Computational Visual Media
DOI 10.1007/s41095-017-0081-9 Vol. 3, No. 3, September 2017, 243–261

Research Article

Optimized evacuation route based on crowd simulation

Sai-Keung Wong[1](✉), **Yu-Shuen Wang**[1], **Pao-Kun Tang**[1], **and Tsung-Yu Tsai**[1]

Abstract An evacuation plan helps people move away from an area or a building. To assist rapid evacuation, we present an algorithm to compute the optimal route for each local region. The idea is to reduce congestion and maximize the number of evacuees arriving at exits in each time span. Our system considers crowd distribution, exit locations, and corridor widths when determining optimal routes. It also simulates crowd movements during route optimization. As a basis, we expect that neighboring crowds who take different evacuation routes should arrive at respective exits at nearly the same time. If this is not the case, our system updates the routes of the slower crowds. As crowd simulation is non-linear, the optimal route is computed in an iterative manner. The system repeats until an optimal state is achieved. In addition to directly computing optimal routes for a situation, our system allows the structure of the situation to be decomposed, and determines the routes in a hierarchical manner. This strategy not only reduces the computational cost but also enables crowds in different regions to evacuate with different priorities. Experimental results, with visualizations, demonstrate the feasibility of our evacuation route optimization method.

Keywords crowd simulation; evacuation; path planning; optimization

1 Introduction

Crowd evacuation is important in building design and city planning. A well-designed evacuation plan contains effective evacuation guidance that can lead people to exits in a short period of time. In this paper, we present an approach to determine optimal evacuation routes taking into account crowd information and positions of obstacles. As crowd distributions can be estimated for some events such as concerts and new year celebrations, evacuation direction signs can be positioned before crowds gather at a place. Furthermore, as surveillance cameras and sensor networks have become more widespread recently, dynamic estimation of a crowd distribution allows our system to dynamically respond, improving its effectiveness.

Mathematical models have been given for computing optimal routes [1]. These models represent an environment as a graph consisting of a set of nodes and a set of edges, which represent intersections and roads, respectively. Most of these models assume a single movement direction of a crowd along the graph edges. They also require crowd movement speed and road capacities when computing an evacuation route. However, crowd speed varies over time, and it depends on the attributes of the evacuees and the degree of crowd congestion. Without a physical simulation that computes the movement of each evacuee, it is difficult to tell whether an evacuation plan is effective. For example, an old person may move more slowly than a young person, evacuees in a group tend to move together to exits, and a crowd has difficulty when joining a main route from a branch route. Such phenomena can only be modelled and observed using crowd simulation. Furthermore, rendering believable animated characters for crowd evacuation is also crucial.

Our goal is to compute an optimal route for crowd evacuation. Given the attributes and distribution of a crowd, exits, road widths, and obstacles, our system determines the evacuation direction for each

1 "National Chiao Tung University", Taiwan, China. E-mail: S.-K. Wong, cswingo@cs.nctu.edu.tw (✉); Y.-S. Wang, yushuen@cs.nctu.edu.tw; P.-K. Tang, tangtony98@hotmail.com; T.-Y. Tsai, disneyanduj@gmail.com.

Manuscript received: 2016-08-28; accepted: 2017-03-10

road to enable evacuation within a short time span. To achieve this aim, we represent the road network as a situation map using a graph, and initially set the evacuation direction of each edge based on a shortest path algorithm. These directions are then iteratively updated according to the results of simulation. Note that the crowd on a road could divide and evacuate in opposite directions. We call a place where this phenomenon occurs a *division point*. Our goal is to ensure that evacuees near a division point take similar time to arrive at respective exits. As the initial route is generally not optimal, our system updates the division points iteratively until all of them become stable. In addition, our approach is adaptive to the results of crowd simulation. It can handle transportation such as buses and elevators. That is, a crowd of people can move to a specific position, wait, and only a portion of them can leave at discrete time. Experimental results demonstrate that a crowd can evacuate more quickly using our optimized routes than those provided by shortest path and weighted graph algorithms. A preliminary version of this work was published in Ref. [2].

2 Related work

2.1 Evacuation

Early evacuation methods were often based on mathematical models. They can be classified into approaches based on static networks, discrete or continuous time dynamic networks, and simple simulation (probabilistic or cellular automata based). We refer readers to the surveys [1, 3] for more details.

Dressler et al. [4] applied network flow techniques to select exits. Hadzic et al. [5] focused on evacuation in buildings on fire. Their model is an extension of a dynamic network flow. Regions that are not reachable are removed from the graph so that the routes determined will not lead people to those regions. Abdelghany et al. [6] employed a genetic algorithm to search for evacuation routes in an area in which a crowd is uniformly distributed. Wang et al. [7] determined evacuation plans by considering the speed of spread of smoke and carbon monoxide in fires. Desmet and Gelenbe [8] adopted a flow-based method to compute evacuation routes and used dynamic exit signs to guide evacuees. Their method is intended for low-headcount evacuations; it does not simulate the movements of the evacuees and their interactions. Tang et al. [9] proposed a pseudo-polynomial-time dynamic programming algorithm to compute evacuation routes for a three-dimensional network. Berseth et al. [10] conducted a study to evaluate how the placement of pillars and doors affects pedestrian flows during evacuation of a building; a similar study was presented in Ref. [11]. Haworth et al. [12] optimized level of service (e.g., crowd density) to improve crowd evacuation. Moussaïd et al. [13] studied movement patterns of crowds in an immersive 3D virtual environment, in which the crowds were controlled by real people.

Given a building and a crowd of agents, Thompson and Marchant [14] estimated the travel time of each agent and rendered a distance map to depict whether a building is well designed. Lovas [15] introduced several measures based on reliability theory to evaluate the importance of escapeway elements. Pelechano and Badler [16] placed *leaders*, persons familiar with the layout, at intersections, to guide evacuees to safety. Ma et al. [17] showed that a few leaders could effectively guide evacuees but leaders may also have a negative impact. Dimakis et al. [18] employed a distributed simulator based on the multi-agent paradigm for managing an emergency. Rodriguez and Amato [19] studied how the types of interaction affect evacuation strategies. Tsai et al. [20] developed a multi-agent evacuation simulation which could support several kinds of agent representing, e.g., children, parents, and persons in authority. Kretz et al. [21] proposed a method to compute the quickest path for each agent using a local simulation. Given that the path is determined according to the information in the current state, congestion potentially occurs in later stages. Agents may move back and forth during evacuation. In contrast, our system computes the evacuation route for each agent using a full simulation, allowing it to achieve a stable result.

When routes are determined, they can be conveyed by wireless devices to guide the evacuees. Inoue et al. [22] suggested the use of mobile phone for navigation. Chen [23] adopted a wireless supervision system to control indicator boards which give directions for evacuation. Such strategies could be integrated into our system to provide a practical tool.

2.2 Crowd simulation

Helbing and Molnár [24] introduced a social force model to simulate pedestrian dynamics. Relative distances between agents are preserved to prevent collisions. To simulate human behavior, van den Berg et al. [25] presented a reciprocal velocity obstacle method to compute feasible movement directions for agents. Karamouzas et al. [26] introduced the idea of predicting future collisions to adjust the movement direction of each agent, to add realism. Using the principle of least effort, the method presented by Guy et al. [27] computes a biomechanically energy-efficient and collision-free trajectory for a crowd.

Collision handling potentially results in deadlock. Agents that collide with each other are likely to collide again in the next time step. To prevent this problem, Curtis et al. [28] set different priorities to agents and arranged that agents with lower priorities would give way to those with higher priorities. Johansson et al. [29] applied evolutionary optimization to compute the parameters used in a social force model, using real videos. The method can be used to calibrate crowd models and group models [30]. Durupınar et al. [31] used psychological parameters to form different crowd models. Such models can be employed to enhance the realism of crowd simulation. Boatright et al. [32] defined steering contexts (e.g., groups crossing, chaos) and adopted machine learning techniques to capture the main characteristics of crowds in each steering context. Li and Wong [33] employed local views of agents and crowd movement to determine the movements of pedestrians. Kapadia et al. [34] presented a variety of techniques to enhance the realism of crowd simulation, including sound perception, multi-sense attention, and understanding of environment semantics.

2.3 Road maps

Maps can be used to guide a crowd to destinations. Accordingly, given a situation layout, one goal is to compute a road map that comprises feasible paths to guide agents. The method presented by Geraerts et al. [35] creates a set of collision free corridors from a situation with static obstacles. These connected corridors can be considered to form a graph so that path finding techniques can be used to determined a route for evacuation. To further improve the approach, van Toll et al. [36] computed a density map based on the distribution of agents. Their method then guides agents to alternative paths to avoid congestion. Makni [37] introduced a hierarchical path planning algorithm. He applied the constrained Delaunay triangulation [38] to represent a scene map and find paths through adjacent triangles. Pettré et al. [39] computed the map from multi-layered and uneven terrains. Bayazit et al. [40] applied a global road map to achieve sophisticated flocking behaviors. Patil et al. [41] computed navigation fields to guide different crowds to their own destinations. Wong et al. [42] used particle swarm optimization to refine predefined evacuation paths based on a fitness function. Our proposed method relies on guidance paths to guide the agent movement.

3 Theory and system overview

We adopt crowd simulation techniques to optimize evacuation routes. A crowd of people is represented by agents and their motions are simulated to determine the time needed for evacuation, i.e., the time taken by the agent to move from its initial position to an exit. Our method is versatile and can deal with different structures for the situation, obstacle positions, and crowd distributions. To determine an optimal evacuation route, the goal is to maximize the rate of agents arriving at the exits in every time period. This requirement can be achieved by expecting the evacuation time of neighbouring agents, even those who evacuate in different ways, to be similar.

We show an experiment in Fig. 1, in which the road to the left exit is narrow, to explain why simulation is important in evacuation route optimization. The evacuation plan determined by the shortest path algorithm is oblivious to the distribution of agents. A good evacuation plan for 1000 uniformly distributed agents may not work well for 100 agents because the congestion problems are different. Instead, our approach is to refine the position of each division point so that the evacuation time of agents around it are similar. To implement this idea, we initially

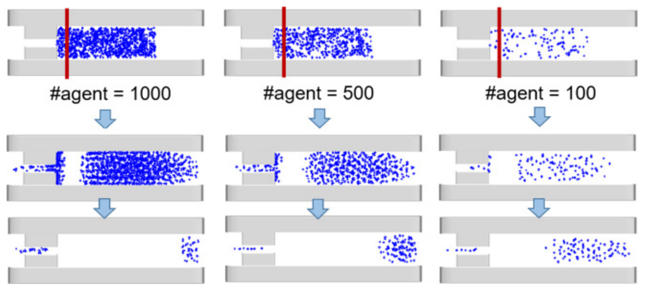


Fig. 1 Optimal evacuation route depends on not only the structure of the situation but also the distribution of people. Simulated agents (blue dots) evacuate to the left or right exit according to the guidance from a shortest path algorithm (weighted by road capacity). The division point (vertical red line) should be updated when the distribution of agents changes: if it is not, the agents do not reach the exits optimally.

