Personal and Ubiquitous Computing (2022) 26:1299–1319
https://doi.org/10.1007/s00779-022-01687-9

ORIGINAL ARTICLE

# Modelling group dynamics for crowd simulations

R. A. Saeed[1] · Diego Reforgiato Recupero[2] · Paolo Remagnino[3]

Received: 11 October 2021 / Accepted: 30 May 2022 / Published online: 15 July 2022

**Abstract**
This paper investigates a new method to simulate pedestrian crowd movement in a large and complex virtual environment, representing a public space such as a shopping mall. To demonstrate pedestrian dynamics, we consider groups of pedestrians of different size, sharing a crowded environment. A pedestrian has its own characteristics, such as *gender*, *age*, *position*, *velocity*, and *energy*. The proposed method uses a multi-group microscopic model to generate real-time trajectories for all people moving in the defined virtual environment. Additionally, a dynamic model is introduced for modelling group behaviour. Based on the proposed method, all pedestrians in each group can continuously adjust their attributes and optimize their path towards the desired visiting targets, while avoiding obstacles and other pedestrians. Simulation results show that the proposed method can describe a realistic simulation of dynamic behaviour.

**Keywords** Crowd dynamics · Multi-group microscopic model · Agent-based model · Pedestrian movement

## 1 Introduction

Simulation of crowd dynamics has been studied in different research fields, including computer graphics, virtual reality, robotics, social science, statistical physics, pedestrian, and evacuation dynamics. Broadly speaking, most techniques can be grouped in two main categories: *macroscopic* and *microscopic*. The *macroscopic* techniques focus on the aggregate behaviours of crowds. On the other hand, *microscopic* techniques are used to create realistic trajectories for individual pedestrians so that they can naturally avoid static and dynamic obstacles, and other pedestrians in the defined virtual environment [1].

✉ R. A. Saeed
raza.saeed@unibz.it

Diego Reforgiato Recupero
diego.reforgiato@unica.it

Paolo Remagnino
p.remagnino@kingston.ac.uk

1 Faculty of Science and Technology, University of Bolzano, Bolzano, Italy

2 Department of Mathematics and Computer Science, University of Cagliari, Cagliari, Italy

3 Department of Computer Science, Kingston University, London, UK

Using a similar analogy, a crowd simulation can be either *homogeneous* or *heterogeneous* [1, 2]. *Homogeneous* crowds correspond to instances where each agent has a very similar behaviour or goal. Many models have been proposed for *homogeneous* crowd simulation, e.g. the *continuum* model [1, 3] and the $flow-based$ models [4, 5]. In *heterogeneous* crowds, each agent in the crowd maintains a distinct, observable identity (see [6]). Many techniques have been proposed for simulating *heterogeneous* crowds [7, 8], e.g. $agent-based$ models [9]. In the $agent-based$ model, each agent motion in the crowd is computed separately [1, 2]. The computation of an agent motion is split in two distinct tasks: *global* and *local* navigation [1, 2]. *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 [1].

Crowd simulations based on the *microscopic* model have been widely used in many applications [6]. However, most methods do not implement the process of independently animating pedestrians in real-time. In addition, a crowd model needs to address the dynamic interactions between pedestrians. Generally, people in a crowd move in groups (see Fig. 1), where the behaviour and dynamics of a pedestrian in a group are connected with other pedestrians, and the interaction between pedestrians can significantly influence

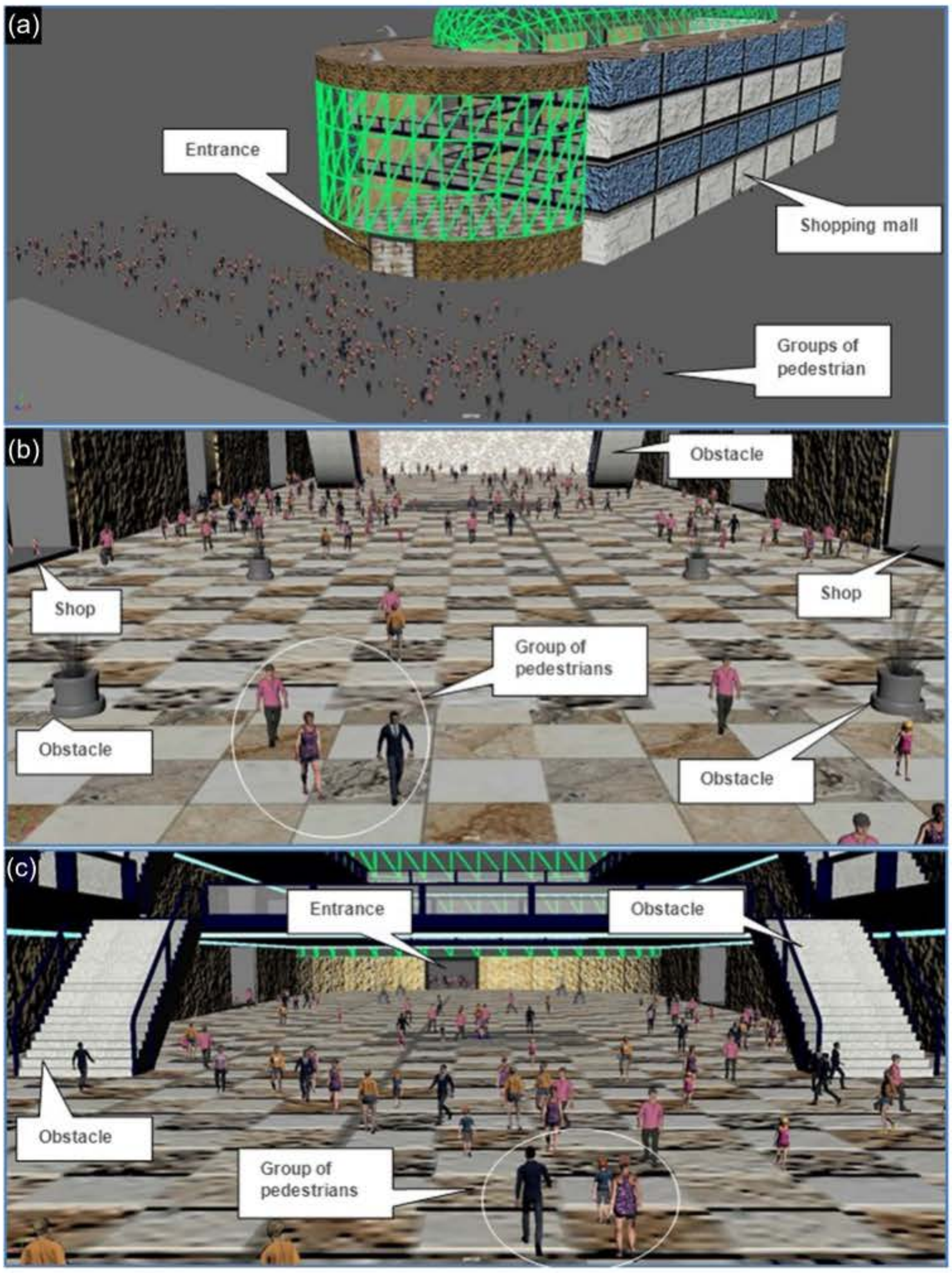

**Fig. 1** Screenshot of the shopping mall: pedestrians move in different directions to reach their desired goal points on the ground floor of the shopping mall while avoiding the static obstacles and pedestrians

crowd behaviour. Therefore, crowd simulation methods with a strong computational performance for predicting each pedestrian's movement in each population in real-time are needed.

Broadly speaking, our contribution is the introduction of a *mesoscopic* model, specifically concerned with people group's dynamics; in particular:

- We propose a new method for simulating pedestrian crowd movement in a large and complex virtual environment in a public place such as a shopping mall,
- A group of pedestrians can have a variable number of pedestrians with different characteristics,
- Pedestrians, either as individuals or in groups, move through the virtual environment in different directions to reach their desired goal points,
- A multi-group *microscopic* model is proposed to generate a real-time trajectory for each pedestrian navigating in the virtual environment.

To describe the group of pedestrians, let us consider a simulated crowd scenario shown in Fig. 1. In this simulated scenario, the crowd is organized in groups. Each group consists of a different number of pedestrian types (male and female) of different ages (old, young, and child) to establish the range of personality variation. Figure 1*b* and *c* illustrate an example, the small marked groups of pedestrians consist of "3" pedestrians. The first group (see Fig. 1*b*) consists of "1" young-man, "1" young-women, and "1" old-man. The second group (see Fig. 1*c*) consists of "1" young-women, "1" old-man, and "1" child. In our model, pedestrians always move with the same group. In any simulation, groups of pedestrians enter the virtual environment through the entrance, and they navigate inside the virtual environment to visit several prefixed target points. Afterward, they move toward the exit to leave the virtual environment.

In the rest of the paper, crowd simulation is the term used to describe the entire dynamics of a scenario, and pedestrians are referred to as *agents*. Our contribution is organized as follows: Section 2 discusses related research, Section 3 formalizes the problem, Section 4 introduces the proposed method, Section 5 introduces the crowd simulation and summarizes the simulation results. Finally, Section 6 closes the paper with final considerations.

## 2 Related work

Simulation of crowd movement and the interaction between pedestrians have been studied in the literature [10]. For example, authors in [3] present an approach for simulating crowds using discrete agents as a single continuous system. Also, a crowd model based on continuum dynamics has been provided by [11] using potential field integrated with global navigation. The crowd flows has been investigated by [12] using the particle methods and continuous pedestrian model. Moreover, the authors in [13] derived the governing equations of motion for simulating different types of pedestrians in the $2D$ environment. Additionally, there are many other approaches to simulate crowd behaviour based on cognitive modelling and behaviour [14], psychological or sociological factors [15], personality models [8], and stress modelling [1, 16].

The effect of groups on crowd movement was investigated in [17] using the least effort cellular automata algorithm. To evaluate the impact of group dynamics on the crowd movement, an agent-based model is proposed by [18, 19].

