# Congestion-aware Evacuation Routing using Augmented Reality Devices

Zeyu Zhang[1]    Hangxin Liu[1]    Ziyuan Jiao[1]    Yixin Zhu[1,2]    Song-Chun Zhu[1,2]

*Abstract*— We present a congestion-aware routing solution for indoor evacuation, which produces real-time individual-customized evacuation routes among multiple destinations while keeping tracks of all evacuees' locations. A population density map, obtained on-the-fly by aggregating locations of evacuees from user-end Augmented Reality (AR) devices, is used to model the congestion distribution inside a building. To efficiently search the evacuation route among all destinations, a variant of $A^\star$ algorithm is devised to obtain the optimal solution in a *single* pass. In a series of simulated studies, we show that the proposed algorithm is more computationally optimized compared to classic path planning algorithms; it generates a more time-efficient evacuation route for each individual that minimizes the overall congestion. A complete system using AR devices is implemented for a pilot study in real-world environments, demonstrating the efficacy of the proposed approach.

## I. INTRODUCTION

U.S. Federal Emergency Management Agency (FEMA) statistics [1] have revealed that 4.68 million in-building fires happened in a 10-year period of 2007 to 2016, along with 27,000 deaths and 139,925 fatal injuries. Although modern buildings often provide detailed 2D floor plans and clear emergency exit signs, tenants in general still take *extra* time in wayfinding during emergencies [2] due to fear, chaos, gridlock traffic, *etc.*, which jeopardizes their evacuation efficiency. In addition, such static information lacks real-time dynamics in terms of congestion, which further impedes the evacuation efficiency. Consequently, additional efforts must be spent on education [3], [4], drills [5], [6], or training in simulations [7], [8] in advance to reduce casualties.

However, such preventative training could be costly in setup; it is also challenging to recreate and emulate the actual (dangerous) scenarios realistically. Such insufficiency urges the needs for an *intuitive* system that can direct naive users with no prior experience or limited training to safely and effectively evacuate during actual emergencies.

In real-world settings, there are two main difficulties that prevent existing methods to form an effective means of evaluation. First, although the evacuation time has been identified as a vital factor [9] to improve evacuation efficiency, the majorities of the prior methods only consider the layout of the building (*e.g.*, the distance between two key points, the width of a hallway), but not the congestion condition during the evacuation. Although the localization and navigation methods developed for indoor mobile robots [10] could be

[1] UCLA Center for Vision, Cognition, Learning, and Autonomy (VCLA) at Statistics Dept. Emails: {zeyuzhang, hx.liu, zyjiao, yixin.zhu}@ucla.edu, sczhu@stat.ucla.edu.
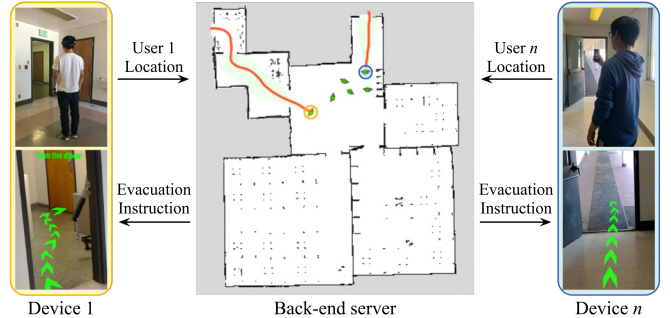[2] International Center for AI and Robot Autonomy (CARA)

Fig. 1: The architecture and data flow of the proposed system. By streaming the locations from the user-end devices (*e.g.*, AR headsets, cellphones), the remote server gathers population distribution of the environment in real-time. The centralized information is used to generate a time-efficient evacuation route and augment that to a user's view.

adapted to guide evacuees [11], [12], and several localization methods using various devices or sensors can effectively provide *individual*'s location, it remains challenging to take the *overall* congestion into account in evacuation routing. Second, the process of perceiving, analyzing, and deciding the evacuation route needs to be in real-time for multiple users, requiring a time-efficient path planning algorithm that can handle a large number of queries.

To address these challenges, this paper leverages the recent advancement of cloud computing with the growing availability and throughput of the indoor wireless network. As illustrated in Fig. 1, by streaming the real-time localization from the edge devices, a population density map can be generated in a remote server by aggregating individual's locations; higher density indicates more congestion (see Fig. 2). A congestion-aware evacuation routing algorithm is devised to provide the most efficient evacuation route by searching among *all* possible destinations. A parallel asynchronous design of the proposed algorithm is further developed to support thousands of path planning queries in real-time. To validate the proposed method and system, a pilot study was conducted using a complete system implemented with the Microsoft HoloLens AR platform.

### A. Related Work

**Localization** of evacuees is a prerequisite in providing an efficient evacuation and modeling the congestion condition. In addition to a typical localization setup in robotics [10], RFID tags [11], info points [12], [13], images [14], and the received WiFi [15], [16] or Bluetooth [17] wireless signals are introduced to improve the localization accuracy under an emergency. However, they require some infrastructures

deployed in advance and extra efforts at gathering localization information from individuals. **Multi-robot localization** methods can obtain each robot's location via a peer-to-peer broadcasting [18], [19], but they are usually slow in convergence, and the communication network between robots could be unreliable in practice. To address such deficiencies, the present work localize edge users using a SLAM-based method and a cloud computing platform to gather individuals' locations in a centralized manner.

