Document Title:	Semi-Automated 3D Geo-coding of Large Urban Structures for Deployment of Effective Emergency Response and Communication
Author(s):	William Ribarsky, Ph.D., Kalpathi Subramanian, Ph.D.
Document No.:	242582
Date Received:	June 2013
Award Number:	2009-SQ-B9-K009

FINAL REPORT

October 1, 2009 - August 31. 2012

Semi-Automated 3D Geo-coding of Large Urban Structures for Deployment of Effective Emergency Response and Communication

NIJ Award 2009-SQ-B9-K009

Drs. William Ribarsky(PI), Kalpathi Subramanian(Co-PI)

Department of Computer Science

The University of North Carolina at Charlotte

Charlotte, North Carolina, 28223-0001, USA

Contents

List of Figures

List of Tables

List of Algorithms

ABSTRACT

The goal of this project relates to effective management of emergencies or disasters that impact large urban structures, such as commercial buildings, arenas or stadiums and their surrounding street networks. The project has resulted in the *Effective Emergency Response and Communication(EERC)* system; the system consists of a *command center* that represents the main control center for managing the emergency, a *mobile application* that runs on multiple mobile devices in the hands of responders in the field, and an *evacuation system* that can run simulations of optimal evacuation patterns in the midst of dynamically changing conditions. The system employs a client-server architecture, using a PostgreSQL server and iPhone/iPod devices for the responders and command center to communicate with each other. Our system departs from traditional GIS based solutions in the following respects: (1) *Direct Interaction:* our system provides direct interaction with a *3D representation of the urban structure*, represented as a 3D graph, including all viewing, navigation and routing so that participants can immediately get understandable views of operations in complex spatial environments, (2) *Situational Awareness:* situational awareness is central to our system. It provides the most current information, including evacuation status, for the buildings, occupants, and first responders inside the buildings and to the commander at the same time; (3) *Scalability:* our system is highly *scalable*, with the employed algorithms using a combination of automated processing and graph simplification schemes that have been used to digitize over 70 campus buildings and surrounding streets as well as buildings at other locations; (4) *Dynamic Changes:* our system supports *dynamically changing conditions* during an emergency, which are input to the system for immediate feedback and interactive visual analysis, that includes providing responders with decision support for actions to be taken. The system has been evaluated through a sequence of training events with emergency and safety personnel and first responders. It is now being put to use in various police and emergency planning operations and training.

1 Executive Summary

We have developed an iPhone-based mobile application for first response and emergency evacuation in an urban setting. With it, police officers can move around in large buildings always knowing where they are and shortest routes to exits or any room in the building. They can also transmit position information to a command post and to fellow officers involved in the emergency. In addition, precise routing directions can be issued from the command post, locations of victims or perpetrators can be shared, and the locations of important objects (e.g., fire extinguishers, electrical closets, etc.) can be transmitted. The system is based on *client-server* architecture, i.e. information between the command post and the responders are mediated and transmitted through a server, typically a geospatial data management server. A complete urban environment can be loaded in the server including building 3D routing graphs, building interior information, and street networks. Any piece can then be provided to the mobile users. As an example, we have constructed and loaded all building and street information for the UNC Charlotte campus (74 buildings) and have modeled several other buildings as well for exercises with collaborators.

During this project, we have completed the following tasks:

1. Starting from building CAD files, we developed tools to semi-automatically construct a 3D network of the building geometry (rooms, hallways, elevators, stairways, entrances, exits). We built a flexible and portable client/server infrastructure that supports applications across desktop and mobile device platforms. The client/server architecture developed was centered around a PostgreSQL/PostGIS database server, which proved powerful and flexible.

2. We designed intelligent and highly scalable data processing capabilities that permit: rapid conversion of 2D building data into 3D networks; and editing of models, for modifications or repairs, updates and data input.

3. We created highly intuitive visualization and interaction capabilities that permit rapid

deployment with and acceptance by users for law enforcement applications. Interactive visualization interfaces were developed for both the desktop or laptop command post environment and the mobile device to be carried by the responding police officer. For the latter, a complete iPhone environment was developed with onboard building layout database (for multiple buildings in a neighborhood) and upload/download access to the database server.

4. We worked with different police departments to insure that the server-based system and its command post and mobile elements work effectively in different environments across a range of tasks. A series of exercises and demonstrations was carried out for law enforcement evaluation, disaster response, and consideration at NIJ meetings.

5. We developed the mobile application. Based on user studies, evaluations by law enforcement experts, and exercises, a complete mobile application for emergency response was developed so that it can be used effectively and efficiently.

6. We evaluated the tools and applications employing a combination of police personnel and student users. Based on the evaluations we introduced improvements to the interfaces and new capabilities, as needed. A series of evaluations, user studies, and post-mortems after exercises have been completed, and the interfaces have been improved as a result.

7. We developed methods for efficient large scale evacuation that take into account the finite capacity and bottlenecks in building routes, whose properties can be included in the 3D graph network. These methods were developed and then substantially extended so that updates could be run in real time (e.g., sudden blockages in evacuation routes). A decision support tool was developed so that the evacuation model results could be quickly turned into actions taken by the emergency commander and the responders under his/her command. The methods have been extended to collections of buildings

in a neighborhood and the evacuation routes around them. Evaluation of this model has been done in a tabletop exercise and then as part of the large exercise that includes both the mobile emergency response system and the real-time evacuation decision support system.

8. We completed data acquisition and processing. We developed 3D graph networks for collection of urban buildings that show the efficiency and effectiveness of our network creation approach for a range of building types and sizes. We developed 3D routing graphs for all the buildings on the UNC Charlotte campus, the paths between the buildings, and the street network around them. We have also done building graphs for individual buildings in other cities. We have organized these routing data for use in the evacuation model. We have proved the flexibility of the PostgreSQL/PostGIS server by adding additional layers of information such as positions of fire extinguishers, room content information, paths for those with disabilities, routes with access problems due to locked doors, and class schedules for the campus buildings. The latter permits the Police Chief or other public safety officer to know the number and distribution of people in a building at any hour of the day.

We are now working with Public Safety officers at this university and for the 17 campus university system on a related project to provide emergency routing and evacuation information. We are in discussion to extend this project to multiple campuses in the university system.

In collaboration with the Charlotte Police Department, we have developed scenarios, including search and rescue, to evaluate the effectiveness of the mobile application and its interface. Our evaluations demonstrate that search and route finding using the mobile device is faster and more accurate than undertaking the same tasks with a map or without any aid. We are also participating in the following exercises:

- We worked on incorporating our urban routing and emergency response system into the Canada-US Experiment Emergency Resiliency Environment (CAUSE-ERE) scenario set for demonstration at the end of June. Several DHS program managers and Canadian emergency response managers will be in attendance for the experiment. The initial experiment involves an earthquake in the Pacific Northwest spanning the U.S.-Canada border. Our system is being used for mobile emergency routing and situational awareness in a downtown Seattle building.

- We conducted a shooter exercise on the UNCC campus on Nov. 18, 2011. In this exercise, a person with a gun is moving through a campus building. Ultimately, the person is cornered and neutralized in the building. This is a re-running of a similar exercise that took place 2 years ago in which UNCC Police, Charlotte Police, emergency medical responders, hostage negotiators, and the university executive team took part. One objective of this current exercise was to use the new capabilities of the mobile urban emergency response and evacuation system to address possible shortcomings in the system, in comparison to previous exercises. Among the issues uncovered(in the earlier exercises) was the lack of situational awareness by the executive team (which must make campus-wide decisions) and the lack of a process to manage lifting of the campus-wide lockdown. We collected extensive data from this event and analyzed it. Overall the training event was successful, as per the feedback from the UNC Charlotte Police and our own post-mortem analysis. There were issues in the system interface that suggested improvements. These have been implemented, in preparation for the next training exercise that has been scheduled for August, 2012.

- We carried out a tabletop evacuation exercise with the UNC Charlotte Police and the Public Safety Department using our novel decision support tool for modeling evacuations as they unfold in real-time. Three relevant campus scenarios involving a cluster of several buildings were carried out with the UNCC Police Chief acting as the emergency commander. Approximately 5000 evacuees were handled in each scenario. The tool was able to handle the sudden changes in evacuation routes as the scenarios unfolded. The feedback from the Chief and police officers was positive. They thought the near real-time responsiveness of the tool, the ability to see the layout of the building and surroundings with clearly marked exits, and the distribution of evacuee population in this context so that deployment of assets could be quickly and accurately determined were big pluses.

- In collaboration with the UNC Charlotte Police, we conducted a second joint training exercise July 27, 2012 that combined the capabilities of both the mobile emergency response system and the real-time evacuation simulation and decision support system. A high-rise building, the 12 story Atkins library, was the site for this exercise. Atkins has features somewhat different from a typical campus office building It is a large and complex environment created from a combination of 4 older buildings. There are large open areas, inaccessible areas (rare books collections, for example) and certain doorways that are normally closed to the public. All these features were reflected in the geographical maps and 3D routing graphs in our system. UNC Charlotte Police's Mobile Command Center (MCC) was used as the command post. Charlotte Fire Department personnel acted as observers. Again a shooter in a building scenario was used to stage the event. In contrast to the earlier training event in 2011, there were very few issues in the performance of the EERC system; the post-mortem evaluation and responses from the SWAT officers were highly positive, especially in transmitting their location and receiving orders from the command post. Because of the symmetric nature of the upper floors of the building, there was some confusion in orientation as the officers stepped off the stairwells as they pursued the target. These issues will be taken into account in future updates of the mobile application system.

- We are in the process of scaling up our evacuation system to handle the whole campus. The initial evaluation will involve a table-top exercise with the UNC Charlotte police, in order to understand computational and other issues, in scaling up the evacuation model from a cluster of 5 buildings(that contributed towards a publication) to more than 70 buildings. Future events will be coordinated with UNC Charlotte police, including those involving the new 49er football stadium that opened in 2013. Key features to measure include interactive routing of occupants and vehicles out of campus via specific routes suggested by the campus officers, and move building occupants to specific holding areas.

2 Introduction

Effective management of natural and man-made disasters are becoming increasingly important in order to prevent or minimize loss of life and property. Emergency management, as currently defined consists of four phases: mitigation, preparedness, response and recovery. Of these mitigation and preparedness refer to activities prior to an emergency, while response and recovery are activities during or after the event. The use of Geographic Information Systems(GIS) is currently the technological tool of choice in managing large-scale disasters[24]. GIS systems centrally contain spatial and other attribute data that are relevant to an event and can be used by emergency responders for timely decision making. The extent to which these systems are used depends on a large number of factors, and most importantly, on available resources at the county level.

Traditional GIS based decision support systems have significant limitations[54]; spatial data is in 3D, while most existing systems do not have full support for 3D geometry structures, attributes and textures, and navigation capabilities. Ideally, the rich capabilities of graphics and visualization systems is yet to be translated into today's GIS systems. A very recent work by Lee et al.[32] on using triangle structures based on Voronoi diagrams exemplifies this need. Here the authors propose this structure as a basis for efficient spatial queries, as well as a means to explore what-if style scenarios, in terms of locating infrastructure and other resources for emergency management. A good discussion of significant impediments to using GIS is presented by Zerger et. al.[52] for emergency management, including limitations in spatial data accuracy and quality, limited numerical and real-time modeling capabilities.

In this project, our focus is on effective management of emergencies or disasters that impact large urban structures, such as commercial buildings, arenas or stadiums and their surrounding street networks. We have built the needed infrastructure and tools to build a system that departs from the more traditional GIS based solutions, in the following ways:

- Our system works works *directly with a 3D representation* of the building structure, specifically a 3D graph. Thus all of the needed viewing, navigation and routing calculations are performed conveniently in 3D and displayed on graphics workstations in *real-time*.