The authors in [20] suggested a multi-group microscopic model based on the interacting particle system coupled with the eikonal equation for describing groups' behaviour. Furthermore, [20] discussed a multi-behaviours microscopic model combined with the social force model and optimal path computation. The authors in [21] provided a literature review from traditional models to recent models to introduce the researchers' crowd simulation models. In a real crowd simulation application, the authors in [2] presented a crowd simulation method for generating the human-like trajectories in the simulated environment based on geometric techniques. In animation and games, the authors in [22] combined steering behaviours to allow the character to navigate their world in a life-like manner. Moreover, the problem of directing and controlling virtual crowds has been addressed by [23]. To perform multiple virtual agents' path planning, the authors in [24] present a multi-agent navigation graph for each agent 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 [1, 14].

In real-world applications, crowd movement simulation provides valuable tools for improving efficiency and safety in public places such as airport terminals [18], shopping malls, train stations [14], and theatres [20]. Moreover, crowd movement simulations have been used to develop a level-of-service concept, design elements of pedestrian facilities, planning guidelines [25], safety planning [2], and support transportation planners or managers to design timetables [20]. Furthermore, realistic simulations based on pedestrian behaviour such as obstacle avoidance, stress response, and avoidance of other pedestrians have many applications such as computer game designers, movies, and virtual environments [2]. Considerable effort has been made on locomotion, realistic pedestrian movement behaviours, path planning, and navigation in large virtual environments [15].

Finally, a crowd manifests itself with dynamics that are a combination of individual, group, and global dynamics in a given environment. Although the complexity of a very busy scene might seem chaotic at first, such as a train station or an airport concourse, individual and group dynamics shape the dynamics of the crowd as a whole and an in-depth study at microscopic or mesoscopic level can discern the underlying dynamics. This duality of individual versus crowd dynamics has been studied by social scientists, who employed nonlinear dynamics to model trajectories and interactions. Notable examples are the research contributions of Epstein [26, 27] and Helbing [28–30].

With respect to the state-of-the-art discussed above, our work proposes a method for simulating the crowd dynamics of pedestrians in a very complex environment. In this method, the crowd consists of a large number of pedestrians in multiple groups with a different number and various types of pedestrians with different intentions. Pedestrians in each group continuously adjust their paths and update their attributes independently in real-time.

## 3 Problem formulation

We study a crowd simulation of pedestrians with different group sizes and category, moving through a virtual environment for a limited amounted of time. Each group of pedestrians visits 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 *gender*, *age*, *position*, *velocity*, and *energy*. 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 pedestrians with a wide variety of characteristics. Based on the different schedule models, pedestrians are appropriately introduced 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; this 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 in a group must move with the group to their destination, while avoiding stationary obstacles and other pedestrians (the dynamic obstacles) when they come in close proximity.

## 4 Proposed method for crowd simulation

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 is introduced in the proposed method for modelling pedestrian's behaviours, where each type of pedestrian has its own attributes. During the simulation, pedestrians move in groups toward their destination points in the simulated working environment, and avoid obstacles and other pedestrians when they move closer. Moreover, the goal point for each group of pedestrians changes 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 tuning their parameters, avoiding collisions, optimizing their paths, and changing their behaviour. In this work, pedestrians use local knowledge to avoid local collisions and interact with other pedestrians. Furthermore, pedestrians use global knowledge for long-term planning and provide goal-directing capability.

In this study, pedestrians continuously adjust their position based on the proposed method in real-time. Therefore, visualizing their motions is important to understand the trajectories that pedestrians follow. For that reason, the virtual environment and the virtual pedestrians are imported into the $3D$ animation software package *Autodesk Maya* [31] via a custom *Python* script. Then, the *Maya*'s key-frame 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 proposed method consists of three main steps: environment setting, motion computation and steering behaviours, and locomotion, described in the three following sections.

### 4.1 Environment setting

The first step is to introduce the working environment, obstacles, pedestrians, and goal points. Moreover, this step identifies the initial state of the pedestrian's attributes: *gender*, *age*, *position*, *velocity*, and *energy*. Also, interactions between pedestrians with the environment have to be defined.

***Remark 1*** During this simulation, the virtual pedestrian's position is defined along the $x$, $y$, 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 0.

#### 4.1.1 Virtual environment and obstacles

The walking area of the virtual environment is considered a two-dimensional ($2D$) workspace ($C$) for all simulations. The workspace centre is located on the origin point, and the continuous workspace is divided into square grid cells $(x = 1, z = 1)\, unit$. In the proposed method, all the grid cells of the given workspace meet the following equation:

$$C = \sum_{x=1}^{w} \sum_{z=1}^{l} C_{(x,z)} \tag{1}$$

where $w$ and $l$ represent the width and length of the simulated workspace and $C_{(x,z)}$ represents the grid cells in the workspace. Afterwards, the obstacle model is imported into the scene to prevent the pedestrians from walking through the space occupied by obstacles. The region of space occupied by the 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 $1 \leq z \leq l$. A given cell can correspond either to a navigable area $C_{(x,z)}$ or to a space occupied by the obstacles $C_{(x,z)} \in C_{obs}$. In this study, the obstacles are distributed at different locations in the free space. A safety zone with radius $R$ is created around the obstacles to avoid the possibility of overlapping the paths traced by pedestrians with the obstacle boundaries. The steps for constructing the workspace and obstacles are illustrated in Algorithm 1, and a brief description of each step is provided as follows:

**Algorithm 1** Generate workspace and obstacles.

```
1: Inputs:
       $n3DObstacles$
2: $import\ the\ workspace\ model\ into\ the\ scene$
3: $width(w), length(l) \leftarrow workspace\ in\ the\ scene$
4: $C_{obs} = [], C_{free} = [],$
5: for $i = -w/2$ to $w/2$ do
6:     for $j = -l/2$ to $l/2$ do
7:         $C(i,j) \leftarrow 0$
8:     end for
9: end for
10: $import\ obstacles\ models\ into\ the\ scene$
11: $move\ obstacles\ to\ different\ locations$
12: $create\ a\ safety\ zone\ around\ obstacles,\ (R = 1)$
13: for $obs = 1$ to $n3DObstacles$ do
14:     $x_1, z_1 \leftarrow obstacles\ model$
15:     for $k = x_1 - r$ to $x_1 + r$ do
16:         for $l = z_1 - r$ to $z_1 + r$ do
17:             if $distance\ between\ (x_1, z_1)\ and\ (k,l) < R$ then $C(k,l) \leftarrow 1$
18:             end if
19:         end for
20:     end for
21: end for
22: for $k = -w/2$ to $w/2$ do
23:     for $l = -l/2$ to $l/2$ do
24:         if $C(i,j) = 1$ then $assign\ C(k,l)\ as\ C_{obs}$
25:         else $assign\ C(k,l)\ as\ C_{free}$
26:         end if
27:     end for
28: end for
```

Step 1: Define the number of obstacles *n3DObstacles*. Algorithm 1-Line(1)

Step 2: Import the models of the workspace and obstacles into the scene. Algorithm 1-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 1-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 1-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 1-Line(7)

Step 6: Create a safety zone around each obstacle with a radius of $R$. 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 is denoted by the matrix $C_{obs(h,l)}$, $(h = 1, 2)$, $(l = 1...nObstacles)$. Algorithm 1-Line(13–21).

Step 7: Set the value of each grid cell of the space occupied by obstacles $C_{obs}$ around each obstacle with a radius of $R$ inside the working environment $C$ to "1". Algorithm 1-Line(17).

Step 8: Create $2D$ array from the grid cell values, where the grid cells $C_{(x,z)}$ can correspond either to the navigable area $C_{(x,z)} \in C_{free}$ or to a space occupied by obstacles $C_{(x,z)} \in C_{obs}$. Algorithm 1-Lines(22–28).

### 4.1.2 Pedestrians

After generating the workspace with obstacles, the virtual pedestrians are introduced into the scene. First, pedestrians are assigned attributes; these include *gender*, *age*, *position*, *velocity*, and *energy*. Algorithm 2 describes this in detail. A brief description of each step follows:

**Algorithm 2** Generate pedestrians.

```
1: Inputs:
       $nGroups,\ nTypes,\ c_{p1}$
2: $nPedestrians_{crowd} \leftarrow []$
3: $Pedestrians_{crowd} \leftarrow []$
4: $nPedestrians_{total} \leftarrow 0$
5: $import\ 3D\ character\ models\ into\ the\ scene$
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:            $get/set\ position\ of\ pedestrians$
15:            $get/set\ velocity\ of\ pedestrians$
16:            $get/set\ energy\ level\ of\ pedestrians$
17:            $Pedestrians_{group}.append(Pedestrian)$
18:        end for
19:    end for
20:    $Pedestrians_{crowd}.append(Pedestrians_{group})$
21:    $nPedestrians_{crowd}.append(nPedestrians_{group})$
22:    $nPedestrians_{total} \leftarrow nPedestrians_{total} + nPedestrians_{group}$
23: end for
```

Step 1: Define the number of groups (*nGroups*), number of pedestrian's type *nTypes*, and constant $c_{p1}$. Algorithm 2-Line(1).

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

Step 3: Generate the crowd with several groups (*nGroups*), where pedestrians in each group ($Pedestrians_{group}$) consist of a different number of pedestrians ($nPedestrians_{group}$). Algorithm 2-Line(6–24).

Step 4: Create pedestrians with their attributes such as (position, velocity, energy level,... etc.) 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 associated characteristics. Algorithm 2-Lines(13–18).

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 2-Line(20–21).

Step 6: Determine the total number of pedestrians in the crowd ($nPedestrians_{total}$) by summing all pedestrians in all groups. Algorithm 2-Line(22).

### 4.1.3 Goal points

Each group of pedestrians is assumed to have different goal points to visit in terms of the number and sequence of the goal points. The goal points in the simulated environment correspond to a specific region in the environment. The steps for formulating the list of goal points for each group in the simulated environment are illustrated in Algorithm 3, and a brief description follows:

Step 1: Define the number of the goal points (*nGoals*) and their locations ($Goals_{group}$) in the crowd. Algorithm 3-Line(1).