Typical **path planning** algorithms (*e.g.*, A⋆, RRT⋆) can be used for evacuation routing by minimizing evacuation distance [20], [21], [22] or maximizing time efficiency [23], [24], [25] to properly handle the planning requests from a large number of agents [26], [27]. However, one of the key assumptions is that the congestion condition is fixed and provided upon planning. To take congestion into account, the field of **multi-agent path finding** has developed various approaches to find collision-free paths for multiple robots/agents [28], [29]. However, such approaches are still brittle in large-scale due to limited capability of handling dynamic changes of the congestion condition, possibly because adaptation to the changes of the congestion condition in real-time is a non-trivial problem, requiring additional engineering efforts. To handle hundreds of requests in real-time with dynamic changes of the congestion condition, the system presented in this paper jointly optimizes both distance and time by considering the overall congestion on-the-fly and developed a paralleled planning scheme.

**AR** technologies have received a considerable amount of attention in various human-robot interaction settings [30], [31], [32]. It affords an intuitive interaction to a naive user by overlaying rich *virtual* visual aids on top of the observed *physical* environment. In literature, several evacuation systems using AR technologies have been proposed, though limited, to provide indoor evacuation assistance. Typical AR devices, such as AR headsets [33], [34], [22], are equipped with depth cameras and IMUs, capable of providing a precise indoor localization. Due to these advantages, this paper implements and validates the system using the state-of-the-art AR headset, Microsoft HoloLens.

### B. Contribution

This paper makes the following three contributions:

1) We propose and implement a real-time population density map that models the *congestion* during the evacuation. The population distribution is aggregated from decentralized user-end devices to a centralized cloud server.

2) We propose and implement an efficient congestion-aware evacuation routing algorithm that simultaneously searches among *all* possible destinations in a *single* pass, providing the most time-efficient route by utilizing the population density map. A parallelized asynchronous solution of the algorithm is also devised to support thousands of path planning queries in real-time.

3) We prototype a complete evacuation system using the Microsoft HoloLens as the user-end device. A pilot study
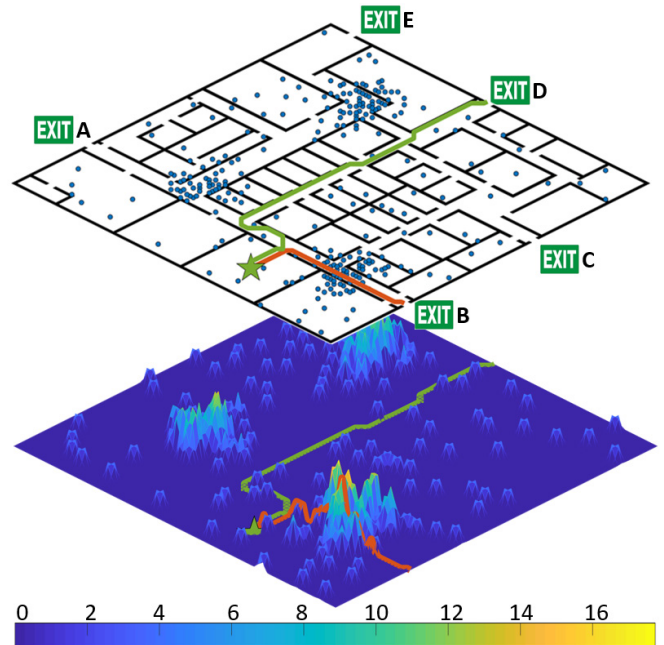


Fig. 2: Evacuation routing with a density map. (Top) The floorplan and the distribution of evacuees (blue dots). (Bottom) The population density map indicates the magnitude of congestion. If a human agent (green star) followed the route generated by a naive planner in terms of the shorted path (in red) to Exit B, s/he would compete with other agents. Instead, the proposed system suggests a further, but more time-efficient evacuation route (in green) toward Exit D.

has been conducted to validate the viability and efficacy of the proposed algorithm and system.

### C. Overview

The remainder of the paper is organized as follows. Section II introduces the construction of the population density map. The congestion-aware evacuation routing algorithm is described in Section III. Section IV evaluates the proposed system in both simulation and real-world scenarios. Section V concludes the paper with discussions.

## II. POPULATION DENSITY MAP

Without the congestion information, individual evacuees would only be able to choose a route by minimizing the distance, incapable of considering the actual time needed. Hence, an effective choice of evacuation route during emergencies should take the congestion into account. This section describes the proposed method that models the congestion as a population density map aggregated from user-end devices.

### A. Localization

Given a 2D floorplan, various SLAM methods could be used for localization. Since devising a SLAM method is out of the scope of this paper, an off-the-shelf solution based on ORB-SLAM [35] is adopted and briefly described below.