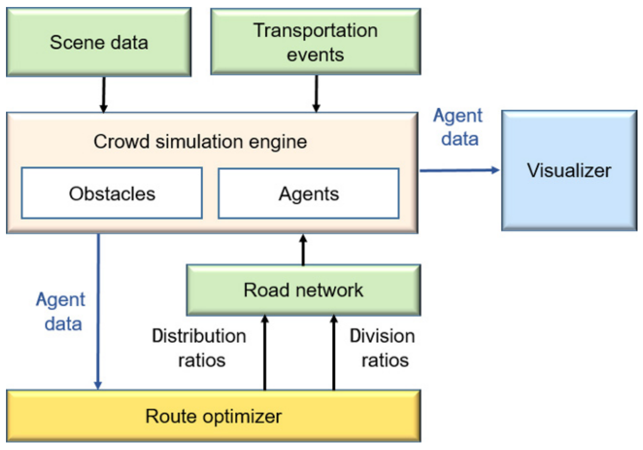


Fig. 2 System architecture.

set evacuation routes according to a shortest path algorithm and then iteratively update the routes until an optimal state is achieved. We also prevent head-on collision of agents when computing the route to prevent rapid increases in evacuation time. Thus, the following constraints should be satisfied:

1. Each edge has at most one division point to divide a crowd into two groups. Agents in a group evacuate in a single direction, while those in different groups go in separate directions.
2. Guidance paths at an intersection cannot across. Otherwise, head-on collisions potentially occur, causing evacuees to take a long time to arrive at the exits.

The input to our system is a scenario map that indicates the positions of obstacles and roads. We transform the map into a graph $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{E})$, where $\boldsymbol{N} = \{\boldsymbol{n}_1, \ldots, \boldsymbol{n}_z\}$ is the set of node positions, $\boldsymbol{n} \in \mathbb{R}^2$, z is the number of nodes, and $\{i, j\} \in \boldsymbol{E}$ indicates node connectivity. We also denote by $\boldsymbol{n}_\mathrm{e} \in \boldsymbol{N}$ the exits that agents should reach. Our goal is to compute the evacuation direction(s) of each edge and the distribution ratios of each node. When agents on an edge should evacuate in opposite directions, we also determine the position of the division point that minimizes the evacuation time.

Figure 2 shows our system architecture. It has three major components: route optimizer, crowd simulation engine, and visualizer. The route optimizer updates the evacuation routes. The crowd simulation engine executes crowd simulation based on the routes, and reports the simulation results. The visualizer renders the agents in different ways, e.g., agents as dots, agents color-coded according to evacuation time, or agents as 3D models.

4 Evacuation route optimization

We initialize the evacuation routes according to the distances of agents to exits and then refine the routes by considering the agents' evacuation time. An optimal state is achieved when neighboring agents who take different routes arrive at their respective exits at similar time.

4.1 Route initialization

We apply a shortest path algorithm to initialize the evacuation routes. To implement this idea, we add to graph $\boldsymbol{G}$ an auxiliary node $\boldsymbol{m}$ connected to all exits $\boldsymbol{n}_\mathrm{e} \in \boldsymbol{N}_\mathrm{e}$. The distances between $\boldsymbol{m}$ and $\boldsymbol{n}_\mathrm{e}$ are set to zero. The cost of an edge $\{i, j\}$ is equal to the edge length:

$$D_{ij} = \|\boldsymbol{n}_i - \boldsymbol{n}_j\| \tag{1}$$

By applying the Dijkstra's algorithm to compute the shortest path tree, we obtain the initial evacuation routes. Note that no cyclic paths can appear as D_{ij} must be positive. A weighted graph algorithm considering edge capacity can be used by setting $D_{ij} = \|\boldsymbol{n}_i - \boldsymbol{n}_j\|/w_{ij}$, where w_{ij} is the minimal width of the road modelled by edge $\{i, j\}$. However, if the edge capacity is not fully utilized, the initial route should also be adjusted.

We set the evacuation direction of each edge according to the shortest path tree. The directions are used to guide the agents' evacuation in the simulation process. For an edge $\{i, j\}$ that is in $\boldsymbol{E}$ but not on the shortest path tree, we set a division point on it because the paths from $\boldsymbol{n}_i$ to $\boldsymbol{m}$ and from

$\boldsymbol{n}_j$ to $\boldsymbol{m}$ do not cross edge $\{i, j\}$. Ideally, the division point should be set based on the distances from nodes $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ to $\boldsymbol{m}$, respectively. But we simply set the point to the middle of $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ as the current route is only used for initialization and will be updated according to the simulation results.

4.2 Route optimization

We have assigned the initial evacuation directions based on the shortest path algorithm. A crowd can move along the evacuation directions to the exits. However, this approach does not consider the actual movements of agents and congestion problems. Therefore, we apply crowd simulation to estimate the evacuation time of each agent. The evacuation time of each local region is then determined as well. The results are used to perform an iterative refinement process in which we (i) refine the division points on edges and (ii) determine the distribution ratios at intersections, to solve the abovementioned problem. The refinement process stops when the system converges or a predefined maximum number of generations is reached. Experimental tests suggest a suitable value for this number is 10.

4.2.1 Refinement of division points

Let $\{i, j\}$ be an edge with a division point q. Our goal is to refine the position of q (denoted by $\boldsymbol{n}_q$) to ensure that the crowds that evacuate in opposite directions from this point arrive at their respective exits simultaneously. The evacuation time at a node on an edge is set to the average evacuation time of the agents at that node. We now derive the formula for updating a division point after each crowd simulation, based on the evacuation time of agents moving along an edge.

Let T_i and T_j be the evacuation time if the two crowds start evacuation from nodes i and j, respectively. Let ΔT_{qi} and ΔT_{qj} be the evacuation time that they take to move from $\boldsymbol{n}_q$ to $\boldsymbol{n}_i$ and

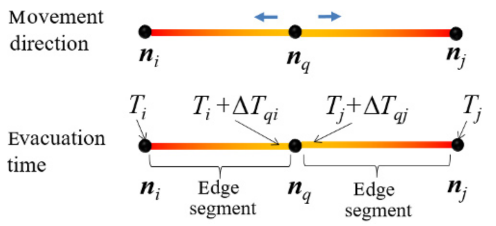


Fig. 3 Evacuation time of agents along an edge divided into two parts at a division point q. The agents are divided into two groups which move in opposite directions.

$\boldsymbol{n}_j$, respectively (see Fig. 3). The division point separates the crowd into two smaller crowds moving in opposite directions. So that the two crowds moving from the division point should arrive at respective exits at the same time, we get the following constraint:

$$T_i + \Delta T_{qi} = T_j + \Delta T_{qj} \tag{2}$$

Typically, T_i and T_j are difficult to estimate and should be determined by simulation. Furthermore, as crowds are seldom congested when they start evacuation, ΔT_{qi} and ΔT_{qj} can be linearly proportional to the distance from $\boldsymbol{n}_q$ to an end of $\{i, j\}$. Let the division ratio $\phi = \|\boldsymbol{n}_i - \boldsymbol{n}_q\| / \|\boldsymbol{n}_i - \boldsymbol{n}_j\|$. After the division point q is updated, the new ϕ' should satisfy the constraint in Eq. (2), so we have:

$$T_i + \frac{\phi'}{\phi}\Delta T_{qi} = T_j + \frac{1-\phi'}{1-\phi}\Delta T_{qj} \tag{3}$$

Here, ϕ'/ϕ and $(1-\phi')/(1-\phi)$ are the changes in the distance ratios in the two opposite directions respectively. Therefore, $(\phi'/\phi)\Delta T_{qi}$ and $(1-\phi')/(1-\phi)\Delta T_{qj}$ are the extra amounts of time for agents moving from the division point to nodes i and j respectively. If the division ratio does not change, we need not update q. Otherwise, given the current state of T_i, T_j, ΔT_{qi}, and ΔT_{qj}, we aim to compute ϕ' to satisfy Eq. (3). Let $\alpha = \Delta T_{qi}/\phi$ and $\beta = \Delta T_{qj}/(1-\phi)$. Then Eq. (3) becomes:

$$T_i + \phi'\alpha = T_j + (1-\phi')\beta \tag{4}$$

Therefore, the division point q is updated using the new division ratio:

$$\phi' = \frac{T_j - T_i + \beta}{\alpha + \beta} \tag{5}$$

Our system refines the position of the division point q and the evacuation routes. If ϕ' is zero, then all agents move along edge $\{i, j\}$ towards node j, while if ϕ' is one, all agents move towards node i. ϕ' may lie outside the interval $[0, 1]$ which implies that the division point should not be on edge $\{i, j\}$. If the valence of node i or j is two, we propagate the division point to the adjacent edge to $\{i, j\}$ in the appropriate direction (see Fig. 4). Otherwise, we handle the problem during refinement of intersections.

4.2.2 Refinement at intersections

To evacuate efficiently, people have to make decisions when they arrive at an intersection. Incoming and outgoing edges are edges that agents move toward and away from an intersection. If an intersection

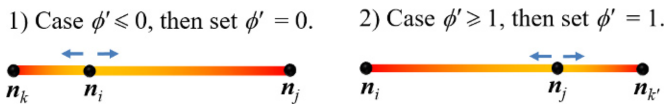

1) Case $\phi' \leqslant 0$, then set $\phi' = 0$.
n_k
n_i
n_j
2) Case $\phi' \geqslant 1$, then set $\phi' = 1$.
n_i
n_j
$n_{k'}$

Fig. 4 Division point propagation for nodes with valence two. Arrows indicate the agent movement direction. Case 1: if the new division ratio $\phi' \leqslant 0$, then it is possible that the agents move faster along edge $\{i, j\}$ than agents along edge $\{k, i\}$. Case 2: if $\phi' \geqslant 1$, then the agents may move faster along edge $\{i, j\}$ than agents along edge $\{j, k'\}$. In either case, the division point should be propagated from $\{i, j\}$ to $\{k, i\}$ or $\{j, k'\}$, respectively.

has at least one incoming and two outgoing edges, we determine distribution ratios of flows to help agents choose the correct evacuation route. We illustrate this idea in Fig. 5. Note that an intersection that has only incoming edges must be an exit. One that has only outgoing edges can be ignored because there are no decisions for agents to make: agents just move along the edges that are closest to them. In the following we consider agents moving along an incoming edge to node j and determine how they should move along the outgoing edge(s) at j. Let $E_{\mathrm{o}}(j)$ be the set of outgoing edges at node j, and let the distribution ratio of edge $e_k \in E_{\mathrm{o}}(j)$ be $\theta_k = D_k / \sum_{i \in E_{\mathrm{o}}(j)} D_i$, where $D_k = 1/T^{\mathrm{m}}_k$ and T^{m}_k is the maximum evacuation time of agents moving along outgoing edge $\{j, k\}$. The idea here is that when the agents reach node j, more agents move to those outgoing edges with lower evacuation time. In summary, the evacuation time of each agent is recorded and used to iteratively update the evacuation route. The division points on edges and distribution ratios at intersections are fine-tuned.