- Our system supports *direct interaction* for all needed queries and updates.
- A key aspect of our system is *situational awareness*; information that is displayed to the incident commander or responders is current, facilitating effective decision support.
- Our system is *highly scalable*; algorithms and techniques used in our system were used to digitize over 70 campus buildings.
- Our system supports *dynamically changing conditions* during an emergency; these are input to the system for immediate feedback and *interactive visual analysis.*

Our system, illustrated in Fig. 1, and termed *Effective Emergency Response and Communication(EERC)*, consists of a *command center* that represents the main control center for managing the emergency, a *mobile application* that runs on multiple mobile devices in the hands of responders in the field, and an *evacuation system* that can run simulations in the midst of dynamically changing conditions. The system uses a client-server architecture, using a PostgreSQL server and iPhone/iPod devices for the responders to communicate with the command center.

2.1 Literature Citations and Review

2.1.1 Building Representations

Effective response to a disaster within a building begins with a good model and representation of the underlying structure. In the event of a disaster(such as fire, chemical leak, explosion), the representation should allow rapid evacuation of the occupants from the structure as well as provide quick and guided access for responders to particular location within the structure (a criminal incident within a building). In considering a 3D representation, there are 2 major considerations, (1) Easy and efficient access to neighborhood and adjacency information, and (2) Efficient computation of routes between any two points within a building. Adjacency and neighborhood information helps determine spatial relationships, and helps build higher level or hierarchical building representations. The ability to efficiently compute routes within a building is directly relevant to evacuation or locating paths to a particular point within a building(for example, location of a shooter or a gas leak within a building).

Topology vs. Geometry. A large amount of work has gone into representing n-dimensional objects using topological models, especially with Boundary Representations(BREPS) [43]. Topological models are good at capturing spatial relationships: for 3D BREPS, the relationships are in terms of the cells or simplices and their neighborhoods. Each object is further abstracted in terms of lower level elements: point, line, face, body. While topological models are good at capturing adjacency relationships, they do not include geometric information, which is needed for evacuation or determining location information from urban structures.

Thus, rather than relying on a pure topological model for representing buildings, researchers have focused on higher level representations. In particular, building 3D topological models between the 3D objects within a building: rooms, corridors, stairways, elevators, entrances, exits, walls. Lee and Kwan[31] developed the Combinatorial Data Model(CDM), that represents topological relationships between 3D objects using a dual graph, that interprets the "meet" relation between 3D objects [20]. This graph is an adjacency graph, representing a spatial partition where two adjacent nodes (polygons in 2D GIS, for instance) are linked by an graph edge. This dual space representation is analogous to spatial triangulations and Voronoi diagrams.

The 3D graph representation represents spatial relationships; for this to be useful in disaster response applications, it needs to be augmented with geometric information. Lee et al. proposed a conversion from their CDM to a Geometric Network Model (GNM); in particular, this adds geometric information so that hallways are "stretched" out (rather than being represented as a single node in the CDM) and rooms connect to the hallway at their appropriate geometric locations. Our work[37] follows a similar final representation, but using a different methodology to perform the conversion.

2.1.2 Generating 3D Building Representations

Generating a 3D representation of a building that is ready for use in emergency response applications consists of the following steps: **Building Input Format and Preprocessing.** The source input data for a building may come from paper maps or digital maps such as satellite images or CAD files. The building geometry needs to be georeferenced (properly located) using maps of the region, either manually or automatically.

Depending on the source of the building data, some amount of preprocessing will be necessary. For example, Wei et al.[50] use a sequence of automatic steps to detect and georeference buildings using LIDAR points or by extracting features from the digital images. Additional parameters such as height, area and roof types were also extracted, resulting in 3D polyhedral building models. Their algorithms were further improved by fusing LIDAR data with the corresponding aerial images. While this approach is effective for determining the external geometry of the building, detailed geometry of the interiors of the building is lacking, which is necessary for emergencies within the building.

Most modern buildings have digital CAD or Shape files, that have detailed descriptions of the building geometry(room layout, materials), utilities (electrical, plumbing), presence of chemicals and other hazardous materials, etc. Integrating such critical information into a building model is useful in an emergency situation. The extensive work by Lee et al.[36], Liu et al.[38] focus on constructing 3D building models beginning with CAD or other detailed sources of building data. When no information is available about a building, then a manual construction of the model is necessary.

In the following sections, we assume a detailed description of the building via a CAD file (or equivalent) is available, and describe techniques to build a 3D building model.

Determining the Adjacency Graph. In this step, all adjacency relationships between 3D units are determined. This will then become the basis for determining the 3D building graph. Lee and Kwan[35] built the Combinatorial Data Model(CDM) for capturing topological relationships within a building structure. Their method assumes rooms and other 3D units are orthogonal and axis aligned (X,Y are the horizontal dimensions and Z the vertical dimensions). Adjacency relationships are determined in 2 steps. In the first step, adjacency relations along the horizontal directions are derived from the topological relationships between 2D polygons on each floor. In the second step, adjacency relationships across floors (vertical dirction) are similarly determined between the 3D units. These two steps result in two separate graphs, $G(h_i) = (V(h_i), E(h_i)$ and $G(v_i) = (V(v_i), E(v_i)$, where $V(.)$ and $E(.)$, represent vertices and nodes of the graph $G(.)$. Combining these graphs results in all required topological relationships. They then derive the node-relation structure from the combined graph. Each of the graphs are represented by an adjacency matrix and an incidence matrix. The adjacency matrices are combined by the addition operator, while incidence matrices are combined by the union operator. The final step in building the adjacency graph involves building the hierarchical network: here each floor's master node (typically the node representing the hallway) connecting to other units are connected across floors to other master nodes.

Medial line Extraction. Once the topological relationships within a building is determined, the next step is to determine paths within the building that will be used for evacuation or routing. Paths typically begin within rooms or entrances and end up at exits or other points within a building. To determine these paths automatically, we need to determine the center or medial line of the corridors/hallways that make up each floors. Routing between floors will be via stairways or elevators.

Voronoi diagram based methods have been used by researchers to compute the medial axis of polygonal or raster data[11, 42, 33]. The Voronoi diagram represents a subdivision of space into regions whose points are closer to a generating vertex than any other element. Algorithms for computing the Voronoi diagram have been well studied[51, 19]. The intersections between the Voronoi edges converge to the polygon centerline. Lee[33] used a simplified medial axis transformation (MAT) to determine the medial lines of polygons representing corridors of the building. The medial axis transformation[10] is a shape representation method; it is computed by either circle fitting methods or using thinning methods. Lee's method computes the straight medial axis transformation, a simplified and linearized version of the MAT; in his method, the angular bisectors of consecutive pairs of convex vertices of the polygon are intersected; repeated passes of this process (succeeding passes use the new rays generated by the bisectors from the previous pass) results in the medial axis of the polygon.

A second class of methods is based on rasterizing the space under the polygon. Potential field based methods are one example in this class. The pixels that intersect with the polygon edges are the sources of potential forces. First proposed by Chuang et al.[15], these methods evaluate the forces at each pixel, using an integration of the incoming forces from all visible pixels. Critical points, where the force sums to zero(or a minimum) are located and tracked along the force direction. These points are connected to make up the centerline[17, 16].

Distance functions have also been the basis for medial line extraction. A distance function is a function representing distance from certain source points, and most often, referring to the distance-to-closest surface, or, Distance From Boundary(DFB). Such distant maps have been used to extract skeletons [23].

Distance fields are usually combined with Dijsktra's algorithms (shortest path, minimum spanning tree) in order to extract the object medial line; the idea in these schemes is to transform the object points (identified in a preprocessing step) into a weighted graph, with the weights being defined by the inverse of the computed distance metric. Then Dijkstra's algorithm is applied to find the shortest path between specified end points. Liu et al.[37] compute a distance field after segmenting the objects from the background, and use minimimum spanning tree algorithms to extract the medial line. Additional refinements were implemented to ensure the medial line is continuous, especially with hallway geometry that might loop on itself. They applied their method to two academic buildings. Chen et al.[14] used a similar approach but modify the shortest path points to the maximal DFB points orthogonal to the path. Bitter et al.[7, 9] use a heuristic that combines the DFB and an additional distance metric, distance from source (DFS). This helps discourage the "hugging corner" problem, that is typical of shortest path based approaches. Similar ideas were explored by Wan et al.[48].

Determining the 3D Building Model Once the topological relationships within the various 3D units in a building is determined, these will be converted into a 3D building network; in conjunction with the medial line extraction procedures, we will be be able to compute routes, that will be useful in evacuation or other location specific spatial queries. This requires a conversion from the adjacency information (adjacency or topological graph) into a 3D building model, generally represented as a graph. Finally, integration of the building network to street networks will complete the integration of the building to the urban or transportation networks.

Lee's method[33] converts the combinatorial data model into a Geometric Network Model(GNM). The straight medial axis transform converts hallway polygons into linear features. Each 3D unit on a floor, if connected to the hallway is projected to the hallway edges in a direction orthogonal to the medial line. The algorithm is $O(NM)$, where N is the number of nodes and M is the number of edges on the medial axis. This procedure is repeated for each floor i, generating a geometric graph $Nh_i = (V(Nh_i), E(Nh_i)$, where as before $V(.)$ and $E(.)$ are the vertices and edges of the graph Nh_i. These floor graphs are combined with the vertical graph $Nv = (V(Nv), E(Nv))$; vertical graphs are connections made via stairways and elevators between floors.

2.1.3 Evacuation Models

The study of computer based evacuation modeling has evolved with both mathematical and algorithmic approaches. These approaches can be categorized as macroscopic and microscopic. Macroscopic approaches focus primarily on a lower bound of egress time. The evacuees are treated as a unit or group of units and moved from source to destination, or safe zone. Interaction between these units can be defined with capacity/congestion rules and the primary goal is to minimize the time it takes for all evacuees to reach an exit. Microscopic approaches use agent based modeling, where each evacuee is governed by unique rules of behavior. Interaction between individuals and their environment can be defined based on spatial and social parameters. These methods take into account individual behavior as well as modulate their behavior based on local constraints, thereby attempting to create more realistic evacuation scenarios.

Capacity Based (Macroscopic). The inputs for this approach are a graph structure and evacuee populations. The output is a route plan with start times, and a location matrix for each evacuee for each time segment defined. This is, therefore, an a priori solution that is evaluated based on the time it takes for each evacuee to reach a safe zone or exit. This evaluation is applied whether the solution is optimal or sub-optimal. The evaluation of the route planning algorithm is based on processing cost. Obviously, the evaluation of a given approach simply based on minimizing the evacuation time does not consider the details of the complexities of the individual evacuees in the process of moving through their environment. For this reason, the application of network flow, capacity, and congestion restraints are applied to add reality to the challenges of moving within the environment to be evacuated.

This is the reason the Network Flow problem is referred to as optimal because it can represent the conditions of each evacuee during an egress process. Of course, the term optimal must be considered from the context of the methods for evacuation planning because flow, capacity, and congestion cannot represent all the factors that effect individual movements within an environment. Heuristic approaches such as the Capacity Constrained algorithms proposed by Lu et al.[40] attempt to find lower cost algorithmic solutions at the further expense of the detail of each evacuees egress. However, these approaches are interesting because they can be evaluated quickly from a user perspective. For example, the route planner algorithm can be modified, executed, and visualized in an animation relatively fast. This can yield a quicker way to analyze the algorithm for individual behaviors, at least from a spatial perspective, as the route plan moves from time step to time step.

Network Flow algorithms were applied to building evacuation planning beginning with work funded and developed under the sponsorship of the National Bureau of Standards, Center for Fire Research in the late '70s and early '80's. An application named EVACNET+ was produced by Francis and Kisko in 1984. Further development of these approaches resulted in EVACNET4 which "takes the network model one provides and determines an optimal plan to evacuate the building in a minimum amount of time. This is done using an advanced capacitated network flow transshipment algorithm, a specialized algorithm used in solving linear programming problems with network structure. From the user's point of view, all the user does is supply the model, ask EVACNET4 to run it, and then examine the results." [21].