The extracted ORB features [36] are rotation-invariant and noise-resistant. The features yield a robust representation to camera condition changes (*e.g.*, auto-gain, auto-exposure,

illumination changes), which is an important property in emergency settings due to the varied lighting and rapid motions in evacuations. A feature matching process is performed to produce a set of monocular keypoints. The DBoW2 algorithm [37] further matches these key points to a keyframe of the environment by searching through the pre-built covisibility graph [38]. A motion-only bundle adjustment [35] uses the best-matched keyframe to determine and optimize the camera pose, thus providing an evacuee's location. This process can be performed on typical AR headsets with built-in cameras (*e.g.*, HoloLens) or smartphones [12], [13], [14].

### B. Population Distribution and Density Map

The population distribution is directly related to congestion during emergencies. After performing the localization on user-end devices (*e.g.*, AR headsets, mobile phones), the population distribution in the environment can be obtained by aggregating all users' locations to the cloud server through the wireless local area networks.

A density map can be further computed based on the population distribution to describe the congestion condition in the environment. The density map is a two-dimensional grid-like map built on top of the floorplan. Fig. 2 shows an example of the density map with its corresponding population distribution; the color bar indicates the magnitude of congestion $\rho$ at location $(x, y)$:

$$\rho_{x,y} = \sum_{(x_i,y_i)\in\mathcal{C}_{x,y}} \frac{\gamma N_{x_i,y_i}}{\sqrt{2\pi}} e^{-\frac{(x_i-x)^2+(y_i-y)^2}{2}}, \qquad (1)$$

where $\mathcal{C}_{x,y}$ is a set of cells which forms a patch centered at coordinate $(x, y)$, $N_{.,.}$ the number of agents, and $\gamma$ the congestion coefficient, describing the influence radius of every agent. The value of $\gamma$ depends on the resolution of the grid map; we set $\gamma = 5$ in our experiment. The path planning algorithm proposed in the subsequent section utilizes this density map to estimate the congestion while planning an evacuation route that better balances the distance and time.

### III. Evacuation Routing with Density Map

Consider the scenario shown in Fig. 2; if a naive planner were used, the agent (green star) would follow the red path to Exit B as it is the shortest. However, though further, it may be more favorable to the agent to follow the green path leading to Exit D with less congestion. This example illustrates the necessity of taking the density map into account in evacuation routing. This section describes the proposed *congestion-aware evacuation routing* algorithm that accounts for the density of other evacuees in a route and performs a centralized traffic control to avoid congestion and reduce competition during the evacuation in an emergency.

At a high level, this algorithm is a variant of the classic A⋆ algorithm, which is designed for pathfinding in multi-destination scenarios. It is modified (i) to search and select the most time-efficient path among *all* candidate destinations in a *single* pass, instead of searching individual destination with multiple passes as classic A⋆ does, and (ii) to avoid explore all of the possible destinations due to the bounded

---

**Algorithm 1:** Congestion-aware Evacuation Routing

**Input** : Grid map: $G_{grid}$
Density map: $G_{density}$
Start point: $src$
Destinations: $DList = \{dst_1, dst_2, \ldots, dst_n\}$

**Output:** A selected path from $src$ to one of the destinations $dst_k$

1 Initialize the open list: $OpenList = \emptyset$
2 Initialize the closed list: $ClosedList = \emptyset$
3 Insert Node($src$, $null$) to $OpenList$
4 **while** $OpenList$ is not empty **do**
5    // Pop node with the least f-score
6    $p \leftarrow OpenList.pop()$
7    // Reach the best destination
8    **if** $p == dst$ **then**
9       **return** $TracePath(dst)$
10    **end**
11    $S \leftarrow FindSuccessors(p, G_{grid})$
12    **foreach** $s \in S$ **do**
13       **foreach** $dst \in DList$ **do**
14          $s.dst \leftarrow dst$
15          $s.g \leftarrow p.g + Cost(p, s, G_{density})$
16          $s.h \leftarrow Heuristic(s, dst)$
17          $s.f \leftarrow s.g + s.h$
18          **if** $Validate(s, OpenList, ClosedList)$ **then**
19             Insert $s$ to $OpenList$
20          **end**
21       **end**
22    **end**
23    Insert $p$ to $ClosedList$
24 **end**

---

cost: during each iteration, the algorithm only explores the grid that has the smallest F-score along the direction of the exits/destinations; see details of the algorithm in below.

**Data Structure:** In practice, the explored space is much smaller compared to the exhausted Dijkstra search or A⋆ search with multiple passes. The algorithm is outlined in Algorithm 1; it is performed based on a data structure *Node*. Each Node $p$ contains:

1) Position ($p.pos$) stores the $(x, y)$ coordinates of the node in the grid map.

2) Parent ($p.parent$) records the $(x, y)$ coordinates from which $p.pos$ comes; $p.parent = null$ indicates current node is the start node.

3) Destination ($p.dst$) keeps the coordinates of the destination.

4) Total cost ($p.c$) represents the total cost from the start position to $p.pos$.