4.3 Hierarchical route optimization

The priority of evacuation in different zones may be different. For example, people may prefer to stay outside rather than inside a building for many reasons concerning fresh air and safety. Accordingly,

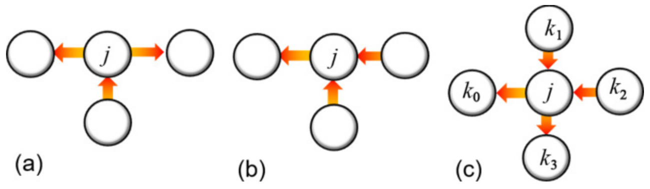


Fig. 5 Crowd flow at an intersection. Simulated agents may split (a) or merge (b) during evacuation. Head-on collision may occur at complex intersections (c). Suppose a crowd of agents attempts to move from k_1 to k_3 and another crowd moves from k_2 to k_0. Our system assigns agents priorities which are used to choose outgoing edges, to overcome this problem.

we present a hierarchical framework to compute evacuation routes in order to help people leave top-priority zones efficiently. Consider a situation with a park including a dome and an outdoor area, both of which are filled with people. When computing evacuation routes for these people, a straightforward approach would simply take the whole scenario into consideration. However, doing so, the computed route may cause people stay in the dome for a long time to minimize overall evacuation time. To prevent such problems, we partition the situation into two different zones and optimize the route hierarchically.

There are two steps: (i) compute the evacuation route for the agents in the dome, and (ii) compute the evacuation route for the agents in the park and the agents leaving from the dome. The first step is performed because of the higher priority of the dome. The optimization process is identical to the one mentioned previously. An additional task is to collect the records of arrival time and exit of each agent.

The first step gives the evacuation route for the agents inside the dome. Then we perform the second step. One can simply assume an exit of the dome to be a teletransporter. When using a crowd simulation to evaluate the routes, the agents inside the dome are popped up at the specific exits in the teletransporter based on the records from the first step. These agents then follow the guidance paths to exits when optimizing routes in the outdoor area. The agents in this area are treated as usual. In other words, the simulations in different zones are separate, but they are linked by the recorded arrival time of the agents in the first step.

Another advantage of this hierarchical route optimization is the reduction of computational costs. As simulations are expensive due to nonlinear crowd behavior and congestion problems, larger numbers of agents often result in much more computation. Therefore, when routes in different zones are optimized separately, the costs can be greatly reduced. The only drawback is that the global evacuation time of the agents becomes longer. However, as mentioned, the priorities of different zones may be different. Our approach allows people to evacuate early from zones that are more dangerous. Note that this hierarchical procedure is not limited to two levels but can be extended to multiple disjoint zones.

5 Crowd simulation

Our evacuation route optimization is driven by a simple physics-based crowd simulation. The crowd simulation model enables the agents to move faster (e.g., run) if there is space for them to do so. We want to emphasize that our method for computing the evacuation route could be used with any other crowd simulation model, e.g., Refs. [24, 25, 33].

Following Ref. [43], the attributes of an agent in our system include position $\boldsymbol{p}_i$, walking speed s_i, and body radius r_i. We represent each agent by a circle, although an ellipse would be better, as an agent needs some space to turn around. Using anthropometric survey data [44], we set randomly the body diameter in the interval [0.3 m, 0.5 m] in our system. We also set the walking speed in the interval [$1\ \mathrm{m}\cdot\mathrm{s}^{-1}$, $1.5\ \mathrm{m}\cdot\mathrm{s}^{-1}$] based on experiments reported in Refs. [45, 46]. These parameters could be dynamically set using real data obtained from smart phones or tickets in an event, allowing the simulation results to be more accurate. Parameters for crowd simulation can also be extracted from crowd videos [47].

Our simulation process updates the position of each agent in a continuous space. However, to efficiently determine the neighbouring agents, the simulation proceeds in an organized manner. We divide the situation into a grid structure, in which the width of each grid cell is set to a value larger than $2\ell_{\mathrm{g}} + v_{\max}\Delta t$, where ℓ_{g} is the maximum radius of agents, $v_{\max}$ is the maximum speed of agents, and Δt is the simulation step size. Therefore, the neighboring agents to a specific agent can be found in an area of 3×3 grid cells.

5.1 Crowd motions

Our system simulates crowd motions according to the guidance of the evacuation routes, obstacles, and crowd density in neighboring regions. Agents move along edges while avoiding collisions until they arrive at exits.

5.1.1 Evacuation direction

To obtain the evacuation direction of each agent, we find the edge $\{i, j\} \in \boldsymbol{G}$ closest to the agent. The temporary destination $\boldsymbol{d} = \boldsymbol{n}_\delta$ is then computed, where $\delta \in \{i, j\}$ is set based on the evacuation direction of edge $\{i, j\}$. Let $\boldsymbol{p}$ be the agent position. Its movement direction is

$$\boldsymbol{v} = \frac{\boldsymbol{d} - \boldsymbol{p}}{\|\boldsymbol{d} - \boldsymbol{p}\|} \tag{6}$$

5.1.2 Spacious regions

People prefer to stay in spacious, less crowded regions so that they can leave more quickly. To prevent local congestion, we first compute a density map [36] for the situation to represent crowd densities and obstacles. The density of each grid cell is determined by the number of agents within it. The map is then applied to adjust the motion of each agent to steer them to more spacious areas: an agent should in the negative gradient direction if the agent's current speed is less than the agent's desired speed. For simplicity, suppose that an agent is located at the grid cell (x, y). The suggested movement direction for the agent is

$$\boldsymbol{v}_{\mathrm{d}} = \sum_{i=-1}^{1} \sum_{j=-1}^{1} (D_{x,y} - D_{x+i,y+j}) \boldsymbol{u}_{x,y}(i, j) \tag{7}$$

where $D_{x,y}$ is the density of grid cell (x, y), and $\boldsymbol{u}_{x,y}(i, j)$ is the relative unit direction of grid cell $(x + i, y + j)$ with respect to (x, y). $\boldsymbol{v}_{\mathrm{d}}$ is normalised to give $\tilde{\boldsymbol{v}}_{\mathrm{d}}$ if it is non-zero; otherwise, $\tilde{\boldsymbol{v}}_{\mathrm{d}}$ is a zero vector.

The velocity of each agent is adjusted to be $\boldsymbol{v}' = s(\boldsymbol{v} + w\tilde{\boldsymbol{v}}_{\mathrm{d}})$, where s is the walking speed of the agent, and w is a weight which controls the influence of the density map. We set $w = 0.25$ if $\boldsymbol{v}$ and $\tilde{\boldsymbol{v}}_{\mathrm{d}}$ point in the same general direction, i.e., $\boldsymbol{v} \cdot \tilde{\boldsymbol{v}}_{\mathrm{d}} > 0$. In this case, the agent can move faster. Otherwise, we set $w = 0$ as during evacuation, people would rather wait than go back if the route in front of them is congested. The new position of the agent becomes $\boldsymbol{p}' = \boldsymbol{p} + \boldsymbol{v}'\Delta t$.

5.2 Collision prevention

We update the positions of all agents in each time step. Given that agents are handled individually, they may collide with each other. A relaxation step is used to prevent this problem. Our system assumes that the agents are patient and give way to one another. Since agents cannot intersect the walls and other obstacles in a building, the one that is closer to an obstacle prefers to stay in position. To implement this idea, we consider each pair of agents at a given time and relax their positions iteratively. Suppose agents a and b potentially collide. Then we relax their positions by

$$\hat{\boldsymbol{p}}_a = \boldsymbol{p}'_a + w_a \boldsymbol{d}, \quad \hat{\boldsymbol{p}}_b = \boldsymbol{p}'_b - w_b \boldsymbol{d}$$
$$\text{with } \boldsymbol{d} = [(r_a + r_b) - \|\boldsymbol{p}'_a - \boldsymbol{p}'_b\|] \frac{\boldsymbol{p}'_a - \boldsymbol{p}'_b}{\|\boldsymbol{p}'_a - \boldsymbol{p}'_b\|} \tag{8}$$

where w_a and w_b are relaxation weights. We normally set $w_a = w_b = 0.5$ if both agents are not in the proximity of obstacles. If one of them is close to an obstacle, we allow this agent to stay at the same position by setting its weight to zero. If both agents are close to obstacles, the one farther from the destination gives way, and its weight is set to one.

In addition to handling collisions between agents, our system also prevents agents from colliding with obstacles. If the updated position $\boldsymbol{p}'$ is inside an obstacle, we restore the agent to its previous position and then move it a small step along the surface of the obstacle. If the collision still occurs, we iteratively reduce the step size until the problem is resolved.

Since the movement of each agent potentially results in new collisions, we repeat the two strategies to relax agent positions several times to avoid the problem.

6 Experimental results

6.1 Performance

We have implemented the above algorithm and ran it on a 3.20 GHz Intel Core i5 CPU with 8 GB RAM. The simulation step size was 1/30 s. To evaluate the performance of our method, we tested the presented system in various scenarios, including indoor and outdoor environments. In examples shown, agents and obstacles are represented by using dots and gray polygons, respectively. The colors of the dots indicate their evacuation time. Overall, our system takes eight generations on average to obtain an optimal evacuation route, as indicated in Fig. 6. Detailed timing statistics for our CPU-based implementation are given in Table 1. Clearly, the computational cost of our crowd simulation depends on the number of agents in a scene, but the cost can be greatly reduced by leveraging GPUs. For the example shown in Table 1, 35% of the crowd update cost is avoided if our hierarchical framework is employed.

6.2 Comparison

In the following, we compare the performance of the proposed method with two other methods: the shortest path algorithm and the weighted graph algorithm.

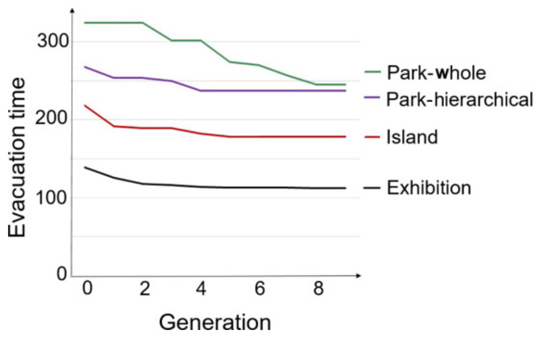


Fig. 6 Our system takes about eight generations to converge based on the evacuation time of the last agent.