Linear programming methods based on Network Flow yield optimal solutions. Though these solutions are optimal their algorithmic cost is high. The Maximum Flow Problem is one network solution that is implemented with costs as high as $O(nm^2)$[40] and yielding a best known cost of $O(nm \lg n^2/m)$, where n is the number of nodes and m is the number of edges in the time expanded graph. Because the processing cost of these polynomial time algorithms is so high, in general they are considered to be impractical. Over the last 20 years 'optimal' plans like the algorithms developed by Hoppe and Targus have been modified as general grid network problems, where network arcs have constant length. The work of Kamiyama et al.[27] applies two initial conditions to the network flow problem. For each vertex the sum of transit times of arcs on any path takes the same value, and for each vertex the minimum cut is determined by the arcs incident to it whose tails are reachable. They refer to these as having uniform path-length and being fully connected. Their work proves that for a 2d grid network the transshipment problem can be solved in $O(n \lg n)$ time.

Our model is based on a heuristic approach, the Capacity Constrained Route Planner algorithm proposed by Shekhar et al.[40]. This approach attempts to find lower cost algorithmic solutions at the expense of the detail of each evacuee's egress. These approaches are interesting because they can be evaluated quickly from a user perspective. For example, the route planner can be modified, executed, and visualized relatively fast. This can yield a quicker way to analyze the algorithm for individual behaviors, at least from a spatial perspective, as the route plan moves from time step to time step.

The work of Shekhar and Yoo[44] compares models relevant to the study of nearest neighbor paths. Also Kim, et.al[29] discuss contraflow in reconfigured networks for emergency route planning. This work gives insight into reconfiguring a network that is damaged and therefore relevant to our work. Since we have only considered multi-directional paths in buildings these methods have not been implemented.

We utilized a heuristic approach, so as to reduce the optimal network flow problem to a generalized shortest path problem. Heuristic approaches have reduced algorithmic cost but are referred to as sub-optimal; however, we find that they work reasonably well for our current scenarios. The inputs for this approach are a graph structure and evacuee populations. The output is a route plan with start times, and a location matrix for each evacuee for each defined time segment. This is therefore an a priori solution that is evaluated based on the time it takes for each evacuee to reach a safe zone or exit.

Agent Based (Microscopic). Microscopic evacuation planning simulates the detailed interaction of individuals with their environment including fellow evacuees. These approaches do not exclude the deterministic path or goal that is the foundation of the route planner, but they primarily rely on behavior rules applied to each evacuee to overcome the “lack of detail” inherent in network flow or heuristics based planners. These methods are also referred to as Agent Based Models, or ABM. They integrate the routing logic and time expansion into one process, therefore they are used for visual evaluation rather than graphical or chart based evaluation. However, the goal remains to minimize the time to evacuate individuals to an exit or safe zone. Certain ABM models can exhibit more complex flow patterns (e.g., swirling or chaotic patterns) than are possible with macroscopic models.

A general description of these approaches can be found in [13]. The most important aspect of ABMs are the rules applied to each evacuee. There are no globally defined rules of behavior but there are common applications of constraints placed on the agents. Castle et. al. [13] include a detailed list of some rules or attributes.

- **Autonomy:** Agents receive and transmit information to make individual decisions.
- **Heterogeneity:** Agents with identical rules/attributes can be grouped.
- **Pro-active/Goal-directed:** Agents can be directed on specific paths.
- **Reactive/Perceptive:** Agents can be programmed with prior knowledge and awareness.
- **Bounded Rational:** Limits must be placed on agents ability to analyze their environment.
- **Communicative:** Agents are required to add and depend on information from each other.
- **Mobility:** Spatial roaming and interaction can be implemented.
- **Adaptive-Learning:** Agents can be trained.

Because of the adaptive parameter of ABM, neural networks and fuzzy-logic (if-then-else)[39] approaches are adapted to building evacuation simulation. Discrete Particle Swarm

Optimization[22] can also be applied. Velocity and spatially based[25] rules of interaction between agents in a simulation are also a common approach.

Neural Networks, Fuzzy Logic Methods. These methods begin with the training of an adaptive neural network based system. The evacuation simulations are conducted numerous times until a predetermined best case scenario is developed by an adaptive learning process from information stored and flowing through the network. Obviously the execution of this process will be time consuming based on the accuracy required by the prerequisite definitions of the simulation. According to Lo et. al. [39], even though computing power has increased significantly over recent years these processes are very expensive from a processing cost perspective. Processing power must be significant for, large scale, real world situations and may be exclusive based on the size of a given scenario.

A general description of the process, described in [39], illustrates an overview of the neural network fuzzy logic approach. The authors propose an Adaptive Network based Fuzzy Inference System (ANFIS), to accomplish the task of predicting pre-evacuation behavior, and a predictive back propagation model to predict evacuation response. An example of a Fuzzy Logic rule applied in their implementation is: “if age is young, fire experience is limited and fire cue is intensive flame or smoke and acquired directly, then initial reaction is to inform others. Other behaviors are defined based on the parameters age, fire experience, and fire cue. Obviously, these rules require knowledge of the demographic of the evacuees which in most cases will be unknown. For the analysis of the ANFIS system questionnaires were distributed to 150 individuals that were evacuated from 3 different fires in Hong Kong, China.

Discrete Particle Swarm Optimization. The PSO algorithms were developed from the study of bird flocks and fish schools to attempt to mathematically analyze the behavior of this phenomena during panic situations. Therefore the PSO is applied in the area of emergency evacuation to attempt to predict the actions of human evacuees during emergency situations in a building evacuation. The work of Fang[22] is an adaption of the Cellular Automata (CA) PSO with a leader function included. The intent of this study is to simulate results from an evacuation scenario which includes one source and one exit to illustrate the feasibility of the proposed approach in representing evacuation phenomena such as jamming and clogging.

In general formulas are applied to represent the velocity of evacuees in successive iterations of the PSO algorithm. The selection of variables from the leader is used as a best performing evacuee and then applied to the other evacuees in the next iteration. The leader, in the case of this example, is simply chosen as the evacuee closest to the exit of the original source destination. An ABM for the interaction of these evacuees is therefore present for the analysis of the actions and behavior of evacuees at choke points in the model. It is important to note that this is simply another mathematical solution to software analysis of evacuee behavior at the agent level. The results could be compared to the same approach using an different ABM such as a Neural Network adaptive approach.

Velocity Obstacles(VO). These methods have their roots in robotics. The relative speed and location of nearby objects are calculated and applied to an algorithm that adjusts individual velocities to avoid collisions. This ABM is a hybrid approach related to particle behavior and collision avoidance. These methods are important if the implementation requires a real time visual solution for analysis. This is not a goal in neural networks or PSO methods.

The following is a general description of a hybrid method from Guy et. al. [25]. The authors' goal is to produce avoidance algorithms for real time simulations. If two planar objects are moving in the same plane they are given known velocities of all other objects that could potentially cause a a collision at some time after a given moment in time. These constraints can be visualized as a cone in the plane. The intersection of the cones represent collision points. The information needed by each object is stored and retrieved efficiently via a K-d tree structure. Each object is then moved in its designated direction for a designation time slice and velocities are adjusted for each object in the plane to avoid collision in the next time slice.

The authors claim a 10 fold reduction in processing cost from previous applications of Velocity Obstacle based methods. They have also created a software package called Clearpath. The package is multi-threaded to improve the Extended VO's real time performance on multi-core processors. This allows the number of simulated objects to be increased for the best real time performance on a given multi-core processing system. Their results indicate guaranteed collision avoidance with nearly 20 frames per sec(fps) on a quad processor machine while processing 25K objects, from a simulated stadium scene.

We believe our system captures sufficient detail using a congestion model. However, because agents can be trained, we are evaluating Q-Learning/SARSA techniques to add human factors to future work.

2.1.4 Visual Analytics

Visual analytics involves effectively combining interactive visual displays with computational transformation, processing and filtering of large data[47]. One focus of visual analytics is real-world problems involving situationally-aware decision support. Andrienko et al.[3] focused on automatic generation of transportation schedules for evacuation from a disaster zone; visual analytic tools were used for verification by human experts. Campbell and Weaver[12] used interactive visualization tools for hospital evacuation scenarios that involved training first responders. The work of Kim et al.[28, 30] focused on use of mobile devices for situationally aware emergency response and training, and thus their approach is similar to our work. They demonstrated their system with an evacuation simulation of the Rhode Island club fire of 2003. Our system is considerably more general and is scalable to large urban buildings and provides the means to interrupt the simulation based on new situational information or dynamic changes.

The use of linked views is an important technique to connect different representations of information within a single visualization, with applications specific to urban structures[41, 26]. Sensor networks are used within buildings to help create interactive visual analytics. Visual analytics tools such as Jigsaw[46] are available for integrating process output data from an application. We are using custom code in our system, therefore toolkits are not utilized.

The work Ivanov et. al[4] is not specific to building evacuations; however, their use of data graphs to interact with maps provided inspiration for our cross platform bar chart displays that interact with our simulation. This work is from a growing segment of the visual analytics referred to as the visual analytics of movement or Geo-analytics. These scientists are focused on the problem of visualizing spatial and temporal datasets in geography and cartography. From our perspective, interactive visual analytics of building evacuations are essential and provide an intuitive interface for emergency planners, and central to our system. We have split our data into two primary parts: an animation and a cross connected interactive display. The animation display contains both spatial and temporal data and the user can view an evacuation as an animation and move back and forth in the animation. The interactive section defines the evacuation from the perspective of significant events and congestion which is a primary concern in evacuation. In many ways these can be considered as Geo-Analytic approaches.

This approach is partially implemented in the visualizations of the route planner activities of our application. As we are concerned primarily about congestion(based on feedback from our users) and this visualization is only applicable to movement, it is simply mentioned here. Continuing work and the use of other evacuation algorithms will result in a need for the scenario build functions to incorporate path analysis.

Ivanov et. al[4] also present an approach where the analysis of movement is visualized by segmenting time. A plot of multiple objects moving in space for a given period of time can cause what they refer to as *worms*. The longer the worm for a given object over a given amount of time on the 2D plane of a map, the faster the object is moving; in other words, speed is mapped (proportionally) into the worm length. In our implementation, we discretize the space traveled by the occupants and set a pre-defined capacity for each segment. Thus, speed is controlled by the capacity within the next segment; occupants have to wait if the next segment is already occupied. As our application becomes more concerned with large outdoor spaces where discretization of space is less practical from a computing perspective these approaches will become invaluable.

3 Methods

3.1 System Design

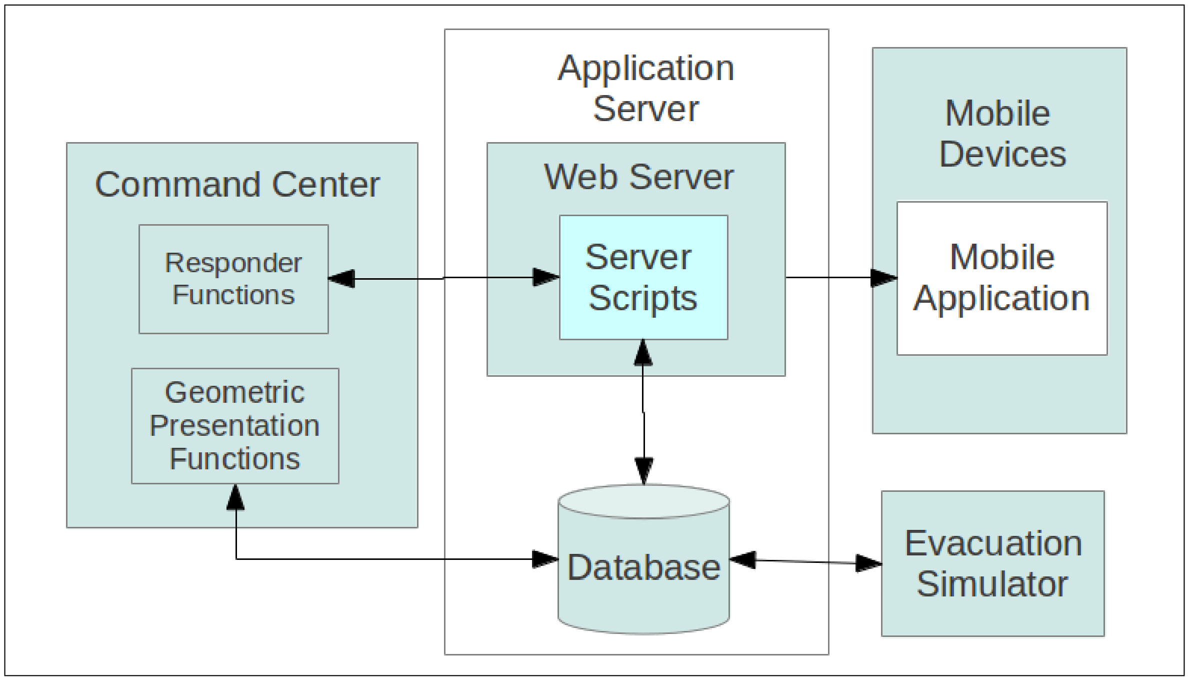


Figure 1: The EERC system consists of 3 primary components, all connected through any available network infrastructure. The Command Center is a desktop application, while the database resides on a remote machine with an Apache Web server for HTTP access. Mobile devices connect through a wireless network and access the system via HTTP.