Step 2: Denote the list of all goal points existing 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; the $y - coordinate$ ($Goals_y$) is set to zero vector. Algorithm 3-Line(3)

Step 3: Create the number of goal points randomly ($nRandom_{goals}$) between [1, *nGoals*] for each group ($g_r$), and also the sequence of the goal points is generated randomly. Algorithm 3-Line(6–14)

Step 4: Create the starting and the final destination points for each group ($g_r$), which are located outside the working environment. The $x - coordinate$ for both the start and the end points are generated randomly between

**Algorithm 3** Generate goal points.

```
1: Inputs:
     $nGroups,\ nGoals$
2: $Goals_{crowd} \leftarrow [\,],\ Goals_{group} \leftarrow [\,],\ nGoals_{crowd} \leftarrow [\,]$
3: $GoalPoints^{list} \leftarrow 3D\,model\,in\,the\,scene$
4: for $g_r = 1$ to $nGroups$ do
5:     $nGoals_{group} \leftarrow 0$
6:     $nRandom_{goals} \leftarrow between\,[1, nGoals]$
7:     for $g_o = 1$ to $nRandom_{goals}$ do
8:         $nRandom_{goalsIndex} \leftarrow between\,[1, nGoals]$
9:         $x - coordinate \leftarrow GoalPoints^{list}[nRandom_{goalsIndex}]$
10:        $y - coordinaten \leftarrow 0$
11:        $z - coordinate \leftarrow GoalPoints^{list}[nRandom_{goalsIndex}]$
12:        $goal_{position} \leftarrow [x - coordinate, y - coordinate, z - coordinate]$
13:        $Goals_{group}.append(goal_{position})$
14:    end for
15:    $create\,the\,starting\,and\,final\,destination\,points$
16:    for $g_p = 1$ to $2$ do
17:          $x - coordinate\ \leftarrow random\,number\,between[(-w/2),(w/2)]$
18:          $y - coordinate\ \leftarrow 0$
19:          $z - coordinate\ \leftarrow l/2 + c_{s1}$
20:        $goal_{position} \leftarrow [x - coordinate, y - coordinate, z - coordinate]$
21:        if $g_p = 1$ then $Goals_{group}.append(first, goal_{position})$
22:        else if $g_p = 2$ then $Goals_{group}.append(last, goal_{position})$
23:        end if
24:    end for
25:    $nGoals_{group} \leftarrow nRandom_{goals} + 2$
26:    $nGoals_{crowd} \leftarrow nGoals_{crowd}.append(nGoals_{group})$
27:    $Goals_{crowd}.append(Goals_{group})$
28: end for
```

$[(-w/2),(w/2)]$, 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 3-Lines(15–24)

Step 5: Formulate the overall goal points' array of the crowd ($Goals_{crowd}$) by concatenating of goal points' array of all groups ($Goals_{group}$). Algorithm 3-Line(27)

### 4.1.4 Initilization of a pedestrian

Each pedestrian is assigned a position and a velocity. Pedestrians and groups are distributed randomly outside the virtual environment. Then, they start to move to reach the goal point in the list of goal points. Steps to compute the initial value of $Position_{p_e}$ and $Velocity_{p_e}$ for each pedestrian are illustrated in Algorithm 4, and a brief description follows:

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 of pedestrians in the crowd $Goals_{crowd}$, and define constants $c_{f1}$, $c_{f2}$ and $c_{f3}$. Algorithm 4-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 depends on the value of constant $c_{f1}$. The starting point represents the first goal point in the goal points list $Goals_{group}$ that belongs to the group $g_r$. Algorithm 4-Lines(6–11).

Step 3: Check the distance between the generated position $Position_{p_e}$ and the position of all the other created pedestrians $Position_{nPede}$; the distance should not reside within a certain value ($c_{f2}$). Algorithm 4-Lines(12–23).

Step 4: Formulate the list of all pedestrian's position in the crowd ($Pedestrians_{crowd}$) by concatenating all randomly created positions of pedestrians $Position_{p_e}$ of all groups. Algorithm 4-Line(24).

**Algorithm 4** Setting the initial state of the pedestrians in the crowd $Pedestrians_{group}$.

```
1: Inputs:
       $nGroups$, $nGoals$, $Pedestrians_{crowd}$,
       $nPedestrians_{crowd}$, $Goals_{crowd}$, $c_{f1}$, $c_{f2}$, $c_{f3}$
2: $Positions_{crowd} \leftarrow []$
3: for $g_r = 1$ to $nGroups$ do
4:     $Pedestrians_{group} \leftarrow Pedestrians_{crowd}[g_r]$, $nPedestrians_{group} \leftarrow nPedestrians_{crowd}[g_r]$
5:     $Goals_{group} = Goals_{crowd}[g_r]$
6:     for $p_e = 1$ to $nPedestrians_{group}$ do
7:         $Position_{p_e} \leftarrow []$
8:           $x - coordinate \leftarrow (Goals_{group}[1][0] + random.random() * cf1)$
9:           $y - coordinate \leftarrow 0$
10:          $z - coordinate \leftarrow (Goals_{group}[1][2] + random.random() * cf1)$
11:        $Position_{p_e} \leftarrow [x, y, z]$, $nPede \leftarrow 1$
12:        while $nPede < P_e$ do
13:            $interfere \leftarrow "True"$
14:            while $interfere \leftarrow "True"$ do
15:                find $distance$ between $Position_{p_e}$ and $Positions_{crowd}[nPede]$
16:                if $distance < c_{f2}$ then
17:                    $create\,another\,random\,position$
18:                else
19:                    $interfere \leftarrow "False"$
20:                end if
21:            end while
22:            $nPede \leftarrow nPede + 1$
23:        end while
24:        $Positions_{crowd}.append(Position_{p_e})$
25:        $update\,pedestrians'\,positions\,Position_{p_e}$
26:        $create\,random\,velocity\,for\,Pedestrians_{group}[p_e]$
27:        $Velocity_{p_e} \leftarrow []$
28:          $v_x - coordinate \leftarrow random\,number * c_{f3}$
29:          $v_y - coordinate \leftarrow 0$
30:          $v_z - coordinate \leftarrow random\,number * c_{f3}$
31:        $Velocity_{p_e} \leftarrow [v_x, v_y, v_z]$
32:        $update\,pedestrians'\,velocity\,Velocity_{p_e}$
33:    end for
34: end for
```

Step 5: Create random velocity values $Velocity_{p_e}$ ($p_e = 1, .., nPedestrians_{group}[g_r]$) 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 4-Lines(25–3).

Step 6: Update position in $Pedestrians_{group}[p_e]$ with $Position_{p_e}$ (see Algorithm 4-Line(25)) and velocity in $Pedestrians_{group}[p_e]$ with $Velocity_{p_e}$ (see Algorithm 4-Line(32)) in the scene.

At intitialization, all pedestrians' characteristics are set, including dynamics' attributes *position* and *velocity*, and *personal* attributes, such as *gender*, *age*, and *energy*. 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 at random.

In this study, pedestrians ($Pedestrians_{crowd}[g_r]$, $g_r = 1, ..., nGroups$) enter the simulated environment one at the same time. Pedestrians in the crowd need to decide whether and when (key-frame *kFrame*) they start to move