5) Heuristic score ($p.h$) indicates how far from $p.pos$ to $p.dst$ heuristically.

6) F-score ($p.f$) depicts the total estimated cost from the start node to $p.dst$. Specifically, we have $p.f = p.c + p.h$.

**Implementation details:** The algorithm maintains two lists; *i.e.*, the $OpenList$ and the $ClosedList$. The $OpenList$ stores nodes waited to be explored, and the $ClosedList$

records nodes already visited. Each is implemented by a heap-based priority queue, which stores all nodes in order, and is further augmented with a hash table to look up one of its elements in constant time. Both $OpenList$ and $ClosedList$ support two operations: (i) *pop()* returns a node with the least F-score and removes the node from the list, and (ii) *insert(p)* inserts the node $p$ into the list. See Algorithm 1 for an outline of the algorithm. Below, we detail some important functions.

- $Node(a, b)$ creates a new $Node$, whose $pos$ field is $a$ and $parent$ field is $b$.
- $TracePath(dst)$ retraces the path from the start position to $dst.pos$ by recursively visiting the $parent$ field.
- *FindSuccessors(p, $G_{grid}$)* looks up the grid map $G_{grid}$ and returns a list of nodes whose position is adjacent to $p$ in the $G_{grid}$. By default, the $parent$ field is set to $p$.
- The heuristic function *Heuristic(s, dst)* is defined as the euclidean distance between $s.pos$ and $dst.pos$.
- *Validate(s, OpenList, ClosedList)* determines whether the node $s$ should be inserted into the $OpenList$. The node $s$ would be added to $OpenList$ if one of the following conditions is satisfied: (i) $s$ is not in the $OpenList$; (ii) a node in the $OpenList$ whose position is as same as $s$ but has a higher heuristic score $f$; and (iii) a node in the $ClosedList$ whose position is as same as $s$ but has a higher total cost $c$. The key ideas behind these conditions are: (i) if the node has not been explored, add it to the $OpenList$; (ii) the node $s$ may have a better solution than the one at the same position in the $OpenList$ to be explored; and (iii) there is a better solution found to reach position $s.pos$, therefore re-explore the position $s.pos$.
- *Cost(p, s, $G_{density}$)* calculates the cost of travelling from $p.pos$ to $s.pos$. The cost $c$ is calculated by

$$c = \beta \cdot \text{dist}(s.pos, p.pos) + (1 - \beta) \cdot G_{density}[s.pos], \quad (2)$$

where $\text{dist}(s.pos, p.pos)$ is the euclidean distance, $G_{density}[s.pos]$ looks up the density map and returns the congestion estimation at position $s.pos$, $\beta$ is a coefficient that balances between distance and congestion estimation.
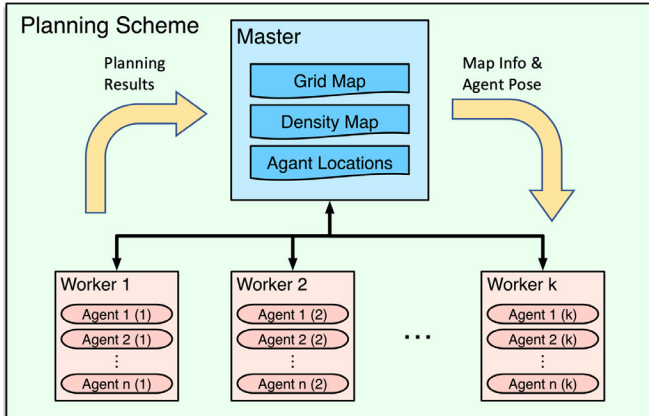


Fig. 3: Illustration of the parallelized asynchronous planning scheme. Both *Workers* and *Master* run in separated processes on a same multi-core machine.

**Paralleled Asynchronous Planning:** A paralleled asynchronous planning scheme is introduced to more efficiently schedule the planning jobs requested by user-end devices in a remote cloud server. Fig. 3 shows the high-level design. The users' job requests are evenly assigned to multiple *workers* (*e.g.*, CPU cores). Every *worker* maintains a list of path planning requests and processes them in turns. A *worker* periodically pushes the result back to the *master*, who maintains user information, updates the density map, and provides the latest population density to the user-end devices. Such a design can significantly improve the planning efficiency for a large number of users' planning requests by fully exploiting the computation power of remote servers.

## IV. EXPERIMENTS

We first evaluated the proposed method in a simulated indoor environment with crowds simulation to validate (i) the necessity of the introduced density map for modeling the congestion in terms of egress time, and (ii) the computational efficiency of the proposed planning algorithm comparing to classical planning algorithms.

The method is implemented in a proof-of-concept system consisting of Microsoft HoloLens as the user-end device and a regular blade server as the remote server. Additional experiments are conducted in the physical environment to qualitatively show the visual aids rendered by the system and quantitatively evaluate the system through a pilot study.

### A. Simulation Setup

A simulated environment (see Fig. 2) as a 2D map is constructed. The environment has a dimension of 100m×100m, which is discretized into 100×100 grids. It simulates a floor-plan of a typical school building that consists of classrooms, offices, study rooms, auditoriums, *etc*. There are five exits (EXIT A-E) in the environment.