The Effective Emergency Response Communication(EERC) system consists of 3 primary components, connected via any available network infrastructure(cellular, IP, etc). The *Command Center* is a desktop application and represents the staging area of any incident and serves as the location that integrates information from a variety of sources. Mobile devices, part of the *Mobile Application*, represent responders and connect through a wireless network and access the system via HTTP. The *Application Server* runs a web server, taking HTTP requests from responder functions on the command center and the mobile application. PHP scripts are implemented as simple channels receiving and passing data onto the database and back to the requesting application, for insert, delete, update, and select requests. Currently, the server is available to the command center and mobile applications through routes from the campus wireless network.

The *Evacuation Simulator* currently shares the data and geometry infrastructure and will run evacuation simulations under dynamically changing scenarios of the urban environment. In the future, it will become a more integral part of the EERC.

3.2 Generating Building Networks

Fig. 2 shows the framework for constructing the 3D building network, starting from the building CAD files. After some preprocessing, the adjacency relationships between different polygon elements in each floor is determined and represented as a 2D graph. Elements relevant to determining evacuation routes such as hallways, stairways, elevators and entrances are interactively identified. Next, centerlines of hallway polygons are extracted. Using the adjacency graph, all rooms adjacent to the hallway are connected to the nearest centerline point, resulting in a 2D network for each floor. Finally, the 2D networks from each floor are linked by the stairways, elevators and the building entrances, resulting in a 3D building network. All processed data is maintained in a PostgreSQL/PostGIS database[2, 1].

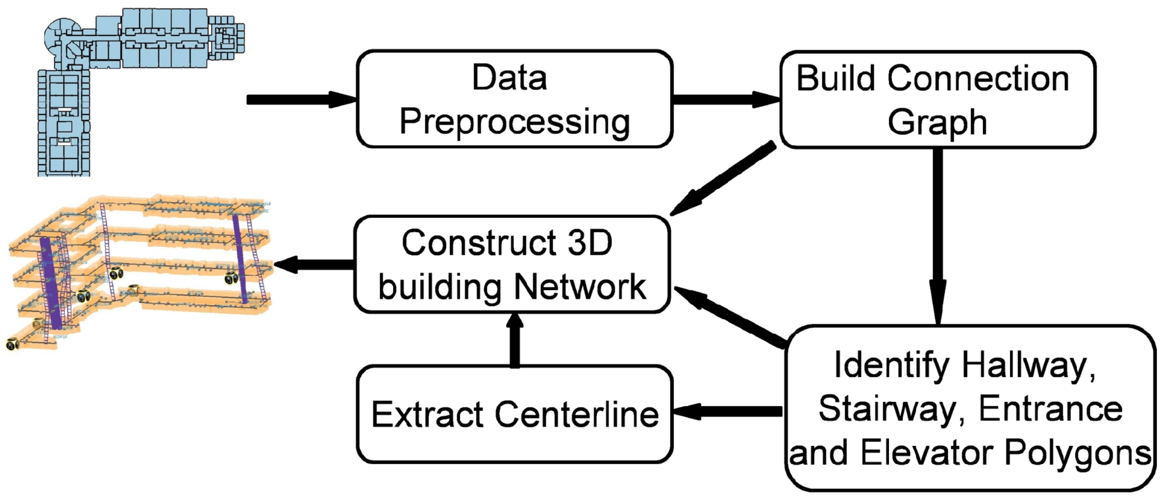

Data Preprocessing
Build Connection Graph
Construct 3D building Network
Identify Hallway, Stairway, Entrance and Elevator Polygons
Extract Centerline

Figure 2: The process of constructing 3D building network.

3.2.1 Data Acquisition and Preprocessing

Data acquisition begins with the CAD building files that will ultimately be incorporated into the PostGIS database. These drawing files are are read by ArcGIS(ESRI) software. Since these files contain data in 2D (each floor is on a separate file), and miss information on the connections between the floors(staircase, elevators), some manual processing is needed to prepare the data for 3D processing and graph construction. Additionally, elimination of data errors and consistency checking is necessary to avoid problems down the line.

Since the CAD files contain no spatial reference information, they must first be *georeferenced* to a base map so as to line up with other datasets. We use the *Spatial Adjustment* tools in ArcGIS to define control points on the base map and point them to the same areas on the CAD building file. These are usually corners of the building or other defining features that will allow the algorithm to line up the data correctly with minimal distortion. Using a building footprint file or a rectified orthophoto allows a georeference to a coordinate system (in our case, NAD1983) and "spatially enable" our data.

Once georeferenced, ArcGIS is used to read the files in their native format and convert to Shapefile format. Four shapefiles are produced for each CAD file, based on geometry type: point, line, multipatch, and polygon. Currently we only use the polygon data, which contain rooms, stairways and elevators. Next, all unneeded data are removed, and the files are "cleansed". A tag in the attribute table of the shapefile identifies the room, stairway, and elevator polygons. These are labeled as RM$, and a simple SQL query is used to select and insert them into their own separate file.

The "cleaned" files are uploaded into a central PostgreSQL with PostGIS database. This database serves as the central data server for the application, and allows updates to be propagated down the line to any device reading from it. We use the database as the input source for the 3D building network construction, visualization, and other analysis tasks of the building.

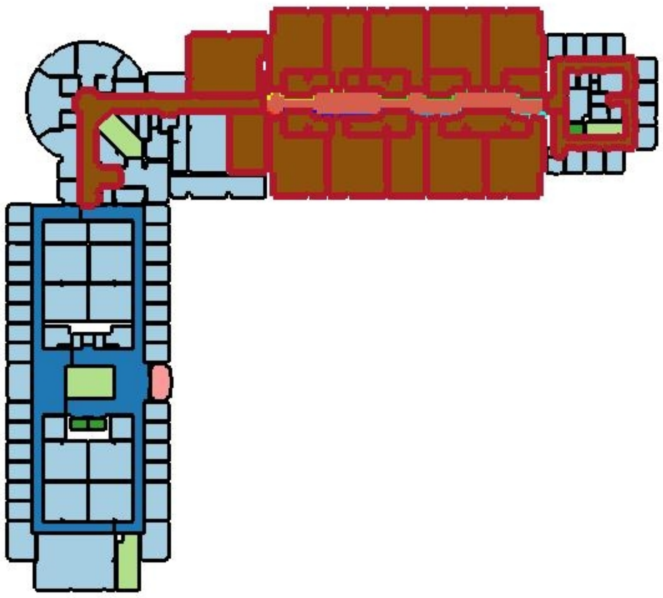


Figure 3: Adjacency Graph Construction. A floor of a building is illustrated with a hallway polygon selected (light red). The automatically selected adjacent rooms and hallway polygon are illustrated (dark orange polygons). Building stairways are shown in green, and entrance to the building is in light red(next to the stairway, middle left)

3.2.2 Adjacency Graph Construction

Construction of the 3D building network requires knowledge of the spatial relationships between the elements that make up each floor and the links between the floors. However, the input database contains simply a collection of polygons that represent these elements at each floor. Thus we begin by analyzing the spatial relationships between the polygonal elements. For the sake of spatial analysis, we consider all polygonal elements to be rooms, except for hallways, stairways, and elevators. The goal is to recover and represent the relationship between these elements in an *adjacency graph*, where nodes represent polygonal elements and edges connect neighboring polygons.

Once the input polygonal data is read from the PostGIS database, each polygon's vertex ordering is checked, so as to be in counter-clockwise order, ensuring consistent orientation. Duplicate vertices, if any, are removed. Then the polygons (which may be concave) are converted into triangles using the Delaunay triangulation[5], to simplify geometry analysis.

An interactive tool is also provided to specify polygons, if the triangulation fails.

Let $\mathbf{T^i_k}(\mathbf{v^k_1}, \mathbf{v^k_2}, \mathbf{v^k_3})$ define triangle k within polygon i, with vertices $v^k_n, n \in (1, 3)$. Similarly, a polygon is represented as $\mathbf{P_k}(\mathbf{v^k_1}, \mathbf{v^k_2}, \ldots, \mathbf{v^k_n})$. Suppose the current polygon $\mathbf{P_i}(\mathbf{v^i_1}, \mathbf{v^i_2}, \ldots, \mathbf{v^i_n})$ is being processed to determine its spatial relationship with polygon $\mathbf{P_j}(\mathbf{v^j_1}, \mathbf{v^j_2}, \ldots, \mathbf{v^j_m})$. There are two steps to constructing the adjacency graph.

- **Triangle-Polygon Relationship.** In the first stage, each boundary vertex $\mathbf{v^j_k}$ of polygon j is evaluated as inside or outside of triangle $\mathbf{T^i_k}(\mathbf{v^k_1}, \mathbf{v^k_2}, \mathbf{v^k_3})$ in polygon i. For computational efficiency, we circumscribe each triangle, with $\mathbf{o^i_k}$ as the center and r^i_k as the radius. A boundary point is inside a triangle if

$$\sqrt{\|\mathbf{o^i_k} - \mathbf{v^j_k}\|^2} < r^i_k + \alpha \tag{1}$$

  where α is a constant value.

  If all boundary points of polygon j fail Eq. 1, it can be safely removed. Otherwise, it could be adjacent to polygon i and is retained.

- **External Triangle-Triangle Relationship.** In the second stage, the triangles of polygon j are analyzed for adjacency. Following the same strategy, circumcircles are constructed for each triangle of polygon j. Adjacency relationship is similarly defined between two triangles $\mathbf{T^i_k}$ and $\mathbf{T^j_k}$ as,

$$\sqrt{\|\mathbf{o^i_k} - \mathbf{o^j_k}\|^2} < r^i_k + r^j_k + \beta \tag{2}$$

  where β is a constant value. Based on Eq. 2, the accuracy of the neighborhood relationship between two polygons can be further improved by reducing β, resulting in identifying additional pairs of adjacent triangle pairs.

Fig. 3 illustrates an example output from our method. Here we see a hallway polygon

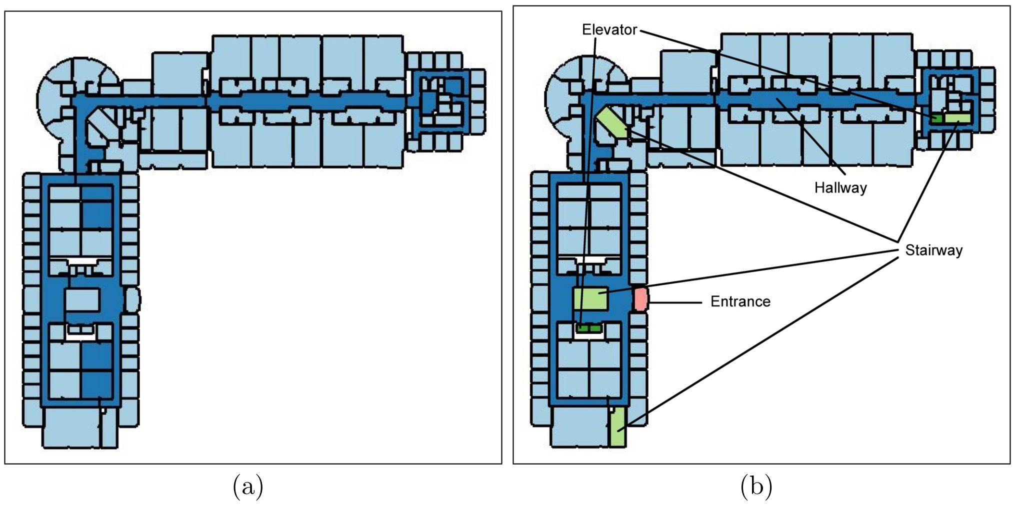


Figure 4: Identifying hallways, stairways, entrances, (a) Hallway polygons in dark blue have automatically been identified, (b) stairways (light green), elevator (dark green) and entrance (light red) were interactively identified, since this information is not part of the input CAD files.