**Algorithm 5** Create a list of random key-frame for activating groups of pedestrians 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: $sort\,Pedestrians_{crowd}\,according\,to\,kFrame_{activate}$
```

from the starting point. Pedestrian enter the virtual environment following a specific schedule, as illustrated in Algorithm 5. Based on this algorithm, each group $g_r$, ($g_r = 1, ..., nGroups$) starts moving at randomly created keyframes ($kFrame_{activate}[g_r]$) between (1, $nFrames*c_{a1}$), where *nFrames* represents the total number of frames and $c_{a1}$ is a constant.

### 4.2 Motion computation and steering behaviours

Pedestrians' dynamics include actions such as *obstacle avoidance*, *pedestrian collision*, *avoidance*, *flocking with neighbouring pedestrians*, *goal − directing*, and combined steering behaviours. Pedestrian behaviours are discussed in the following subsections. In each key frame, pedestrians move from the current position to the new updated position at a variable walking pace. The new updated position and velocity are calculated based on the steering behaviours with the current pedestrian's position and velocity.

Obstacle avoidance To direct pedestrians to avoid the collision, we employed an extended version of a previous method developed by us, called boundary node method (*BNM*) which is used to generate a collision-free path for each pedestrian when they move close to the obstacles in real-time. The details of the *BNM* method are given in [32, 33].

Using the global path planning in real-time contexts is difficult for modelling pedestrians' behaviours 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. The nodes are denoted by $p(q), (q = 1...9)$, the centroid node $p(5)$ represents the pedestrian's location, as illustrated in Fig. 2, and 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 Fig. 2*b*, 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. The *BNM* method uses the length and width of obstacles to generate a collision-free path. However, the height of obstacles (that would allow pedestrians to identify them) is not taken into account by this method. More details on the *BNM* method for path planning and its applicability to various problems are provided in [32]. While the distance between the pedestrian's position and centre of the obstacle is less than ($r$+$R$), the pedestrian interferes with the safety zone around obstacles, where $R$ and $r$ represent

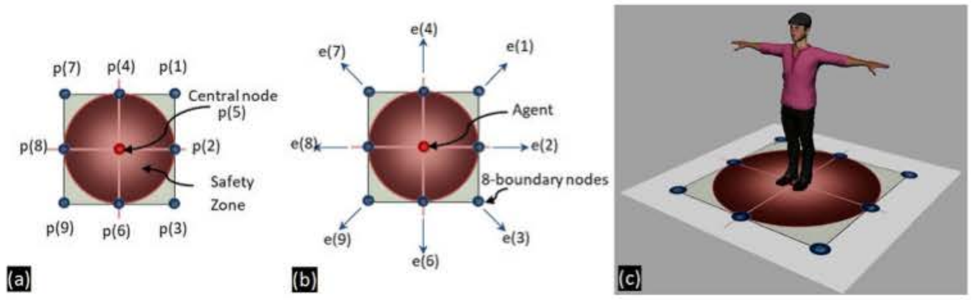

**Fig. 2** A nine-node quadrilateral element with a safety zone (***a***) along with its motion directions (***b***) and simulated pedestrian in a virtual environment (***c***)

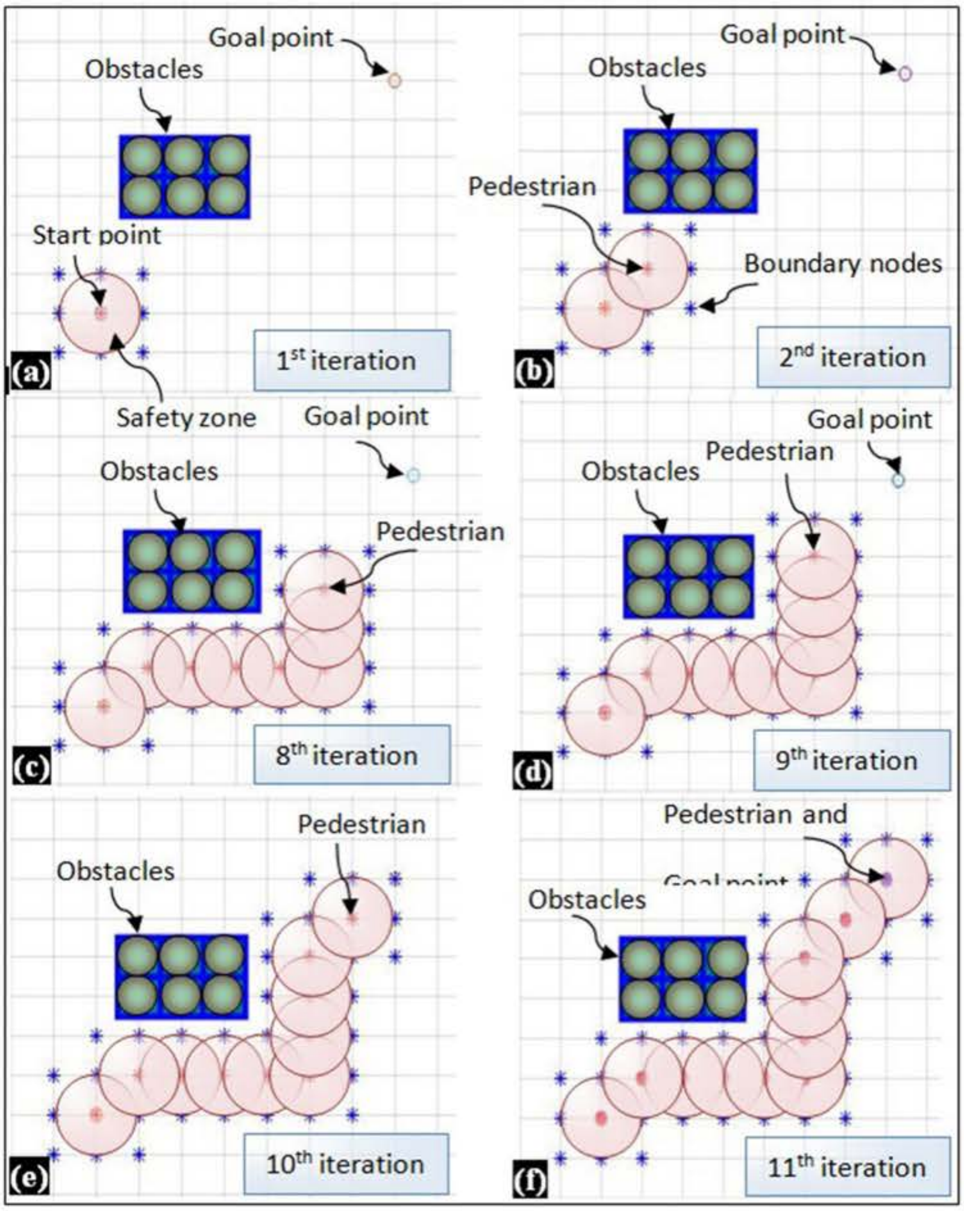

**Fig. 3** Obstacle avoidance using *BNM*: demonstrates the single pedestrian exploration in a *2D* environment with obstacles

the radius of the safety zone around the obstacle and pedestrian, respectively. Based on the obstacles avoidance procedure, the pedestrian and boundary nodes are steered to avoid obstacles and change their motion direction by selecting a new position in the free space $C_{free}$.

In order to explain the obstacle avoidance and demonstrate how the pedestrians are steered to avoid obstacles and change their moving direction with the help boundary nodes, let us consider a very simple example with only one pedestrian, as shown in Fig. 3. In the first iteration of the simulation, $t = 1$, the pedestrian ($p(5)$) is initially located at the start point and the boundary nodes $p(1 \rightarrow 4)$ and $p(6 \rightarrow 9)$ are generated around the pedestrian's position. This is illustrated in Fig. 3*a*, where the boundary nodes are represented by blue star objects and pedestrian is represented by the red star object. At the iteration number $t = 2$, as the pedestrian wants to move forward with the boundary nodes from the current $Position_{p_e}$ (see Fig. 3*a*) to the new $Position_{p_e}$ (see Fig. 3*b*), all the obstacles in the working environment are examined for possible collisions. Let us assume that the distance between the pedestrian position and the center of the obstacle is greater than the sum of the radius of the safety zone around the obstacle and pedestrian $(R + r)$. In this case, there is no interference with the obstacle, and the pedestrian can safely change its position to a new one. In the third iteration $t = 3$, as the pedestrian moves toward the goal position, nodes $p(1)$, $p(4)$, and $p(7)$ interfere with the safety zone around the obstacle. In this case, the pedestrian needs to investigate the workspace to find the next position without colliding against obstacles. This implies that the pedestrian can only move in the $x$-direction, in this case: therefore, the movement of the pedestrian is restricted to either the left or right direction. Changing the motion direction depends on values $E(2)$ and $E(8)$ at the location $p(2)$ and $p(8)$. The values of $E(2)$ and $E(8)$ are calculated based on the potential function. The procedure of calculating the potential value of $E$ based on the new proposed potential function is illustrated in detail in our previous work [32]. The results of the calculated potential values of $E(2)$ and $E(8)$ based on the potential function show that $E(2) < E(8)$. This implies that the next position of the pedestrian must be in the right direction. The same procedure is repeated for the iteration number $t = 3 - 6$ by shifting the pedestrian to the right until it passes the block of obstacles. At $t = 8$, as illustrated in Fig. 3c, the pedestrian moves forward without collision with any obstacles. Similarly, for the iterations $t = 9 - 11$, the pedestrian moves forward until it reaches the goal position as shown in Fig. 3$d - f$.

Pedestrian collision avoidance During the crowd simulation, pedestrians need to stay together and maintain the group while pedestrians must keep a certain distance (spaces) between themselves. The study of group behaviors in a crowd simulation can help in better understanding and respecting different personal spaces. However, the culture, relationship between pedestrians, and crowd density are essential factors affecting personal space. Besides, in certain circumstances, identifying the minimum personal space requirements is needed to protect health and limit the spread of potential disease.

Each moving pedestrian represents a dynamic obstacle for the remaining pedestrians in the scene. 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 $r$, and pedestrians should keep a certain distance from the other neighbouring pedestrians. We assume that the distance between pedestrians should not be less than ($2r$). Suppose that the distance between current and neighbouring pedestrian drops below a specific value ($2r$). In such a case, the current pedestrian changes its motion direction in surrounding areas to prevent the collision. For this situation, we have proposed an appropriate reactive behaviour, called pedestrian collision avoidance, for avoiding collision between pedestrians walking closer. The current pedestrian steers to either left or right for a newly calculated position, located on the third vertex of an equilateral triangle. The overall process of pedestrian collision avoidance is illustrated in Algorithm 6, 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 6-Line(1).

Step 2: Check for interference between current pedestrian $Position_{p_e}$ and all pedestrians $Position_{p_e}$

**Algorithm 6** Pedestrian collision avoidance.