Table 1 Timing. NUM: initial number of agents. ATC: average time per step of crowd update. ATG: average training time per generation

Scenario	NUM	ATC	ATG
Exhibition	25,000	14 ms	27 s
Island	20,000	10 ms	50 s
Park, hierarchical scheme	20,000	10 ms	83 s
Park, whole scene scheme	30,000	15 ms	76 s

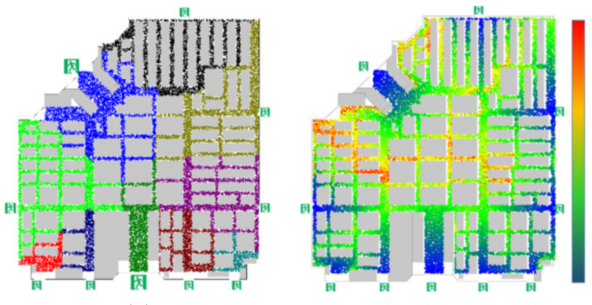
6.2.1 *Exhibition example*

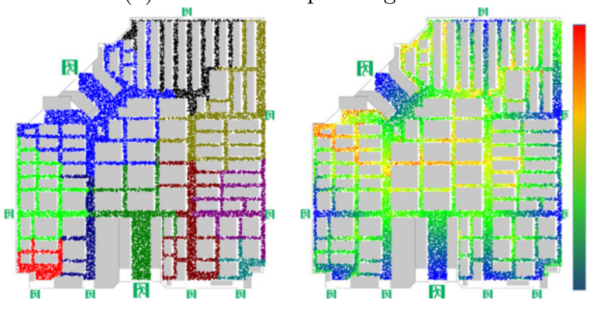
Figure 7 visualizes the evacuation time for agents that follow the routes determined by the shortest path algorithm, the weighted graph algorithm, and our method. In this example, 25,000 agents were evenly distributed in the scene. As the shortest path algorithm guides agents to the nearest exits, congestion potentially occurs. It often appears when the agents move from a branch to a main route because of collisions. The weighted graph algorithm guides agents to wide roads but congestion may occur on narrow roads. In contrast, while our approach may require a portion of the crowd to move to further exits, it disperses the crowd so as to avoid congestion, helping them to evacuate efficiently.

Figure 8 shows the numbers of unevacuated agents for the three methods as a function of time. At the beginning of the simulation, the numbers of remaining agents were similar for the three methods as similar numbers of agents reached the exits. As time went by, our method caused agents to utilize the exits more effectively. However, some exits were not utilized well in the other two methods: see Fig. 9.

6.2.2 *Island example*

In this example, 20,000 agents were unevenly distributed on an island according to the placement of facilities (Fig. 10). As there are three exits at


(a) The shortest path algorithm


(b) The weighted graph algorithm

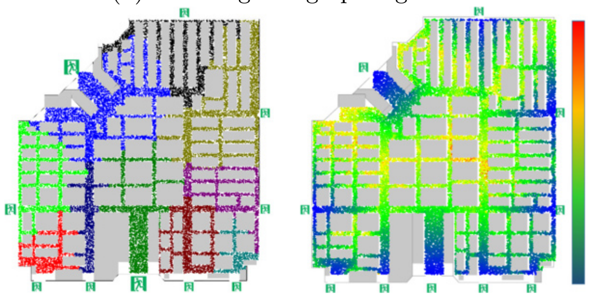

(c) The proposed method

Fig. 7 Exhibition example. We compare the results determined by: (a) the shortest path algorithm, (b) the weighted graph algorithm, and (c) our proposed method. Left: color indicates agents assigned to the same exit. Right: color indicates evacuation time of each agent. Agents who follow guidance determined by our method evacuate faster than in the other two cases.

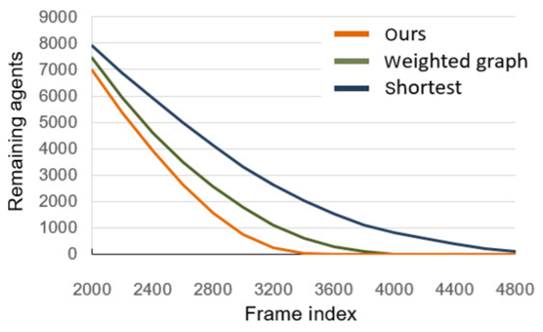


Fig. 8 Exhibition example. Unevacuated agents for the shortest path algorithm, the weighted graph algorithm, and our method.

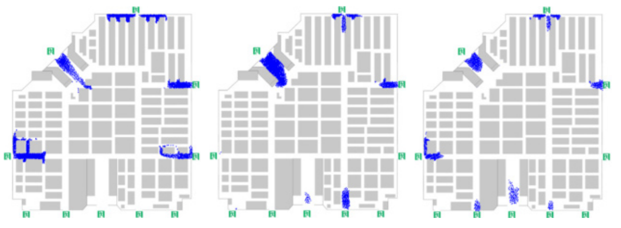


Fig. 9 Exhibition. Snapshots of unevacuated agents for each method at frame 2800. Left to right: the shortest path algorithm (4122 remaining), the weighted graph algorithm (2549 remaining), and our method (1559 remaining). The lower exits were not utilized well by the shortest path algorithm.

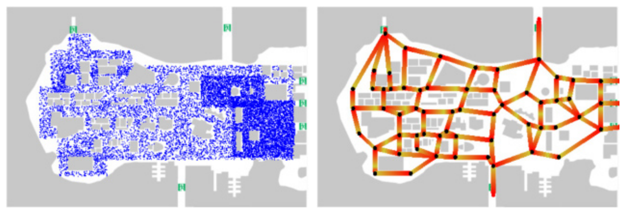
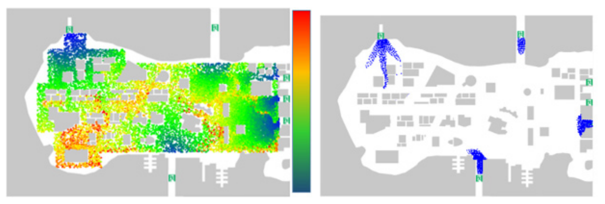
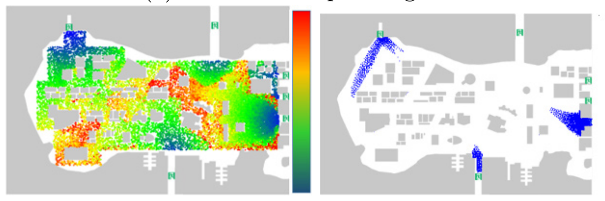


Fig. 10 Left: distribution of agents on the island. Right: road map.


(a) The shortest path algorithm


(b) The weighted graph algorithm

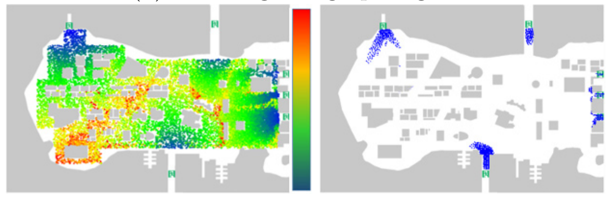
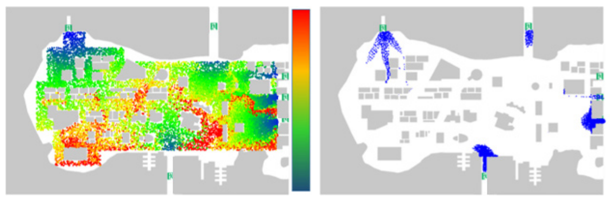
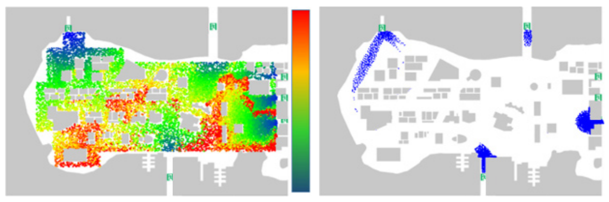

(c) The proposed method

Fig. 11 Island example. Left: distribution of the agent travel time. Right: remaining agents at frame 4500. Agents are moving to all six exits in our method, but some exits are no longer utilized in the other two methods.

the right of the island, most agents can evacuate efficiently. Figure 11 shows the distribution of agent travel time. The results determined by our proposed method and the other two methods are not greatly different. However, we still can see long queues at the top left at time step 4500 of the simulation for the other two methods (see Fig. 11(right)). In contrast, agents that follow our guidance are much closer to the exits at the same time step. We conducted another experiment in which the widths of the roads at the bottom and lower right exits were reduced by around one-half. Figure 12 shows the distribution of agent travel time. As the widths of two exits were reduced, it took longer for the agents to leave the island. Figure 13 shows how the numbers of remaining agents vary with time for each method. The shortest path algorithm does not adapt well to changes in road widths.

Figure 14 shows a 3D of the island. Such 3D animated agents help users to visualize and evaluate the realism of the entire evacuation process. This example shows that the agents are moving in a regular pattern as expected. In the future, we hope to conduct a user study in a virtual reality environment to assess the result quality.


(a) The shortest path algorithm


(b) The weighted graph algorithm

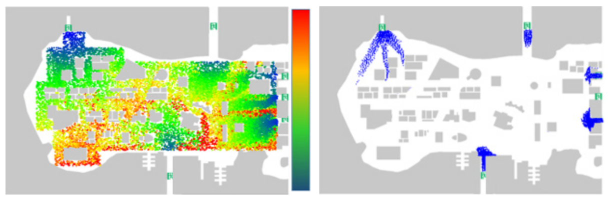

(c) The proposed method

Fig. 12 Island modification: widths of the roads at the bottom and lower right exits were reduced by around one-half. Left: distribution of agent travel time. Right: the remaining agents at frame 4500.

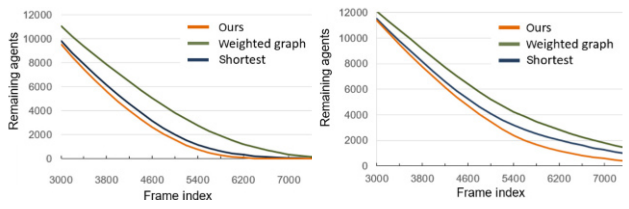


Fig. 13 Island example, numbers of remaining agents over time. Left: normal exit widths. Right: reduced widths of bottom and lower right exits.

6.2.3 *Island with transportation*

In some cases evacuation may require public transportation. We tested our system on an island, in which agents leave the island in discrete groups whenever vacant ferries arrive at the pier. Specifically, we assume the top left exit in Fig. 15 to be a pier. The gate in front of the pier opens once every 1000 time steps and then closes after 200 agents board the ferry. The initial distribution of agents was the same as in the previous example. Compared to the results from the shortest path method (see Fig. 15(a)), our method guides more agents to other exits and successfully minimizes the evacuation time (see Fig. 15(b)), as it takes into account that agents have to wait for ferries during evacuation.