(in light red) that has been selected. The neighboring polygons (in deep orange) that were identified are displayed, including an adjacent hallway polygon.

3.2.3 Identification of Hallways, Stairways, Elevators, Entrances

After the adjacency graph is constructed, we next identify polygons that represent hallways, stairways, and elevators. This is needed prior to centerline extraction, as well as for building network construction. Hallway polygons are typically adjacent to a large number of rooms(for example, corridors leading to offices in a commercial building); thus, its corresponding node has more outgoing edges (high degree node) than other nodes and can easily be segmented from the rest of the graph. In addition, we provide interaction tools to let a user modify the results, if needed. Stairway and elevator polygons are similar to room polygons. If these are part of a building CAD file, they can be automatically identified and marked appropriately; otherwise, our system will let the user specify these types of polygons. Fig. 4 illustrates the process. In Fig. 4(a), automatic processing identifies most of the hallway polygons. In Fig. 4(b), stairways, elevators and entrances are interactively specified, as this information

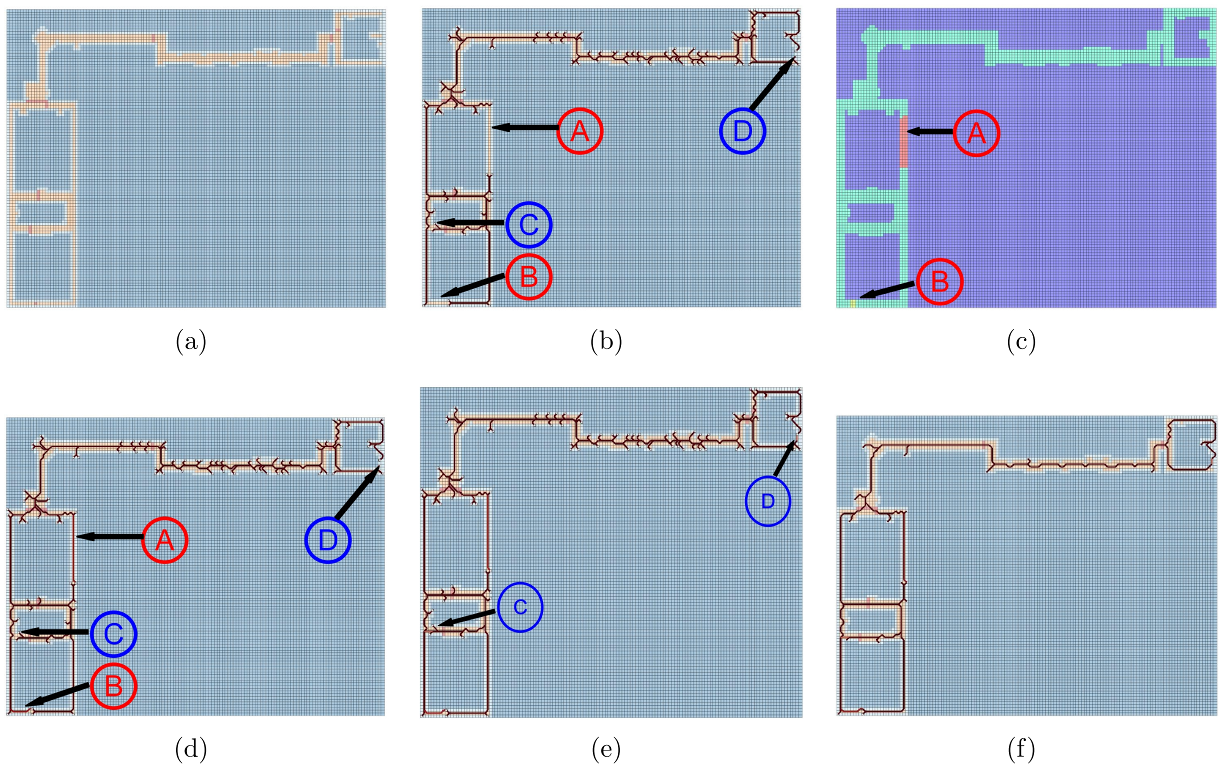


Figure 5: The process of centerline extraction. (a) Construct grids covering hallway polygons, (b) Centerline extraction by using minimum-spanning tree; (c) Identify missing segments through "circle-rolling" method, (d) Recover centerline segments in uncovered regions, (e) Manually repair small gaps, (f) Fulling centerline after pruning spurious branches. Red circles (A) and (B) denote large missing segments which can be detected by "circle-rolling" method, and blue circles (C) and (D) represents small gaps, which can be interactively repaired.

is missing from the input CAD files. elements

3.2.4 Centerline Extraction

A key application of emergency management within buildings is the determination of evacuation routes. These typically involve hallways, stairways, elevators and entrances/exits. Automatic construction of evacuation routes is facilitated by centerline extraction algorithms (also proposed by Lee[31]). Relevant methods for centerline extraction include those based on Voronoi diagram, potential field and distance field.

Voronoi Diagram. The Voronoi diagram represents a subdivision of space into regions whose points are closer to a generating vertex than any other element. Centerline algorithms using the Voronoi diagram[11, 42, 34] begin by sampling the polygon boundary and constructing the Voronoi diagram. The intersections between the Voronoi edges converge to the polygon centerline, as the boundary sampling rate is increased. One problem with this method is the difficulty in joining centerline segments from separate but adjacent *hallway* polygons.

Potential Field. Potential field based methods begin by uniformly subdividing the space underlying the polygon. The pixels that intersect with the polygon edges are the sources of potential forces. First proposed by Chuang et al.[15], these methods evaluate the forces at each pixel, using an integration of the incoming forces from all visible pixels. Critical points, where the force sums to zero(or a minimum) are located and tracked along the force direction. These points are connected to make up the centerline[17, 16]. Our experiments with this method displays instabilities, due to difficulties in visibility determination, especially among the more complex concave *hallway* polygons.

Distance Transform. Similar to the potential field method, these approaches use a *distance function*, a signed function from certain source points. Points with maximum distance(local maxima) are extracted and organized into the centerline by using shortest path[53, 6, 8] or minimum spanning tree [49] algorithms.

We chose to use the distance transform based method, for the following reasons, (1) outside of the distance field calculation, the centerline extraction algorithm is quite efficient, (2) the centerline is guaranteed to be inside the structure, (3) it avoids visibility determination after pixelization and segmentation of object and background pixels, and (4) Separate but adjacent polygons can easily be joined by detecting shared boundary points. Our approach is illustrated in Fig. 5.

1. **Grid construction.** The space containing the polygons is first pixelized by uniform subdivision. Boundary pixels are identified (Fig. 5(a)) as the intersection between grid and the polygon edges. Seed points are selected inside the polygons and a region

growing algorithm is used to identify the object pixels (yellow). the boundary pixels are changed into object pixels (red) if they belong to both (adjacent) polygons. Thus, the adjacent hallway segments can easily be merged.

2. **Centerline-Tree Construction.** The distance transform is performed on object pixels by considering the boundary pixels as source points. Then, the minimum-spanning tree can be built by using the inverse distance value as the weight, and Wan's[49] algorithm is used to output the centerline-tree.

3. **Recovering Missing Centerline Segments.** Parts of the centerline segments may be disconnected (see the marked circles in Fig. 5(b)) in the centerline-tree because the hallway is a loop, while the centerline structure is a tree. These gaps need to be identified and reconnected. To do this automatically, we use a *rolling-circle* scheme. A circle is centered at a centerline point $\mathbf{c}$ and radius equal to $r = 1.414 \times DFB(\mathbf{c})$ where $DFB(\mathbf{c})$ is the distance value of the current point. This circle is rolled along the centerline and its radius is dynamically varied, based on its current location. The regions not covered by the rolling circle are potentially broken segments (Circles A,B in Fig. 5(c)).

   The missing segments are repaired by determining two centerline points on either side of the uncovered region. Ideally, a shortest-path connecting these two lines will be the optimal route. However, it is likely to traverse the same path extracted using Wan's[49] algorithm. We propose two metrics to fix this:

   - Rather than directly extract missing centerline segments from the two selected points, we identify a point in the uncovered region with a local maximum in distance value, as well as being the farthest from one of the selected points. Then Wan's algorithm is used to connect these two points, followed by similarly recovering the gap with the second selected point.

- Next, the weight of minimum-spanning tree is modified. We add a penalty value to ensure that the selected path is always the shortest, by computing the distance from the seed point (DFS) to build the minimum-spanning tree. Thus, the weight is defined as

$$weight = 1/DFB + DFS \tag{3}$$

Fig. 5(d) shows the recovery of two centerline segments(circles A, B).

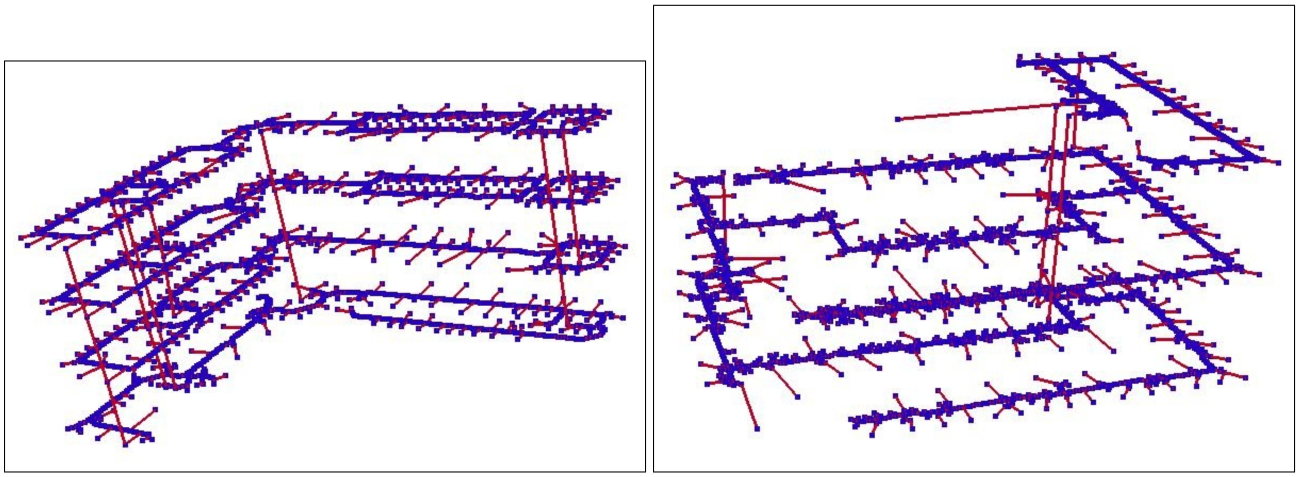
4. **Interactive Centerline Repair.** Small gaps (circles C, D) cannot be detected through the circle-rolling method. At present, we provide an interactive tool to let a user manually fill in these gaps. It is efficient since these are small gaps and the user is only required to click two points; our system automatically searches for the two extracted centerline points closest to the user's choice and the centerline gap is filled. Fig. 5(e) shows the repaired results, where two short centerline segments fill the gaps in circle C and D.