```
1: Inputs:
       $Positions_{crowd}$,
       $newPosition_{p_e}$,$currentPosition_{p_e}$,$nPedestrians_{total}$
2: for $p_e = 1$ to $nPedestrians_{total}$ do
3:     find $distance\ between\ new\ Position_{p_e}\ and\ Positions_{crowd}[p_e]$
4:     if $distance < (2 \times r)$ then
5:         $\Delta x = newPosition_{p_e}[0] - Positions_{crowd}[p_e][0]$
6:         $\Delta z = newPosition_{p_e}[2] - Positions_{crowd}[p_e][2]$
7:         if $random.random() < 0.5$ then $rotate_{angle} \leftarrow 60.0$
8:         else $rotate_{angle} \leftarrow -60.0$
9:         end if
10:        $\alpha \leftarrow rotate_{angle}/180 * math.pi$
11:        $new_x \leftarrow newPosition_{p_e}[0] + math.cos(\alpha) * \Delta x + math.sin(\alpha) * \Delta x$
12:        $new_y \leftarrow 0$
13:        $new_z \leftarrow newPosition_{p_e}[2] + math.sin(-\alpha) * \Delta x + math.cos(\alpha) * \Delta z$
14:        $newPosition_{p_e} \leftarrow [new_x, new_y, new_z]$
15:        find $distance\ between\ new\ Position_{p_e}\ and\ Positions_{crowd}[p_e]$
16:    else if $distance = 0$ then
17:        $newPosition_{p_e} \leftarrow currentPosition_{p_e}$
18:    end if
19: end for
```

$(p_e = 1, ..., nPedestrians_{total})$ in the crowd, if the distance between current pedestrian $Position_{p_e}$ and the other pedestrians in the crowd is less than $2 \times r$, then the interference will occur and a new position $Position_{p_e}$ needs to be calculated. Algorithm 6-Line(2-6).

Step 3: Suppose a collision happens between pedestrians, the proposed method implements *pedestrian collision avoidance* to avoid collision between pedestrians. 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. The pedestrian will choose $right - hand\,side$ or $left - hand\,side$ to avoid the collision based on the random value. Algorithm 6-lines(7–13).

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 crowd $Positions_{crowd}$.

Flocking behaviour We have considered a sort of cooperation among pedestrians moving with the same group. The pedestrians' steering behaviours in the same group demonstrate how pedestrians react, and pedestrians outside of the group are ignored. We considered steering behaviours such as *separation*, *cohesion*, and *alignment* related to the pedestrians. These behaviours influence the pedestrians' motion, more precisely, their position and velocity. The *separation* steering behaviour gives a pedestrian the ability to maintain a certain separation distance from other pedestrians in the same group, as shown in Fig. 4*a*. The *separation* steering behaviour is used to prevent pedestrians from crowding together. Moreover, the *cohesion* steering behaviour gives a pedestrian the ability to cohere with the other pedestrians, as shown in Fig. 4*b*. At the same time, the *alignment* steering behaviour tends to align pedestrians with its neighbouring pedestrians, as demonstrated in Fig. 4*c*. For details on the steering behaviours, see [22, 34].

Goal-directed behaviour Pedestrians in each group walk towards a well-defined goal at a certain point in the virtual environment modelled by a goal-directed behaviour. The distance between the current position of the pedestrian and the current active goal is calculated. If the distance between these two points is less than a certain value, then a new goal point is assigned to the current group. Suppose the current goal point is the last destination point. In that case, the group will terminate the simulation, and this group will not be active anymore. Otherwise, the *goal_directed* steering behavior calculates the $goal_{vector}$ and normalizes the $goal_{vector}$ by applying a weighting $goal_{weight}$, as illustrated in Algorithm 7.

Combining behaviours Combining behaviours can happen in two ways: (1) switching, (2) blending [22]. As the walking circumstance changes in the simulated environment, the pedestrian may "switch" between behavioural modes. Alternatively, these behaviours, which are acting in parallel, are commonly "blended" together. For example, in normal situations, as the pedestrians move around through the simulated environment toward their destination, they blend both flocking and goal-directed behaviours to a single steering force vector to allow the group to walk toward their goals. Furthermore, suppose that a group of pedestrians approaches an obstacle or other pedestrians in the same or different groups. This situation leads to a behavioural switch from moving to collision avoidance. All pedestrians try to avoid collisions with obstacles and pedestrians, and pedestrians do not prefer to remain with the group while avoiding obstacles. The proposed method uses a *Python* script to evaluate pedestrians' behaviours, to compute a new state of pedestrians, and also generates $3D$ scenes and views in the animation software package *Autodesk Maya* [31], as illustrated in the following subsection.

### 4.3 Locomotion

The proposed method computes the pedestrians' movement taking into account the steering behaviours, and generates locomotion for each pedestrian in the virtual environment in real-time. In this study, the *Maya*'s key-frame is used for $3D$ animating motion for a large crowd of multi-groups of pedestrians walking around through the virtual environment toward their destination points. As an implementation of the pedestrian movement, we used different types of pedestrians for this simulation. Then, we imported pedestrians into *Maya* via *Python* scripts. Subsequently, *Maya* allows the imported models to control the pedestrians. Based on the proposed method, the position and the velocity of each virtual pedestrian in the scene are updated in real-time.

## 5 Crowd simulation

This section describes the environment and the settings that we have considered for the proposed method.

In this simulation, pedestrians in different group sizes enter the virtual environment through the main entrance. The entrance's width is "15" *unit* (we assume that all distances in this study are measured in *unit* of length). The width $w$ and the length $l$ of the simulated environment are set to "105" and "330" *unit*, respectively. Inside the simulated environment, there are "8" shops (goal points) that the pedestrians want to visit ("4" shops on both the right- and left-hand sides). Each goal point corresponds to a specific

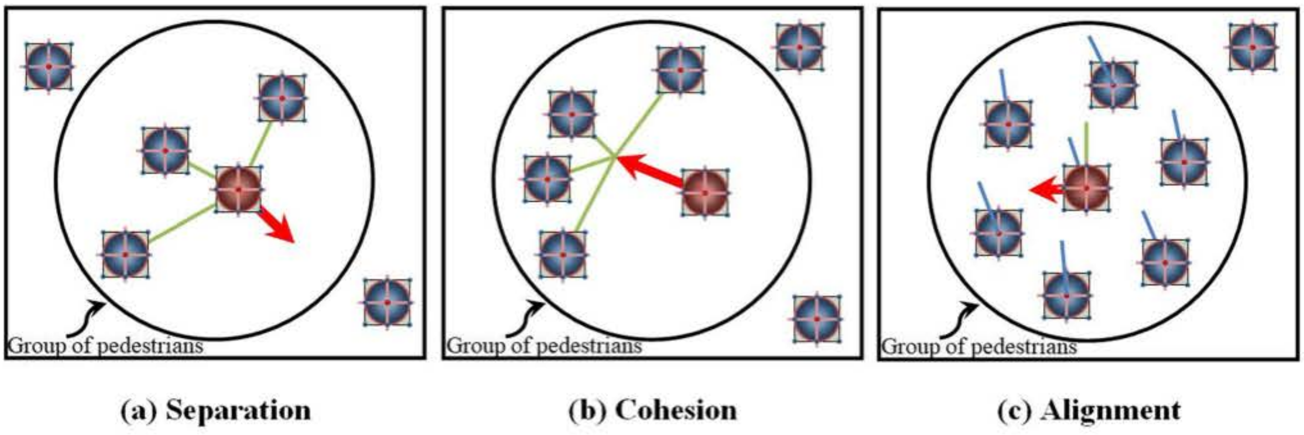

**Fig. 4** Three different types of steering behaviours: (**a**) separation, (**b**) cohesion, and (**c**) alignment. A similar figure is provided by [22]

**Algorithm 7** Goal-directed behavior.

