electronics

Article

A Smart Evacuation Guidance System for Large Buildings

Van-Quyet Nguyen [1,*], Huy-The Vu [1], Van-Hau Nguyen [1] and Kyungbaek Kim [2,*]

[1] Faculty of Information Technology, Hung Yen University of Technology and Education, Hung Yen 160000, Vietnam
[2] Department of Artificial Intelligence Convergence, Chonnam National University, Gwangju 61186, Korea
* Correspondence: quyetict@utehy.edu.vn (V.-Q.N.); kyungbaekkim@jnu.ac.kr (K.K.)

Abstract: In large buildings, during the situation of fire or other hazards, a smart evacuation guidance system needs to fully consider manifold aspects of hazards to guide evacuees through exit gates as fast as possible by dynamic and safe routes. In this paper, we propose a smart evacuation guidance system with a dynamic evacuation routing approach by using the LCDT (Length-Capacity-Density-Trustiness) weighted graph model and partial view (PV) information which represents the hazard intensity and the crowd congestion information of a group of sections/floors in the building. The proposed system is designed as a distributed system with multiple layers of computing by using smart indicators. Given such a system, we develop an efficient distributed approach, so-called LCDT&PV, to find out effective evacuation routes dynamically. We then propose an estimating congestion strategy in order to improve the efficiency of dynamic evacuation routes. To validate the proposed system, we implement a simulator to compare the proposed evacuation routing approach with baseline approaches. Experimental results show that the proposed approach reduces up to 30% of the total evacuation time compared with others. Furthermore, through the results of initial smart indicator implementation, which can interact with the simulator, we show the viability of the proposed system.

Keywords: smart evacuation guidance system; smart indicators; smart building; intelligent evacuation system

check for updates

Citation: Nguyen, V.-Q.; Vu, H.-T.; Nguyen, V.-H.; Kim, K. A Smart Evacuation Guidance System for Large Buildings. *Electronics* **2022**, *11*, 2938. https://doi.org/10.3390/electronics11182938

Academic Editor: Dimitris Kanellopoulos

Received: 17 August 2022
Accepted: 13 September 2022
Published: 16 September 2022

1. Introduction

As a result of the development of the Internet of Things (IoT) as well as many achievements of modern technology, smart buildings have been coming to reality with the support of multiple smart devices such as smart indicators, smart sensors, and smart cameras. These smart devices can help to build management systems to gather the essential information to make the right decisions in emergency situations such as fire building events. In which, emergency evacuation systems play an important role in safely evacuating people to the exit gate as soon as possible. One of the main roles of emergency evacuation systems is finding effective evacuation routes, which is not a trivial problem due to the uncertainty of the hazardous conditions and the possibility of congestions, to reduce evacuation time and help people pass through exit gates as fast as possible.

There have been several studies targeting the weighted graph based approaches using IoT data in smart buildings to dynamically find the evacuation routes more effectively as the situations can easily changes in such conditions [1–3]. Dimakis et al. [1] proposed a building evacuation system that evaluated the optimal evacuation routes in real-time based on updating the hazard intensity between decision nodes (indicators). Wong et al. [2] proposed an optimized evacuation route algorithm based on crowd simulation using division points that divide evacuees into two groups and evacuate in opposite directions. Lujak et al. in [3] proposed a distributed evacuation guidance for large smart buildings with building conditions and hazard intensity consideration based on a smart sensor network and personal mobile devices.

Most evacuation systems considered distributed approaches to find evacuation routes due to massive data generated by a large number of devices (e.g., sensors, cameras) [3]. Cristóbal-Salas et al. [4] proposed a strategy using distributed client-server system to compute the suitable routes for individuals depending on their health and their locations in a small city. Recently, Nguyen et al. [5] proposed a distributed system by implementing the LCDT-based (Length, Capacity, Density, and Trustiness) algorithm and a caching strategy. The system used global view information which is about the hazard intensity and the crowd congestion in the whole building, to build the evacuation routing algorithm. However, there is still room for improvement, because (1) using global view information might not be efficient for finding evacuation routes in a high-rise building (e.g., the evacuation routes on the tenth floor might not affect congestion at the second floor in a short period of time) and (2) implementing of weight calculation in *Smart Guidance Agents* might take a high cost (i.e., processing time) due to handling of massive data gathered by cameras and sensors in a short time.

In this paper, we present a design and implementation of an emergency evacuation system that uses a dynamic evacuation routing approach and smart indicators as edge computing nodes, and then direct the evacuees to exit gates. The proposed system is designed as a distributed system with multiple layers of computing that provides an efficient routing approach by using partial view information which represents the hazard intensity and the crowd congestion information of a group of sections/floors in the building. We design smart indicators to capture people density by applying a pre-trained convolutional neural network model, track danger areas using temperature and smoke sensors, and show directions. The gathered information from smart indicators is provided to the *Smart Guidance Agents* via a Web API for finding effective routes.

Our work as described in this paper makes the following contributions.

- An emergency evacuation system is designed with multiple computation layers using the information provided by smart indicators.
- An approach for dynamic evacuation routing is proposed, so-called LCDT&PV, which improves the LCDT-based approach [5] by using partial view information.
- An estimating congestion strategy is proposed to improve the efficiency of the evacuation routes.
- A simulator is implemented to compare the proposed evacuation routing approach with baseline approaches. Experimental results show that our approach reduces up to 30% of the total evacuation time compared with others.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 describes terms, definitions, and modeling hazard conditions and building conditions using the LCDT-based weighted graph model. Section 4 presents a smart evacuation guidance system by using the LCDT&PV approach for dynamic evacuation routing with the support of smart indicators. Section 5 conducts the experiments to evaluate the proposed system. Finally, conclusions are given in Section 6.

This paper is an extension of work originally presented in The 10th International Conference on Smart Media and Applications (SMA 2021) [6].

2. Related Work

There have been several techniques targeting the smart evacuation system. Dimakis et al. [1] proposed a building evacuation system that computes the optimal evacuation routes in a real-time manner. The system used smart sensors to obtain physical length and hazard intensity, then used them to make a weighted graph. Based on network flow techniques, Dressler et al. [7] proposed a method to determine the right exit gates for evacuees during emergency situations. Abdelghany et al. [8] proposed a framework to evacuate people in large-scale pedestrian facilities with multiple exit gates. Several designs of intelligent indoor emergency evacuation systems are presented in [9–13].

There are a lot of IoT-based evacuation systems that have been developed for providing real-time information to evacuees in case of fire events happening in the buildings [4,14–19].

Alfredo et al. [4] took advantage of a distributed client-server system to create a strategy to compute the suitable routes for individuals depending on their health and their locations in a small city amid disaster events such as floods, storms, or fires. In such events, information of each individual is collected by client software, and server computes the best routes and recommends to evacuees via the client software. However, this strategy did not consider hazardous conditions as well as congestion routes. Zualkernan et al. [14] proposed an IoT-based emergency evacuation system to obtain information about situations inside a building during a fire emergency in order to reduce turmoil and guide evacuees to safe exits. The system utilizes smartphones to recommend safe routes for evacuees. In case evacuees have no smartphone, the exit signs can dynamically change their state in accordance with the situation.