5. **Spurious branch removal.** In this step, a threshold is used to eliminate spurious branches associated with centerline extraction. The threshold is based on the point count of the branch segments. However, this process will only affect the centerline branches originally extracted (not from steps 3 and step 4) as well as having no offset branches. Fig. 5(f) illustrates the final centerline by pruning spurious branches.

3.2.5 3D Building Network Construction

To construct the 3D building network, we first build the 2D networks for each floor. This is accomplished by using the adjacency graph and the extracted centerline points. Rooms connected to hallway polygons are determined from the adjacency graph. Their centroids are then connected (they become graph edges) to the nearest centerline point.

After the 2D networks are constructed for each floor, the 3D network is constructed by linking corresponding stairways and elevators in different floors(represented by graph



(a) (b)

Figure 6: Example networks of 2 campus buildings. Blue line segments represent the hallways, while the red segments represent connections to the rooms (offices) as well as the connections between the floors via stairways and elevators

edges). These are then linked to the building entrances, enabling analytic tools to compute evacuation routes between any two nodes in the network.

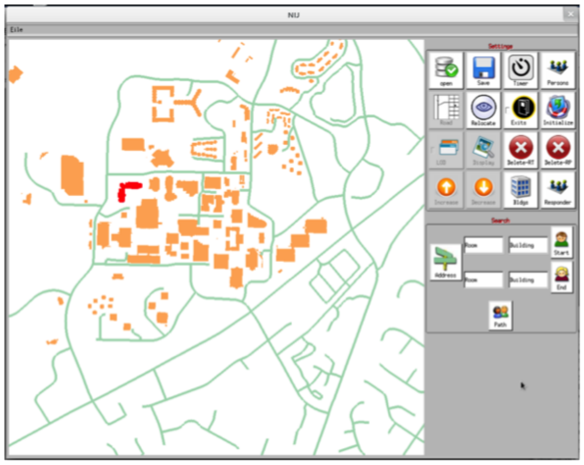
Fig. 6 illustrates two building networks resulting from all of the automated and interactive processing. The blue points represent the centerline, while the red segments are connections(graph edges) to the adjoining rooms. Red segments between floors represent stairways and elevators.

3.3 A Mobile Communication System for Rapid Emergency Mitigation and Response

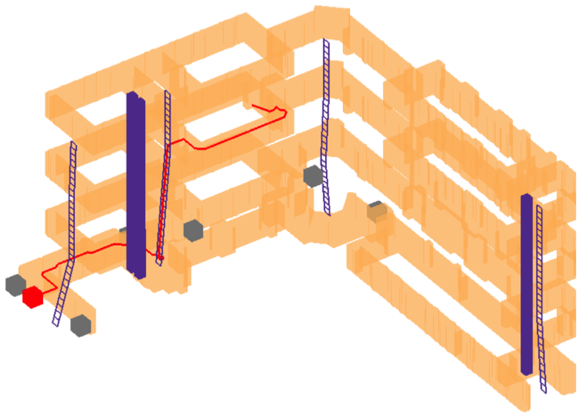
3.3.1 Command Center

The command center application is currently a desktop application. We have used our campus office and residential buildings as the first test case for the system. Thus, the command center loads these datasets and presents an interactive 3D view of the campus and associated buildings. The command center application has the following functions:

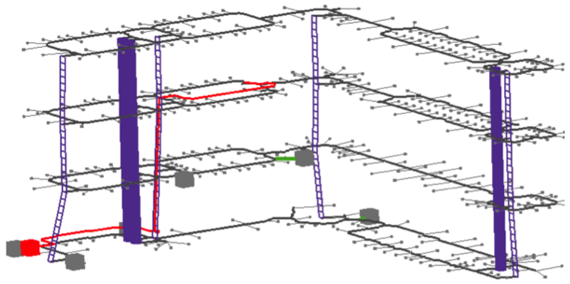
1. **Geometry Functions.** The geometry functions load the campus from polygon data in



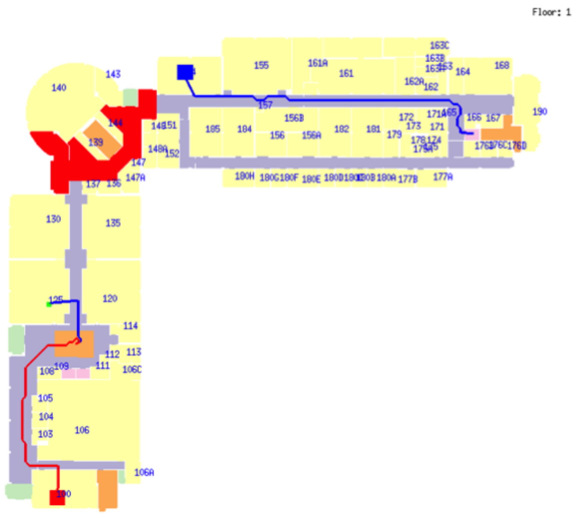
(a) Campus View



(b) Hallway View

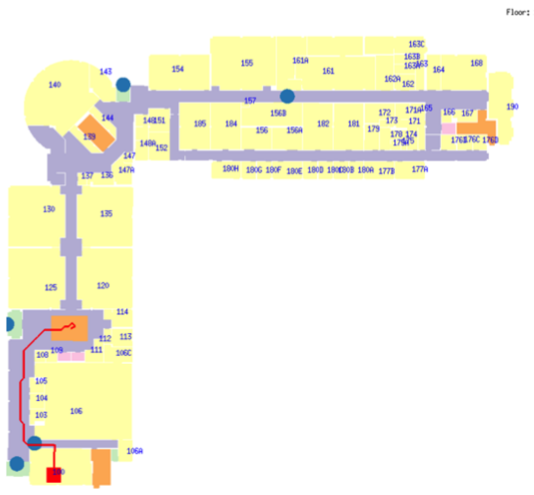


(c) Building Graph View

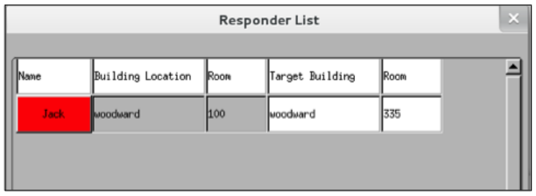


(d) Floor View

Figure 7: Command Center: 3D Views



(a)


Responder List

Name	Building Location	Room	Target Building	Room
Jack	woodward	100	woodward	335

(b)

Figure 8: Responder List. The responder, 'Jack' shown in the responder list(a) is represented by the red icon in the 2D floor view(b)

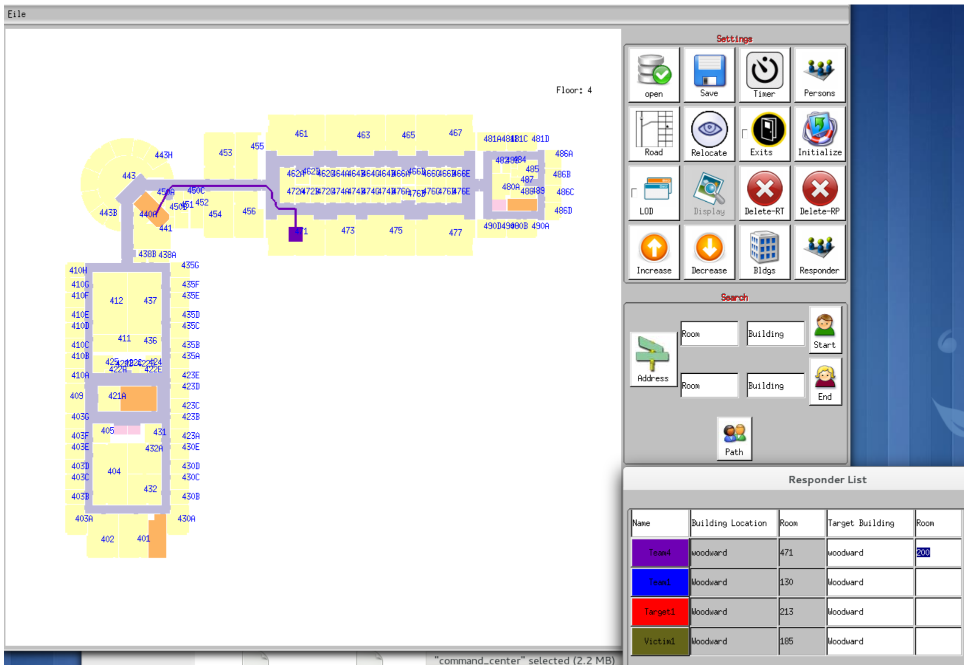
a PostgreSQL database and present them to the user in 2D or 3D. The entire campus layout is presented and the user can select buildings of interest to zoom in on and manipulate.

2. **Views.** Fig. 7 shows 4 different views: campus view of UNC Charlotte and surrounding street network, an office building emphasising hallways, a 3D graph view that is used in generating routes and paths, and a 2D view of a floor within the building.

3. **Responder Functions.** Responder functions communicate with the database through a web server using PHP. There are several functions here:

   - Responders are entered and listed in a display widget along with the device id of there mobile unit. Their location is indicated in the building in the 2D/3D view. Fig. 8 shows an example: responder Jack has been entered in the responder list in Fig. 8(b), and is seen as the red icon in Fig. 8(a).

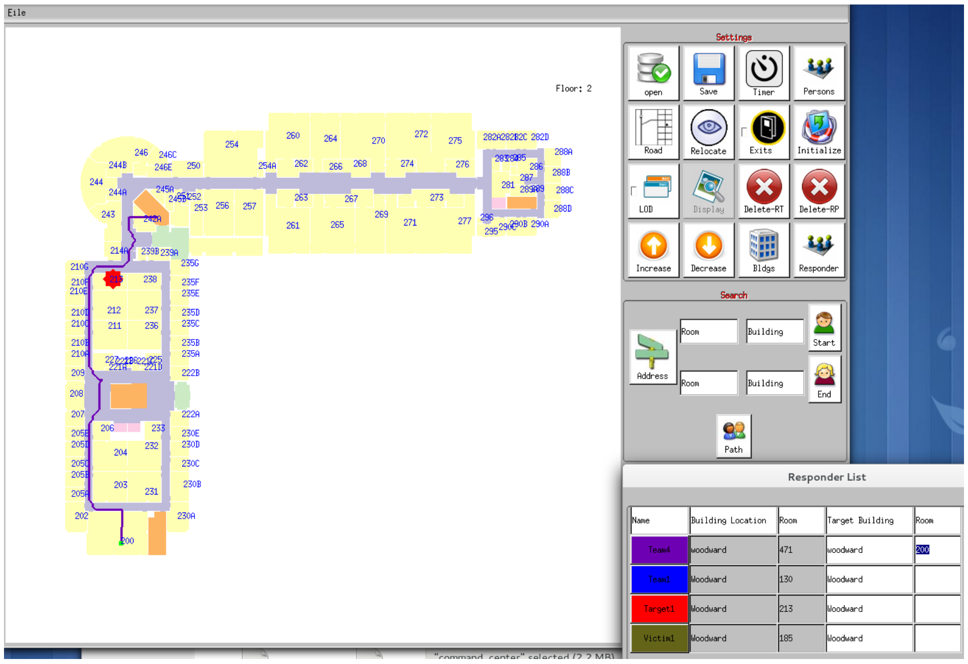
   - Paths are sent to the database in the form of responder id, start, end, and a node list that form a specified route as described by the user. The command center



Responder List

Name	Building Location	Room	Target Building	Room
Team4	woodward	471	woodward	200
Team1	Woodward	130	Woodward	
Target1	Woodward	213	Woodward	
Victim1	Woodward	185	Woodward	

(a) Path from Rm. 471 (fourth floor) to stairwell.

