represents by two-points, the first point is the entrance of the goal, and the second point represents the inside of the goal area.

In the simulated scenario, pedestrians' attributes and their movement are investigated in each frame $keyframe$, where the maximum number of keyframes ($nFrames$) set to "500". In the proposed method, a set of functions is created to get and set the values of pedestrian's attributes in real-time. Based on the proposed method, initially groups of pedestrians g_r, ($g_r = 1, ..., nGroups$, $nGroups$ ="4") with different types ($nTypes$="7") created in the front of the virtual environment around the starting point. The initial position of each pedestrian $Position_{p_e}$ ($p_e = 1, .., nPedestrians_{group}[g_r]$) in each group g_r is generated randomly. Afterward, the initial walking velocity for each pedestrian $Velocity_{p_e}$ ($p_e = 1, .., nPndividuals_{group}[g_r]$) generated randomly between $(0, c_{f3})$. The constant c_{f3} represents the maximum walking speed ($mSpeed$), and it has been limited to "2" $units/timestep$. In the simulated environment, groups of pedestrians move in different directions to reach their desired goal points while avoiding obstacles and other pedestrians in the scene.

In this study, each pedestrian is considered independently, and all pedestrians in the same group have the same independent goal points. A list of goal points has been created for each group, as shown in Figure 6.1, where the goal points and sequences are generated randomly. Both starting and ending points are created randomly outside the virtual environment, and these points are added to the list of the goal points. The value of the $x - coordinate$ for the starting and ending points are generated randomly

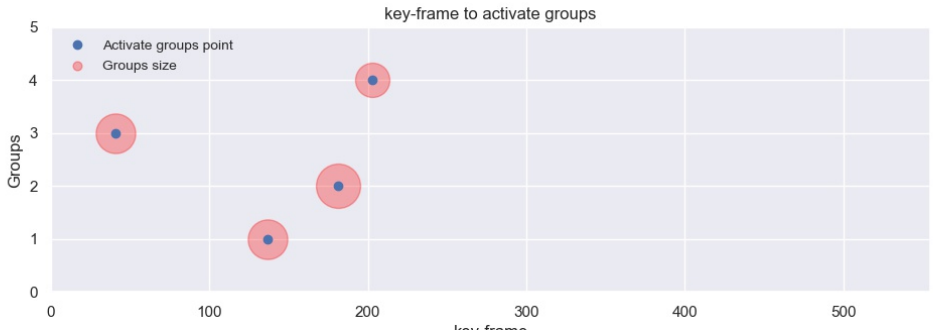


Figure 6.2: keyframes for activating groups to move in the simulated environment.

between [0, w]. In addition, the value of $z - coordinate$ for the starting and ending points are fixed to $l + c_{s1}$, it is assumed that the constant c_{s1} is set to "15". Moreover, the goal point at the centre of the scene (w/2,l/2) represents the first service point. The x and $z - coordinates$ of the entrance are set to $w/2 + 7.5$ and l, respectively. As shown in Figure 6.1, the origin point (0,0,0) of the virtual environment transformed from the centre of the scene to the bottom left corner. The proposed method updates the current group's goal point if the distance between the current pedestrian and the current goal point drops below a certain value C_{g1}. It is assumed that the value of C_{g1} is set to $1.5units$.

In this simulation, each group g_r, ($g_r = 1, ..., nGroups, nGroups$ ="4") starts to move to enter the simulated environment at a randomly generated $keyframe$ ($kFrame_{activate}[g_r]$) between (1, $nFrames$*c_{a1}), where $nFrames$ represent the maximum number of frames and the value of constant c_{a1} is set to "0.5". The activated $keyframe$ for all groups illustrated in Figure 6.2, as shown in the figure, the blue circle objects represent the $keyframe$ for activating the groups to move, where the groups $1, 2, 3 and 4$ are activated at $keyframes$ 41, 137, 181, $and 203$, respectively. The red circles in the graph represent the size of the groups in terms of the number of pedestrians, the bigger circle represents the group with a higher number of pedestrians. Afterwards, at each $keyframe$ ($keyframe = 1, ..., nFrames$, $nFrames$ ="500"), the position and the velocity of pedestrians are computed by using the proposed method. The simulation results of the pedestrians' positions and velocity for all groups are shown in Figures 6.3 and 6.4.

The achieved results shown in Figure 6.3 represent pedestrians' position in each group after the simulation runs for "500" $keyframe$; red circle objects represent the pedestrians' positions. From the obtained results shown in Figure 6.3, it is observed that the proposed method can create the trajectories for each pedestrian to move from the starting point to the final destination point in the virtual environment. This study considers the safety-zone around obstacles to avoid the possibility of overlapping the

Figure 6.3: Pedestrian' trajectory: showing the pedestrians' movement towards their goal points in a simulated environment.

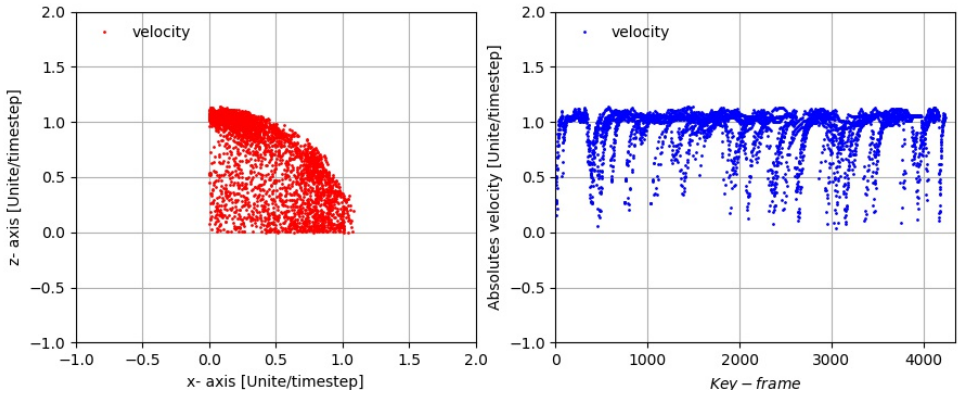


Figure 6.4: Pedestrian' velocity: change the walking velocity of all pedestrians.

trajectories traced by pedestrians with obstacle boundaries. The proposed method provides a collision-free path for pedestrians to reach their goal points (see Figure 6.1). Each new position allocated after the current pedestrians' position and pedestrians start to change their direction as they move closer to obstacles or other pedestrians in the scene.

Figure 6.4 presents the simulation results of the calculated pedestrians' velocity based on the proposed method. The left-hand side of the graph represents the pedestrians' velocity in x and $z - directions$, and the right-hand side represents the absolute velocity of all pedestrians in the scene. We obtained "4235" values of pedestrians' velocity from the simulation results, where the maximum value of the pedestrian's velocity reaches to 1.139 $units/timestep$. The mean and the standard deviation of the calculated velocity are equal to 0.929 $units/timestep$ and 0.205, respectively. It can be observed from the simulation results that the pedestrian's velocity decrease as the pedestrians come closer to the goal points. As shown in Figure 6.4, the minimum velocity is achieved near the goal points.

Each pedestrian has a personal space requirement with a $radius, r$ (see Figure 4.5) to avoid pedestrians' collisions. The radius of the personal space r set to $0.9\,unit$. Pedestrians should not interfere with the space of the other neighbouring pedestrians. The minimum distance between pedestrians set to $2 \times r$ $units$. At each $keyframe$, the minimum distance between the current pedestrian and all other pedestrians is calculated, and the obtained results presented in Figure 6.5. The mean and the standard deviation of the calculated minimum distance are equal to 3.915 $units$ and 2.038, respectively.

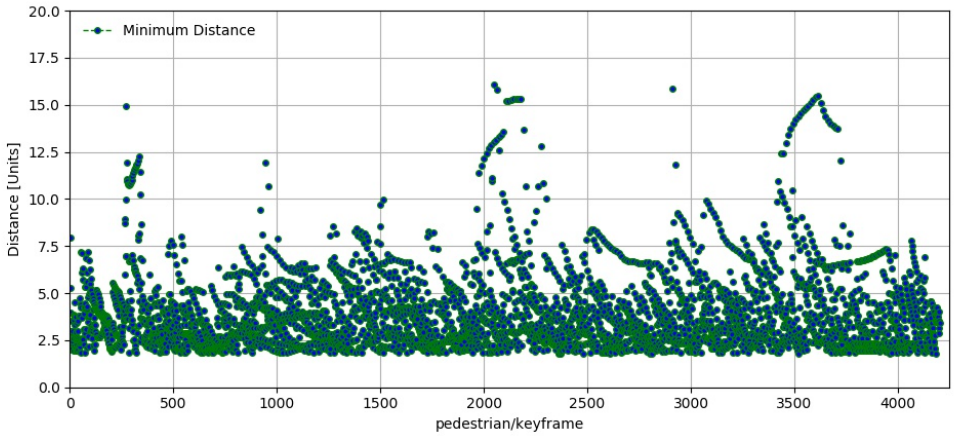


Figure 6.5: The minimum distance between pedestrians.

The obtained results showed that the minimum distances do not fall below $1.8017\ units$. The study of group behaviours can help to better understanding and respecting different cultures' personal space. Even identify the minimum personal space requirements needed to protect health and limit the spread of the disease.