Recently, there have been several studies focusing on applying AI (Artificial intelligence) and ML (Machine Learning) techniques to detect fire events and find efficient evacuation routes in the smart buildings [15,18–20]. However, these studies have a lack of consideration in scalability of their systems due to large smart buildings. Lee et al. [15] proposed a new paradigm to develop assistance technology, whereby the systems developed based on the paradigm using IoT sensors integrated inside the buildings to collect data about hazard signs such as smoke or fire. The system then used a machine learning algorithm to determine the safe routes for evacuees. Wehbe et al. [18] presented a BIM-based smart evacuation guidance system for smart buildings which can detect fire early, collect and analyze the hazard condition based on sensor data, and guide evacuees to the exit with the optimal evacuation paths. Their system utilized IoT and smart technology to detect a fire early and reduce false detection. Then, AI and ML techniques are applied for providing the smart selection of evacuation paths in real time manner. Saini et al. [19] proposed an intelligent evacuation system that efficiently guides evacuees to a safer location while significantly reducing direct fire exposure by combining IoT layer, fog layer, and cloud layer. In their system, the IoT layer can capture the hazard condition in the smart building, and the location of evacuees to track their movements. The fog layer is employed to detect emergency events in the buildings by applying the Support Vector Machine (SVM) technique. Then, the cloud layer is employed an evacuation routing algorithm by using building conditions and evacuees' information to generate efficient routes to direct evacuees to the exit.

Thus, although there have been many studies focusing on the design and implementation of emergency evacuation systems, the development of smart evacuation systems is still in its infancy stage.

3. Preliminaries

3.1. Terms and Definitions

We consider a network of smart indicators in the building as an undirected graph $G = (V, E)$, where V is a finite set of nodes, and E is a finite set of edges, $E \subseteq V \times V$. A node in the graph G represents a smart indicator, called indicator node . An edge (v_i, v_j) represents a route segment in the building between two adjacent smart indicators v_i and v_j.

To evaluate evacuation routes, we assign the evacuation costs on every edge of G. We can define a weighted graph of G as follows.

Definition 1. ***Weighted graph:*** *A graph $G^w = (V, E, \omega)$ is a weighted graph if it is an undirected graph, and the function ω defined on E maps every edge $e \in E$ to a real number.*

3.2. LCDT-Based Weighted Graph Model

A weight value assigned to each edge of graph G indicates the impact of disaster conditions and building conditions to the evacuation time on a route segment. The higher the value of the weight, the more evacuation time is needed. In order to estimate the weight of a route segment, we consider four factors: (1) physical length, (2) physical width, (3) hazard intensity, and (4) people density. Obviously, physical length is a fixed value, and the longer the physical length, the more evacuation time is needed. For the physical width, it is used together with physical length to model the capacity (the number of people that can move together) of a route segment. We now consider a route segment $R_{(i,j)}$ between two adjacent smart indicators v_i and v_j, which has the physical length $L_{(i,j)}$ and the physical width $H_{(i,j)}$. Assume that an evacuee needs a minimum lane space $X \times Y$ (e.g., $X = Y = 1.0$ m) for moving, where X, Y are the physical length and the physical width of the lane space, respectively. Therefore, the capacity of the route segment C(i,j) can be calculated by using Equation (1).

$$C_{(i,j)} = (L_{(i,j)}/X) \times (H_{(i,j)}/Y) \tag{1}$$

Next, we consider the hazard intensity on a route segment, which is affected by a fire event. We model it by calculating trustiness of location $T_{(i,j)}$ based on the data received from a smoke sensor and a temperature sensor located at the route segment. Since this is not the main topic of this paper, we prefer to implement a sensor-based approach for calculating trustiness of location presented in [21]. Similarly, the density, $D_{(i,j)}$, on the route segment can be obtained by implementing a CNN-based approach of people counting from a single image captured by a camera [22].

Finally, we used LCDT model [5] to calculate the weight of every route segment $R_{(i,j)}$ as shown in Equation (2).

$$\omega_{(i,j)} = \frac{L_{(i,j)}}{T_{(i,j)} \times (C_{(i,j)} - D_{(i,j)} + 1)} \tag{2}$$

In Equation (2), $T_{(i,j)} \in [0, 1]$, if $T_{(i,j)} = 1$, there is no affected of disaster event on $R_{(i,j)}$. In contrast, $T_{(i,j)} = 0$ means that, the route segment $R_{(i,j)}$ has been strongly affected by disaster event. As the result, when $\omega_{(i,j)}$ get an infinity value, the evacuation routes should not include $R_{(i,j)}$.

4. Proposed Smart Evacuation Guidance System

This section presents a design and implementation of the proposed smart evacuation guidance system for large buildings.

4.1. Overview of System Architecture

Figure 1 presents the architecture of the proposed emergency evacuation system. The system consists of three main modules: *Smart Indicator*(s), *Smart Guidance Agent*(s) (SGAs), and *Smart Coordinator*. In which, the *Smart Indicators* interact with *Smart Guidance Agents* via a Web API using the HTTP protocol. While *Smart Guidance Agents* work with *Smart Coordinator* in the same application, namely *Smart Guidance Application*. We describe more details about these modules as follows.

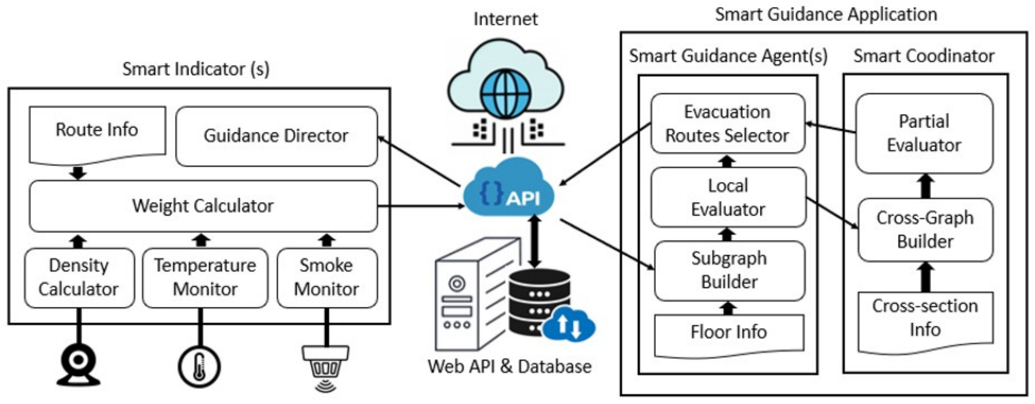


Figure 1. System Architecture.