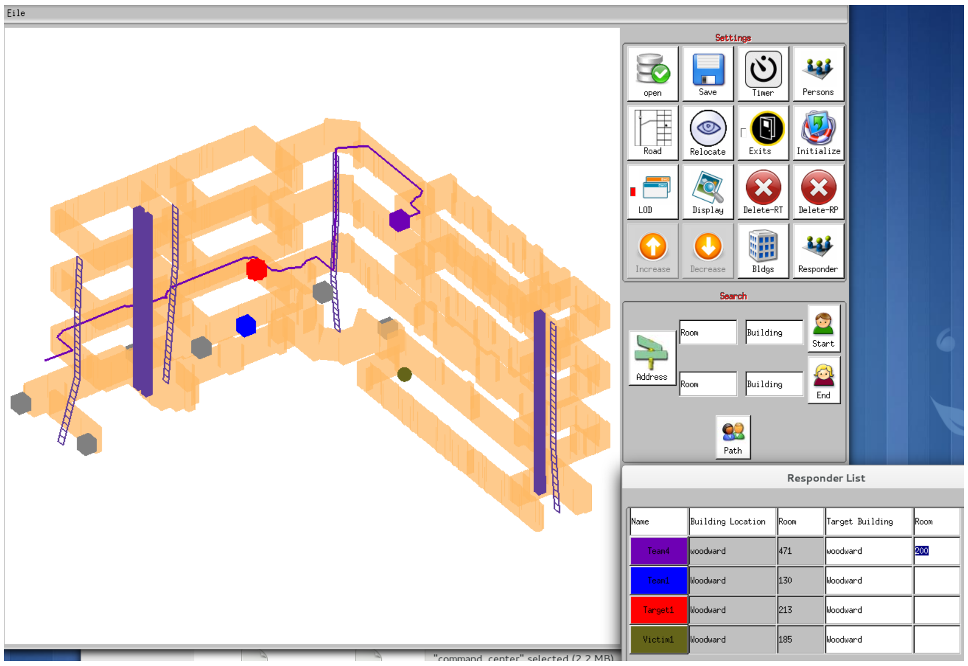


Responder List

Name	Building Location	Room	Target Building	Room
Team4	woodward	471	woodward	200
Team1	Woodward	130	Woodward	
Target1	Woodward	213	Woodward	
Victim1	Woodward	185	Woodward	

(b) Path continuing from stairwell to Rm. 200(second floor)

Figure 9: Command Center(2D Views): Path from Rm. 471(fourth floor) to Rm. 200 (second floor)



Responder List

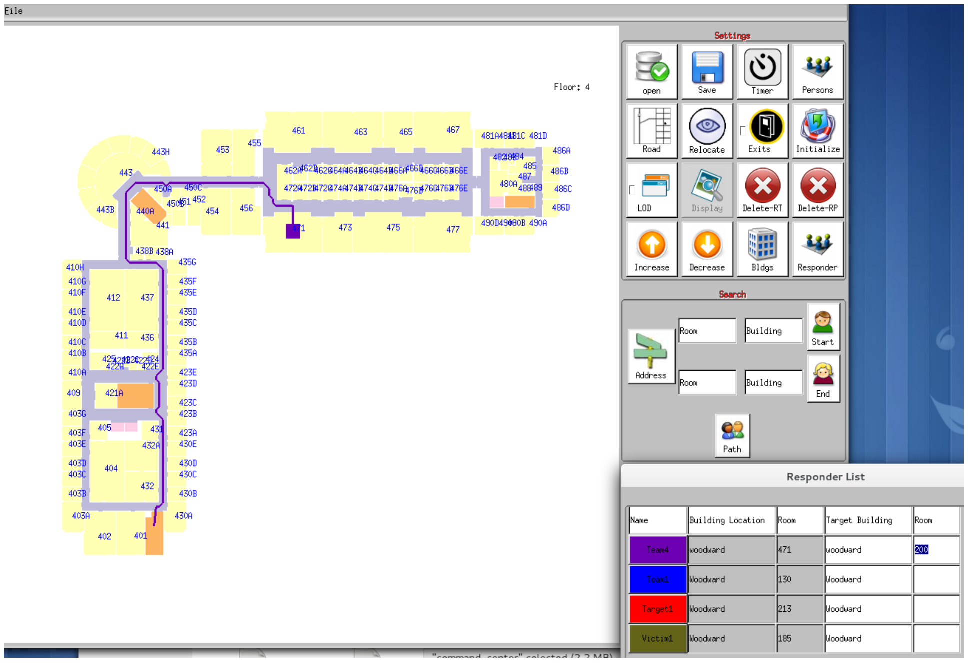
Name	Building Location	Room	Target Building	Room
Team4	woodward	471	woodward	200
Team1	Woodward	130	Woodward	
Target1	Woodward	213	Woodward	
Victim1	Woodward	185	Woodward	

Figure 10: Command Center(3D View): Path from Rm. 471(fourth floor) to Rm. 200(second floor)

application calculates a shortest path based on building geometry and egress centerline calculations referred to as the building network. The hallway and floor views in Fig. 7 show an example path from a room in the third floor to an exit in the first floor of a campus building.

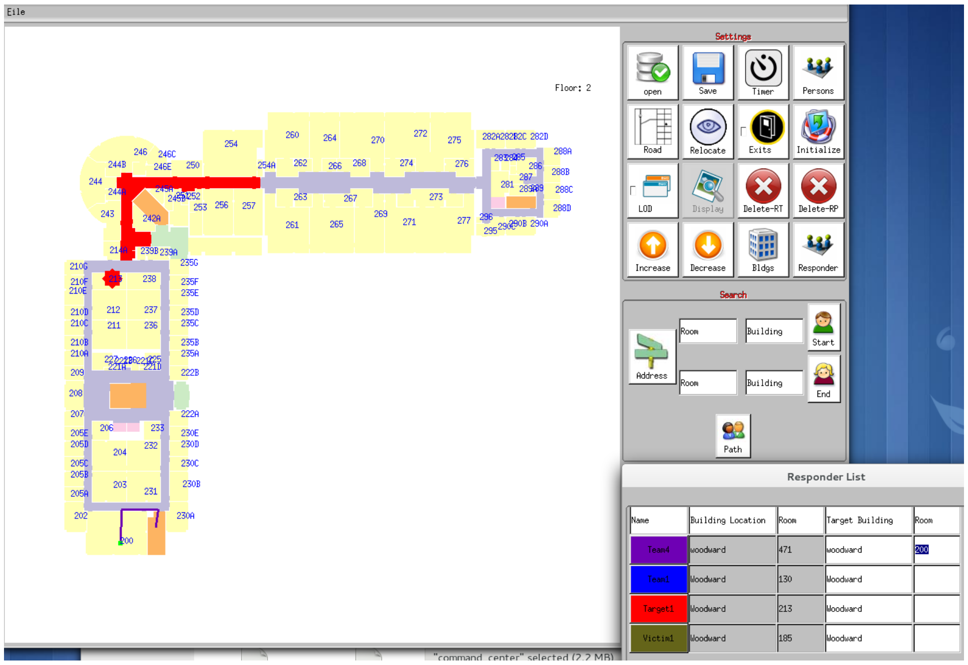
- Blocked areas, such as building hallways, are identified by interaction with the 2D/3D view and sent to the database. This effects path production and is accessed by the mobile application for display and use by responders using mobile devices.
- Victims, responders, assailants, etc. are shown on the 2D/3D view with unique icons.

4. **Inserting Blockages.** During an emergency, certain areas of a structure may become unusable or dangerous and accessibility can be restricted. The command center application permits any section of the building to be blocked. Once an area has been blocked, then all routing will be calculated to avoid that area. Figs. 9,10 show a path in



Name	Building Location	Room	Target Building	Room
Team4	woodward	471	woodward	200
Team1	Woodward	130	Woodward	
Target1	Woodward	213	Woodward	
Victim1	Woodward	185	Woodward	

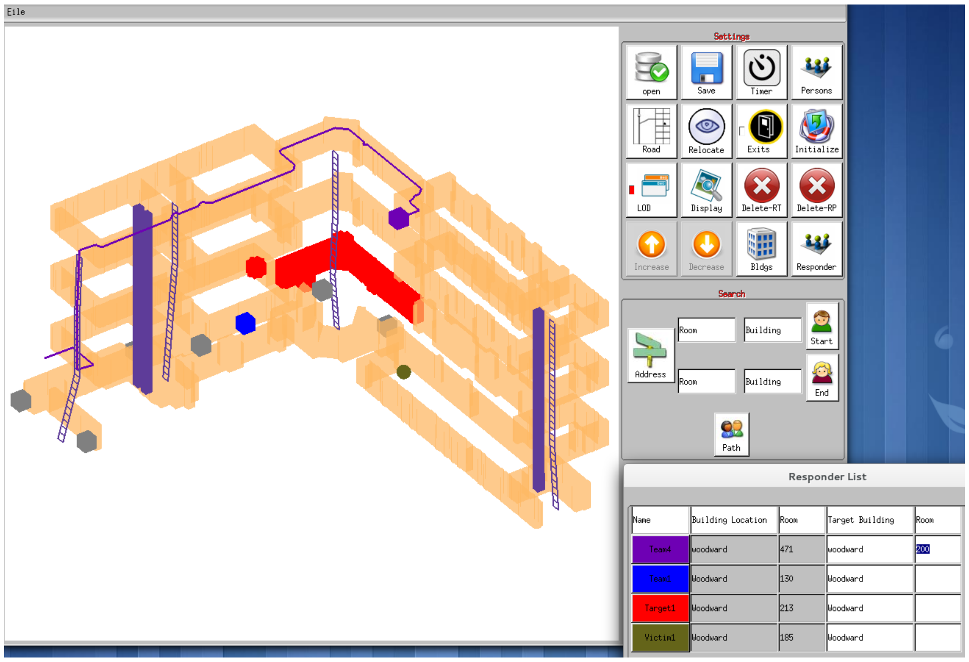
(a) Path from Rm. 471 (fourth floor) to stairwell.



Name	Building Location	Room	Target Building	Room
Team4	woodward	471	woodward	200
Team1	Woodward	130	Woodward	
Target1	Woodward	213	Woodward	
Victim1	Woodward	185	Woodward	

(b) Path continuing from stairwell to Rm. 200(second floor)

Figure 11: Command Center With a Blockage on the 2nd Floor(2D Views): Path from Rm. 471(fourth floor) to Rm. 200 (second floor)



Responder List

Name	Building Location	Room	Target Building	Room
Team4	woodward	471	woodward	200
Team1	Woodward	130	Woodward	
Target1	Woodward	213	Woodward	
Victim1	Woodward	185	Woodward	

Figure 12: Command Center With a Blockage on the 2nd Floor(3D View): Path from Rm. 471(fourth floor) to Rm. 200(second floor)

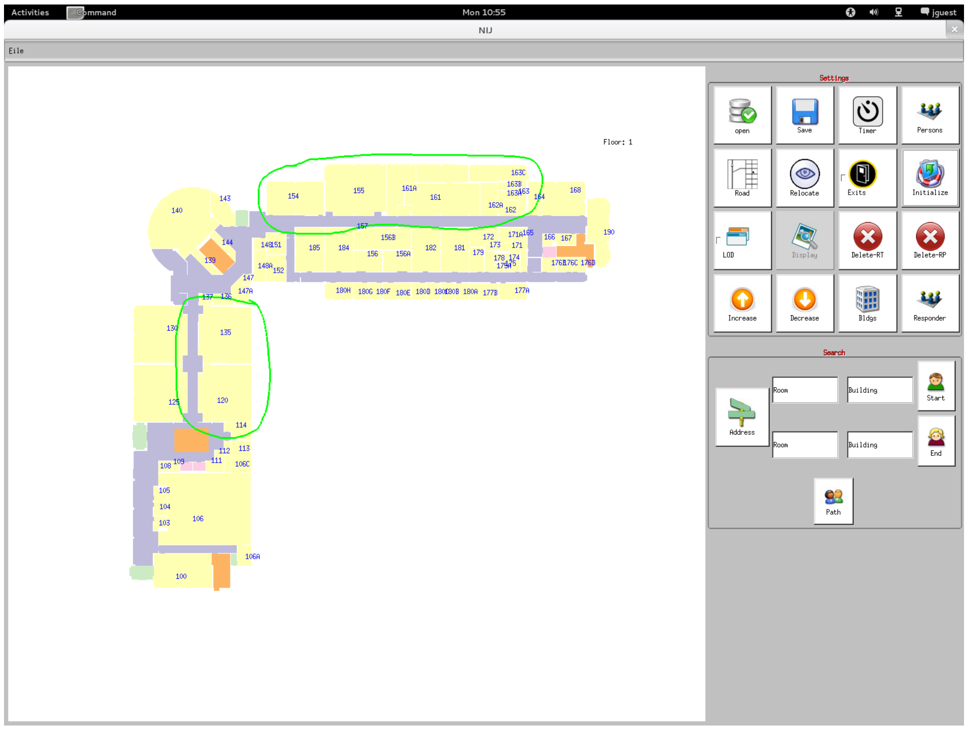


Figure 13: Command center view of a floor of a building with marked areas(in green) to be searched by responder teams

2D and 3D from room 471 on the fourth floor to room 200 on the second floor, leading to an exit. A blockage is now introduced on a second floor stairwell. Figs. 11,12 now show the same path that avoids the second floor stairwell; the path goes to the end of the hallway on the fourth floor and down an alternate stairway(as seen clearly in the 3D view of Fig. 12 to room 200.