```
1: Inputs:
       $current\, Goals_{group}$, $current\, Position_{p_e} (p_e \leftarrow index)$,
       $goal_{weight}$, $C_{g1}$
2: $Goal_i \leftarrow Goals_{index}[g_r]$
3: if ($distance\, between\, current\, Position_{p_e}\, and\, current\, Goals_{group}$) $< C_{g1}$ then
4:     $Goals_{index}[g_r] \leftarrow Goals_{index}[g_r] + 1$
5:     if $Goals_{index}[g_r] \leftarrow length(Goals_{group})$ then
6:         $terminate[g_r] \leftarrow "on"$, $active_{group}[g_r] \leftarrow "off"$
7:     end if
8: end if
9: $goal_{vector} =\leftarrow subtracting(current\, Goals_{group}, current\, Position_{p_e})$
10: $goal_{vector} \leftarrow normalizing(goal_{vector}, goal_{weight})$
11: ....
12: $Combining\, with\, other\, behaviors,\, if\, there\, is$
13: ....
```

region (shop) in the shopping mall environment. Moreover, there are "2" general service points. These points are considered a temporary goal point. We add them to the list of the goal points when the group needs them. 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 service points are located at (0,0,0) and (0,0,−135). Sometimes, pedestrians consider undertaking other activities during the visit, 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.

The simulated environment is confined by walls and other stationary objects are present in the scene. For example, there are several vases in the walking area of the shopping mall at different locations (see Fig. 5). In this study, the vases are used as static obstacles, and pedestrians need to pay attention and keep a certain distance. We assume that there are four static obstacles located at $[(-20, 0, 80), (20, 0, 120), (-20, 0, 120)$, and $(20, 0, 80)]$ and a circular safety zone is created around each obstacle with a radius of 4 *units*. Moreover, there are two stairs (static obstacles) on the ground floor of the shopping mall, and the centre of obstacles is located at $[(-30, 0, -35), (30, 0, -35)]$, where a circular safety zone is created around these obstacles as well with a radius $R$ of 7 *units*. While pedestrians move close to the obstacles, they keep a safety distance.

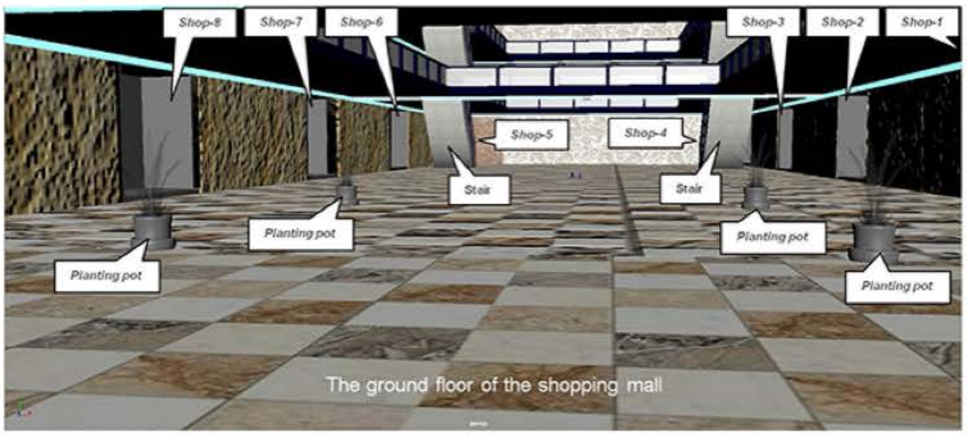

**Fig. 5** Screenshot of the shopping mall with obstacles. The walking area on the ground floor of the shopping mall with the *3D* obstacles is shown

In this study, pedestrians are divided 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 [35] are shown in Fig. 6. Each pedestrian has a different energy level, i.e. an *old* pedestrian has a lower energy level than a *young* pedestrian. The energy level of a person will decrease as he/she walks through the environment. Pedestrians rest at the sitting place in a service point when their energy level goes below the minimum energy level. During the simulation, each pedestrian is considered a dynamic obstacle for other pedestrians. Each pedestrian has a personal space requirement with a radius, *r*, where other pedestrians cannot interfere. 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*r units*, such that pedestrians do not interfere with the other neighbouring pedestrians. In this work, the proposed methods are implemented in *Python*, and the crowd simulation was carried out on a PC with Intel(R) core(TM) i5-8300*H* CPU 2.30*GHz*, 8*RAM*. The source code is available per request by sending an email to the authors.

## 5.1 Crowd simulation results

In our simulations, we present pedestrian movement in a crowd made of "200" different small groups, each with a distinct set of 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 ("696"). Pedestrians' group size distributions in the crowd are illustrated in Table 1. As shown in the table, the groups' size is different in terms of the number of pedestrians. For example, "8" groups (4% of the whole number groups) are seen in the crowd. The number of each pedestrian's type (types of pedestrian="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 2. 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).

Seven different types of characters (see Fig. 6) are used to generate "200" groups of pedestrians with "696" pedestrians. Initially, all groups of pedestrians are scattered in the front of the virtual environment around the starting point, as shown

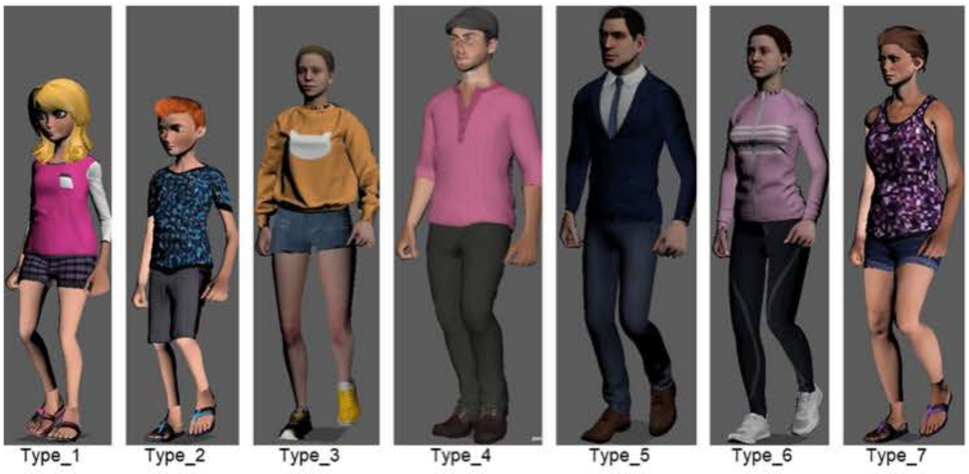

**Fig. 6** Shows different *3D* characters animation obtained from *Mixamo* [35]

**Table 1** 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 |
| Percent | 0.04 | 0.145 | 0.325 | 0.23 | 0.19 | 0.045 | 0.005 |

**Table 2** 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 |
| Percent | 0.155 | 0.134 | 0.139 | 0.132 | 0.142 | 0.149 | 0.148 |

in Fig. 7*a*. 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, groups were appropriately introduced into the shopping mall environment based on the defined schedule.

Each group starts moving into the mall at a randomly created key-frame between (1, $nFrames*c_{a1}$), where $c_{a1}$ is set to 0.5. Figure 8 illustrates the activation's' $key - frames$ of all groups; the blue point objects represent the $key - frames$ 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 $key - frame$, pedestrians' attributes and their positions are calculated and updated independently in real-time. The maximum number of frames *nFrames* is set to "200" $key - frames$. The simulation results of the pedestrians' routes for all groups at each $key - frame$ are illustrated in Fig. 7*b*. The figure shows that the proposed method allows pedestrians to navigate the walking area from the starting point to the final destination point. The results show that the proposed method prevents pedestrians from colliding with the obstacles in the scene by using the obstacle avoidance method, as illustrated in Fig. 7*b* "Obstacles' locations". As the pedestrians move closer to an obstacle or other pedestrians, they need to change their trajectory to avoid potential collisions. Our method exploits the high performance of *BNM* for collision avoidance. Moreover, our *BNM* method makes sure pedestrians never get stuck behind obstacles or against walls, always guaranteeing a collision-free path.

Different screenshots of the simulation results from different viewpoints are presented in Figs. 1 and 9. As shown in Fig. 1*b*&*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. Different types of walking states are demonstrated in Fig. 9*b*, a closeup of a 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 pedestrians are terminated

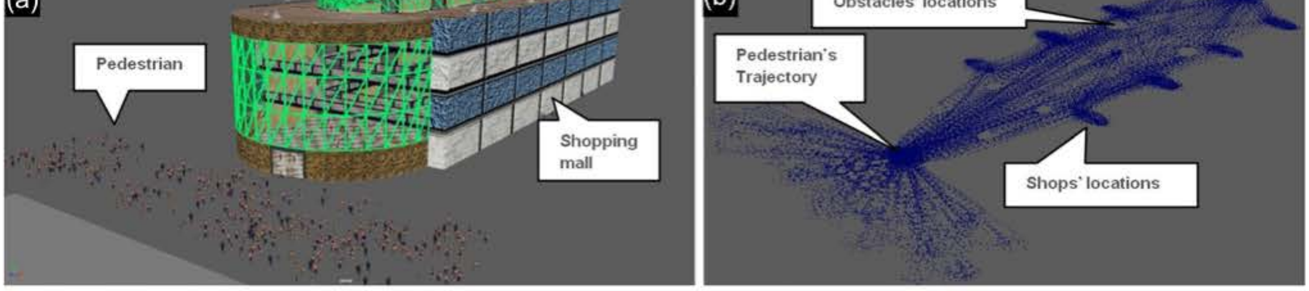

**Fig. 7** Screenshot of pedestrian flow: (***a***) shows the initial configuration of the simulation, and (***b***) the pedestrians' trajectories in the final stage of the simulation

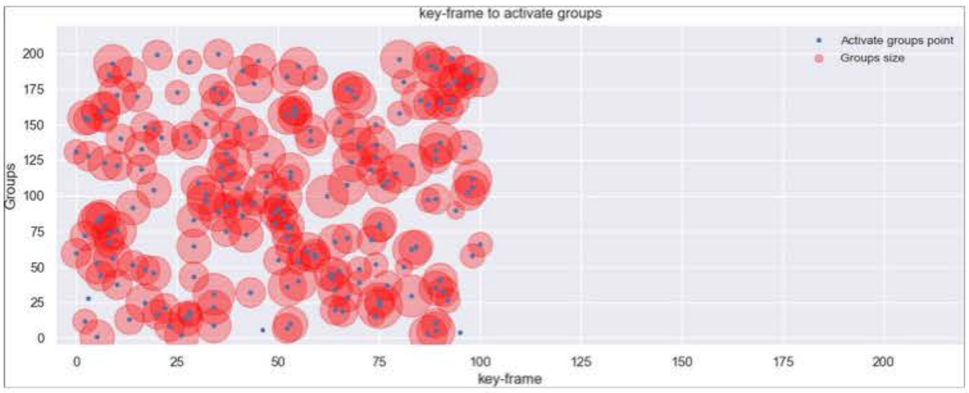

**Fig. 8** Key-frames for activating groups of pedestrians to move in the virtual environment

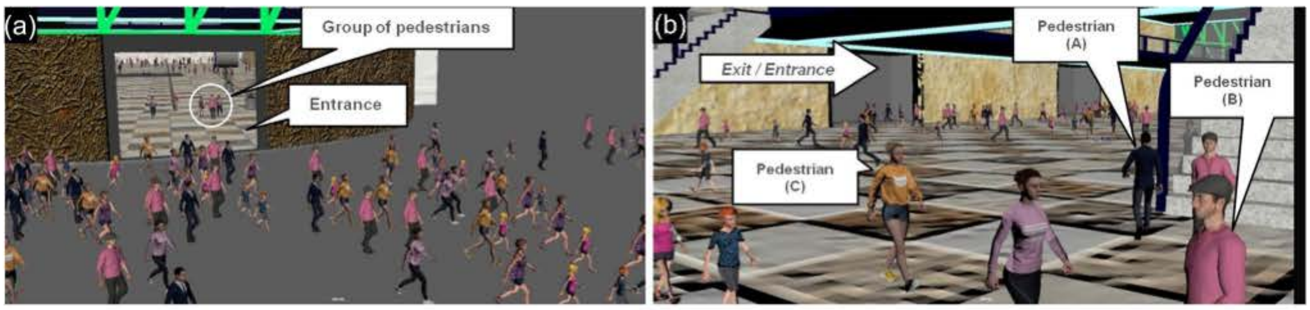