In this simulation, each group consists of several pedestrians' type, where each type of pedestrian has a different energy level $energy_{p_e}$, as illustrated in Table 6.2. Based on the proposed method, the pedestrians' energy level is calculated, and the obtained results are presented in Figure 6.6. As illustrated in the figure, each raw represents the change in the energy level of each pedestrian. For example, the first group consist of 4 pedestrians $Type_1$, $Type_3$, $Type_5$, and $Type_7$ (see Table 6.1). Each pedestrian in this group has the initial level of energy $energy_{p_e}$ (maximum $energy_{p_e}$ level) as starts to move to enter the simulated environment at randomly created key-frame ($keyframe\ 41$, see Figure 6.2). The maximum energy level for the $Type_1$, $Type_3$, $Type_5$, and $Type_7$ are equal to 8 * C_e, 12 * C_e, 16 * C_e, and 14 * C_e, respectively (see Table 6.2), the value of C_e is set to 25 in this simulation. As the pedestrians walk through the environment toward their goal points, the energy level will decrease for all pedestrians, as shown in Figure 6.6. The energy level of $Type_1$ will reach below the minimum $energy_{p_e}$ level first (see Figure 6.6) because $Type_1$ has a lower energy level than other pedestrians in the group. In this situation, all pedestrians in the group will visit the service point in the walking area after finishing the current task (see Figures 6.1 and 6.3). Meanwhile, the pedestrian stays in the service point; their energy will increase again. They are staying in the service point until their energy reaches the maximum level, and then they will start again to move to visit the rest of the goal points.

In order to illustrate the pedestrian's movement in the simulated environment,

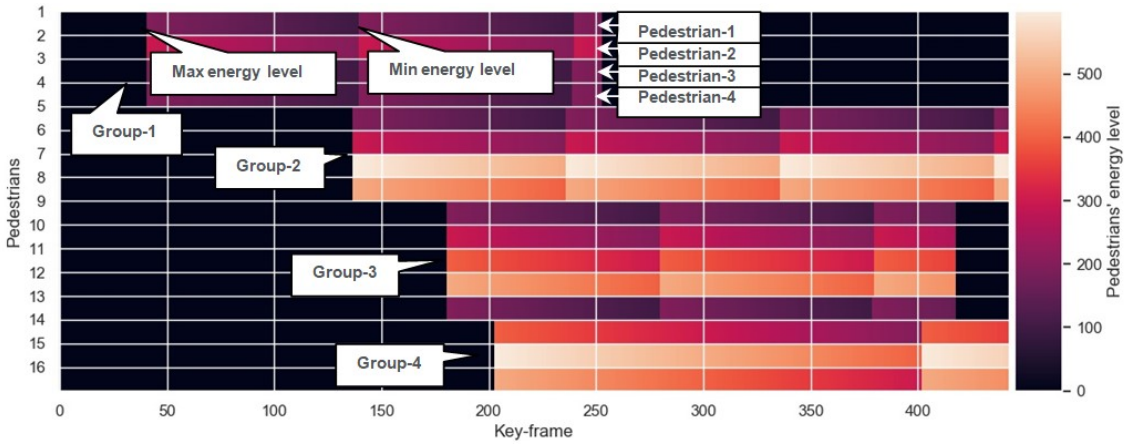


Figure 6.6: Pedestrian' energy level: change of the pedestrians' energy level at each $keyframe$.

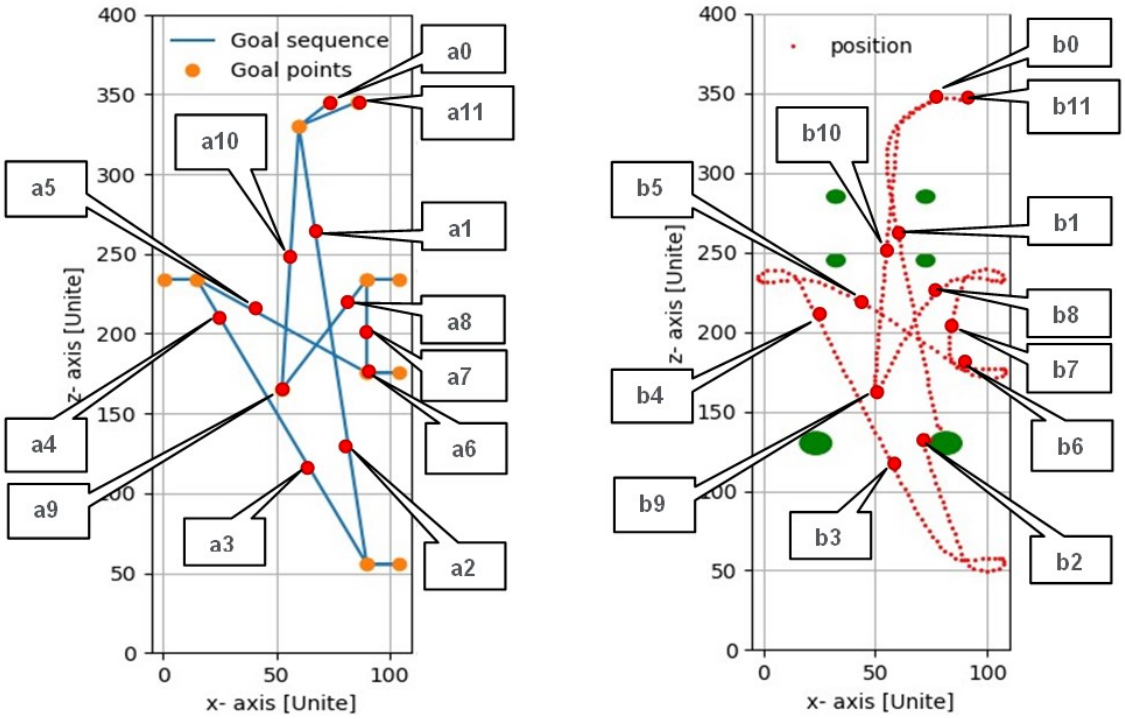


Figure 6.7: Pedestrian's trajectory: the graph shows pedestrians' movement in different directions in the simulated walking area at different $keyframes$: 65, 100, 145, 165, 200, 230, 240, 270, 287, and 310. The red circle objects represent the pedestrians' trace in the virtual environment at each $keyframe$. The left graph shows the generated planned pedestrian's trajectory, and the right graph shows the simulation result of the pedestrian's trajectory in the virtual environment after the simulation runs for 500 $keyframe$.

Table 6.2: Maximum and minimum energy level for each type of pedestrian

Types	***Type***$_1$	***Type***$_2$	***Type***$_3$	***Type***$_4$	***Type***$_5$	***Type***$_6$	***Type***$_7$
$max\,energy$	8 * C_e	10 * C_e	12 * C_e	20 * C_e	16 * C_e	18 * C_e	14 * C_e
$min\,energy$	4 * C_e	5 * C_e	6 * C_e	10 * C_e	8 * C_e	9 * C_e	7 * C_e

different stages of the simulation results are presented in Figure 6.7. A pedestrian (we consider a group that consists of one pedestrian) activates to move to enter the walking area at a randomly created keyframe, as shown in the Figures 6.7 left ($a0$) and 6.7 right ($b0$). Afterwards, the pedestrian enters the walking area through the entrance at $keyframe\ 65$, as illustrated in the Figures 6.7 left ($a1$) and 6.7 right ($b1$), then pass through the walking area to reach the goal points while avoiding obstacles. The simulation results in Figures 6.7 left ($a2$) and 6.7 right ($b2$) present the time that the pedestrian comes closer to an obstacle at $keyframe\ 100$, where the pedestrian turns to the left to pass the obstacle without collision. As the pedestrian reach to the first goal point, the current goal point update to the second goal point in the list of the goal points, and then the pedestrian move toward the new goal point, as shown in Figures 6.7 left ($a3$) and 6.7 right ($b3$) at $keyframe\ 145$. The pedestrian keeps changing the motion directions in the walking area to determine a collision-free path and track the goal points. The simulation results at several locations at $keyframe\ 165, 200, 230, 240,\ and\ 270$ are presented in Figures 6.7 left ($a4 \rightarrow a8$) and 6.7 right ($b4 \rightarrow b8$). In this simulation, the pedestrian visits the service point in the middle of the simulated environment, as shown in Figures 6.7 left ($a9$) and 6.7 right ($b9$) at $keyframe\ 287$. Subsequently, the pedestrian moves toward the entrance/exit to leave the working environment, as illustrated in Figures 6.7 left ($a10$) and 6.7 right ($b10$) at $keyframe\ 310$. At the last stage of the simulation, the pedestrian continues moving until it reaches the final destination point located at the $End-point$ as shown in Figures 6.7 left ($a11$) and 6.7 right ($b11$).

The next simulated scenario demonstrates the construction and configuration of the proposed method for simulating pedestrian crowd movement in a virtual environment of public spaces such as a shopping mall. The simulation results with a higher number of small groups are discussed and presented in the following subsection.

6.1.2 Simulation of Real-Life Crowd Movements

This section investigates the implementation of the proposed method for simulating pedestrian crowd movement in a large and complex virtual environment of public spaces such as a shopping mall for a limited time. To demonstrate a realistic pedestrian movement through a virtual environment, we consider different groups with various types of pedestrians ($family$, $friends$, etc.) in the crowd. Whereas each type of pedestrian has its own attributes such as $gender$, age, $position$, $velocity$, $energy$, etc. In this scenario, many groups of pedestrians are appropriately introducing into the shopping mall. Each

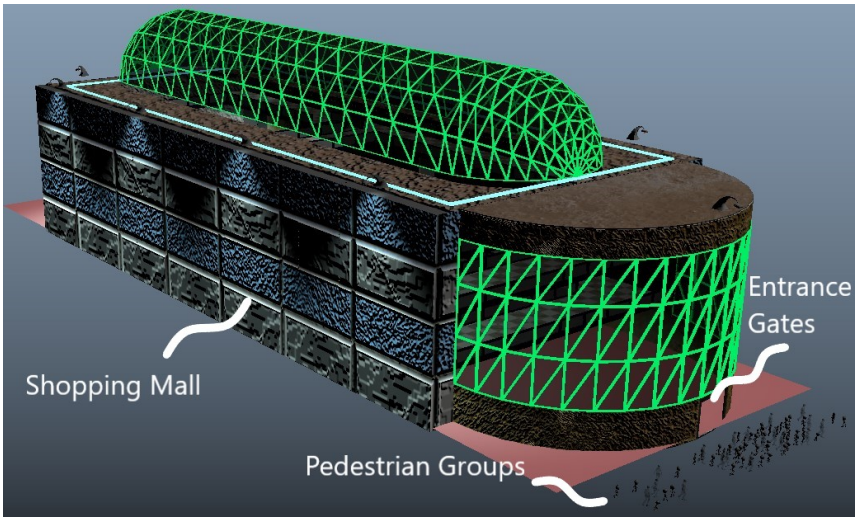


Figure 6.8: A large-scale $3D$ model of the commercial mall populated by virtual groups of pedestrians [6].

group's intention is different for visiting the number of shops in the shopping area with different visiting sequences. At each step in the simulation, virtual pedestrians adjust their attributes and optimize their paths independently in real-time. Moreover, they avoid stationary obstacles and other pedestrians in the virtual environment when they move closer.