The microscopic behaviors of the evacuees, such as the speed of their movement affected by others, are also crucial for a proper simulation in order to more realistically estimate the overall evacuation efficiency. Given the population
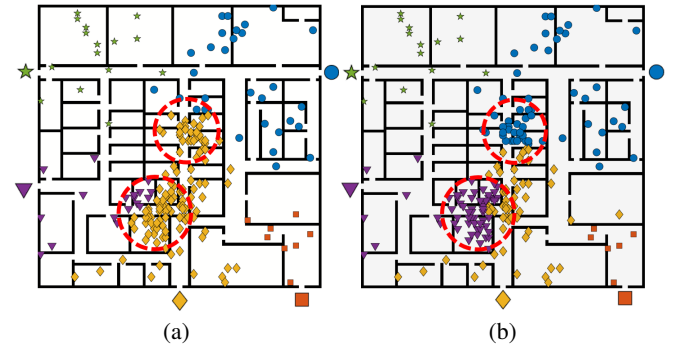


Fig. 4: Qualitative comparison of routing results. The agents' marker types are assigned based on the exits they are directed to. (a) Assigning the agents to their closest exit leads to congestion and competition. (b) The proposed method directs the agents in the dashed circles to another exit, though further, to avoid congestion; thus is more time-efficient.
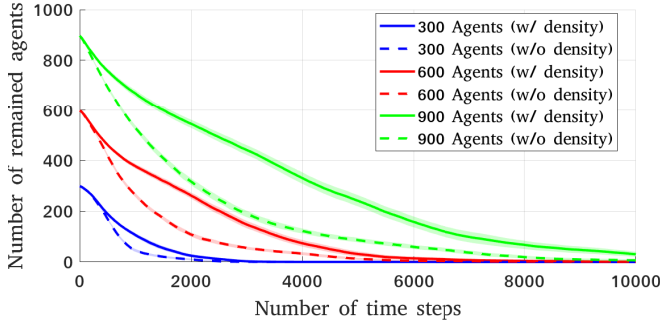
Fig. 5: Quantitative comparison of the average number of remained agents at each time step. The shaded color strips indicate the 98% confidence interval over 100 trails.

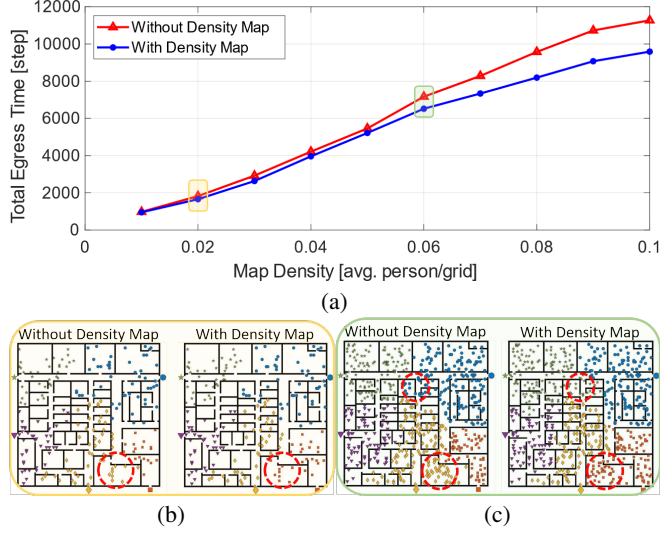

(a)



(b)                          (c)

Fig. 6: (a) Average total egress time vs. map density—average number of people per grid—of the scenario. Each point represents an average of 250 runs. (b)(c)The points in yellow and green shaded box denote examples of planning results for 0.02 and 0.06 map density, respectively.

distribution, the evacuees' movement is simulated by their gait length using the SLIP model [39], [40], which has been widely used to simulate bipedal locomotion; the maximum allowable human gait length is computed to bound an evacuee's movement and velocity.

### B. Evacuation Efficiency

Using the simulation environment, the effectiveness of the congestion modeling is evaluated by the total egress time–the time for all agents to evacuate from the environment. Fig. 4 highlights the differences of the routing results between simply finding the closest exits in terms of the distance and the proposed method that directs agents to avoid congestion by considering both distance and time. Figs. 5 and 6 further quantitatively showcase the efficacy of the proposed method; it yields a faster evacuation at every step and the final egress time, especially when the population density is high.