We point out that our method is adaptive to events that can be simulated. It is not limited to public transportation in an urban city but also elevators in a building. While previous methods usually make several assumptions when computing the optimal evacuation route, the current system can work well in most situations.

6.2.4 *Park example*

Suppose there are some facilities in a park. During evacuation, people would prefer to leave buildings for reasons of fresh air and safety. We thus set-up a park scenario that contains a dome and an outdoor area. In addition, we placed 10,000 and 20,000 agents inside and outside the dome respectively. With the aim of evacuating agents in the dome first, we computed routes using our hierarchical framework. The evacuation time and dome exit of each agent in the dome is first computed, and then used in the optimization of evacuation routes outside the dome. Figure 16 shows both the initial distributions and the evacuation time of the agents. While agents take longer to evacuate the park if they follow the guidance routes determined by the hierarchical framework, they can leave the dome earlier than if following routes computed by considering the whole scene.

6.3 Comparison with a GA-based method

We also compared our method with a method based on a genetic algorithm (GA) [6] using an example from that paper of a rectangular hall with length 250 m and width 200 m having ten exit gates of similar length. Two different configurations were

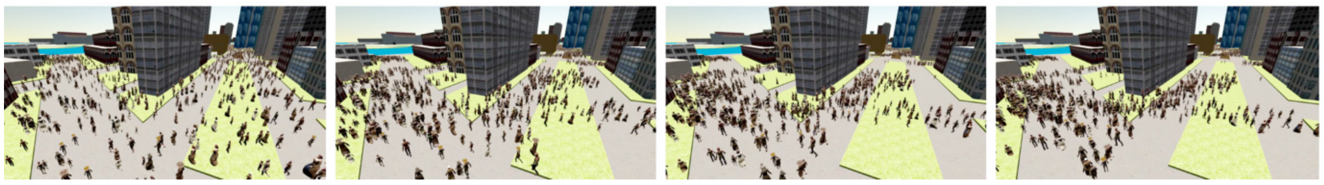


Fig. 14 3D view of the island scenario. Agents split into groups and move to their respective exits. Some agents form queues gradually as they are moving.

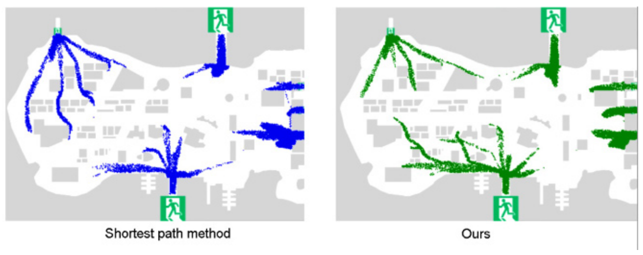


Fig. 15 Evacuation by ferry. Agents have to take ferries to leave the top left corner, causing evacuation from this direction to be slow. Our system thus guides more agents to other exits.

employed in Ref. [6], as shown in Fig. 17, the latter having a rectangular obstacle. Furthermore, we also introduced the use of safe zones as it is not appropriate to ignore all the agents after they reach the exits: they need to take time to move away from the exits so that they do not block the exits. Thus, in our experiments, we also recorded the average evacuation time of agents when the safe zones were employed. The agents were randomly generated in a uniform manner. The agents could move faster (at most two times faster than their desired walking speed) if there was room for them. The goal was to compute the routes giving lowest average evacuation time of the agents .

The GA method divided the hall into 25 subregions evenly and the agents in each subregion were assumed to follow the same evacuation plan. Thus, the GA method attempts to solve a discrete optimization problem, finding the best combination for the subregions. The GA method has higher computation cost for a greater number of subregions.

In the GA method, an individual represents a route plan for the agents. In general, a certain number of individuals are required, e.g., 10, 20, of 40 particles as proposed in Ref. [6]. The fitness of each individual is evaluated by crowd simulation. Mutation and crossover are employed to generate new individuals. Our method requires a road map that is represented as a graph. During the training

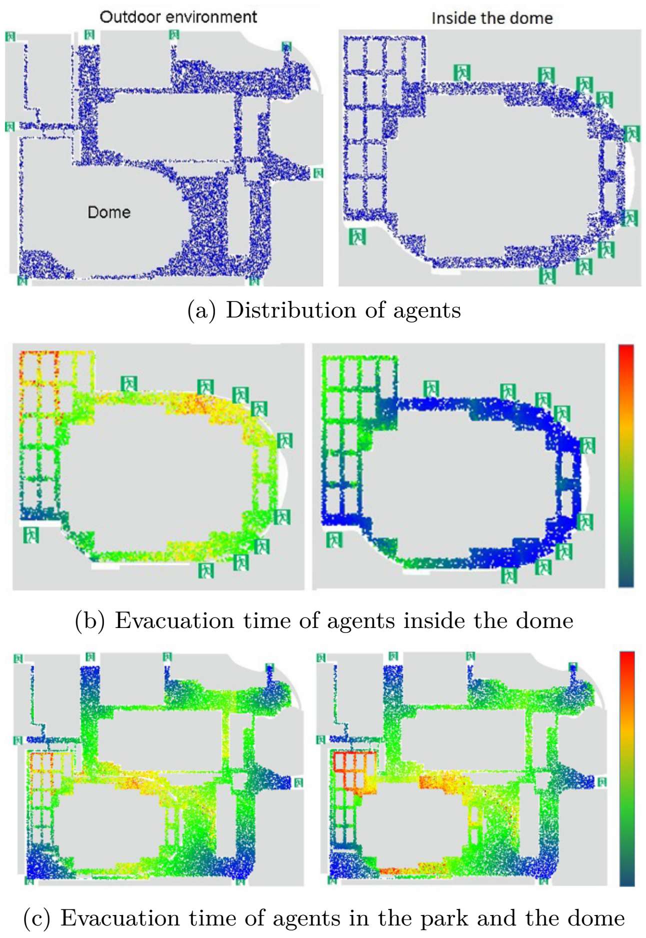


Fig. 16 Park example, showing evacuation time of each agent guided by our optimized evacuation routes. Left, right: routes computed by global and hierarchical schemes respectively. Routes computed hierarchically allow agents to leave the dome faster, but they take longer to leave the park because of congestion in the park.

process of our method, the agents followed their allotted paths. In both methods, agents could move directly to their respective exits if there was no obstacle between them and the exits.

Figure 18 shows the initial and optimal routes in our method. We could perform local adjustment to reassign agents to their exits after the training process is completed. The idea of local adjustment is as follows. If two agents i and j are assigned different

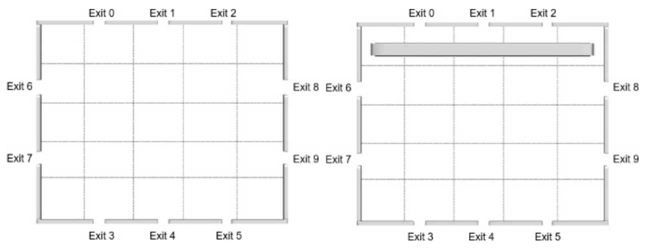


Fig. 17 Example hall, maps for comparisons with the GA method. Left: environment without an obstacle. Right: environment with a rectangular obstacle.

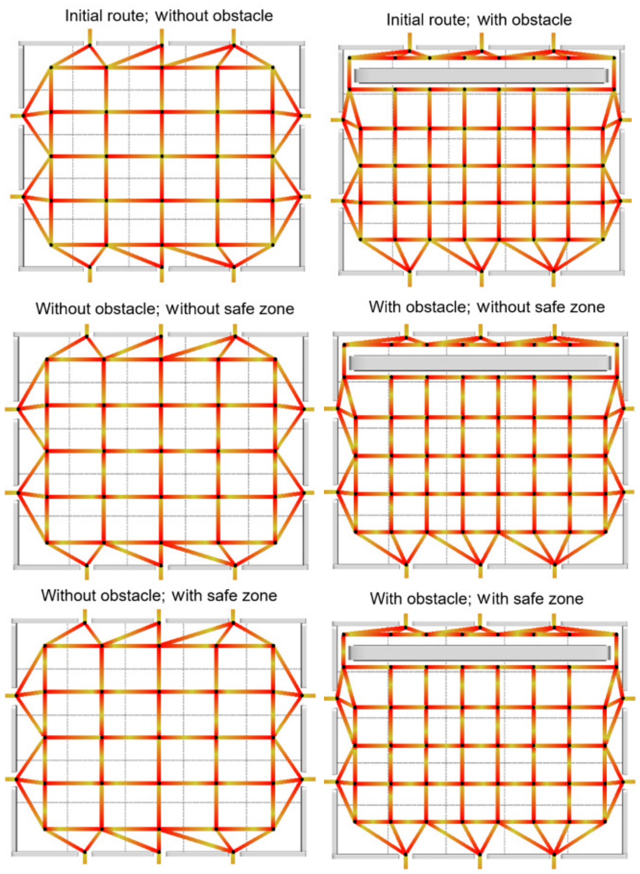

Initial route; without obstacle
Initial route; with obstacle
Without obstacle; without safe zone
With obstacle; without safe zone
Without obstacle; with safe zone
With obstacle; with safe zone

Fig. 18 The initial routes and optimal routes in our method. Top row: initial routes. Middle and bottom: optimal routes. The computed routes with and without safe zones are similar to each other.

exits, if $t_i > t_j + \|\boldsymbol{p}_i - \boldsymbol{p}_j\|/s_i$, then i is assigned the same exit as j and vice versa; here, t_i and t_j are the evacuation time of i and j before local adjustment, respectively, while s_i is the walking speed of i.

Table 2 compares average evacuation time for 25,000 agents for the GA method and the proposed method. The GA method attempts to group the different subregions for an exit while the proposed method attempts to adjust the division points in the graph to achieve an optimal route. Because our method and the GA are optimization techniques, both can compute paths with similar fitness value. However, our method provides faster evacuation than the GA method in general, with an improvement in average evacuation time of 5% to 10%.

The time complexity for the simulation part of the GA method is at least $O(N_\mathrm{A}M_\mathrm{A}T_\mathrm{S})$, where T_S is the computation cost of a crowd simulation, N_A is the number of individuals, and M_A is the number of training iterations. But M_A also depends on the number of subregions. In Ref. [6], the GA method took around 19 iterations to converge using 10 individuals. In our method, the time complexity is $O(M_\mathrm{O}T_\mathrm{S})$, where M_O is the number of training iterations. Our method converges in around five iterations, so for this example $M_\mathrm{A} > M_\mathrm{O}$. Thus, our method has much lower cost than the GA method.

Figures 19 and 20 show snapshots during simulation in two different configurations. An important finding is that the GA method is able to group non-adjacent subregions but our method is able to group adjacent agents. Figure 21 shows the number of remaining agents during simulation.