In this study, a multi-level shopping mall environment is considered, as shown in Figure 6.8. For simplicity, we assume that the pedestrians are moving within a pedestrianized area on the ground floor of the shopping mall (see Figure 6.9). The other floor of the mall is not accessible by pedestrians. Pedestrians' activities are categorized into walking through the large environment model and shopping activities. The shopping

Figure 6.9: Screen-shot of the shopping mall with obstacles: shows the walking area on the ground floor of the shopping mall with the $3D$ obstacles.

Figure 6.10: Shows different $3D$ characters animation obtained from $Mixamo$ [7]

mall consists of several shops, as shown in Figure 6.9, and the shops are located on both sides of the shopping area to emulate the real-world scenario.

In this scenario, pedestrians formulated in groups before entering the shopping mall. Each group consists of multiple pedestrian types ($male$ and $female$) of different ages (old, $young$, and $child$) to establish the range of personality variation. Several examples of the virtual pedestrian [7] that have been used in this simulation are shown in Figure 6.10. Each pedestrian has a different energy level, i.e., an old pedestrian has a lower energy level than a $young$ pedestrian. The energy level will decrease as they walk through the environment. Pedestrians rest at the sitting place in a service point when their energy level ($energy_{p_e}$) reaches below the minimum energy level. Furthermore, during the visit, sometimes pedestrian considers undertaking other activities, and it can happen between two sequential visits. Examples of activities in the proposed model include having food and using other mall services. In the simulated scenario, pedestrians in the same group have the same goal points ($shops$) to visit, and each group is assumed to have a different list of goal points. Each goal point ($nGoals$) corresponding to a specific region ($shop$) in the shopping mall environment. The detail of the simulation method is described in Section 6.1.

Apart from pedestrians and shops, the simulated environment consists of walls, obstacles, and other regions that not accessible by pedestrians in the crowd. For example, there are several planting pots in the walking area of the shopping mall at different locations (see Figure 6.9). In this study, the planting pots are used as obstacles to prevent pedestrians from walking through, and pedestrians need to pay attention and keeps a certain distance. Nevertheless, in reality, the planting pot maybe uses for decorating the mall. Many other things can be used for preventing pedestrians from accessing a particular area and making pedestrians change their motion direction, such as barriers, signs, etc. In this study, we defined the positions of the obstacles at different locations inside the shopping mall. Then a safety zone around each obstacle is created to avoid the possibility

Table 6.3: Pedestrians' group size distributions in the crowd, where pedestrian is denoted by p.

	1p/ group	***2p/ group***	***3p/ group***	***4p/ group***	***5p/ group***	***6p/ group***	***7p/ group***
Number	8	29	65	46	38	9	1
Percentage	0.04	0.145	0.325	0.23	0.19	0.045	0.005

Table 6.4: Pedestrians type contribution in the crowd.

	Type$_1$	***Type***$_2$	***Type***$_3$	***Type***$_4$	***Type***$_5$	***Type***$_6$	***Type***$_7$
Number	108	93	97	92	99	104	103
Percentage	0.155	0.134	0.139	0.132	0.142	0.149	0.148

of overlapping the paths traced by pedestrians with obstacle boundaries. An abounding circle represents a safety zone that the pedestrians can not enter during their motion. The safety zone radius is constant, R, where the value of R depends on the obstacles' size.

In this simulation, we present the pedestrians' movement in the crowd with "200" different small groups, where each group has distinct goal points at known locations. The total number of pedestrians in the crowd is the sum of all pedestrians in all groups, which is equal to "696" pedestrians ($nPedestrians_{total}$="696"). Pedestrians' group size distributions in the crowd are illustrated in Table 6.3. As shown in the table, the groups' size is different in terms of the number of pedestrians. For example, in the condition of pedestrian walking alone, "8" groups (4% of the whole number groups ($nGroups$), where $nGroups$ ="200") are seen in the crowd. The number of each pedestrian's type ($nTypes$="7") in each group is generated randomly between $[0, N_c]$, and we assumed that $N_c = 1$, and the obtained results are presented in Table 6.4. From the table, it is observed that the total number of each type of pedestrian in the crowd is different. For example, the total number of the first type of pedestrian ($Type_1$) is "108" pedestrians (15.5% of the total number of pedestrians in the crowd ($nPedestrians_{total}$), where $nPedestrians_{total}$=696).

In this simulation, seven different types of characters (see Figure 6.10) are used to generate "200" groups with "696" pedestrians. Initially, all groups of pedestrians scattered in the front of the virtual environment around the starting point, as shown in Figure 6.11a. In this scenario, the $x - coordinate$ for the starting and ending points for pedestrians is generated randomly between $[-50, w + 50]$. Moreover, the value of $z - coordinate$ is set to $l + c_{s1}$, where c_{s1} is determined randomly between $[10, 60]$. Afterwards, the groups of pedestrians were appropriately introduced into the shopping mall environment based on the schedule models (see Figure 6.12a). Each group starts to move to enter the mall at randomly created keyframe ($kFrame_{activate}[g_r]$) between (1,

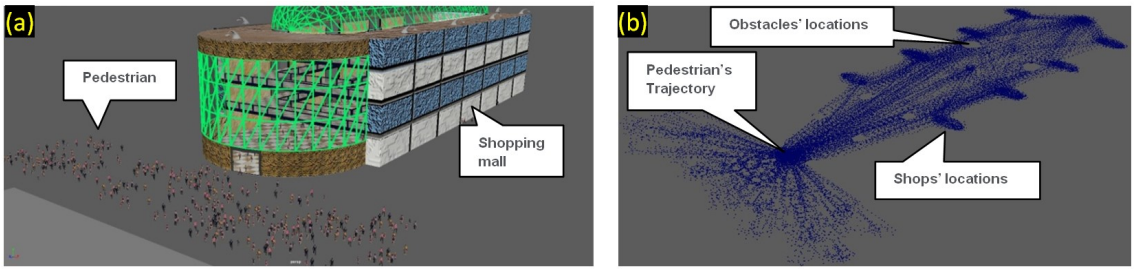


Figure 6.11: Screen-shot of pedestrian flow: a) shows the initial configuration of the simulation, and b) the pedestrians' trajectories in the final stage of the simulation.

$nFrames$*c_{a1}), where c_{a1} is set to 0.5.

Figure 6.13 illustrates the activation's' $keyframes$ of all groups; the blue circle objects represent the $keyframes$ for activating groups to move. The red circle represents the groups' size in terms of the number of pedestrians; a larger circle represents a group with a higher number of pedestrians.

At each $keyframe$, pedestrians' attributes and their positions are calculated and updated independently in real-time. The maximum number of frames $nFrames$ is set to "200" $keyframes$. The simulation results of the pedestrians' trajectories for all groups at each $keyframe$ are shown in Figure 6.11b. The figure shows that the proposed method allows pedestrians to navigate in the walking area from the starting point to the final destination point. The results have shown that the proposed method prevents pedestrians from colliding with the obstacles and pedestrians in the scene by using the obstacle and pedestrian avoidance method. As the pedestrians move closer to the obstacle and other pedestrians, they need to change their motion direction to avoid collision problems. The proposed method uses BNM for collision avoidance because this method has a high computational performance. Moreover, pedestrians never get stuck in the local minima behind the obstacles, and the BNM method guarantees the collision-free path.

Different screen-shot of the simulation results from different viewpoints presented in Figures 6.12 and 6.14. As shown in Figures 6.12b&c, pedestrians attempt to walk in different directions to reach their desired goal points while avoiding obstacles and other pedestrians in the scene. All pedestrians remain with the group during the simulation. Diffident types of walking states are demonstrated in the zoom-in Figure 6.14b of the particular part of the scene. For example, a pedestrian (A) who has walked alone and changing his motion direction to avoid stairs on his right. Moreover, a pedestrian (B) has a limited space to move after leaving the $shop - 6$. At the same time pedestrian (C) has a free moving space to move. At the end of the simulation, all groups terminated in front of the shopping mall, as illustrated in Figure 6.14a. The figure showed that the most