**Fig. 9** Screenshot of the simulation: showing the pedestrian's movement towards the goal points in a scene

in front of the shopping mall, as illustrated in Fig. 9*a*. The figure showed that the most crowded area in the virtual environment is found in front of the mall, where the distance between pedestrians is very small. After the simulation runs for "200" $key-frames$, the computed mean and standard deviation reached "0.905" and "0.216", respectively. The increasing number of groups has the reverse effect on the pedestrians' mean velocity in the crowd. This is because increasing the pedestrians normally slow down the pedestrians' movement. In our simulations, pedestrians change their position and velocity based on various steering behaviours. In each $key-frame$, the new position of each pedestrian is calculated based on the proposed method. If a pedestrian does not interfere with existing obstacles or pedestrians, then the pedestrian's current position will update to the new determined position. If, on the other hand, a pedestrian interferes with obstacles in the scene or pedestrians, the proposed method finds 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 Figs. 10, 11, and 12.

Figure 10 demonstrates the situation when a pedestrian tries to pass an obstacle (the stairs). At this point, the pedestrian checks the path for collision with obstacles. If the pedestrian were to move along the current direction, it would collide with the obstacle, as shown in the figure. Therefore, the pedestrian changes his direction of motion accordingly. The *BNM* method is employed to find a collision-free path and guide the pedestrian to turn left and then right to overcome the obstacle. Afterwards, the pedestrian moves undisturbed to reach its destination point. Similar actions are taken when two or more pedestrians are in danger of colliding. An illustrative example of combined obstacle and pedestrian avoidance is shown in Figs. 11 and 12. In this example, we consider a group of two pedestrians moving forward to their goal point. Firstly, at $key-frame$ 94, as shown in Fig. 11*a*, if the pedestrians moved along the current direction, they would collide with the obstacle (a vase) because the obstacle blocks the path. Therefore, the proposed method calculates their new direction of motion as shown in Fig. 11*b*&*c* at $key-frame$ 95 and 96.

Secondly, after the pedestrians pass the obstacle, a collision may happen between pedestrians at the $key-frame$ 97. When the distance between pedestrians goes below

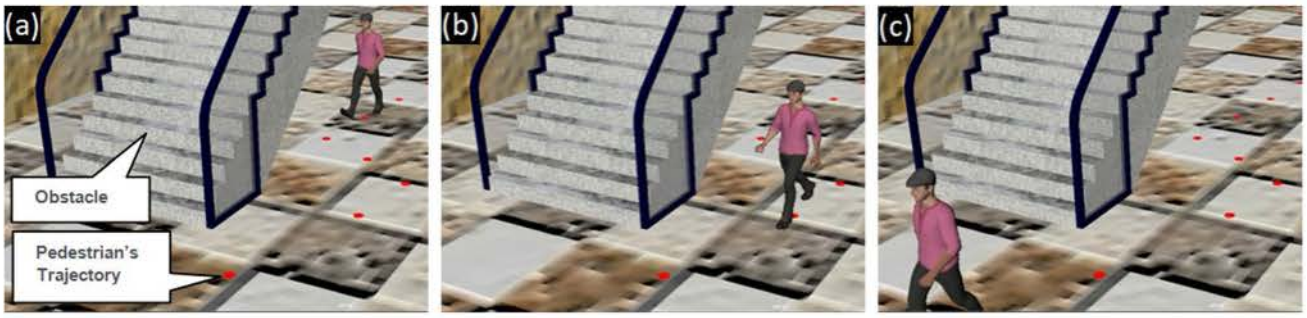

**Fig. 10** Screenshot of the simulation: shows a single pedestrian trying to move through the walking area toward the goal point, and he starts to change his motion direction near the safety area around the stairs at $key - frames$ (***a***) 118, (***b***) 122, and (***c***) 124

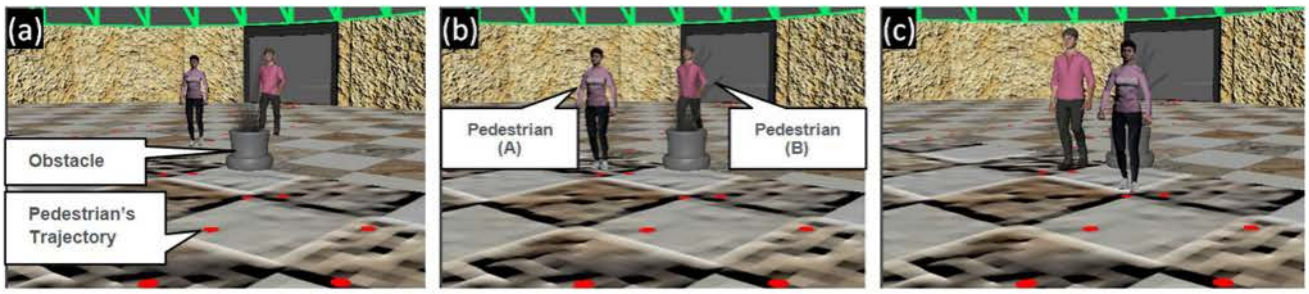

**Fig. 11** Screenshot of the simulation: shows a group of pedestrians, consists of two pedestrians (pedestrian *A* and pedestrian *B*), using the obstacle avoidance method to pass the obstacle. Afterwards, pedestrians keep moving toward their desired goal points: (***a***) collision avoidance between 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 $keyfram = 96$

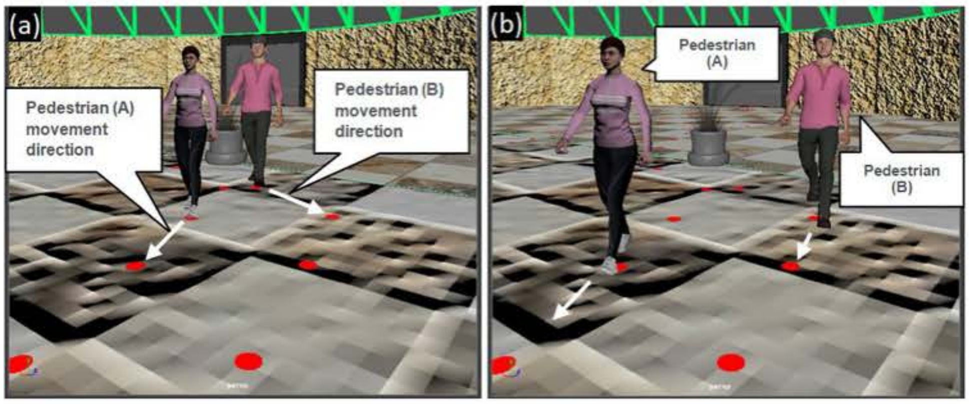
(a)
(b)
Pedestrian (A) movement direction
Pedestrian (B) movement direction
Pedestrian (A)
Pedestrian (B)

**Fig. 12** Screenshot 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 $key - frame$ 97. (***b***) Pedestrian *B* changes the motion direction to avoid pedestrian *A* without any collision at $key - frame$ 98, and then they keep moving to reach their goal point

$2 \times r$, then pedestrian (*B*) employs the avoidance method to resolve this situation (see Fig. 12*a*). Based on this method, pedestrian (*B*) has to change his motion direction to the right to avoid collision with its neighbouring pedestrian (*A*). Then pedestrians (*A*) and (*B*) keep moving at $key - frame$ 98 to reach their destination point, as shown in Fig. 12*b*. 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 pedestrians can avoid collisions and reach their goal points by using the proposed method.

In order to illustrate the pedestrian's movement in the simulated environment, different screenshot views of a simple illustrative example is presented in Fig. 14 at different stages of the simulation, as shown in Fig. 13. A pedestrian (we consider a group that consists of one pedestrian)

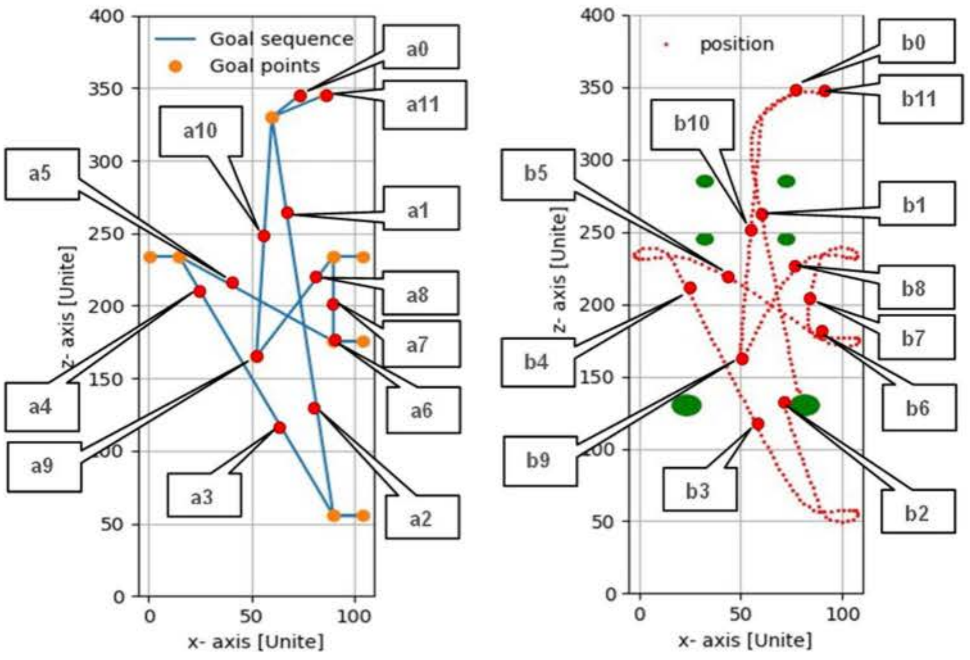

**Fig. 13** Pedestrian's trajectory: the graph shows pedestrians' movement in different directions in the simulated walking area at different *keyframes*: 65, 100, 145, 165, 200, 230, 240, 270, 287, and 310. The red circle objects represent the pedestrians' trace in the virtual environment at each key-frame. 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 *keyframes*

activates to move to enter the walking area at a randomly created key-frame, as shown in Figs. 14*a*, 13 left ($a0$) and 13 right ($b0$). Afterwards, the pedestrian enters the walking area through the entrance at key-frame "65", as illustrated in Figs. 14*a*, 13 left ($a1$) and 13 right ($b1$), then pass through the walking area to reach the goal points while avoiding obstacles. The simulation results in Figs. 14*b*, 13 left ($a2$) and 13 right ($b2$), present the time that the pedestrian comes closer to an obstacle at key-frame "100", where the pedestrian turns to the left to pass the obstacle without collision. As the pedestrian reaches the first goal point, the current goal point updates to the second goal point in the goal point list. Then the pedestrian moves toward the new goal point, as shown in Fig. 14*c*, 13 left ($a3$) and 13 right ($b3$) at key-frame "145". The pedestrian keeps changing the motion direction in the walking area to determine a collision-free path and track the goal points. The simulation results at several locations at key-frame "165", "200", "230", "240", and "270" are presented in Figs. 14$d \rightarrow h$, 13 left ($a4 \rightarrow a8$) and 13 right ($b4 \rightarrow b8$). In this simulation, the pedestrian visits the service point in the middle of the simulated environment, as shown in Fig. 14*i*, 13 left ($a9$) and 13 right ($b9$) at key-frame "287". Subsequently, the pedestrian moves toward the entrance/exit to leave the working environment, as illustrated in Figs. 14*j*, 13 left ($a10$) and 13 right ($b10$) at key-frame "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 Figs. 13 left ($a11$) and 13 right ($b11$).

The performance of our implementation was evaluated by simulating pedestrian movements in the virtual environment with a different number of groups (from $1 \rightarrow 10$ *groups*) and different group sizes. For each pedestrian in each group, the proposed method is used to generate pedestrians' routes as shown in Fig. 1($a$). We performed "200" independent runs for each simulated scenario for "200" *frames*. At each independent run, the average values for the number of pedestrians and the total computational time required to generate the pedestrians' route are calculated. The obtained results are presented in Fig. 15. The results demonstrate that the average value of the computational time is roughly linear, not increasing significantly with increasing the number of pedestrians in the scene. Moreover, the obtained results show that the proposed method can generate the routes for pedestrians navigating in the virtual environment to visit several destination points within a reasonable computational time.