6.4 Comparison with a dynamic route adjustment method

Yang et al. [48] proposed a dynamic method which adjusts the crowd movement at runtime. Space is discretized as a grid. This method computes an attractive value for each grid cell based on four factors: relative distance to the closest exit (S_1), aisle regions (S_2), interaction between agents and obstacles (R), and movement orientation of agents in the local region (D). Here, agents are dynamic obstacles. The first two values can be determined once the environment is given but the last two values must be computed at each simulation time step. The total attractive value is computed as $\exp(k_\mathrm{s}(S_1 + S_2) + k_\mathrm{r}R + k_\mathrm{d}D)$, where k_s, k_r, and k_d are user-defined coefficients in $[0, 1]$. The value of S_1 for a grid cell depends on the shortest distance to the closest exit and the maximum shortest distance of all the grid cells to the exits. S_2 is set for a grid cell if it is in an aisle region. R is a non-positive value that makes agents tend to move away. See Ref. [48] for how to compute S_1, S_2, R, and D. To determine the movement direction of an agent, a

Table 2 Comparison between the GA method and the proposed method: average evacuation time of 25,000 agents, standard deviations in parentheses. Desired walking speeds of agents were randomly initialized in a given interval. LA: local adjustment

		Without safe zone (s)			With safe zone (s)		
	Walking speed (m·s^{-1})	GA	Ours	Ours + LA	GA	Ours	Ours + LA
Without obstacle	[0.42, 0.72]	69.71 (36.03)	66.06 (34.65)	64.87 (33.79)	71.85 (36.31)	68.32 (35.09)	66.91 (34.05)
	1.0	37.13 (19.08)	35.79 (18.23)	34.52 (17.87)	38.51 (19.34)	36.67 (18.81)	35.79 (18..09)
With obstacle	[0.42, 0.72]	85.38 (56.65)	77.26 (45.08)	76.62 (44.92)	87.37 (56.51)	80.78 (46.07)	79.72 (45.32)
	1.0	45.60 (30.01)	42.56 (25.42)	41.34 (24.29)	46.84 (30.06)	43.17 (24.48)	42.75 (24.23)

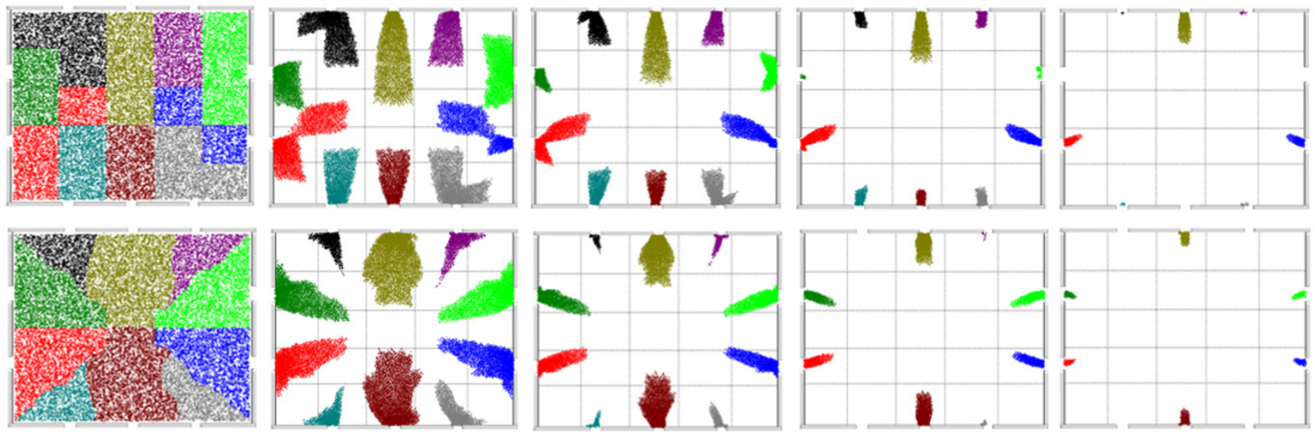


Fig. 19 Evacuation of 25,000 agents without obstacles. Agents were removed once they reached exits. Top: GA, bottom: our method. Left to right: snapshots at frames 0, 1000, 2000, 3000, and 4000. Our method makes agents move nearly in a straight line at frame 2000 to avoid collision at the exits. Clogging behaviors occurred at frame 2000 for exits 0, 5, 7, and 8 in the GA method.

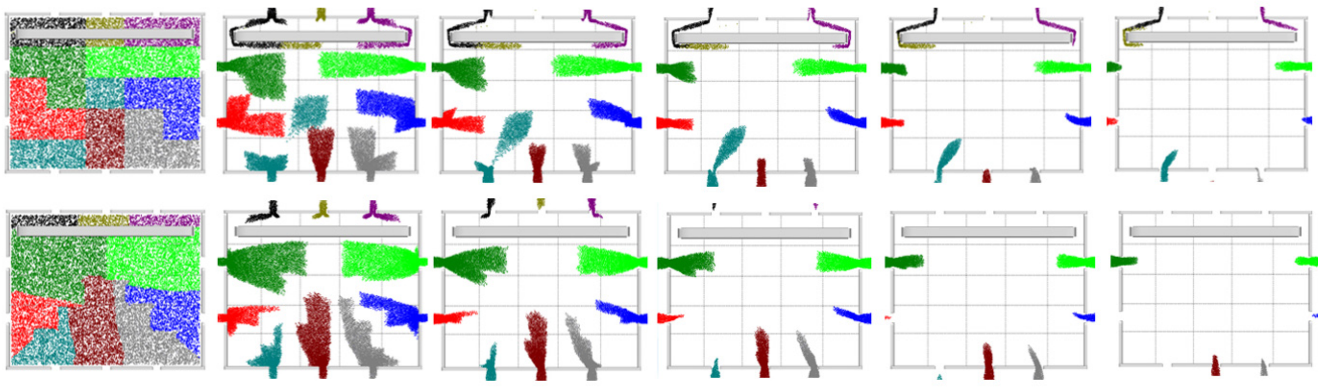


Fig. 20 Evacuation of 25,000 agents with obstacles and safe zones. Agents were simulated even after reaching exits. Top: GA, bottom: our method. Left to right: snapshots at frames 0, 1000, 2000, 3000, and 4000. In the GA method, the agents in the same subregion moved to the same exit. Thus, some agents below the obstacle took a long route to move to exit 1.

discrete choice probability model is computed based on the attractive values of the neighboring cells. An agent tends to move to a position with higher attractive value. The method relies on the aisle regions to attract agents and then distributes the agents to the exits.

We implemented the method in Ref. [48] and applied our crowd simulation technique to drive

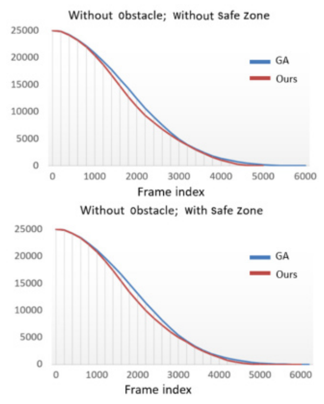


Fig. 21 Number of remaining agents over time during simulation using the GA and our method.

the crowd movement. Figure 22 shows a two-exit environment with 21 rectangular obstacles and two exits. There are two message boards in the top corridor. 35,000 agents were randomly generated inside the region; some were densely generated in two regions. Let A and B be the left and right exits, respectively, with crowd densities ρ_A and ρ_B. When an agent moves to a message board region, the agent will decide whether to move to the left or right exit according to the density difference between the two exits. For example, an agent in the middle left aisle moves to exit B if $\rho_A > k_\rho \rho_B$, where k_ρ is user defined. k_ρ was set to 1.5 in our experiment. Figure 23 shows the total attractive value map which was proposed in Ref. [48].

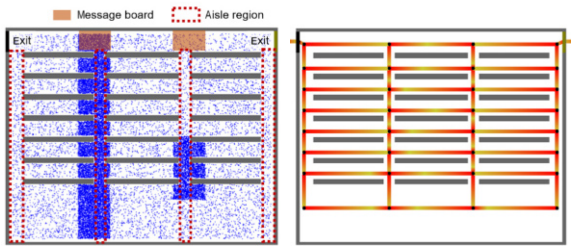


Fig. 22 Two-exit environment. Left: 35,000 agents (blue dots). Right: route computed by our method.

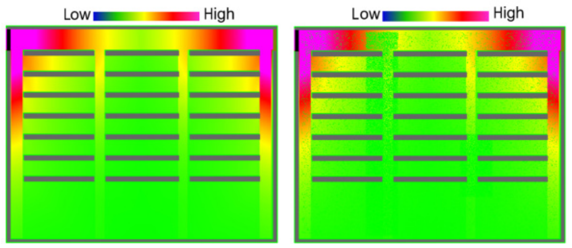


Fig. 23 Total attractive value map in the dynamic method. Left: initial total attractive value map. Right: total attractive value map for higher R (for easy visualization). In regions occupied by agents, total attractive values are lower.

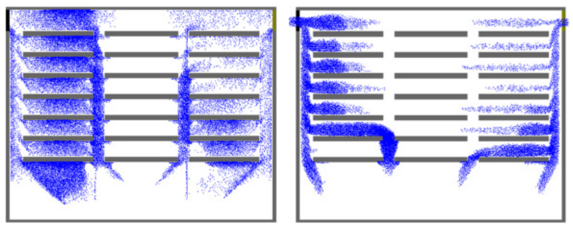


Fig. 24 Snapshots for the two-exit environment at frame 2000. Left: dynamic method from Ref. [48]; $k_s = 0.008$, $k_r = 0.00015$, and $k_d = 0.00025$. Agents tend to move along the aisle regions as their total attractive value is higher than for neighboring regions. Right: our method.

Figure 24 shows snapshots at frame 2000 of simulation results for the two-exit environment. Figure 25 shows the evacuation time distribution for the agents. Figure 26 shows the number of remaining agents over time for the dynamic method and our method. Our method took around 15,000 frames to evacuate all agents, but the dynamic method took over 20,000 frames. The major weaknesses of the dynamic method are: (i) some agents need to move to the aisle regions before they can select an exit, (ii) agents may not move consistently along a direction because their movement directions are computed using a probabilistic model, and (iii) it is not intuitive how to tune the coefficients k_s, k_r, and k_d to optimize the final result. The advantage

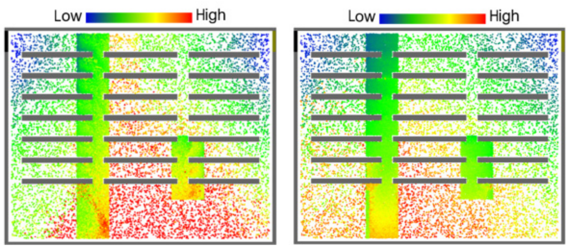


Fig. 25 Evacuation time distribution for the agents. Left: dynamic method. Right: our method.