### C. Computation Efficiency

Fig. 7 shows the average run time (in log-scale) of 50 trials to generate the evacuation routing for up to 1,000
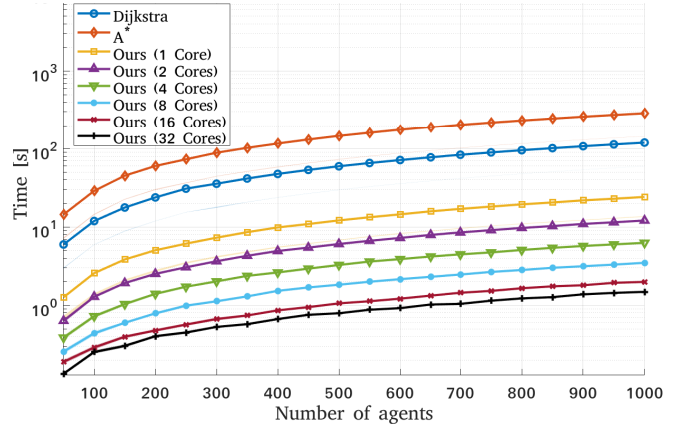


Fig. 7: The proposed method is more time-efficient compared to Dijkstra and repeated A$^\star$ path planning algorithm in a single-core setting. It can be further paralleled to multi-core computing setup with a significant performance gain, achieving a real-time performance (around 1 second) to serve 1,000 users using 32 cores.

agents, wherein we compare the proposed method with A$^\star$ algorithm and Dijkstra algorithms. Although the proposed algorithm runs significantly faster than classic approaches using a single core, it still takes more than 10 seconds to serve 1,000 users. This insufficiency is the core motivation to introduce the paralleled asynchronous planning scheme to take advantage of modern multi-core computers. The proposed algorithm can be easily paralleled and distributed into different cores; the computation time of using a different number of cores (2-32) is shown in Fig. 7. In our experiments, the proposed method can achieve real-time performance to serve 1,000 users with 32 cores.

### D. System Prototype

The prototype system adopts the state-of-the-art AR head-mount display, Microsoft HoloLens, as the user-end device. Compared to other available AR headsets, HoloLens is the first untethered AR head-mounted display that allows the user to move freely in the space without being constrained by cable connections, which is particularly crucial for evacuation applications. Integrated with 32-bit Intel Atom processors, HoloLens equips with an IMU and multiple spatial-mapping cameras to perform low-level computation onboard. Using Microsoft's Holographic Processing Unit, the users can realistically view the augmented contents. The holograms displayed on its screen are created by Unity3D, via which various visual effects can be rendered.

A back-end server is deployed to handle computationally intense jobs (*e.g.*, density map construction, path planning). The system in the server hosts a task scheduler, which coordinates the computation sources and assigns the new-coming user to different worker threads (see Fig. 3). Once the remote server receives users' locations, an evacuation path for every user is generated (see Section III) and communicated back to the corresponding user-end device.

As a proof-of-concept prototype, the system incorporates two types of visual guidance: (i) the evacuation path to direct
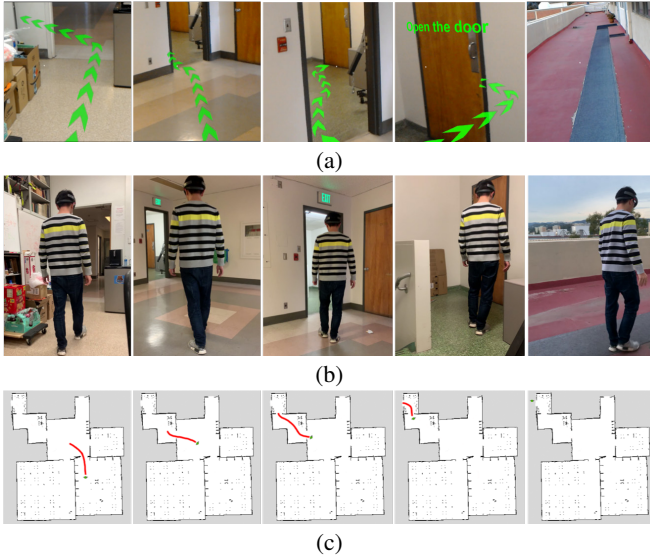
(a)



(b)



(c)

Fig. 8: An example of the evacuation process using an AR headset in real world. (a) Planned path from the user's egocentric view using a Hololens AR headset. (b) Third-person view. (c) Localization in the physical world.

a user to the most-efficient exit, which is updated on-the-fly as the user's location changes, and (ii) the visual symbols instruct the users to interact with the environment during the evacuation; for instance, as shown in the top row of Fig. 8, a 3D text, "Open the door," is augmented to the door to guide users' critical actions, further improving a user's reliance.

### E. Human Study

We conducted a pilot study by recruiting 16 participants to evaluate the effectiveness of the AR evacuation system in a between-subject setting ($N = 8$ for each group). Half of the participants were in the baseline group and provided only a 2D physical floorplan; they were asked to find a path to evacuate in the physical world without any additional information. The other half of the participants were in the AR group, using the AR headset HoloLens with the proposed evacuation system installed. Each subject has no familiarization with the physical environments. The subjects in AR group have no prior experience using AR devices.

Fig. 8 depicts examples of the evacuation process, where the planned route from the user's egocentric view is shown in the top row, a third-person view is shown in the middle row, and a real-world localization is shown in the bottom row. Fig. 9 compares the results between the two groups. The difference of escape time is statistically significant; $t(15) = 1.0$, $p = 0.0056$. Participants in AR group takes a significantly less time (median: 40 seconds) to evacuate. In contrast, the baseline group requires much more time (median: 80 seconds) with a larger variance in terms of the evacuation time. This result indicates the efficacy of using the AR evacuation system in a real-world evacuation scenario.