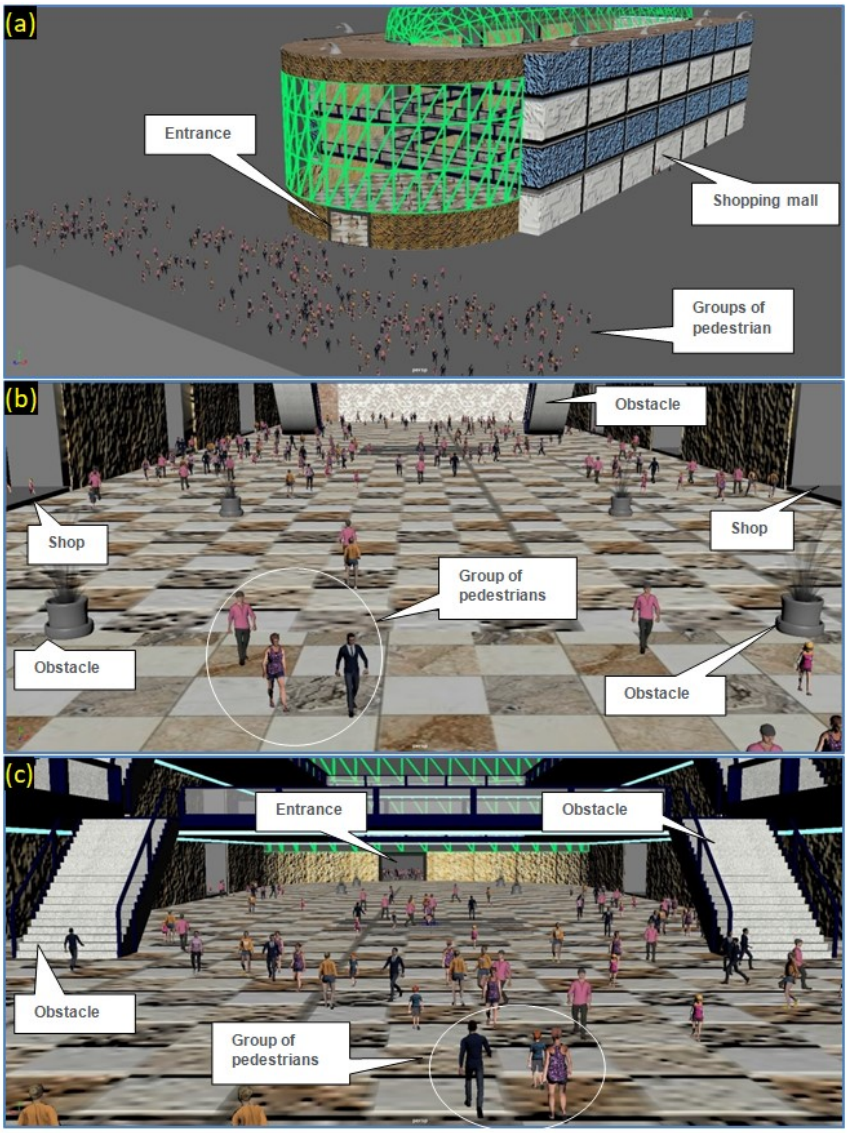


Figure 6.12: Screen-shot of crowd simulation: groups of pedestrians move in different directions to reach their desired goal points while avoiding the static obstacles and pedestrians in the scene.

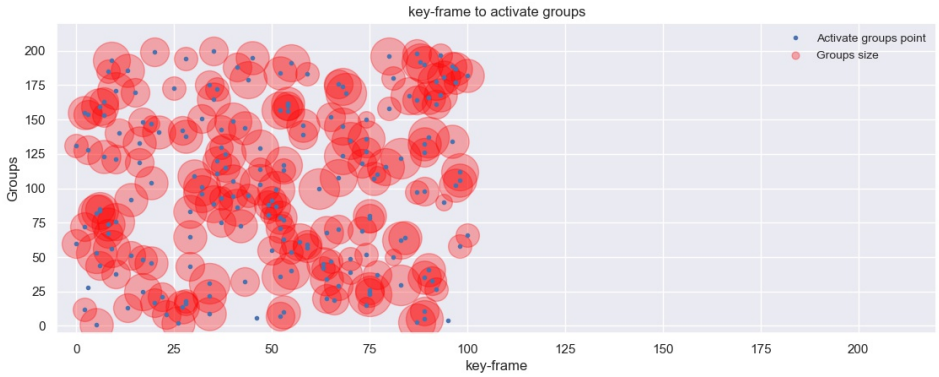


Figure 6.13: Keyframes for activating groups of pedestrians to move in the virtual environment.

Figure 6.14: Screen-shot of the simulation: showing the pedestrian's movement towards the goal points in a scene.

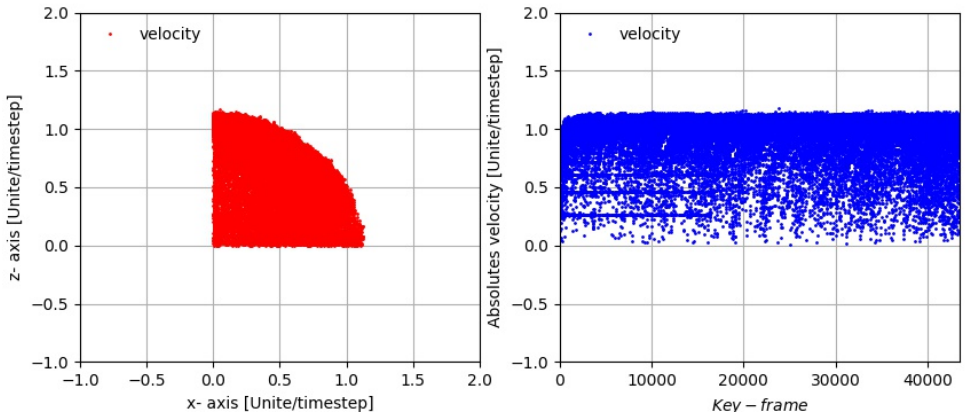


Figure 6.15: Pedestrian' velocity: change the walking velocity of all pedestrians.

Figure 6.16: Screen-shot of the simulation: shows a single pedestrian trying to move through the walking area toward the goal point, and he starts to change his motion direction near the safety area around the stairs at $keyframes$ a)118, b)122, and c) 124.

crowded area in the virtual environment is found in front of the mall, where the distance between pedestrians is very small.

The calculated results of the pedestrians' velocity of all groups of pedestrians using the proposed method are shown in Figure 6.15. the left-hand side of the graph represents the pedestrians' velocity in x and $z - directions$, and the right-hand side represents the absolute velocity of all pedestrians in the scene. After the simulation runs for "200" $keyframes$, the computed mean and standard deviation is reached to "0.905" and "0.216", respectively. By comparing the obtained results with the pedestrians' velocity in Subsection 6.1.1, it can be concluded that the increasing number of groups has the reverse effect on the mean velocity of the pedestrians in the crowd. This is because increasing the groups of pedestrians normally slow down the pedestrians' movement.

In this simulation, pedestrians change their position and velocity based on various steering behaviours. In each $keyframe$, the new position of each pedestrian is calculated based on the proposed method. If the pedestrian does not interfere with the obstacles and other pedestrians, then the pedestrian's current position will update to the new determined position (see Figure 4.2). Simultaneously, if the pedestrian interferes with the obstacles or pedestrians, the proposed method will find another pedestrian position based on the obstacle and pedestrian avoidance methods. Then the current pedestrian's position will update with the newly calculated position. Different illustrative examples for obstacle and pedestrian avoidance methods are presented in Figures 6.16, 6.17, and 6.18.

Figure 6.16 demonstrates the situation when the pedestrian tries to pass an obstacle (stair). In this step of the simulation, the pedestrian checks the path for collision with obstacles. If the pedestrian moves in the current direction, they will collide with the obstacle, as shown in the figure. Therefore the pedestrian needs to change his motion direction. The proposed method uses the BNM method to find the collision-free path and guide the pedestrian to turns his direction to the left and then to the right to pass the

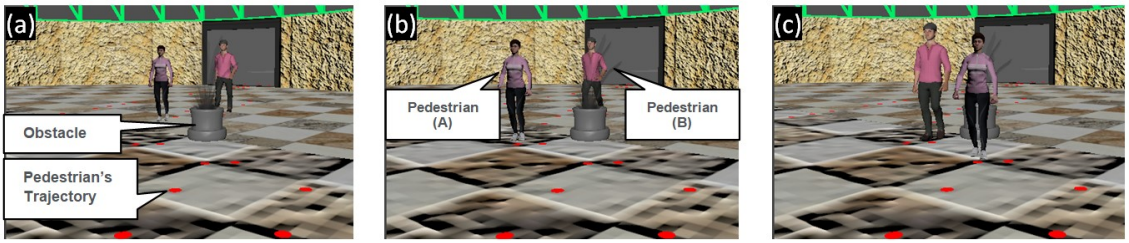


Figure 6.17: Screen-shot of the simulation: shows a group of pedestrians, consists of two pedestrians (pedestrian A and pedestrian B), that using the obstacle avoidance method to pass the obstacle. Afterwards, pedestrians keep moving toward their desired goal points: a) collision avoidance between pedestrians and obstacle at $keyfram = 94$, b) collision avoidance between pedestrian A and obstacle at $keyfram = 95$, and c) collision avoidance between pedestrian B and obstacle at $keyfram = 96$.

obstacle. Afterwards, the pedestrian keeps moving to reach his destination point. When a pedestrian updating its position with the newly calculated position, there is a possibility to collide with the other pedestrians in the scene. An illustrative example of obstacle avoidance and pedestrian avoidance is introduced to address this problem, as shown in Figures 6.17 and 6.18. In this example, we consider a group of pedestrians that consist of two pedestrians moving forward to their goal point.

Firstly, at the $keyframe\,94$, as shown in Figure 6.17a, on condition that the pedestrians move in the current direction, after coming closer to the obstacle (planting pot), they will collide with the obstacle because the obstacle blocks the path. Therefore, the proposed method uses the BNM method to find a new position for pedestrians. Afterwards, pedestrians change their positions to the newly calculated positions, as demonstrated in Figures 6.17b&c at the $keyframe\,95$, and 96.

Secondly, after the pedestrians pass the obstacle, a collision may happen between pedestrians at the $keyframe\,97$. If the distance between pedestrians reaches below $2 \times r$, then pedestrian (B) uses the pedestrian avoidance method to resolve this condition (see Figure 6.18a). Based on this method, a pedestrian (B) has to change his motion direction to the right to avoid collision with a neighbouring pedestrian (A). Then pedestrians (A) and (B) keep moving at $keyframe\,98$ to reach their destination point, as demonstrated in Figure 6.18b. The red circle objects represent the pedestrians' movement, which can be seen clearly near the obstacle in the walking area. The results show that the groups of pedestrians can avoid collisions and reach their goal points by using the proposed method.