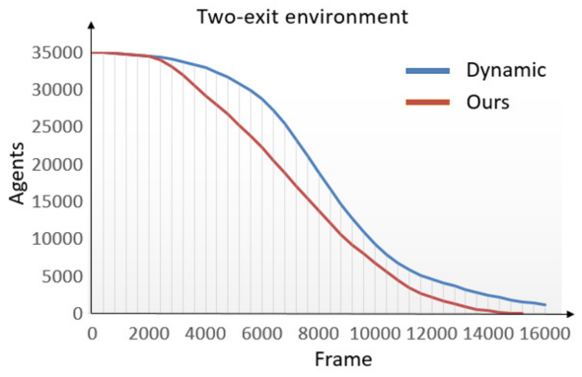


Fig. 26 Number of remaining agents in the two-exit environment. Our method performs better than the dynamic method [48].

of the dynamic method is that it allows agents to dynamically adjust their movement directions. Therefore, the method can handle events that are not known before simulation.

6.5 Comparisons with other methods and discussion

Various computational techniques have been proposed for computing evacuation routes [3]. We give a brief discussion of recent studies. Hamacher et al. [49] suggested computing a quickest flow based on a dynamic network flow model to obtain a lower bound for the evacuation time. It shows that the lower bound is not easily achievable because interaction of agents is ignored when the quickest flow is computed. An upper bound for the evacuation time is obtained by simulation based on a cellular automaton model. Tang et al. [9] proposed a method to compute evacuation routes for a crowd initially at a single node based on a computational network composed of an undirected graph. The method requires the input of parameters (expected travel time and variance) for traversing each edge. Furthermore, there are critical nodes and a deadline is associated with each critical node. Before the deadline, certain evacuation operations must be performed at the critical nodes. The method attempts to compute the routes so that the probability of a successful evacuation is greater than a given threshold. The method has limitations. First, it considers the crowd as initially located at a certain node. Second, the parameters are not easily estimated. The travel time along an edge may be time varying as it depends greatly on the crowd density. Of course, empirical experiments could be performed to collect such data.

Zhong et al. [50] proposed an evolutionary algorithm-based method for evacuation planning in open regions. A region is subdivided into a set of sub-regions. An agent-based crowd simulation model is employed to find heuristic rules used to evaluate the scores of exits for each sub-region. The method computes evacuation routes based on the heuristic rules and then assigns the routes to the agents in the sub-regions. The evacuation routes are evolved by employing Cartesian genetic programming.

Our method relies on the results of crowd simulation to drive the computation of division points and distribution ratios. Compared with the study in Ref. [9], our method does not require the input of the travel time and variance along each edge, but we do not consider critical nodes in our method. Unlike the study in Ref. [50], our method computes evacuation routes for a road network but may not be applicable to open regions. However, the proposed method can easily adapt to scheduled events because it adjusts the routes based on the simulation results. Overall, we believe that our work can inspire new approaches in the future.

6.6 Different crowd models

The presented system is flexible in choice of crowd simulation model. To verify this, we also integrated the social force [24] and reciprocal velocity obstacle (RVO) [25] models into our framework. We found that the optimal evacuation routes determined by the three methods were similar, although the estimated evacuation time of the agents differed between the models. In general, compared with the shortest paths, the last agent simulated in these models saves 24% to 35% of evacuation time when guided by our optimal route.

Our proposed method assumes that the crowd is located in an area with a road network. Because our method computes the routes based on the results of crowd simulation, we do not need to know crowd movement speeds in different road segments as a prior. It would be difficult to collect such data in certain regions before a simulation, for example, from regions where crowds exhibit clogging behaviours (see Fig. 27). Because of clogging behaviour, an agent may take longer to reach an exit than its neighbouring agents.

The inputs to our system are a scenario map and the distribution of a crowd. The former is easy to obtain as it simply indicates the positions of the

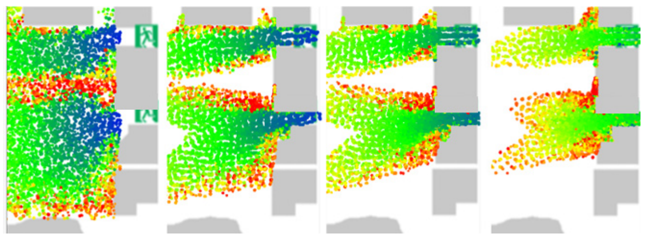


Fig. 27 Clogging behaviour in the island example. Agents are color-coded for tracking. Left to right: snapshots at frames 0, 500, 1000, and 2000. Although some red agents were near the exits, they took longer to reach the exits, because the roads were filled with moving agents preventing these red agents from moving to the exits.

obstacles. The latter could be more difficult in some scenarios. Fortunately, people in urban cities usually have smart phones and their locations can be sensed by global positioning system or Wi-Fi signals. Surveillance cameras can also be used to estimate crowd distribution [8]. To guide the crowd to evacuate efficiently, one simple approach is to set up dynamic signs close to intersections. Another approach would be to send individual evacuation routes to each smart phone to guide each user to an exit. However, at the moment, we simply assume that all the required inputs can be obtained and people in the real world will follow the given guidance during evacuation. We hope to extend our system to a practical level in future. The proposed method can be used under different conditions, such as various walking speeds, but the evacuation time spans in real and simulation states could differ.

Given that crowd distribution estimation can be achieved by analyzing surveillance videos or Wi-Fi adapters (giving cell phone positions), extending our work to handle dynamic changes and thus to update evacuation routes is not a problem. Specifically, the simplest way is to compute a new evacuation route whenever the crowd distribution is updated. We also could improve performance and retain temporal coherence of routes by setting the result from the previous state as an initial guess of the current state when updating the route.

The proposed method can be integrated with existing crowd modeling methods as long as the evacuation time of agents can be estimated during simulation. We also believe that combining other factors, such as sound perception and multi-sense attention, would be helpful in crowd evacuation in the real world.

The proposed method is dedicated for an environment with a road network so that it can compute evacuation routes for users. This network can be easily obtained if the environment has thin and long corridors. If the environment consists of large open areas, the areas can be decomposed based on a Voronoi diagram with respect to the roads connecting to these areas, after which the presented method is able to compute optimal evacuation routes.

6.7 Limitations

Our system relies on the results of crowd simulation. One of the strengths is its flexibility. However, crowd behaviors are often difficult to simulate because of various psychological issues. Thus, no simulation models can be considered perfect. For example, people could become aggressive and cut into lines frequently if they do not feel good during evacuation. Our current system simply assumes that people are patient and willing to follow given guidance to evacuate. Such simulation can work well in the situations such as the end of a concert or a happy new year party. But it is not able to simulate crowd behaviors when a serious earthquake occurs. Accordingly, the evacuation route computed by our system is not guaranteed to be optimum. Furthermore, because the entire simulation must be executed, the computation cost may be too high for a large scene. A hierarchically-based route may not work properly if the major exits of a zone with higher priority are blocked. The resulting route for this zone would be undesirable because the agents would not be able to move away from the zone. In this case, the agents should not be guided to such blocked exits. Our crowd simulation model is not based on the movement of real crowds because we do not consider the response time of the agents. In real life, people would take a short time to respond to the surrounding environment. “Stop, wait-and-go” behavior is not modeled accurately in our implementation. Because our method relies on adjusting the division points along edges and the distribution ratios at junctions, it is not suitable for computing evacuation routes for open regions. We intend to consider applying data driven approaches to computing optimal routes.

7 Conclusions

We have presented a novel approach to combine crowd simulation and evacuation route optimization. Our system can handle crowds with various attributes and environments that are difficult for previous methods. Given a scene map and a crowd distribution, we employ a shortest path algorithm to compute an initial evacuation route. The route is then iteratively refined according to the results of crowd simulation until an optimal state is achieved. Thanks to the integration of simulation, our approach is adaptable to elevators and public transportation that commonly occur in evacuation. Experimental results demonstrate the flexibility of the proposed method.

Although our system assumes that a crowd distribution in a scene is known in advance, the distribution is not difficult to estimate because of widely used smart phones and surveillance cameras. When the optimal evacuation route is computed, a simple and practical way is to send the route to each smart phone and guide users to exits. In the future, we plan to discuss such evacuation guidance with potential users, seek their feedback, and eventually develop a practical application. It would be compelling to compute evacuation routes combined with traffic [51] and layout design [52]. Finally, it would be important to improve the crowd simulation based on human factors and the data extracted from crowd videos [47].

Acknowledgements

The authors thank the reviewers for their constructive comments. This work was supported in part by "the Ministry of Science and Technology of Taiwan" under Grant MOST 102-2221-E-009-083-MY3, Grant MOST 103-2221-E-009-122-MY3, and Grant MOST 104-2221-E-009-051-MY3.

References

[1] Hamacher, H. W.; Tjandra, S. A. Mathematical modelling of evacuation problems: A state of the art. Fraunhofer-Institut für Techno-und Wirtschaftsmathematik, 2001.

[2] Wong, S.-K.; Wang, Y.-S.; Tang, P.-K.; Tsai, T.-Y. Optimized route for crowd evacuation. In: *Pacific Graphics Short Papers*. Grinspun, E.; Bickel, B.; Dobashi, Y. Eds. The Eurographics Association, 2016.

[3] Schadschneider, A.; Klingsch, W.; Klüpfel, H.; Kretz, T.; Rogsch, C.; Seyfried, A. Evacuation dynamics: Empirical results, modeling and applications. In: *Encyclopedia of Complexity and Systems Science*. Meyers, R. A. Ed. Springer New York, 3142–3176, 2009.

[4] Dressler, D.; Groß, M.; Kappmeier, J.-P.; Kelter, T.; Kulbatzki, J.; Plümpe, D.; Schlechter, G.; Schmidt, M.; Skutella, M.; Temme, S. On the use of network flow techniques for assigning evacuees to exits. *Procedia Engineering* Vol. 3, 205–215, 2010.

[5] Hadzic, T.; Brown, N.; Sreenan, C. J. Real-time pedestrian evacuation planning during emergency. In: Proceedings of the IEEE 23rd International Conference on Tools with Artificial Intelligence, 597–604, 2011.

[6] Abdelghany, A.; Abdelghany, K.; Mahmassani, H.; Alhalabi, W. Modeling framework for optimal evacuation of large-scale crowded pedestrian facilities. *European Journal of Operational Research* Vol. 237, No. 3, 1105–1118, 2014.

[7] Wang, H.-R.; Chen, Q.-G.; Yan, J.-B.; Yuan, Z.; Liang, D. Emergency guidance evacuation in fire scene based on pathfinder. In: Proceedings of the 7th International Conference on Intelligent Computation Technology and Automation, 226–230, 2014.

[8] Desmet, A.; Gelenbe, E. Capacity based evacuation with dynamic exit signs. In: Proceedings of the IEEE International Conference on Pervasive Computing and Communication Workshops, 332–337, 2014.

[9] Tang, H.; Elalouf, A.; Levner, E.; Cheng, T. C. E. Efficient computation of evacuation routes on a three-dimensional geometric network. *Computers & Industrial Engineering* Vol. 76, 231–242, 2014.