## V. CONCLUSION AND DISCUSSION

This paper proposes a congestion-aware evacuation routing solution using augmented reality. To alleviate the overall
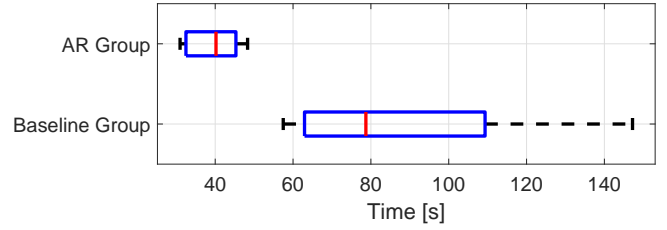


Fig. 9: Box plot for all participants' escape time in two groups. Subjects in AR group take a significantly less time.

congestion during emergencies, the proposed system simultaneously tracks of all evacuees' locations in real-time to provide a time-efficient evacuation path for each individual on-the-fly. The proposed method adopts the idea of edge computing that leverages the computation power on both user-end devices and the remote server. To further accelerate the proposed method, a parallel asynchronous planning scheme is devised to fulfill the demands of thousands of planning queries from evacuees. The simulated experiments demonstrated the effectiveness and efficiency of the proposed method by achieving real-time evacuation routing for 1,000 users. The proposed method is also implemented on a physical system using a Microsoft HoloLens and a remote server. A pilot study has been conducted to demonstrate the efficacy of the proposed method further.

Below, we discuss four related topics in greater depth.

**Alternating between different routes:** The proposed method updates evacuation routes as the density map updates. When the density map changes dramatically, *e.g.*, a large group of users suddenly connect to the system, the system may suggest a completely different route and even alternate back-and-forth. In this case, a possible solution would be imposing extra constraints (*e.g.*, a discount factor) when re-routing an evacuee to another exit. Alternatively, the density map only updates gradually as evacuees move around; thus, minor disturbance of the system is unlikely to cause such an alternation.

**Integrating predictive modules in routing:** In future work, it is possible to integrate a predictive model (*e.g.*, Hidden Markov Model) for evacuees' movements, therefore achieving a better estimation of the density map. Such a predictive module could improve evacuation efficiency by resolving potential congestion in advance.

**Network accessibility:** The presented system currently relies heavily on the wireless local area network (*e.g.*, WiFi) due to the required communications between the back-end server and HoloLens front ends. Under the situation where such infrastructure is not available, alternative forms of infrastructure (*e.g.*, the wireless ad-hoc network, BlueTooth) could be easily adapted to support the communication.

**Adapting to other robot systems:** The proposed method and the paralleled asynchronous planning architecture could be further adapted to other large-scale multi-agent robot systems, which can leverage the power of distributed edge computing and a centralized back-end server.

REFERENCES

[1] "The united state federal emergency management agency statistics." Accessed: 2019-02-01.

[2] A. Schwering, J. Krukar, R. Li, V. J. Anacta, and S. Fuest, "Wayfinding through orientation," *Spatial Cognition & Computation*, vol. 17, no. 4, pp. 273–303, 2017.

[3] K. Shiwaku and R. Shaw, "Proactive co-learning: a new paradigm in disaster education," *Disaster Prevention and Management: An International Journal*, vol. 17, no. 2, pp. 183–198, 2008.

[4] V. Kholshchevnikov, D. Samoshin, A. Parfyonenko, and I. Belosokhov, "Study of children evacuation from pre-school education institutions," *Fire and Materials*, vol. 36, no. 5-6, pp. 349–366, 2012.

[5] J. Ma, W. Song, W. Tian, S. M. Lo, and G. Liao, "Experimental study on an ultra high-rise building evacuation in china," *Safety Science*, vol. 50, no. 8, pp. 1665–1674, 2012.

[6] C. Xudong, Z. Heping, X. Qiyuan, Z. Yong, Z. Hongjiang, and Z. Chenjie, "Study of announced evacuation drill from a retail store," *Building and Environment*, vol. 44, no. 5, pp. 864–870, 2009.

[7] M. Xi and S. P. Smith, "Simulating cooperative fire evacuation training in a virtual environment using gaming technology," in *Virtual Reality (VR)*, 2014.

[8] C. Li, W. Liang, C. Quigley, Y. Zhao, and L.-F. Yu, "Earthquake safety training through virtual drills," *IEEE Transactions on Visualization and Computer Graph (TVCG)*, vol. 23, no. 4, pp. 1275–1284, 2017.

[9] C. Arbib, H. Muccini, and M. T. Moghaddam, "Applying a network flow model to quick and safe evacuation of people from a building: a real case.," *RSFF*, vol. 18, pp. 50–61, 2018.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[11] L. Chittaro and D. Nadalutti, "Presenting evacuation instructions on mobile devices by means of location-aware 3d virtual environments," in *International conference on Human computer interaction with mobile devices and services*, 2008.