Smart Indicators. A smart indicator is responsible for calculating the weight of a road segment (e.g., corridor segment) where it is located and showing the direction(s) to guide evacuees to pass through a safe exit gate. Here, the weight of a road segment represents the impact of building conditions and disaster conditions on that segment. We use the temperature and smoke information gathering by *Temperature Monitor* module and *Smoke Monitor* module. The people density on the road segment can be calculated by implementing a convolutional neuron network based image processing [13]. The people density calculating is performed by the *Density Calculator* module. For the capacity of the road segment, we can obtain it by estimating the maximum number of people who can move on the given segment at the same time based on the physical length and width of that segment information stored in the *Floor Info* file. Supposing that a person needs at least 1.0 square meters (m^2) of space for moving. Thus, a given road segment with 10 m of length and 2 m of width has a capacity of 20 (people). After calculating the weight of the road segment, smart indicators update the weight to the database by calling a REST API. This information is used by SGAs to build a weighted graph for finding efficient evacuation routes.

Another important role of a smart indicator is to show the direction provided by a *Smart Guidance Agent*. To do this, the *Guidance Director* module frequently checks the sign sent by the SGA from the Web API every a period of time (e.g., 5 s in our case).

Smart Guidance Agents. A *Smart Guidance Agent* is designed to control the direction of smart indicators in a specific region (e.g., a floor). To do this, the SGA first builds a weighted subgraph relying on the structure of a network of smart indicators in the given floor using the information stored in the *Floor Info* file and the weights of road segments sent by smart indicators under its management. Then, it runs a shortest path algorithm (e.g., Dijkstra algorithm) to find efficient routes from every smart indicator to stairs of the floor its management. This step is performed by the *Local Evaluator* module. The *Local Evaluator* also calculates and sends the weights among stairs to the *Smart Coordinator* module for building weighted cross-graphs. Finally, the SGA use the *Evacuation Routes Selector* module to choose the best evacuation route for its smart indicators by combining the weight from smart indicators to stair-nodes in the floor (i.e., smart indicators located at stairs) and the weight among stair-nodes of several floors sent by *Partial Evaluator* module.

Smart Coordinator. A *Smart Coordinator* has two main functions: (1) building weighted cross-graphs based on the structure of network stair-nodes among several floors (e.g., each of five floors) using the information stored in the *Cross-section* Info file and the weights among stair-nodes sent by the *Local Evaluators*; (2) finding the shortest path from a stair-node to other stair-nodes in the lower floors for every weighted cross-graph. This information is used by SGA to select efficient routes for smart indicators.

Web API. We implement a Web API application to provide functions to support updating the weight of road segments calculated by smart indicators as well as sending the guidance directions from SGAs to the smart indicators. The Web API application is published to a host on the Internet so that both smart indicators and SGAs can access through the HTTP protocol. The information is packed in JSON format before sending the request to the Web server.

4.2. Design of Smart Indicators

This section presents the implementation of smart indicators which is one of the main units of the proposed systems. As shown in Figure 2, each indicator is composed of three main components including a Raspberry PI, sensors, and an indicator panel. In particular, we use Raspberry PI 3B+ https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/, accessed on 16 March 2022, as a central processing unit. It is responsible for calculating the weight of a road segment by using information collected from sensors and provided by servers. To do that, we build a Raspberry PI OS on it for performing algorithms.

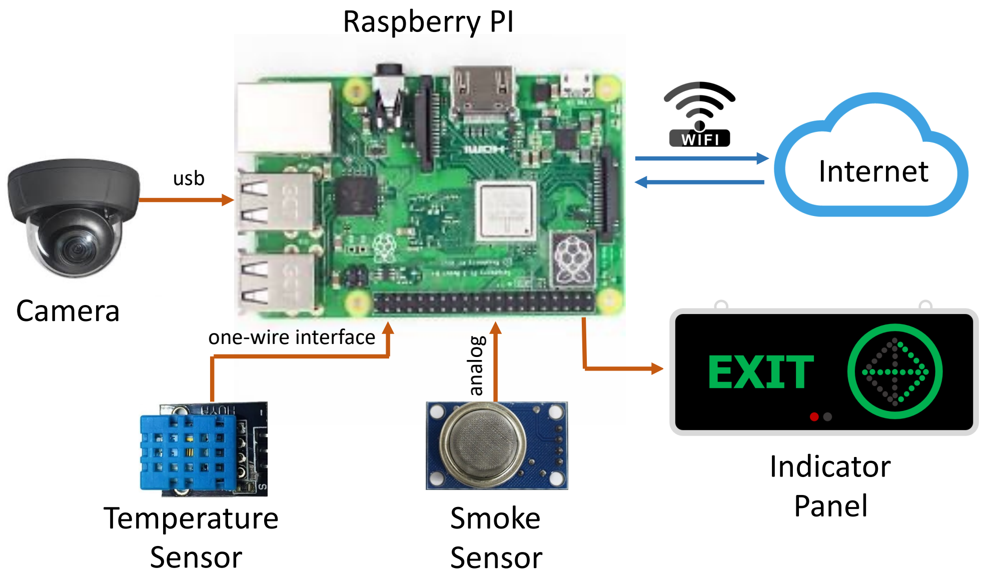


Figure 2. Hardware Components of a Smart Indicator.

For sensors, we first use a USB camera to capture images which are sent to Raspberry PI for calculating people density. Specifically, we adopt pre-trained CNN in [23] for counting people from these images. We then use MQ135 as a gas sensor for monitoring air quality where the smart indicator is located. This component can detect gases such as Ammonia (NH_3), sulfur (S), Benzene (C_6H_6), CO_2, and other harmful gases and smoke. Furthermore, we adopt a DHT11 sensor to measure temperature. This sensor has a one-wire interface enabling easily connects to Raspberry PI.

In addition, an indicator panel is designed to show four directions corresponding to east, west, north, and south, as shown in Figure 2. To do that, we use 37 super bright LEDs placed into four groups and controlled by a Python program running on Raspberry PI. Furthermore, we attach two LEDs (Red and Green) to the smart board to represent the status of disaster events. In which, the Red LED will be ON status during the disaster event, otherwise, it will be OFF and the Green LED will be ON. We also implement a program to send the calculated weight of the road segment and receive the sign via REST APIs every 5 s during a fire event happen. Note that, the smartboard can connect to the Internet via Wifi or Ethernet connection.

4.3. A Distributed Approach for Finding Dynamic Evacuation Routes Using LCDT&PV

In this section, we present a distributed approach for finding optimal evacuation routes that provides partial view information of evacuation routes while distributing the overall route computation to SGAs. To do this, we first define a distributed weighted graph based on the concept of the weighted graph in the previous section. We then propose the algorithms for local and global evaluation of evacuation routes in SGAs and GC, respectively.