[10] Berseth, G.; Usman, M.; Haworth, B.; Kapadia, M.; Faloutsos, P. Environment optimization for crowd evacuation. *Computer Animation and Virtual Worlds* Vol. 26, Nos. 3–4, 377–386, 2015.

[11] Haworth, B.; Usman, M.; Berseth, G.; Kapadia, M.; Faloutsos, P. Code: Crowd optimized design of environments. In: Proceedings of the 29th International Conference on Computer Animation and Social Agents, 2016.

[12] Haworth, B.; Usman, M.; Berseth, G.; Kapadia, M.; Faloutsos, P. Evaluating and optimizing level of service for crowd evacuations. In: Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games, 91–96, 2015.

[13] Moussaïd, M.; Kapadia, M.; Thrash, T.; Sumner, R. W.; Gross, M.; Helbing, D.; Hölscher, C. Crowd behaviour during high-stress evacuations in an immersive virtual environment. *Journal of the Royal Society Interface* Vol. 13, No. 122, 20160414, 2016.

[14] Thompson, P. A.; Marchant, E. W. A computer model for the evacuation of large building populations. *Fire Safety Journal* Vol. 24, No. 2, 131–148, 1995.

[15] Lovas, G. G. On the importance of building evacuation system components. *IEEE Transactions on Engineering Management* Vol. 45, No. 2, 181–191, 1998.

[16] Pelechano, N.; Badler, N. I. Modeling crowd and trained leader behavior during building evacuation. *IEEE Computer Graphics and Applications* Vol. 26, No. 6, 80–86, 2006.

[17] Ma, Y.; Yuen, R. K. K.; Lee, E. W. M. Effective leadership for crowd evacuation. *Physica A: Statistical Mechanics and its Applications* Vol. 450, 333–341, 2016.

[18] Dimakis, N.; Filippoupolitis, A.; Gelenbe, E. Distributed building evacuation simulator for smart emergency management. *The Computer Journal* Vol. 53, No. 9, 1384–1400, 2009.

[19] Rodriguez, S.; Amato, N. Behavior-based evacuation planning. In: Proceedings of the IEEE International

Conference on Robotics and Automation, 350–355, 2010.

[20] Tsai, J.; Fridman, N.; Bowring, E.; Brown, M.; Epstein, S.; Kaminka, G.; Marsella, S.; Ogden, A.; Rika, I.; Sheel, A.; Taylor, M. E.; Wang, X.; Zilka, A.; Tambe, M. ESCAPES: Evacuation simulation with children, authorities, parents, emotions, and social comparison. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems, Vol. 2, 457–464, 2011.

[21] Kretz, T.; Grosse, A.; Hengst, S.; Kautzsch, L.; Pohlmann, A.; Vortisch, P. Quickest paths in simulations of pedestrians. *Advances in Complex Systems* Vol. 14, No. 5, 733–759, 2011.

[22] Inoue, Y.; Sashima, A.; Ikeda, T.; Kurumatani, K. Indoor emergency evacuation service on autonomous navigation system using mobile phone. In: Proceedings of the 2nd International Symposium on Universal Communication, 79–85, 2008.

[23] Chen, C.-Y. The design of smart building evacuation system. *International Journal of Control Theory and Applications* Vol. 5, No. 1, 73–80, 2012.

[24] Helbing, D.; Molnár, P. Social force model for pedestrian dynamics. *Physical Review E* Vol. 51, 4282–4286, 1995.

[25] Van den Berg, J.; Lin, M.; Manocha, D. Reciprocal velocity obstacles for real-time multi-agent navigation. In: Proceeding of the IEEE International Conference on Robotics and Automation, 1928–1935, 2008.

[26] Karamouzas, I.; Heil, P.; van Beek, P.; Overmars, M. H. A predictive collision avoidance model for pedestrian simulation. In: *Motion in Games*. Egges, A.; Geraerts, R.; Overmars, M. Eds. Springer-Verlag Berlin Heidelberg, 41–52, 2009.

[27] Guy, S. J.; Chhugani, J.; Curtis, S.; Dubey, P.; Lin, M.; Manocha, D. PLEdestrians: A least-effort approach to crowd simulation. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 119–128, 2010.

[28] Curtis, S.; Zafar, B.; Gutub, A.; Manocha, D. Right of way. *The Visual Computer* Vol. 29, No. 12, 1277–1292, 2013.

[29] Johansson, A.; Helbing, D.; Shukla, P. K. Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. *Advances in Complex Systems* Vol. 10, No. supp02, 271–288, 2007.

[30] Lee, K. H.; Choi, M. G.; Hong, Q.; Lee, J. Group behavior from video: A data-driven approach to crowd simulation. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 109–118, 2007.

[31] Durupınar, F.; Güdükbay, U.; Aman, A.; Badler, N. I. Psychological parameters for crowd simulation: From audiences to mobs. *IEEE Transactions on Visualization and Computer Graphics* Vol. 22, No. 9, 2145–2159, 2016.

[32] Boatright, C. D.; Kapadia, M.; Shapira, J. M.; Badler, N. I. Generating a multiplicity of policies for agent steering in crowd simulation. *Computer Animation and Virtual Worlds* Vol. 26, No. 5, 483–494, 2015.

[33] Li, F.-S.; Wong, S.-K. Animating agents based on radial view in crowd simulation. In: Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology, 101–109, 2016.

[34] Kapadia, M.; Pelechano, N.; Allbeck, J.; Badler, N. *Virtual Crowds: Steps toward Behavioral Realism*. Morgan & Claypool Publishers, 2015.

[35] Geraerts, R.; Overmars, M. H. The corridor map method: A general framework for real-time high-quality path planning. *Computer Animation & Virtual Worlds* Vol. 18, No. 2, 107–119, 2007.

[36] Van Toll W. G.; Cook IV, A. F.; Geraerts, R. Real-time density-based crowd simulation. *Computer Animation & Virtual Worlds* Vol. 23, No. 1, 59–69, 2012.

[37] Mekni, M. Hierarchical path planning for situated agents in informed virtual geographic environments. In: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, Article No. 30, 2010.

[38] Kallmann, M.; Bieri, H.; Thalmann, D. Fully dynamic constrained delaunay triangulations. In: *Geometric Modeling for Scientific Visualization*. Brunnett, G.; Hamann, B.; Müller, H.; Linsen, L. Eds. Springer-Verlag Berlin Heidelberg, 241–257, 2004.

[39] Pettré, J.; Laumond, J. P.; Thalmann, D. A navigation graph for real-time crowd animation on multilayered and uneven terrain. In: Proceedings of the 1st International Workshop on Crowd Simulation, Vol. 43, No. 44, 194, 2005.

[40] Bayazit, O. B.; Lien, J. M.; Amato, N. M. Better group behaviors in complex environments using global roadmaps. *Artificial Life 8* Vol. 8, 362, 2003.

[41] Patil, S.; van Den Berg, J.; Curtis, S.; Lin, M. C.; Manocha, D. Directing crowd simulations using navigation fields. *IEEE Transactions on Visualization and Computer Graphics* Vol. 17, No. 2, 244–254, 2011.

[42] Wong, S.-K.; Tang, P.-K.; Li, F.-S.; Wang, Z.-M.; Yu, S.-T. Guidance path scheduling using particle swarm optimization in crowd simulation. *Computer Animation & Virtual Worlds* Vol. 26, Nos. 3–4, 387–395, 2015.

[43] Müller, M.; Heidelberger, B.; Hennix, M.; Ratcliff, J. Position based dynamics. *Journal of Visual Communication and Image Representation* Vol. 18, No. 2, 109–118, 2007.

[44] Donelson, S. M.; Gordon, C. C. 1995 matched anthropometric database of U.S. marine corps personnel: Summary statistics. Technical Report. Geo-Centers INC Newton Centre MA, 1996.

[45] Aspelin, K. Establishing pedestrian walking speeds. Portland State University, 5–25, 2005.

[46] TranSafety Inc. Study compares older and younger pedestrian walking speeds. *Road Management & Engineering Journal*, 1997.

[47] Bera, A.; Kim, S.; Manocha, D. Online parameter

learning for data-driven crowd simulation and content generation. *Computers & Graphics* Vol. 55, 68–79, 2016.

[48] Yang, L.; Zhu, K.; Liu, S. Cellular automata evacuation model considering information transfer in building with obstacles. In: *Pedestrain Dynamics and Evacuation.* Peacock, R. D.; Kuligowski, E.; Averill, J. D. Eds. Springer Science+Business Media Springer, 317–326 2011.

[49] Hamacher, H.; Heller, S.; Klein, W.; Köster, G.; Ruzika, S. A sandwich approach for evacuation time bounds. In: *Pedestrian and Evacuation Dynamics.* Peacock, R.; Kuligowski, E.; Averill, J. Eds. Boston: Springer, 503–513, 2011.

[50] Zhong, J.; Cai, W.; Luo, L. Crowd evacuation planning using cartesian genetic programming and agent-based crowd modeling. In: Proceedings of the Winter Simulation Conference, 127–138, 2015.

[51] Lin, W.-C.; Wong, S.-K.; Li, C.-H.; Tseng, R. Generating believable mixed-traffic animation. *IEEE Transactions on Intelligent Transportation Systems* Vol. 17, No. 11, 3171–3183, 2016.

[52] Feng, T.; Yu, L.-F.; Yeung, S.-K.; Yin, K.; Zhou, K. Crowd-driven mid-scale layout design. *ACM Transactions on Graphics* Vol. 35, No. 4, Article No. 132, 2016.

Sai-Keung Wong received his M.Phil. and Ph.D. degrees in computer science from the Hong Kong University of Science and Technology in 1999 and 2005, respectively. Since August 2014, he has been an associate professor with the Department of Computer Science and the Institute of Multimedia Engineering, "National Chiao Tung University", Hsinchu, Taiwan, China. His research interests include computer graphics, virtual reality, and human–computer interaction. Dr. Wong received the Distinguished Mentor Award in 2011 and the Excellent Teaching Award in 2014.

Yu-Shuen Wang received his B.S. and Ph.D. degrees from the Department of Computer Science and Information Engineering, "National Cheng-Kung University", in 2004 and 2010, respectively. He is currently an associate professor of the Department of Computer Science at the "National Chiao Tung University" (http://people.cs.nctu.edu.tw/~yushuen/). He leads the Computer Graphics and Visualization Lab at the Institute of Multimedia Engineering. His research interests include computer graphics, computational photography, and visualization.

Pao-Kun Tang is a master student in the Department of Computer Science at the "National Chiao Tung University", Taiwan, China. His research interests include computer graphics, animation, and human–computer interaction.

Tsung-Yu Tsai is a master student in the Department of Computer Science at the "National Chiao Tung University", Taiwan, China. His research interests include computer graphics, crowd simulation, and human–computer interaction.