[12] A. Mulloni, H. Seichter, and D. Schmalstieg, "Indoor navigation with mixed reality world-in-miniature views and sparse localization on mobile devices," in *International Working Conference on Advanced Visual Interfaces*, 2012.

[13] A. Mulloni, H. Seichter, and D. Schmalstieg, "Handheld augmented reality indoor navigation with activity-based instructions," in *International conference on human computer interaction with mobile devices and services*, 2011.

[14] J. Ahn and R. Han, "An indoor augmented-reality evacuation system for the smartphone using personalized pedometry," *Human-Centric Computing and Information Sciences*, vol. 2, no. 1, p. 18, 2012.

[15] A. Alnabhan and B. Tomaszewski, "Insar: Indoor navigation system using augmented reality," in *ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, 2014.

[16] M. Penmetcha, A. Samantaray, and B.-C. Min, "Smartresponse: Emergency and non-emergency response for smartphone based indoor localization applications," in *International Conference on Human-Computer Interaction*, 2017.

[17] C. Shao, B. Islam, and S. Nirjon, "Marble: Mobile augmented reality using a distributed ble beacon infrastructure," in *International Conference on Internet-of-Things Design and Implementation*, 2018.

[18] S. I. Roumeliotis and G. A. Bekey, "Distributed multirobot localization," *IEEE transactions on robotics and automation*, vol. 18, no. 5, pp. 781–795, 2002.

[19] E. D. Nerurkar, S. I. Roumeliotis, and A. Martinelli, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *International Conference on Robotics and Automation (ICRA)*, 2009.

[20] J. Kim and H. Jun, "Vision-based location positioning using augmented reality for indoor navigation," *Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 954–962, 2008.

[21] K. Liu, G. Motta, and T. Ma, "Xyz indoor navigation through augmented reality: a research in progress," in *International Conference on Services Computing*, 2016.

[22] G. Gerstweiler, K. Platzer, and H. Kaufmann, "Dargs: Dynamic ar guiding system for indoor environments," *Computers*, vol. 7, no. 1, p. 5, 2018.

[23] Y. Zu and R. Dai, "Distributed path planning for building evacuation guidance," *Cyber-Physical Systems*, vol. 3, no. 1-4, pp. 1–21, 2017.

[24] S.-K. Wong, Y.-S. Wang, P.-K. Tang, and T.-Y. Tsai, "Optimized evacuation route based on crowd simulation," *Computational Visual Media*, vol. 3, no. 3, pp. 243–261, 2017.

[25] H. Liu, B. Xu, D. Lu, and G. Zhang, "A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm," *Applied Soft Computing*, vol. 68, pp. 360–376, 2018.

[26] P. Velagapudi, K. Sycara, and P. Scerri, "Decentralized prioritized planning in large multirobot teams," in *International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[27] S. Liu, D. Sun, and C. Zhu, "A dynamic priority based path planning for cooperation of multiple mobile robots in formation forming," *Robotics and Computer-Integrated Manufacturing*, vol. 30, no. 6, pp. 589–596, 2014.

[28] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[29] H. Ma, T. Kumar, and S. Koenig, "Multi-agent path finding with delay probabilities," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 3605–3612, AAAI Press, 2017.

[30] M. F. Zaeh and W. Vogl, "Interactive laser-projection for programming industrial robots," in *IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2006.

[31] M. Zolotas, J. Elsdon, and Y. Demiris, "Head-mounted augmented reality for explainable robotic wheelchair assistance," in *International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[32] H. Liu, Y. Zhang, W. Si, X. Xie, Y. Zhu, and S.-C. Zhu, "Interactive robot knowledge patching using augmented reality," in *International Conference on Robotics and Automation (ICRA)*, 2018.

[33] U. Rehman and S. Cao, "Augmented reality-based indoor navigation using google glass as a wearable head-mounted display," in *International Conference on Systems, Man, and Cybernetics*, 2015.

[34] J. Sánchez, Á. Carrera, C. Iglesias, and E. Serrano, "A participatory agent-based simulation for indoor evacuation supported by google glass," *Sensors*, vol. 16, no. 9, p. 1360, 2016.

[35] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *Transactions on Robotics (T-RO)*, vol. 33, no. 5, pp. 1255–1262, 2017.

[36] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *International Conference on Computer Vision (ICCV)*, 2011.

[37] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *Transactions on Robotics (T-RO)*, vol. 28, no. 5, pp. 1188–1197, 2012.

[38] H. Strasdat, A. J. Davison, J. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual slam," in *International Conference on Computer Vision (ICCV)*, 2011.

[39] R. Blickhan, "The spring-mass model for running and hopping," *Journal of biomechanics*, vol. 22, no. 11-12, pp. 1217–1227, 1989.

[40] H. Geyer, A. Seyfarth, and R. Blickhan, "Compliant leg behaviour explains basic dynamics of walking and running," *Proceedings of the Royal Society B: Biological Sciences*, vol. 273, no. 1603, pp. 2861–2867, 2006.