Definition 2. ***Distributed Weighted Graph***: *A distributed weighted graph, G^d, for a weighted graph $G^w = (V, E, \omega)$ is a set of N fragments (subgraphs) $\{G_i = (V_i, E_i, \omega) | i \leq N\}$ and a cross-graph $G_c = (V_c, E_c, \omega)$, where $V_i, V_c \subset V; E_i \subset E$; and $E_c = \{(u, v) | u, v \in V_c\}$.*

In our context, N subgraphs are corresponding to the number of floors in a group of floors as well as the number of SGAs in that group. G_i represents a network of smart indicators at the floor ith, and G_c represents a network of smart indicators located at stairs and exit gates in the smart building.

4.3.1. Partial Evacuation Routing Approach

In this section, we present a strategy using LCDT-based evacuation routing approach without global view information (i.e., using the hazard intensity and the crowd congestion in the whole building) but with partial view information (i.e., using the hazard intensity and the crowd congestion information in a group of sections in the building). We have a key observation is that the status of hazard intensity and crowd congestion of floor A might not affect finding the evacuation routes in a floor B far from A (e.g., the tenth floor is far from the second floor). Meanwhile, several adjacent floors are affected by each other. Therefore, we improve the LCDT-based evacuation routing approach by using the partial view information instead of the global view information. The main idea of the LCDT-based partial evacuation routing algorithm is summarized in four main steps as the following:

- Step 1: Choose a parameter K $(2 < K < N)$ which indicates how many sections or floors (i.e., a large floor can be split into several sections) can be used as a group of sections or floors for partial evaluation in the Smart Coordinator, in which N is the number of sections or floors.
- Step 2: For each section/floor, the SGA builds a LCDT-based weighted graph for the subgraph of indicators. Then, it runs a procedure, so-called ***LocalEval***, to find options of evacuation routes to from every indicator node to stair nodes.
- Step 3: For each group of sections or floors, the SC builds a weighted cross-graph. The number of weighted cross-graphs, Cg, can be calculated by using a simple equation below:
$$f(x) = \begin{cases} N/K + 1, & \text{if } N\%K \geq K/2 \\ N/K, & \text{otherwise} \end{cases} \tag{3}$$
Thus, each cross-graph is constructed using K adjacent sections/floors except the last cross-graph has $K \pm N\%K$ sections/floors.
- Step 4: For each weighted cross-graph, the SC runs a procedure, so-called ***Partial-WeightEval***, to find effective routes among stair-nodes. Then, it runs a procedure, so-called ***PartialDensityEval***, to estimate the number of evacuees following a stair node for each given interval. Finally, the results are sent to SGAs for selecting evacuation routes.
- Step 5: Select effective routes from every smart indicator in each section/floor to the stair-node or exit-node on the lowest floor of each section. This step is performed by the Evacuation Routes Selectors module in SGAs by running ***TotalWeightEval*** procedure and ***RoutesSelector*** procedure.

In the following subsections, we present the implementation of the procedures of LCDT&PV approach in detail.

4.3.2. Implementation of Local Evaluation Algorithm

As we mentioned in the previous section, the Local Evaluator in each SGA is responsible for finding the options of evacuation routes for very indicator in a given subgraph. We have to find the effective routes from all nodes in the subgraph to stair nodes or exit nodes. Algorithm 1 illustrates the procedure of finding the options for every indicator in a subgraph G_i.

In Algorithm 1, we first set an empty set of next options to the *NextOptions* variable of every *Indicator Node* in G_i. We then run the Dijkstra algorithm on the weighted graph G_i with source nodes are a set of *Stair Nodes* S_i, and the destination nodes are all nodes in V_i, resulting in a complexity of $O(|S| \cdot |E_i| \cdot log|V_i|)$, with $|S|$ is the number of *Stair Nodes* in the G_i (every G_i has the same number of *Stair Nodes*) (lines 4–5). We get the weight and the next node from each indicator to each *Stair Node* during the searching by using *GetWeight* function and *GetNextNode* function (*not shown*), respectively (lines 7–8). These information are kept to be used for selecting the final evacuation route for each indicator in *Evacuation Route Selection* phase (line 9). Furthermore, the weights among pairwise of *Stair Nodes* are calculated and stored in a weight matrix W (lines 10–12). The matrix W

represents the weighted-based shortest distances between *Stair Nodes* in S_i ($i > 1$) or from *Stair Nodes* to Exit Nodes in case of evaluation on G_1 (exit gates on the first floor). Once the local evaluation phase is done, W is sent to the *Smart Coodinator*.

Algorithm 1 ***LocalEval***—*Local Evaluation*

```
Input: A subgraph $G_i = (V_i, E_i, S_i)$ with $S_i$ is a set of stair nodes
Output: $W$ is a matrix which contains the weight between pairs of stair nodes.
 1: for each $v$ in $V_i$ do
 2:   $v$.NextOptions $\leftarrow \emptyset$;
 3: end for
 4: for each $s$ in $S_i$ do
 5:   $T \leftarrow$ Dijkstra($s$, $G_i$); // T: Shortest Path Tree
 6:   for each $v$ in $V_i$ do
 7:     $w_{(v,s)} \leftarrow$ GetWeight($v$,$s$,$T$);
 8:     $u \leftarrow$ GetNextNode($v$,$s$,$T$);
 9:     $v$.NextOptions.Add($< u, w_{(v,s)}, s >$);
10:     if ($v \in S_i$ and $v \neq s$) then
11:       $W_{(s,v)} \leftarrow w_{(v,s)}$;
12:     end if
13:   end for
14: end for
15: return $W$;
```

4.3.3. Implementation of Partial Evaluator Module

This module performs evaluation of the effective evacuation routes from every *Stair Node* to every *Exit Node*. The module runs two main procedures: *PartialWeightEval* and *PartialDensityEval*. The idea of *PartialWeightEval* in Algorithm 2. The input of this algorithm is a cross-graph, G_c, which is built as the following: (1) each *Exit Node e* is connected to every *Stair Node* in G_1; (2) each *Stair Node* in G_i, where $i > 1$, will be connected to each other; (3) the connections among *Stair Nodes* in two adjacent floors are set up based on layout information of the building which is stored in *Cross-Section Info* component of SC; (4) the weights of the edges in G_c are assigned based on the cost matrices W received from SGAs. After constructing the cross-graph, we apply the Dijkstra algorithm to find the effective evacuation routes from every *Exit Node* to all *Stair Nodes* in G_c (lines 1–2). The weights from from every *Stair Node* to every *Exit Node* are calculated and stored in a weight matrix ω (lines 3–4). These weights will be sent back to the corresponding SGA.