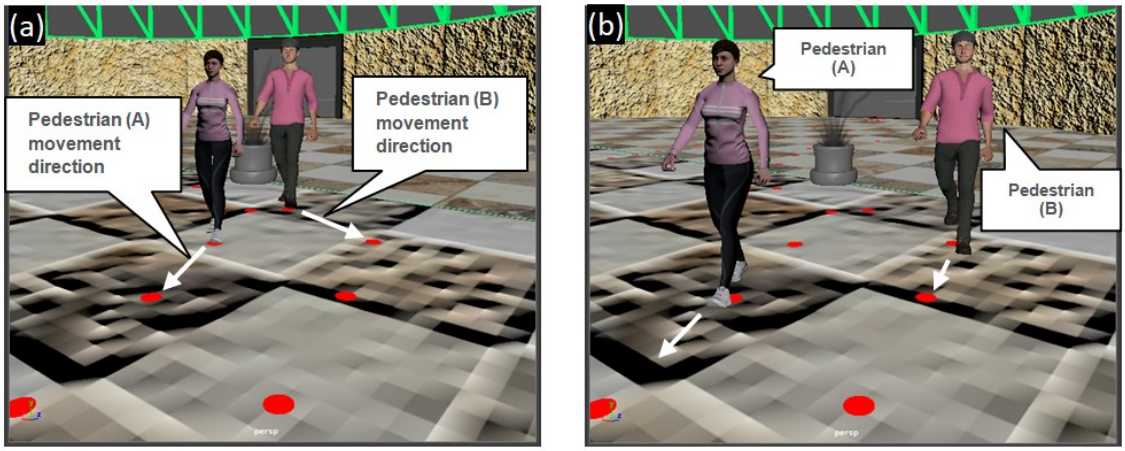


Figure 6.18: Screen-shot of the simulation: shows a group of pedestrians, consists of two pedestrians (pedestrian A and B), passing each other using the pedestrian avoidance method. a) collision avoidance takes place between pedestrians A and B at $keyframe$ 97. b) pedestrian B changes the motion direction to avoid pedestrian A without any collision at $keyframe$ 98, and then they keep moving to reach their goal point.

6.1.3 Comparison with Different Methods

This section presents the comparison between the proposed method and the related methods based on the evaluation criteria described in [121]. However, setting uniform evaluation criteria for different categories of models is quite difficult. The authors in [121] collected and analyzed the information provided by the papers describing different models to simulate crowds in the years 2000-2020, and they presented their results in Table 4 [121]. This table shows the group behaviour of the models that have been used previously in the field of crowd simulation. In order to compare the proposed method with these models, the group behaviours of the proposed method for simulating the crowd movement are defined and presented in Table 6.5. In this study, each pedestrian in each group is influenced by other pedestrians in the same group. The pedestrian-to-pedestrian influencing relationships inside a group are referred to as intra-group structure. However, a group of pedestrians in the crowd is not influenced by other groups, and the relationships s group-to-group are referred to as inter-group relationship [129]. As shown in Table 6.5, the proposed method focus on intra-group interactions such as group formation, group structure, group cohesion, group cooperation, peer behaviour, Intra-group Emotion Contagion. Moreover, this method pays less attention to the inter-group relationship.

Table 5 [121] presents the obtained simulation results compared with the other models that have been used previously in the years 2000-2020. By comparing the $agent-based$ models with the obtained results based on the proposed method, one can conclude that the

Table 6.5: Ability of simulating group dynamics with the proposed method for the crowd simulation (GA: Group Avoidance, GI: Group–pedestrian Interaction, InterE: Inter-group Emotion Contagion. LF: Leader–Follower, GF: Group Formation, GS: Group Structure, GC: Group Cohesion, GCo: Group Cooperation, PB: Peer Behavior, IntraE: Intra-group Emotion Contagion).

Method	**Inter-Group**			**Intra-Group**						
	GA	**GI**	**InterE**	**LF**	**GF**	**GS**	**GC**	**GCo**	**PB**	**IntraE**
Proposed method	×	×	×	×	✓	✓	✓	✓	✓	✓

Table 6.6: The mean and the standard deviation Std of the pedestrians' number (NOP), and the computational time (CT) (in $seconds$, $[S]$) required to find the trajectories for the pedestrians in each group.

nGroups	**No. of Pedestrians (NOP)**		**Computational Time (CT)**	
	Mean$_{NOP}$	**Std**$_{NOP}$	**Mean**$_{CT}$	**Std**$_{CT}$
1	6.101	2.405	3.540	1.348
2	12.681	3.384	6.910	1.860
3	18.703	4.332	10.475	2.349
4	24.920	4.996	13.860	2.607
5	31.231	5.375	17.400	2.900
6	36.775	5.581	21.320	3.170
7	42.475	6.532	24.335	3.272
8	49.077	7.175	27.830	3.906
9	56.590	7.667	31.810	4.006
10	61.838	7.701	35.375	4.105

proposed method can simulate different scales of crowds (small scale, pedestrians < 200, medium scale, 200 < pedestrians < 500, large scale, 500 < pedestrians < 1000). In contrast, there are other $agent-based$ models that cannot simulate large-scale crowds [121].

6.1.4 Statistical Analysis

To evaluate the performance of the proposed method, we implemented the proposed method for simulating the crowd movement in the virtual environment with different groups (from $1 \rightarrow 10\, groups$) and different scales. For each simulation instance, we perform "200" independently runs for "200" $frames$. For each group, the proposed method is used to generate pedestrians' trajectories that navigating in the virtual environment shown in Figure 6.8. The mean and the standard deviation (Std) of pedestrians' number of each group is calculated. The results are presented in Table 6.6, and the graphical representation of pedestrians' number of each group is illustrated in Figure 6.19a. For each simulation scenario presented in

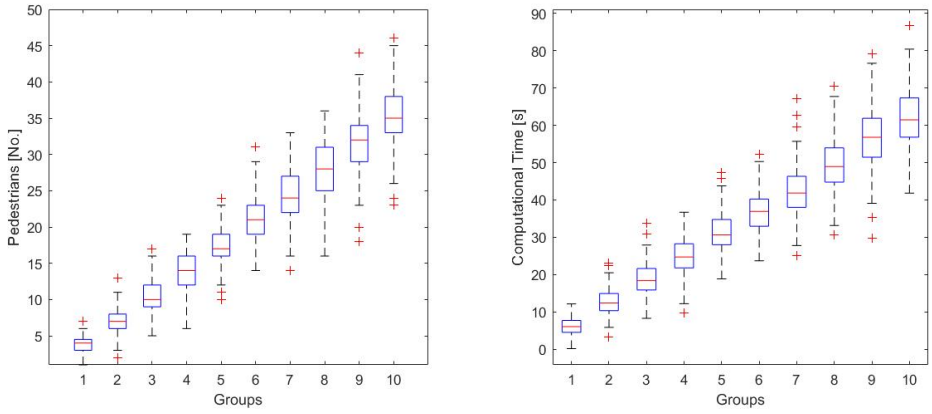


Figure 6.19: Simulation and performance evaluation: (a) the number of pedestrians, and (b) the computational time required to generate the pedestrians' trajectories in each group using the proposed method.

Table 6.6, the generated pedestrians in different groups are located randomly at the front of the simulated environment. Then, each group g_r, ($g_r = 1, ..., nGroups$) starts to move to enter the simulated environment at a specific $keyframe$ number ($kFrame_{activate}[g_r]$=1). After that, pedestrians continuously move forward to reach their desired goal points, where the number of goal points and their sequence are defined as follow: $[[37.5, 0.0, 10.5], [-37.5, 0.0, -51], [37.5, 0, -109.5], [-37.5, 0.0, 10.5]]$ (see $Goals_{group}$[1] in Figure 4.1). In this scenario, pedestrians' attributes and their movement calculated in each $keyframe$, where the maximum number of $keyframes$ ($nFrames$) is set to 200 $keyframes$. All other parameters are the same as the previous simulations.

At each independent run, the proposed method calculated the total computational time required to generate the pedestrians' trajectories and avoid obstacles and pedestrians in the scene. Table 6.6 presents the calculated mean and the standard deviation (Std) of the total computational time, and the graphical representation of the simulation results illustrated in Figure 6.19b. The obtained results reveal that the proposed method can generate the trajectories for the groups of pedestrians navigating in the virtual environment to visit several goal points within a reasonable computational time. Moreover, the obtained results reveal that the mean value of the computational time is not increased significantly with increasing the number of pedestrians in the crowd (see Table 6.6). The computational time depends on the number of pedestrians, the number of goal points and their ordering, the complexity of geometric problems, pedestrian avoidance, etc.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In the first part of this thesis, a new developed method, called the Boundary Node Method (BNM), is presented for solving the path planning problem. The developed method is used to find a collision-free path for a mobile robot through a sequence of waypoints that the robot has to traverse from the starting point to the goal point without colliding with any obstacles. The BNM method can generate a path safely and efficiently. However, the path is not optimal in terms of the total path length. An additional new developed method, called Path Enhancement Method (PEM), is used on top to generate an optimal or close-to-optimal collision-free path. The developed method uses an optimization technique to generate a collision-free path in a relatively short computational time. The computational time required to solve the path planning problem does not significantly increase the environment's complexity. Moreover, this method does not work through random operations. There is no uncertainty in generating points, which leads to finding the final solution for the problem without variation in the solution. The developed method has been successfully used in several working environments with different degrees of complexity. The obtained results show that the developed method can provide the shortest collision-free path for a mobile robot within a relatively short computational time. In order to validate the performance of the developed method, the simulation results compared with the results of the existing path planning methods. The comparison results reveal that the proposed method achieved better performance for solving the path planning problem in terms of the computational times and the path length.