## 6 Conclusions

This study develops a new method for simulating pedestrian crowd movement in a virtual environment. We have considered a public space such as a virtual working environment. A large number of virtual pedestrians of

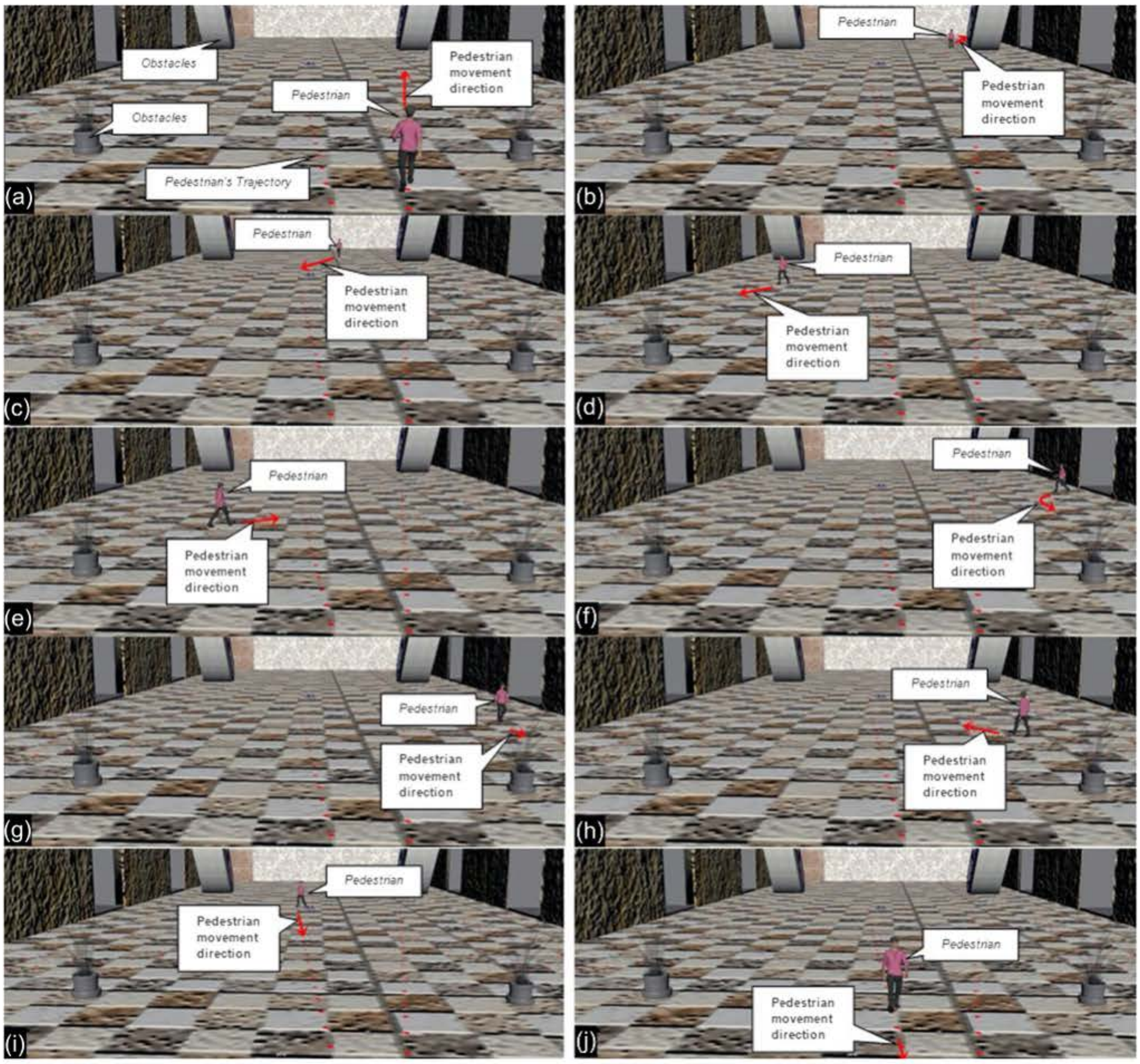

**Fig. 14** Screenshot of simulation at different stages: pedestrian' trajectory in the simulated walking area at different *keyframes*: (***a***) 65, (***b***) 100, (***c***) 145, (***d***) 165, (***e***) 200, (***f***) 230, (***g***) 240, (***h***) 270, (***i***) 287, and (***j***) 310. The red circle objects represent the pedestrians' trace in the virtual environment at each *keyframe*

different sizes and types move as groups inside the virtual environment. Each group follows different directions to reach their goal points. The proposed method generates a real-time route for each pedestrian walking in the virtual environment. The developed method demonstrates how the pedestrians adapt their dynamics to avoid static obstacles and other pedestrians. Based on our method, each pedestrian in each group constantly adjust their paths and parameters in real-time. The obtained results show that the developed method performs well to generate the pedestrian crowd movement with "200" different small groups of pedestrians in a given virtual environment. After 200 *keyframes*, we obtained "4235" values for pedestrian velocity, where the maximum value of the pedestrian's velocity reaches 1.14 *units/timestep*. The mean and the standard deviation of the calculated velocity are equal to 0.905 *units/timestep* and 0.216, respectively. We concluded that the increasing number of groups of pedestrians has the reverse effect on the pedestrian mean velocity in the crowd.

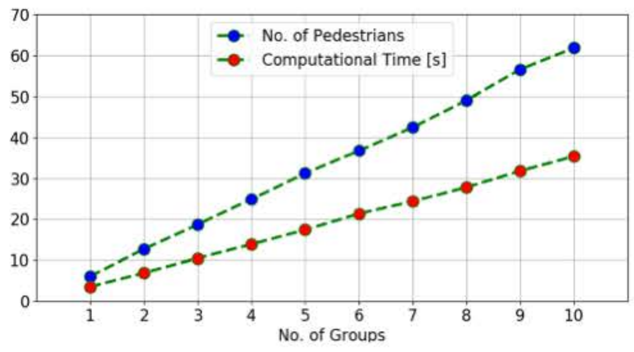

**Fig. 15** The average value of the number of pedestrians and the computational time required to find the routes for pedestrians in each group

**Funding** Open access funding provided by Libera Università di Bolzano within the CRUI-CARE Agreement.

## Abbreviations

| | |
|---|---|
| *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 3*D* 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 |
| $Position_{pe}$ | pedestrian's position |
| $Velocity_{pe}$ | pedestrian's velocity |
| $Energy_{pe}$ | 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* |

## References

1. Ali S, Nishino K, Manocha D, Shah M (2013) Modeling, simulation and visual analysis of crowds: a multidisciplinary perspective. In: Modeling, simulation and visual analysis of crowds. Springer, pp 1–19
2. Guy SJ (2012) Geometric collision avoidance for heterogeneous crowd simulation
3. Narain R, Golas A, Curtis S, Lin MC (2009) Aggregate dynamics for dense crowd simulation. In: ACM SIGGRAPH Asia 2009 papers. pp 1–8
4. Hughes RL (2003) The flow of human crowds. Annual Review of Fluid Mechanics 35(1):169–182
5. Izadinia H, Saleemi I, Li W, Shah M (2012) 2 t: multiple people multiple parts tracker. In: European conference on computer vision. Springer, pp 100–114
6. Lim CK, Tan KL, Zaidan AA, Zaidan BB (2020) 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
7. Durupinar F, Pelechano N, Allbeck J, Gudukbay U, Badler NI (2009) How the ocean personality model affects the perception of crowds. IEEE Computer Graphics and Applications 31(3):22–31
8. Guy SJ, Kim S, Lin MC, Manocha D (2011) Simulating heterogeneous crowd behaviors using personality trait theory. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation. pp 43–52
9. Degond P, Appert-Rolland C, Moussaid M, Pettré J, Theraulaz G (2013) A hierarchy of heuristic-based models of crowd dynamics. Journal of Statistical Physics 152(6):1033–1068

10. Piccoli B, Tosin A (2009) Pedestrian flows in bounded domains with obstacles. Continuum Mechanics and Thermodynamics 21(2):85–107
11. Treuille A, Cooper S, Popović Z (2006) Continuum crowds. ACM Transactions on Graphics (TOG) 25(3):1160–1168
12. Etikyala R, Göttlich S, Klar A, Tiwari S (2014) Particle methods for pedestrian flow models: From microscopic to nonlocal continuum models. Mathematical Models and Methods in Applied Sciences 24(12):2503–2523
13. Hughes RL (2002) A continuum theory for the flow of pedestrians. Transportation Research Part B: Methodological 36(6):507–535
14. Shao W, Terzopoulos D (2007) Autonomous pedestrians. Graphical Models 69(5–6):246–274
15. Pelechano N, Allbeck JM, Badler NI (2007) Controlling individual agents in high-density crowd simulation
16. Kim S, Guy SJ, Manocha D, Lin MC (2012) Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In: Proceedings of the ACM SIGGRAPH symposium on interactive 3D graphics and games. pp 55–62
17. Sarmady S, Haron F, Talib AZH (2009) Modeling groups of pedestrians in least effort crowd movements using cellular automata. In: 2009 Third Asia international conference on modelling & simulation. IEEE, pp 520–525
18. Cheng L, Reddy V, Fookes C, Yarlagadda PK (2014) 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, vol 2. IEEE, pp 161–167
19. Manenti L, Manzoni S (2011) Crystals of crowd: Modelling pedestrian groups using mas-based approach. In : WOA. pp 51–57
20. Mahato NK, Klar A, Tiwari S (2018) Particle methods for multi-group pedestrian flow. Applied Mathematical Modelling 53:447–461
21. Yang S, Li T, Gong X, Peng B, Hu J (2020) A review on crowd simulation and modeling. Graphical Models 111:101081
22. Reynolds CW (1999) Steering behaviors for autonomous characters. In: Game developers conference, vol 1999. Citeseer, pp 763–782
23. Patil S, Van Den Berg J, Curtis S, Lin MC, Manocha D (2010) Directing crowd simulations using navigation fields. IEEE Transactions on Visualization and Computer Graphics 17(2):244–254
24. Sud A, Andersen E, Curtis S, Lin M, Manocha D (2007) Real-time path planning for virtual agents in dynamic environments. In: 2007 IEEE virtual reality conference. IEEE, pp 91–98
25. Helbing D, Farkas IJ, Molnar P, Vicsek T (2002) Simulation of pedestrian crowds in normal and evacuation situations. Pedestrian and Evacuation Dynamics 21(2):21–58
26. Epstein JM (2014) AgentZero: Toward Neurocognitive foundations for generative social sciences. Princeton University Press, Princeton
27. Smaldino PE, Epstein JM (2015) Social conformity despite individual preferences for distinctiveness. Royal Society Open Science 2:140437
28. Helbing D (1992) A fluid-dynamic model for the movement of pedestrians. Complex Systems 6(5):391–415
29. Helbing D, Molnar P (1995) Social force model for pedesrtian dynamics. Physical Review E 51(5):4282–4286
30. Helbing D (2010) Quantitative sociodynamics: Stochastic methods and models of social interaction processes. Springer, New York
31. Autodesk (2018) Maya. https://www.autodesk.com/maya/. Accessed June 2019
32. Saeed RA, Recupero DR, Remagnino P (2020) A boundary node method for path planning of mobile robots. Robotics and Autonomous Systems 123:103320
33. Saeed RA, Recupero DR (2019) Path planning of a mobile robot in grid space using boundary node method. Proceedings of the 16th international conference on informatics in control, automation and robotics, ICINCO 2019, vol 2. pp 159–166
34. Reynolds CW (1987) Flocks, herds and schools: A distributed behavioral model. In: Proceedings of the 14th annual conference on Computer graphics and interactive techniques, pp 25–34
35. Adobe (2018) Mixamo. https://www.mixamo.com/. Accessed June 2019