Algorithm 2 ***PartialWeightEval***—*Partial Weight Evaluation*

```
Input: $G_c = (V_c, E_c)$; $\Phi$ is a set of exit nodes and $\Phi \in V_c$
Output: $\omega$ is a matrix which contains the weights from stair nodes to exit nodes.
1: for each $e$ in $\Phi$ do
2:   $T_c \leftarrow$ Dijkstra($e$, $G_c$);
3:   for each $s$ in $V_c \setminus \Phi$ do
4:     $weight \leftarrow$ GetWeight($s$,$e$,$T_c$);
5:     $\omega_{(s,e)} \leftarrow weight$;
6:   end for
7: end for
8: return $\omega$;
```

Next, the *PartialDensityEval* procedure performs the estimation of the number of evacuees following a stair node (1) passing through the exit nodes and (2) remaining between the stair node and the exit node, after each interval, Δt. These data are important in selecting evacuation routes in the later step, because selecting evacuation routes based on current status only might cause a congestion in the near future (after Δt). Therefore, the effective evacuation routes will help a lot of evacuees passing through exit nodes and reduce the number of evacuees going into crowed areas.

Algorithm 3 presents the implementation of the *PartialDensityEval* procedure. The input of this algorithm includes a weighted cross-graph, $G_c = (V_c, E_c)$; Φ is a set of exit nodes; ω is a set of weight on E_c; D is a set of people density on E_c; Δt is the update interval. The output consists of (1) a matrix ΔD containing the number of evacuees from the previous nodes of every stair node v passing through every exit node e after Δt and (2) a matrix Θ containing the number of evacuees from the previous nodes of every stair node v remaining from v to every exit node e after Δt. Firstly, we run a Dijkstra algorithm to find out the shortest path tree from every exit node (line 1–2). Then, for each stair node v in $V_c \setminus \Phi$, we estimate the number of evacuees following v, which will pass through exit nodes or remain on route segment from v to exit nodes, after Δt. To do that, we estimate the length in which evacuees can run after Δt with an average velocity, $\overline{v}$ (e.g., $\overline{v}$ = 3.0 m/s) along with a speed reduction parameter, α, depending on the weight of route segment (line 8). There are three main case of estimation: (1) all of evacuees from a previous node of v pass through a exit note e (lines 9–11); (2) a portion of evacuees from a previous node of v pass through a exit note e (lines 12–20); (3) no evacuees from a previous node of v pass through a exit note e (lines 21–27). Note that, the estimated number of evacuees from all previous nodes of v passing through e or remaining in (v, e) are summed up into $\Delta D_{(v,e)}$ and $\Theta_{(v,e)}$, respectively (line 30–31). Finally, the results are returned to the ΔD and Θ.

4.3.4. Implementation of Evacuation Routes Selector

This is the last phase in finding the optimal evacuation routes, and it is illustrated in Algorithms 4 and 5.

Algorithm 4 calculates the weight-based distances from each node $v \in V_i$ to all exit nodes based on the weight matrices W and ω (lines 1–7). The result is return to the weight matrix W (line 8).

Algorithm 5 selects the next indicator for each indicator node based on its point to the exit nodes. The point from an indicator to an exit node is calculated based on the weight-based point and the density-based point (lines 3–8). Here, $W^{min}_{(v)}$ is the minimum value of weights from the indicator v to exit nodes; $\Delta D^{max}_{(s)}$ is the maximum value of the number of evacuees from previous nodes of stair node s passing through exit nodes; $\Theta(s)^{min}$ is the minimum value of the number of evacuees from previous nodes of stair node s going into route segments (v, e). For each indicator node v, the next indicator is selected based on the highest point from v to the exit nodes (lines 9–11).

Algorithm 3 ***PartialDensityEval***—*Partial Density Evaluation*

Input: $G_c = (V_c, E_c)$; Φ is a set of exit nodes; ω is a set of weight on E_c; D is a set of people density on E_c; Δt is the update interval.

Output: ΔD is a matrix containing the number of evacuees from the previous nodes of every stair node v passing through every exit node e after Δt; Θ is a matrix containing the number of evacuees from the previous nodes of every stair node v remaining from v to every exit node e after Δt.

```
1:  for each $e$ in $\Phi$ do
2:    $T_c \leftarrow$ Dijkstra($e$, $G_c$); // Shortest Path Tree
3:    for each $v$ in $V_c \setminus \Phi$ do
4:      $\Delta D_{(v,e)} \leftarrow 0; \Theta_{(v,e)} \leftarrow 0;$
5:      $v.preNodes \leftarrow$ GetPreNodes($v, T_c$);
6:      for each $u$ in $v.preNodes$ do
7:        $L_{(u,e)} \leftarrow L_{(u,v)} + L_{(v,e)};$
8:        $\Delta L_{(u,e)} \leftarrow \overline{v}.\left(1 - \alpha.\frac{\omega_{(u,v)} + \overline{\omega}_{(v,e))}}{2}\right).\Delta t;$
9:        // all evacuees between node u and node v pass through e
10:       if ($\Delta L_{(u,e)} > L_{(u,e)}$) then
11:         $\Delta D_{(u,e)} \leftarrow D_{(u,v)};$
12:       else
13:         // if having evacuees pass through e
14:         if ($\Delta L_{(u,e)} > L_{(v,e)}$) then
15:           $\Delta D_{(u,e)} \leftarrow \frac{\Delta L_{(u,e)} - L_{(v,e)}}{L_{(u,v)}}.D_{(u,v)};$
16:           if ($\Delta L_{(u,e)} > L_{(u,v)}$) then
17:             $\Theta_{(v,e)} \leftarrow D_{(u,v)} - \Delta D_{(u,e)};$
18:           else
19:             $\Theta_{(v,e)} \leftarrow \frac{L_{(v,e)}}{L_{(u,v)}}.D_{(u,v)};$
20:           end if
21:           //if having no evacuees pass through e
22:         else
23:           if ($\Delta L_{(u,e)} > L_{(u,v)}$) then
24:             $\Theta_{(u,e)} \leftarrow D_{(u,v)};$
25:           else
26:             $\Theta_{(u,e)} \leftarrow \frac{\Delta L_{(v,e)}}{L_{(u,v)}}.D_{(u,v)};$
27:           end if
28:         end if
29:       end if
30:       $\Delta D_{(v,e)} \leftarrow \Delta D_{(v,e)} + \Delta D_{(u,e)};$
31:       $\Theta_{(v,e)} \leftarrow \Theta_{(v,e)} + \Theta_{(u,e)};$
32:     end for
33:   end for
34: end for
35: return $\Delta D, \Theta$
```

Algorithm 4 ***TotalWeightEval***—*Total Weight Evaluation*

```
Input: $G_i = (V_i, E_i)$, $w$ is the weight matrix on $G_i$, $\{\omega_{(s,e)}\}$, with $s \in S_i$ and $e \in \Phi$.
Output: $W$: a matrix representing weights from every indicator node to every every exit
   note.
 1: for each $v$ in $V_i \setminus \Phi$ do
 2:   for each $opt$ in $v$.NextOptions do
 3:     for each $e$ in $\Phi$ do
 4:       $W_{(v,opt,e)} \leftarrow w_{(v,opt)} + opt.weight2S + \omega(opt.s, e)$;
 5:     end for
 6:   end for
 7: end for
 8: return $W$;
```