From the obtained results, it was observed that the proposed method generates a path that consists of straight lines between waypoints with sharp turns. In real applications, when the robot follows a path in the workspace, it may not be able to make a sharp turn, and it is not the safest path for the robot. In order to improve the path concerning the robot dynamics, the cubic spline method is used to construct a continuous smooth path that connects the starting point to the goal point.

The application of the developed method is extended further for solving the multi-goal path planning problem (MTP). This method is used to find the shortest collision-free path connecting a given set of goal points scattered randomly in a $2D$ workspace without colliding with any obstacles. In this study, the shortest collision-free path is determined in a two-step. First, we applied the Genetic Algorithm (GA) to find an optimal sequence of the goal points. Second, we used the BNM method to construct a collision-free path between every pair of sequenced goal points. Subsequently, the PEM method is used to obtain an optimal or near-optimal collision-free path from the initial feasible path by minimizing the overall path length. The performance of the developed method has been tested on many different workspace scenarios with varying layouts of obstacles. The simulation results show that the developed method has been efficiently generated an optimal or near-optimal collision-free path that is connecting a given set of goal points. Each robot find its way independently without collision with static obstacles or other robots in the system. The developed method is then extended further to solve the multi-goal path planning problem for multiple mobile robot systems. This method is used to find the shortest collision-free path connecting every pair of sequenced goal points. The simulation results demonstrate the effectiveness of the developed method for constructing the multi-goal path for multi-robot systems. Moreover, the results reveal that all robots reached their final destination goal within a reasonable computational time without collision with either static obstacles or other robots.

Furthermore, to verify the performance of the developed method for solving path-planning problem, several experimental tests have been performed on the $e-puck$ robot with different obstacle configurations and various positions of the goal points. The experimental results showed that the proposed method could construct the shortest collision-free path and direct the real physical robot to the final destination point.

In the second part of this study, a new method is developed for simulating pedestrian crowd movement in a virtual environment, where the first part of this study concerning the generation of the shortest collision-free path is used. In this study, the virtual pedestrians in many groups navigated in the virtual environment with different directions to reach their distinct destination points. The developed method uses the multi-group $microscopic$ model for generating a real-time trajectory for each pedestrian in the crowd. Additionally, an $agent-based$ model is introduced into the developed method for modelling the pedestrians' behaviours. Each pedestrian in the crowd has a particular set of data that represents the characteristics of the pedestrian. Moreover, various steering behaviours are introduced, and several techniques have been presented for combining steering behaviours to a single steering force to allow the pedestrians to walk toward their goal points. The developed method demonstrates how the pedestrians choose their path with their group in a virtual environment toward their goal points while avoiding static obstacles and other pedestrians. Based on this method, each pedestrian in each group constantly adjust their paths toward the desired goal point and updated their attributes independently in real-

time. The obtained results demonstrate that the developed method has been well applied to generate the pedestrian crowd movement in a virtual environment. Furthermore, it is concluded that the developed method had achieved good results in terms of safety and accuracy.

7.2 Future Works

This study opens several new directions for future research:

1. During this study, the robot has been stimulated by a simple nine-node quadrilateral element, better results might be obtained if we attempt to use other element types such as a nine-node circular element (see Figure 7.1a) a seventeen-node quadrilateral element (see Figure 7.1b), and a thirteen-node octagonal element (see Figure 7.1c).

2. Further study is required to address several research issues related to autonomous navigation of mobile robots in unknown environments, where the robot does not have full knowledge about its environment.

3. Another possible direction for future work is to simulate crowd movement in an environment with dynamic obstacles and goals. Moreover, the developed method can be extended for further simulation to capture the attractive and repulsive effects acting on the pedestrians. Sometimes pedestrians are attracted by other persons (e.g., friends) or objects (e.g., window displays). Additionally, the application of the developed method can be extended to describe more complex desired behaviour and other social phenomena.

4. Extend the developed method further to investigate the robot's trajectory navigating through the crowd of people.

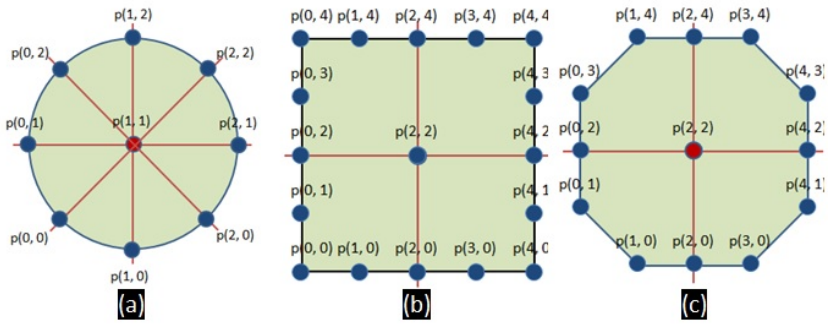


Figure 7.1: Different types of elements to simulate a mobile robot, such as a) nine-node circular element, b) seventeen-node quadrilateral element, and c) thirteen-node octagonal element.

Bibliography

[1] Craig W Reynolds. Steering behaviors for autonomous characters. In *Game developers conference*, volume 1999, pages 763–782. Citeseer, 1999.

[2] Adem Tuncer and Mehmet Yildirim. Dynamic path planning of mobile robots with improved genetic algorithm. *Computers & Electrical Engineering*, 38(6):1564–1572, 2012.

[3] Qing Li, Wei Zhang, Yixin Yin, Zhiliang Wang, and Guangjun Liu. An improved genetic algorithm of optimum path planning for mobile robots. In *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*, volume 2, pages 637–642. IEEE, 2006.

[4] Amir Hossein Karami and Maryam Hasanzadeh. An adaptive genetic algorithm for robot motion planning in 2d complex environments. *Computers & Electrical Engineering*, 43:317–329, 2015.

[5] Changan Liu, XH Yan, CY Liu, and GD Li. Dynamic path planning for mobile robot based on improved genetic algorithm. *Chinese Journal of Electronics*, 19(2):2010–2014, 2010.

[6] kofta55. Commercial mall 3d model. `https://free3d.com/3d-model/commercial-mall-689388.html`, 2019.

[7] Adobe. Mixamo. `https://www.mixamo.com/`, 2018.

[8] N Leena and KK Saju. A survey on path planning techniques for autonomous mobile robots. *IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)*, 8:76–79, 2014.

[9] Jihee Han and Yoonho Seo. Mobile robot path planning with surrounding point set and path improvement. *Applied Soft Computing*, 57:35–47, 2017.

[10] P Victerpaul, D Saravanan, S Janakiraman, and J Pradeep. Path planning of autonomous mobile robots: A survey and comparison. *Journal of Advanced Research in Dynamical and Control Systems*, 9:1535–1565, 2017.

[11] Nan Chao, Yong-kuo Liu, Hong Xia, Abiodun Ayodeji, and Lu Bai. Grid-based rrt$*$ for minimum dose walking path-planning in complex radioactive environments. *Annals of Nuclear Energy*, 115:73–82, 2018.

[12] Alex Shum, Kirsten Morris, and Amir Khajepour. Direction-dependent optimal path planning for autonomous vehicles. *Robotics and Autonomous Systems*, 70:202–214, 2015.

[13] Yong Zhang, Dun-wei Gong, and Jian-hua Zhang. Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing*, 103:172–185, 2013.

[14] Azzeddine Bakdi, Abdelfetah Hentout, Hakim Boutami, Abderraouf Maoudj, Ouarda Hachour, and Brahim Bouzouia. Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control. *Robotics and Autonomous Systems*, 89:95–109, 2017.

[15] Tomás Lozano-Pérez and Michael A Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[16] Michael Brand, Michael Masuda, Nicole Wehner, and Xiao-Hua Yu. Ant colony optimization algorithm for robot path planning. In *Computer Design and Applications (ICCDA), 2010 International Conference on*, volume 3, pages V3–436. IEEE, 2010.

[17] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[18] Jussi Rintanen. Introduction to automated planning. *Lecture notes of the AI planning course, Albert-Ludwigs-University Freiburg*, 2006.

[19] Chaymaa Lamini, Said Benhlima, and Ali Elbekri. Genetic algorithm based approach for autonomous mobile robot path planning. *Procedia Computer Science*, 127:180–189, 2018.

[20] Thi Thoa Mac, Cosmin Copot, Duc Trung Tran, and Robin De Keyser. Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*, 86:13–28, 2016.

[21] Hsu-Chih Huang and Ching-Chih Tsai. Global path planning for autonomous robot navigation using hybrid metaheuristic ga-pso algorithm. In *SICE Annual Conference (SICE), 2011 Proceedings of*, pages 1338–1343. IEEE, 2011.

[22] Oscar Montiel, Ulises Orozco-Rosas, and Roberto Sepúlveda. Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. *Expert Systems with Applications*, 42(12):5177–5191, 2015.

[23] Marco A Contreras-Cruz, Victor Ayala-Ramirez, and Uriel H Hernandez-Belmonte. Mobile robot path planning using artificial bee colony and evolutionary programming. *Applied Soft Computing*, 30:319–328, 2015.

[24] WANG Song, Hong-xing LI, and Yi-nong ZHANG. Path planning of mobile robot based on genetic bee colony algorithm. *DEStech Transactions on Computer Science and Engineering*, 16(5):615–620, 2016.

[25] Hong Liu, Bin Xu, Dianjie Lu, and Guijuan Zhang. A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm. *Applied Soft Computing*, 2018.

[26] Jitin Kumar Goyal and KS Nagla. A new approach of path planning for mobile robots. In *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, pages 863–867. IEEE, 2014.