Algorithm 5 ***RoutesSelector***—*Evacuation Routes Evaluation*

```
Input: $V_i$; $\Phi_i$ ; $W$; $\Delta D$; $\Theta$
Output: The next nodes for all indicator nodes in $V_i$
 1: for each $v$ in $V_i \setminus \Phi_i$ do
 2:   for each $e$ in $\Phi_i$ do
 3:     $P_{(v,e)} \leftarrow 0$;
 4:     for each $opt$ in $v$.NextOptions do
 5:       $s \leftarrow opt.s$
 6:       $P^W_{(v,e)} \leftarrow \frac{W^{min}_{(v)}}{W_{(v,opt,e)}}$;
 7:       $P^D_{(v,e)} \leftarrow \frac{1}{2} \cdot \left( \frac{\Delta D_{(s,e)}}{\Delta D^{max}_{(s)}} + \frac{\Theta^{min}_{(s)}+1}{\Theta_{(s,e)}+1} \right)$;
 8:       $point \leftarrow \beta.P^W_{(v,e)} + (1-\beta).P^D_{(v,e)}$;
 9:       if ($point > P_{(v,e)}$) then
10:         $P_{(v,e)} \leftarrow point$;
11:         $v$.next $\leftarrow opt$;
12:       end if
13:     end for
14:   end for
15: end for
```

4.4. Implementation of Communication Module

We implement a Web API application to provide functions to support updating the weight of road segments calculated by smart indicators as well as sending the guidance directions from SGAs to the smart indicators as shown in Tables 1 and 2.

Table 1. Description of Indicators APIs.

API	Description
GET api/Indicators/GetIndicators	Get information of all indicators
GET api/Indicators/GetIndicator/{id}	Get information of a specific indicator
PUT api/Indicators/PutIndicators	Update information for a list of indicators
PUT api/Indicators/PutIndicator/{id}	Update information for a specific indicator
POST api/Indicators/PostIndicator	Add a indicator to the database
DELETE api/Indicators/DeleteIndicator/{id}	Delete a specific indicator from the list of indicators
PUT api/Indicators/UpdateDirections	Update directions for a list of indicators
PUT api/Indicators/UpdateConditions	Update conditions for a list of indicators

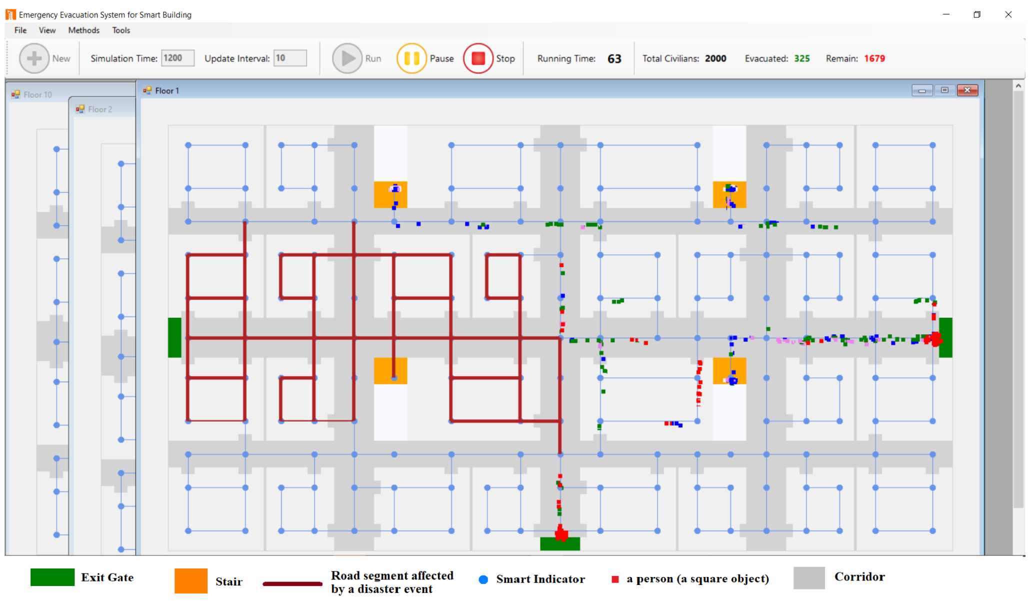
Table 2. Description of Conditions APIs.

API	Description
GET api/Conditions/GetConditions	Get information about temperature, smoke, amout of people that captured by indicators
GET api/Conditions/GetCondition/{id}	Get information about temperature, smoke, amout of people that captured by a specific indicator
PUT api/Conditions/PutCondition/{id}	Update information about temperature, smoke, amout of people that captured by a specific indicator
POST api/Conditions/PostCondition	Add information about temperature, smoke, amout of people that captured by a specific indicator
DELETE api/Conditions/DeleteCondition/{id}	Delete all conditions related to a given indicator

5. Evaluation and Results

We implemented a simulator as a Smart Guidance Application to show how the proposed system works. In this application, we implemented the LCDT-based evacuation routing algorithms using partial view information including the LCDT&PV basic approach and an improved approach with density estimation, namely LCDT&PV + DE approach. To evaluate the efficiency of the proposed approaches, we also re-implemented the baseline approaches including the LCDT-based evacuation routing algorithm using global view information, namely LCDT&GV [5]; Shortest-Path-Length based approach using the nearest stairs to pass through exits, namely Nearest Stairs; Length-Capacity-based approach which was mentioned in [2], namely LC; and Length-Trustiness-based approach [1], namely LT. These methods use the weighted graph models for finding effective evacuation routes, but they are different in weighting methodology. Specifically, the Nearest Stairs approach uses only physical length from indicators to stairs in a building as the weights on the weighted graph. Meanwhile, the LT approach considers both the physical length and the hazard intensity (trustiness of location) of the road segments to assign the weights for the weighted graph. For the LC approach, the physical length and the capacity of the road segments are used to build the weighted graph. For the LCDT-based approaches, we assign the weights for the weighted graph by using the physical length, capacity, density, and trustiness of the road segments. While the LCDT&GV approach takes the relation of all regions (floors) in a building into account, the LCDT&PV approach considers the relation of a few adjacent regions in a building along with an estimating congestion strategy to provide relevant information to find evacuation routes. Furthermore, we also implemented and deployed a smart indicator that interacts with the simulator.

We simulate a synthetic building with ten-story (ten floors). Each has four stair gates and 187 smart indicators, and the first floor has three exit gates as shown in Figure 3. The area of each floor is approximately 8300 square meters. We simulate a fire event that affected a stair on the first floor as well as two exit gates and their close regions. We assign randomly trustiness values in the range [0.2, 0.8] for road segments affected by the fire event. We also simulate the fire spreading by randomly updating trusted values and expanding the affected regions. We generate 2000 objects simulating people in the building. The location of people is generated randomly near the locations of smart indicators. We choose the interval for updating the status of every smart indicator to be 10s. The moving step of people is updated every 200 ms.


Emergency Evacuation System for Smart Building

Simulation Time: 1200 Update Interval: 10

Running Time: 63 Total Civilians: 2000 Evacuated: 325 Remain: 1679

Figure 3. An Evacuation Guidance Simulator.