[27] Yanrong Hu and Simon X Yang. A knowledge based genetic algorithm for path planning of a mobile robot. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 4350–4355. IEEE, 2004.

[28] Bing Fu, Lin Chen, Yuntao Zhou, Dong Zheng, Zhiqi Wei, Jun Dai, and Haihong Pan. An improved a* algorithm for the industrial robot path planning with high success rate and short length. *Robotics and Autonomous Systems*, 2018.

[29] Nadia Adnan Shiltagh and Lana Dalawr Jalal. Optimal path planning for intelligent mobile robot navigation using modified particle swarm optimization. *International Journal of Engineering and Advanced Technology*, 2(4):260–267, 2013.

[30] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96:59–69, 2014.

[31] Akshay Kumar Guruji, Himansh Agarwal, and DK Parsediya. Time-efficient a* algorithm for robot path planning. *Procedia Technology*, 23:144–149, 2016.

[32] Shuzhi Sam Ge and Yan Juan Cui. New potential functions for mobile robot path planning. *IEEE Transactions on robotics and automation*, 16(5):615–620, 2000.

[33] Gene Eu Jan, Ki Yin Chang, and Ian Parberry. Optimal path planning for mobile robot navigation. *IEEE/ASME Transactions on mechatronics*, 13(4):451–460, 2008.

[34] Mitul Saha, Tim Roughgarden, Jean-Claude Latombe, and Gildardo Sánchez-Ante. Planning tours of robotic arms among partitioned goals. *The International Journal of Robotics Research*, 25(3):207–223, 2006.

[35] Christian Wurll, Domink Henrich, and Heinz Wörn. Multi-goal path planning for industrial robots. *International Conference on Robotics and Application (RA'99), Santa Barbara, USA*, 1999.

[36] Emile Glorieux, Pasquale Franciosa, and Dariusz Ceglarek. Coverage path planning with targetted viewpoint sampling for robotic free-from surface inspection. *Robotics and Computer-Integrated Manufacturing*, 61:101843, 2020.

[37] Jan Faigl. An application of self-organizing map for multirobot multigoal path planning with minmax objective. *Computational intelligence and neuroscience*, 2016, 2016.

[38] Chutian Sun. A study of solving traveling salesman problem with genetic algorithm. In *2020 9th International Conference on Industrial Technology and Management (ICITM)*, pages 307–311. IEEE, 2020.

[39] Liu Hongyun, Jiang Xiao, and Ju Hehua. Multi-goal path planning algorithm for mobile robots in grid space. In *Control and Decision Conference (CCDC), 2013 25th Chinese*, pages 2872–2876. IEEE, 2013.

[40] Steven N Spitz and Aristides AG Requicha. Multiple-goals path planning for coordinate measuring machines. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 3, pages 2322–2327. IEEE, 2000.

[41] Kevin Vicencio, Brian Davis, and Iacopo Gentilini. Multi-goal path planning based on the generalized traveling salesman problem with neighborhoods. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2985–2990. IEEE, 2014.

[42] Vojtěch Vonásek and Robert Pěnička. Space-filling forest for multi-goal path planning. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1587–1590. IEEE, 2019.

[43] Ali Noormohammadi-Asl and Hamid D Taghirad. Multi-goal motion planning using traveling salesman problem in belief space. *Information Sciences*, 471:164–184, 2019.

[44] Zhong Yu, Liang Jinhai, Gu Guochang, Zhang Rubo, and Yang Haiyan. An implementation of evolutionary computation for path planning of cooperative mobile robots. In *Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No. 02EX527)*, volume 3, pages 1798–1802. IEEE, 2002.

[45] P Th Zacharia and NA Aspragathos. Optimal robot task scheduling based on genetic algorithms. *Robotics and Computer-Integrated Manufacturing*, 21(1):67–79, 2005.

[46] Martin Bonert, LH Shu, and Beno Benhabib. Motion planning for multi-robot assembly systems. *International Journal of Computer Integrated Manufacturing*, 13(4):301–310, 2000.

[47] Yiqing Huang, Zhikun Li, Yan Jiang, and Lu Cheng. Cooperative path planning for multiple mobile robots via hafsa and an expansion logic strategy. *Applied Sciences*, 9(4):672, 2019.

[48] Shuang Liu, Dong Sun, and Changan Zhu. Coordinated motion planning for multiple mobile robots along designed paths with formation requirement. *IEEE/ASME transactions on mechatronics*, 16(6):1021–1031, 2010.

[49] Keerthi Sagar, Dimiter Zlatanov, Matteo Zoppi, Cristiano Nattero, and Sreekumar Muthuswamy. Multi-goal path planning for robotic agents with discrete-step locomotion. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 58172, page V05AT08A033. American Society of Mechanical Engineers, 2017.

[50] Stefania Pellegrinelli, Nicola Pedrocchi, Lorenzo Molinari Tosatti, Anath Fischer, and Tullio Tolio. Multi-robot spot-welding cells for car-body assembly: Design and motion planning. *Robotics and Computer-Integrated Manufacturing*, 44:97–116, 2017.

[51] Jakub Hvězda, Miroslav Kulich, and Libor Přeučil. Improved discrete rrt for coordinated multi-robot planning. *arXiv preprint arXiv:1901.07363*, 2019.

[52] Rahul Kala. Sampling based mission planning for multiple robots. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 662–669. IEEE, 2016.

[53] Haoyao Chen, Dong Sun, and Jie Yang. Global localization of multirobot formations using ceiling vision slam strategy. *Mechatronics*, 19(5):617–628, 2009.

[54] Ryan Luna and Kostas E Bekris. Efficient and complete centralized multi-robot path planning. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3268–3275. IEEE, 2011.

[55] Saad Ali, Ko Nishino, Dinesh Manocha, and Mubarak Shah. Modeling, simulation and visual analysis of crowds: a multidisciplinary perspective. In *Modeling, simulation and visual analysis of crowds*, pages 1–19. Springer, 2013.

[56] Stephen J Guy. Geometric collision avoidance for heterogeneous crowd simulation. 2012.

[57] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Transactions on Graphics (TOG)*, 25(3):1160–1168, 2006.

[58] Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C Lin. Aggregate dynamics for dense crowd simulation. In *ACM SIGGRAPH Asia 2009 papers*, pages 1–8. 2009.

[59] Roger L Hughes. A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological*, 36(6):507–535, 2002.

[60] Roger L Hughes. The flow of human crowds. *Annual review of fluid mechanics*, 35(1):169–182, 2003.

[61] Hamid Izadinia, Imran Saleemi, Wenhui Li, and Mubarak Shah. 2 t: multiple people multiple parts tracker. In *European Conference on Computer Vision*, pages 100–114. Springer, 2012.

[62] CK Lim, KL Tan, AA Zaidan, and BB Zaidan. A proposed methodology of bringing past life in digital cultural heritage through crowd simulation: a case study in george town, malaysia. *Multimedia Tools and Applications*, 79(5):3387–3423, 2020.

[63] Funda Durupinar, Nuria Pelechano, Jan Allbeck, Ugur Gudukbay, and Norman I Badler. How the ocean personality model affects the perception of crowds. *IEEE Computer Graphics and Applications*, 31(3):22–31, 2009.

[64] Stephen J Guy, Sujeong Kim, Ming C Lin, and Dinesh Manocha. Simulating heterogeneous crowd behaviors using personality trait theory. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 43–52, 2011.

[65] Pierre Degond, Cécile Appert-Rolland, Mehdi Moussaid, Julien Pettré, and Guy Theraulaz. A hierarchy of heuristic-based models of crowd dynamics. *Journal of Statistical Physics*, 152(6):1033–1068, 2013.

[66] Lin Cheng, Vikas Reddy, Clinton Fookes, and Prasad KDV Yarlagadda. Impact of passenger group dynamics on an airport evacuation process using an agent-based model. In *2014 International Conference on Computational Science and Computational Intelligence*, volume 2, pages 161–167. IEEE, 2014.

[67] Avneesh Sud, Erik Andersen, Sean Curtis, Ming Lin, and Dinesh Manocha. Real-time path planning for virtual agents in dynamic environments. In *2007 IEEE Virtual Reality Conference*, pages 91–98. IEEE, 2007.

[68] Chao Cao, Peter Trautman, and Soshi Iba. Dynamic channel: A planning framework for crowd navigation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5551–5557. IEEE, 2019.

[69] Sachin Patil, Jur Van Den Berg, Sean Curtis, Ming C Lin, and Dinesh Manocha. Directing crowd simulations using navigation fields. *IEEE transactions on visualization and computer graphics*, 17(2):244–254, 2010.

[70] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, 1987.

[71] Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. *Graphical models*, 69(5-6):246–274, 2007.

[72] Naveen K Mahato, Axel Klar, and Sudarshan Tiwari. Particle methods for multi-group pedestrian flow. *Applied Mathematical Modelling*, 53:447–461, 2018.

[73] Lorenza Manenti and Sara Manzoni. Crystals of crowd: Modelling pedestrian groups using mas-based approach. In *WOA*, pages 51–57, 2011.

[74] Dirk Helbing, Illes J Farkas, Peter Molnar, and Tamás Vicsek. Simulation of pedestrian crowds in normal and evacuation situations. *Pedestrian and evacuation dynamics*, 21(2):21–58, 2002.

[75] Carsten Burstedde, Kai Klauck, Andreas Schadschneider, and Johannes Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3-4):507–525, 2001.

[76] Siamak Sarmady, Fazilah Haron, and Abdullah Zawawi Hj Talib. Modeling groups of pedestrians in least effort crowd movements using cellular automata. In *2009 Third Asia International Conference on Modelling & Simulation*, pages 520–525. IEEE, 2009.

[77] Nuria Pelechano, Jan M Allbeck, and Norman I Badler. Controlling individual agents in high-density crowd simulation. 2007.

[78] Benedetto Piccoli and Andrea Tosin. Pedestrian flows in bounded domains with obstacles. *Continuum Mechanics and Thermodynamics*, 21(2):85–107, 2009.

[79] Rahul Kala, Anupam Shukla, and Ritu Tiwari. Robotic path planning in static environment using hierarchical multi-neuron heuristic search and probability based fitness. *Neurocomputing*, 74(14-15):2314–2335, 2011.

[80] Mansoor Davoodi, Fatemeh Panahi, Ali Mohades, and Seyed Naser Hashemi. Clear and smooth path planning. *Applied Soft Computing*, 32:568–579, 2015.

[81] Brendan Englot and Franz Hover. Multi-goal feasible path planning using ant colony optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2255–2260. IEEE, 2011.

[82] Mitul Saha, Gildardo Sánchez-Ante, and J-C Latombe. Planning multi-goal tours for robot arms. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 3, pages 3797–3803. IEEE, 2003.

[83] Stefan Edelkamp, Morteza Lahijanian, Daniele Magazzeni, and Erion Plaku. Integrating temporal reasoning and sampling-based motion planning for multigoal problems with dynamics and time windows. *IEEE Robotics and Automation Letters*, 3(4):3473–3480, 2018.

[84] Purushothaman Raja and Sivagurunathan Pugazhenthi. Optimal path planning of mobile robots: A review. *International Journal of Physical Sciences*, 7(9):1314–1320, 2012.

[85] Omar Souissi, Rabie Benatitallah, David Duvivier, AbedlHakim Artiba, Nicolas Belanger, and Pierre Feyzeau. Path planning: A 2013 survey. In *Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on*, pages 1–8. IEEE, 2013.

[86] Zeng Bi, Yang Yimin, and Yuan Wei. Hierarchical planning approach for mobile robot navigation under the dynamic environment. In *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*, pages 372–376. IEEE, 2008.

[87] MA Porta Garcia, Oscar Montiel, Oscar Castillo, Roberto Sepúlveda, and Patricia Melin. Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Applied Soft Computing*, 9(3):1102–1110, 2009.

[88] BK Patle, DRK Parhi, A Jagadeesh, and Sunil Kumar Kashyap. Matrix-binary codes based genetic algorithm for path planning of mobile robot. *Computers & Electrical Engineering*, 67:708–728, 2018.

[89] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 500–505. IEEE, 1985.

[90] J-O Kim and Pradeep K Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, 8(3):338–349, 1992.

[91] Ahmed Hussain Qureshi and Yasar Ayaz. Potential functions based sampling heuristic for optimal path planning. *Autonomous Robots*, 40(6):1079–1093, 2016.

[92] Lucas Janson, Brian Ichter, and Marco Pavone. Deterministic sampling-based motion planning: Optimality, complexity, and performance. *The International Journal of Robotics Research*, 37(1):46–61, 2018.

[93] Wei Wang, Lei Zuo, and Xin Xu. A learning-based multi-rrt approach for robot path planning in narrow passages. *Journal of Intelligent & Robotic Systems*, 90(1-2):81–100, 2018.

[94] Ezequiel Di Mario, Zeynab Talebpour, and Alcherio Martinoli. A comparison of pso and reinforcement learning for multi-robot obstacle avoidance. In *2013 IEEE Congress on Evolutionary Computation*, pages 149–156. Ieee, 2013.

[95] Zhuang Wang and Jiejin Cai. Probabilistic roadmap method for path-planning in radioactive environment of nuclear facilities. *Progress in Nuclear Energy*, 109:113–120, 2018.

[96] Petr Vaněk, Jan Faigl, and Diar Masri. Multi-goal trajectory planning with motion primitives for hexapod walking robot. In *2014 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 2, pages 599–604. IEEE, 2014.

[97] Kai Li Lim, Kah Phooi Seng, Lee Seng Yeong, Li-Minn Ang, and Sue Inn Ch'ng. Uninformed pathfinding: A new approach. *Expert Systems with Applications*, 42(5):2722–2730, 2015.

[98] Kevin Hernandez, Bladimir Bacca, and Breyner Posso. Multi-goal path planning autonomous system for picking up and delivery tasks in mobile robotics. *IEEE Latin America Transactions*, 15(2):232–238, 2017.

[99] Laura Burke. "conscientious" neural nets for tour construction in the traveling salesman problem: the vigilant net. *Computers & operations research*, 23(2):121–129, 1996.

[100] Christian Wurll and Dominik Henrich. Point-to-point and multi-goal path planning for industrial robots. *Journal of Robotic Systems*, 18(8):445–461, 2001.

[101] Paraskevi Th Zacharia, Elias K Xidias, and Nikos A Aspragathos. Task scheduling and motion planning for an industrial manipulator. *Robotics and computer-integrated manufacturing*, 29(6):449–462, 2013.

[102] Jan Faigl, Miroslav Kulich, and Libor Přeučil. A sensor placement algorithm for a mobile robot inspection planning. *Journal of Intelligent & Robotic Systems*, 62(3-4):329–353, 2011.

[103] Miroslav Kulich, Jan Faigl, and Libor Preucil. Cooperative planning for heterogeneous teams in rescue operations. In *Safety, Security and Rescue Robotics, Workshop, 2005 IEEE International*, pages 230–235. IEEE, 2005.

[104] Tim Danner and Lydia E Kavraki. Randomized planning for short inspection paths. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 971–976. IEEE, 2000.

[105] Radu-Emil Precup, Emil M Petriu, Mircea-Bogdan Radae, Emil-Ioan Voisan, and Florin Dragan. Adaptive charged system search approach to path planning for multiple mobile robots. *IFAC-PapersOnLine*, 48(10):294–299, 2015.

[106] PK Das, HS Behera, and BK Panigrahi. Intelligent-based multi-robot path planning inspired by improved classical q-learning and improved particle swarm optimization with perturbed velocity. *Engineering science and technology, an international journal*, 19(1):651–669, 2016.

[107] Stefan Edelkamp and Junho Lee. Multi-robot multi-goal motion planning with time and resources. In *Annual Conference Towards Autonomous Robotic Systems*, pages 288–299. Springer, 2019.

[108] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[109] Stephen J Guy, Jatin Chhugani, Changkyu Kim, Nadathur Satish, Ming Lin, Dinesh Manocha, and Pradeep Dubey. Clearpath: highly parallel collision avoidance for multi-agent simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 177–187, 2009.

[110] Jamie Snape, Jur Van Den Berg, Stephen J Guy, and Dinesh Manocha. Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5917–5922. IEEE, 2009.

[111] Jur van den Berg, Jason Sewall, Ming Lin, and Dinesh Manocha. Virtualized traffic: Reconstructing traffic flows from discrete spatio-temporal data. In *2009 IEEE Virtual Reality Conference*, pages 183–190. IEEE, 2009.

[112] Xiaogang Jin, Jiayi Xu, Charlie CL Wang, Shengsheng Huang, and Jun Zhang. Interactive control of large-crowd navigation in virtual environments using vector fields. *IEEE Computer Graphics and Applications*, 28(6):37–46, 2008.

[113] Julien Pettré, Jan Ondřej, Anne-Hélène Olivier, Armel Cretual, and Stéphane Donikian. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 189–198, 2009.

[114] Barbara Yersin, Jonathan Maïm, P Ciechomski, Sébastien Schertenleib, and Daniel Thalmann. Steering a virtual crowd based on a semantically augmented navigation graph. In *Proc. The First International Workshop on Crowd Simulation (V-CROWDS'05), Lausanne, Switzerland*, pages 169–178, 2005.

[115] George Kingsley Zipf. Human behavior and the principle of least effort. 1949.

[116] Ioannis Karamouzas, Peter Heil, Pascal Van Beek, and Mark H Overmars. A predictive collision avoidance model for pedestrian simulation. In *International workshop on motion in games*, pages 41–52. Springer, 2009.

[117] G Keith Still. *Crowd dynamics*. PhD thesis, University of Warwick, 2000.

[118] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[119] Raghavender Etikyala, Simone Göttlich, Axel Klar, and Sudarshan Tiwari. Particle methods for pedestrian flow models: From microscopic to nonlocal continuum models. *Mathematical Models and Methods in Applied Sciences*, 24(12):2503–2523, 2014.

[120] Sujeong Kim, Stephen J Guy, Dinesh Manocha, and Ming C Lin. Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In *Proceedings of the ACM SIGGRAPH symposium on interactive 3D graphics and games*, pages 55–62, 2012.

[121] Shanwen Yang, Tianrui Li, Xun Gong, Bo Peng, and Jie Hu. A review on crowd simulation and modeling. *Graphical Models*, 111:101081, 2020.

[122] Konstantin Yakovlev, Egor Baskin, and Ivan Hramoin. Grid-based angle-constrained path planning. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 208–221. Springer, 2015.

[123] Autodesk. Maya. `https://www.autodesk.com/maya/`, 2018.

[124] JetBrains. Pycharm. `https://www.jetbrains.com/pycharm/`, 2018.

[125] Jared Go, Thuc D Vu, and James J Kuffner. Autonomous behaviors for interactive vehicle animations. *Graphical Models*, 68(2):90–112, 2006.

[126] James M Hyman. Accurate monotonicity preserving cubic interpolation. *SIAM Journal on Scientific and Statistical Computing*, 4(4):645–654, 1983.

[127] Imen Chaari, Anis Koubaa, Hachemi Bennaceur, Adel Ammar, Maram Alajlan, and Habib Youssef. Design and performance analysis of global path planning techniques for autonomous mobile robots in grid environments. *International Journal of Advanced Robotic Systems*, 14(2):1729881416663663, 2017.

[128] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.

[129] Fasheng Qiu and Xiaolin Hu. Modeling group structures in pedestrian crowd simulation. *Simulation Modelling Practice and Theory*, 18(2):190–205, 2010.