Figure 4 illustrates the comparison of the LCDT&PV approach and the LCDT&PV + DE approach with other baseline approaches. We found that the LCDT&PV approaches are more effective than others overall. Specifically, to guide all people passing through exit gates, the needed simulation times are 352 s, 371 s, 406 s, 468 s, 498 s, and 509 s corresponding to LCDT&PV + DE, LCDT&PV, LCDT&GV, LT, LC, and Nearest Stairs, respectively. Thus, our LCDT&PV approach can save approximately 8% to 25% of the total evacuation time compared to other baseline approaches. Meanwhile, the LCDT&PV + DE approach can improve 5% of the total evacuation time compared to the LCDT&PV basic approach.

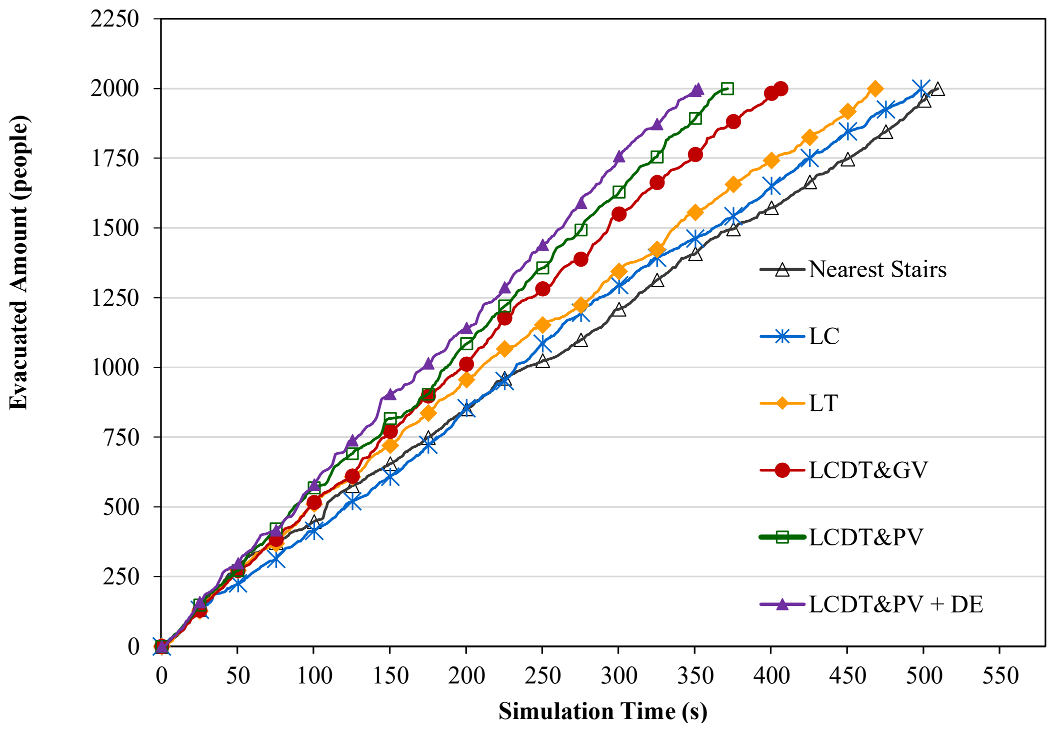


Figure 4. Comparison of the effectiveness between the LCDT&PV and other baseline approaches.

We also run experiments with varied K for partial evaluation in the LCDT&PV approach. The experimental results are described in Figure 5. We can see that the effectiveness of our LCDT&PV approach is slightly different between the values of K. In this case, K equals 3 is the best choice to run our LCDT&PV algorithm. In practice, identifying a suitable K for partial evaluation depends on several information such as the area of the buildings, the distribution of people in the building, and the impact of the fire event.

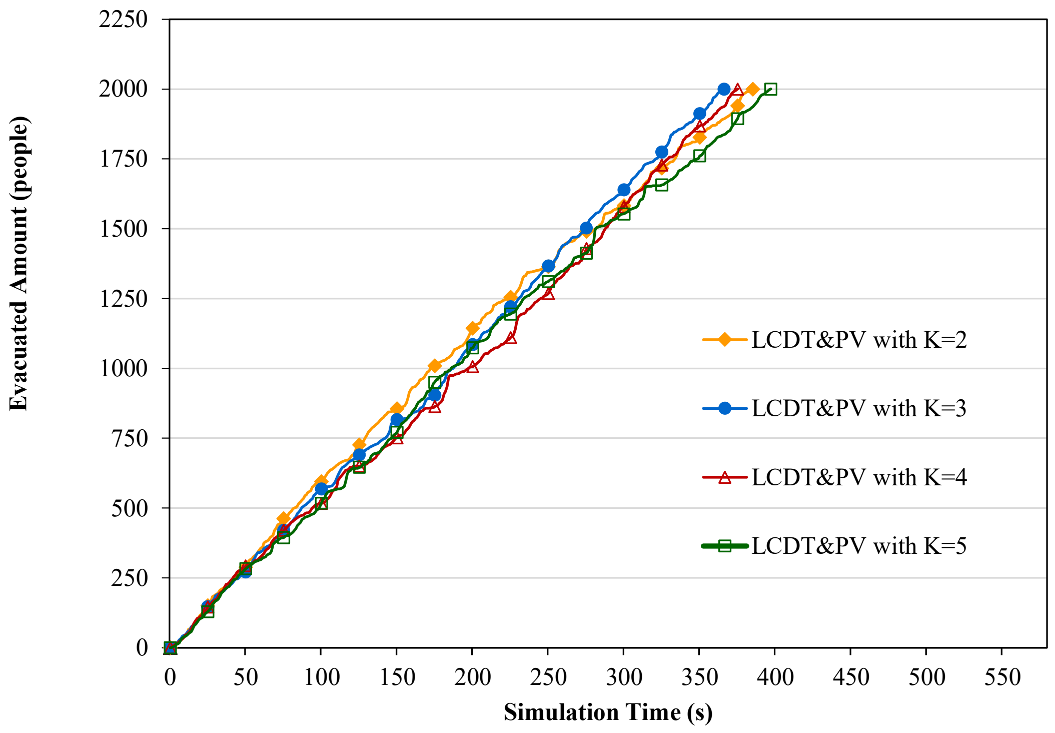


Figure 5. Comparison of the effectiveness with varied K in the LCDT&PV approach.

6. Conclusions

This paper presented a design and implementation of an emergency evacuation system that uses a dynamic evacuation routing approach and smart indicators as edge computing nodes, and then directs the evacuees to exit gates. While most traditional building evacuation guidance systems are concerned with routing based on the physical length of the road segment, the proposed system uses a dynamic routing approach that not only generates effective routes for evacuees but also quickly updates routes as the disaster status and the crowd congestion could change during the evacuation time. Moreover, the proposed system is designed as a distributed system with multiple layers of computing that provides an efficient routing approach using partial view information which represents the hazard intensity and the crowd congestion information of a group of sections/floors in the building. To achieve this, we first designed the architecture of a smart evacuation guidance system for large buildings. We then implemented smart indicators to capture people density using a pre-trained convolutional neural network model, tracked danger areas using temperature and smoke sensors, and showed directions. The gathered information from smart indicators was provided to the *Smart Guidance Agents* via a Web API for finding effective routes. Most importantly, an estimating congestion strategy was proposed to improve the efficiency of the evacuation routes. Finally, we implemented a simulator to compare the proposed evacuation routing approach with baseline approaches. Experimental results showed that the proposed approach reduces up to 30% of the total evacuation time compared with others. Through the results of initial smart indicator implementation, we showed the viability of the proposed system.

Author Contributions: Conceptualization, V.-Q.N. and K.K.; methodology, formal analysis, and writing—original draft preparation, V.-Q.N.; resources, visualization, writing—review, and editing, V.-H.N. and H.-T.V.; supervision, project administration, and funding acquisition, V.-Q.N. and K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the Innovative Human Resource Development for Local Intellectualization support program(IITP-2022-RS-2022-00156287) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: There is no conflict of interest.

References

1. Dimakis, N.; Filippoupolitis, A.; Gelenbe, E. Distributed building evacuation simulator for smart emergency management. *Comput. J.* **2010**, *53*, 1384–1400. [CrossRef]
2. Wong, S.K.; Wang, Y.S.; Tang, P.K.; Tsai, T.Y. Optimized evacuation route based on crowd simulation. *Comput. Vis. Media* **2017**, *3*, 243–261. [CrossRef]
3. Lujak, M.; Billhardt, H.; Dunkel, J.; Fernández, A.; Hermoso, R.; Ossowski, S. A distributed architecture for real-time evacuation guidance in large smart buildings. *Comput. Sci. Inf. Syst.* **2017**, *14*, 257–282. [CrossRef]
4. Cristóbal-Salas, A.; Santiago-Vicente, B.; Pérez-Castañeda, D.; Solis-Maldonado, C.; Luna-Sáncez, R.A.; Tchernykh, A.; Nesmachnow, S. A Customized Evacuation Route Planning Algorithm for Highly Risky Smart Cities. In Proceedings of the 2019 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), Ixtapa, Mexico, 13–15 November 2019; pp. 1–5.
5. Nguyen, V.Q.; Nguyen, H.D.; Huynh, Q.T.; Venkatasubramanian, N.; Kim, K. A scalable approach for dynamic evacuation routing in large smart buildings. In Proceedings of the 2019 IEEE International Conference on Smart Computing (SMARTCOMP), Washington, DC, USA, 12–15 June 2019; pp. 292–300.
6. Nguyen-Van, Q.; Kim, K. Design and Implementation of Smart Indicators based Emergency Evacuation System for Smart Buildings. In Proceedings of the 10th International Conference on Smart Media and Applications, Gunsan, Korea, 9–11 September 2021; pp. 1–6.
7. 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 Eng.* **2010**, *3*, 205–215. [CrossRef]
8. Abdelghany, A.; Abdelghany, K.; Mahmassani, H.; Alhalabi, W. Modeling framework for optimal evacuation of large-scale crowded pedestrian facilities. *Eur. J. Oper. Res.* **2014**, *237*, 1105–1118. [CrossRef]
9. Zhang, Q.; Chen, T.; Lv, X.Z. New framework of intelligent evacuation system of buildings. *Procedia Eng.* **2014**, *71*, 397–402. [CrossRef]
10. Xu, M.; Hijazi, I.; Mebarki, A.; Meouche, R.E.; Abune'meh, M. Indoor guided evacuation: TIN for graph generation and crowd evacuation. *Geomat. Nat. Hazards Risk* **2016**, *7*, 47–56. [CrossRef]
11. Angel, A.S.; Jayaparvathy, R. Design and implementation of an intelligent emergency evacuation system. In Proceedings of the International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC), Phuket, Thailand, 11 January 2017; pp. 13–17.
12. Wächter, T.; Rexilius, J.; König, M.; Hoffmann, M. Dynamic Evacuation System for the Intelligent Building Based on Beacons and Handheld Devices. In Proceedings of the International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), Hangzhou, China, 3–5 December 2021; pp. 117–124.
13. Karagöz, E.; Tecim, V. An Integrated Model and Application for Smart Building Systems with Artificial Intelligence. In *The Impact of Artificial Intelligence on Governance, Economics and Finance*; Springer: Berlin/Heidelberg, Germany, 2022; Volume 2, pp. 15–40.
14. Zualkernan, I.A.; Aloul, F.A.; Sakkia, V.; Al Noman, H.; Sowdagar, S.; Al Hammadi, O. An IoT-based emergency evacuation system. In Proceedings of the IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), Bali, Indonesia, 5–7 November 2019; pp. 62–66.
15. Lee, J.; Koo, D.; Tadesse, D.; Jain, A.; Shettar, S.; Kim, D. Emergency Evacuation Assistance. In Proceedings of the 14th International Conference on Ubiquitous Information Management and Communication (IMCOM), Taichung, Taiwan, 3–5 January 2020; pp. 1–6.
16. Choi, M.; Cho, S.J.; Hwang, C.S. Relieving Bottlenecks during Evacuations Using IoT Devices and Agent-Based Simulation. *Sustainability* **2021**, *13*, 9465. [CrossRef]
17. Fang, H.; Lo, S.; Lo, J.T. Building fire evacuation: An IoT-aided perspective in the 5G era. *Buildings* **2021**, *11*, 643. [CrossRef]
18. Wehbe, R.; Shahrour, I. A BIM-Based Smart System for Fire Evacuation. *Future Internet* **2021**, *13*, 221. [CrossRef]

19. Saini, K.; Kalra, S.; Sood, S.K. Disaster emergency response framework for smart buildings. *Future Gener. Comput. Syst.* **2022**, *131*, 106–120. [CrossRef]
20. Gomaa, I.; Adelzadeh, M.; Gwynne, S.; Spencer, B.; Ko, Y.; Bénichou, N.; Ma, C.; Elsagan, N.; Duong, D.; Zalok, E.; et al. A framework for intelligent fire detection and evacuation system. *Fire Technol.* **2021**, *57*, 3179–3185. [CrossRef]
21. Nguyen, V.Q.; Kim, K. Study on Location Trustiness based on Multimodal Information. In Proceedings of the 4th International Conference on Smart Media and Applications (SMA), Danang, Vietnam, 10–13 January 2016.
22. Sindagi, V.A.; Patel, V.M. A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognit. Lett.* **2018**, *107*, 3–16. [CrossRef]
23. Zhang, Y.; Zhou, D.; Chen, S.; Gao, S.; Ma, Y. Single-image crowd counting via multi-column convolutional neural network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 589–597.