5. **Search Areas.** The command center can also direct responders to search specific areas of the building and mark them off as 'searched' upon confirmation from responders. Fig. 13 shows an example with two marked areas (in green) to be searched by responder teams in the building. These are immediately transmitted to the responders' mobile devices.

3.3.2 Mobile Responder System

The mobile responder system currently is based on iPhone/iPod/iPad devices. It consists of the following functions:

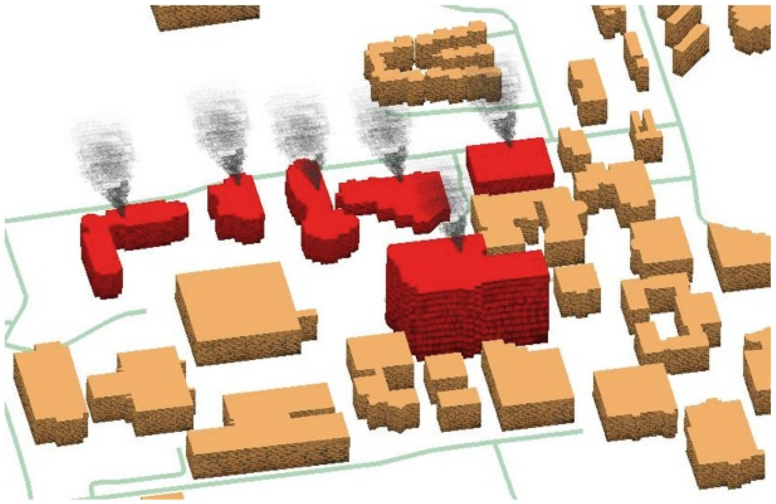


Figure 14: Mobile Appliation: An example cluster of buildings, with multiple selected buildings(in red)

1. **Campus View.** A high level view of the entire collection of buildings is provided. The user may then select a particular building. Fig. 14 shows part of a campus cluster

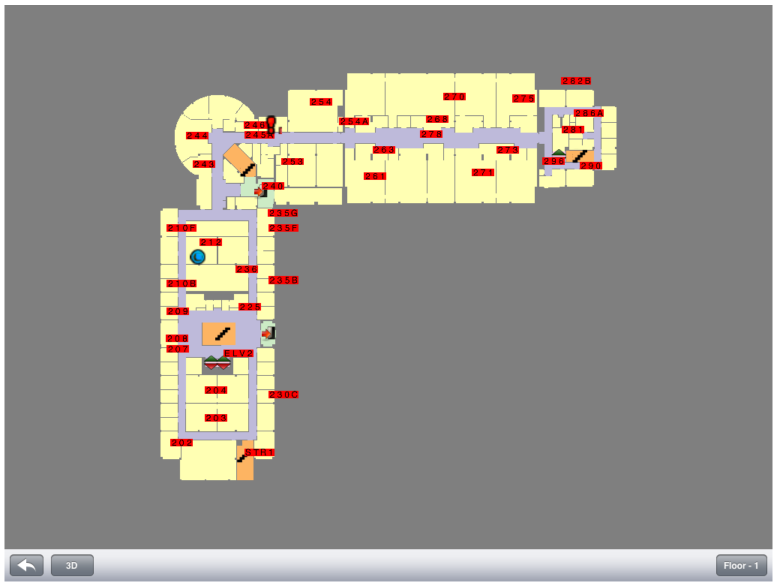


Figure 15: Mobile Application:2D floor view of campus building. Rooms are numbered, stairways are marked by icons(black on orange), exit(arrow), hallways are in light blue.

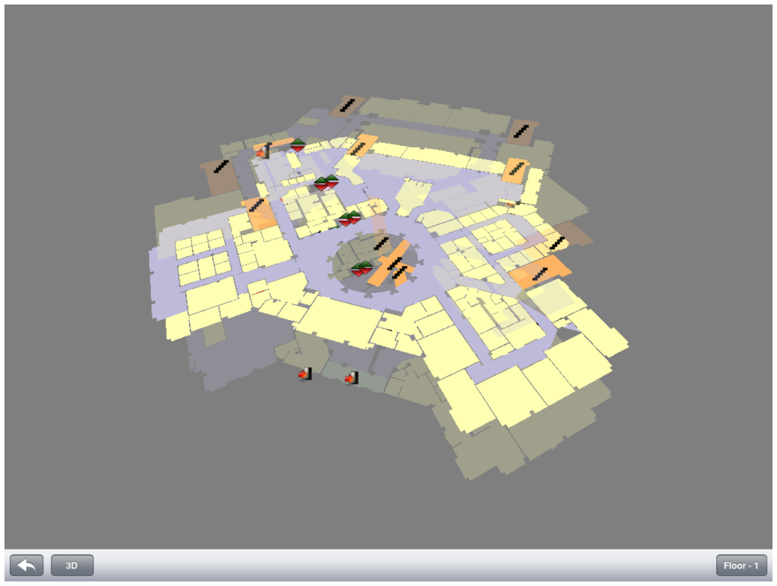


Figure 16: Mobile Application: 3D perspective view of a campus building. The currently selected floor is highlighted, while the remaining floors are deemphasised(partially transparent). Stairways, elevators, hallways, entrances/exits are marked as in the 2D view.

of 70 buildings at UNC Charlotte.

2. **2D Views.** The mobile application can display each floor of a building, as seen in Fig. 15. Rooms are numbered and stairways, elevators and entrances/exits are clearly marked by icons, and hallways are color coded. Switching between floors requires simple gestures on the interface.

3. **3D Views.** 3D perspective views are supported; an example is shown in Fig. 16. All critical building elements are marked as in the 2D view. In general, the 3D view is more heavily used in the command center, as the responders are more concerned with their current location and the path to their target on the current floor.

4. **Routing Features.** The application can receive the target location of a room to direct the responder. The room number is visually displayed in the top left corner of the screen. In addition to receiving a target location the application will determine the quickest route to that room. As the responders update their location inside of the building the path will be recalculated and the updated path is displayed. Fig. 17 illustrates a path from room 456 (4th floor) to room 200(2nd floor). Both 2d(floor views) and 3D views of the path are shown.

5. **Blockages.** The command center can block certain parts of the building based on received reports, such as smoke in the hallway or stairwell, danger due to presence of a target, etc. The blocked areas are immediately transmitted to the responder. All routes are recalculated to avoid the blocked areas of the building. Fig. 18 illustrates the same source-destination from Fig. 17 but with a section of the second floor blocked(red highlighted areas), but now the routes avoid the blocked areas (hazard).

6. **Location Reporting.** The application allows responder to interactively touch the screen at any location. The application drops a pin at that location similar to the iOS maps functionality. The 2D touch location on the screen is converted to geographical

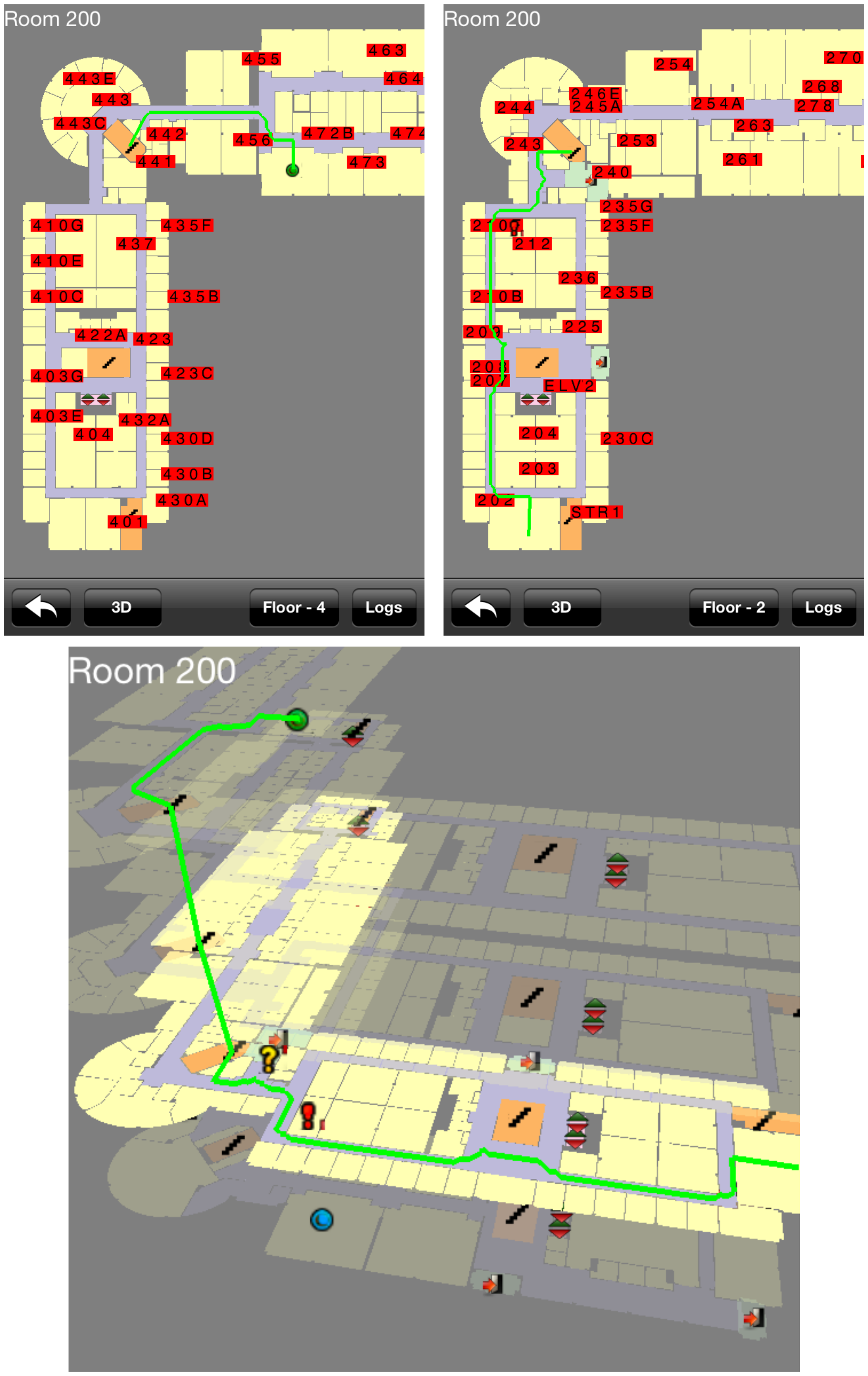


Figure 17: Mobile Application: Routing Features. Top row shows a path from room 456 on the 4th floor to room 200 on the second floor. Left image displays part of the route on the fourth floor, while the right images displays the path on the second floor, traveling via a stairway between the two floors. Bottom image illustrates the entire path in the 3D view (floor 2 is highlighted in this view)

.

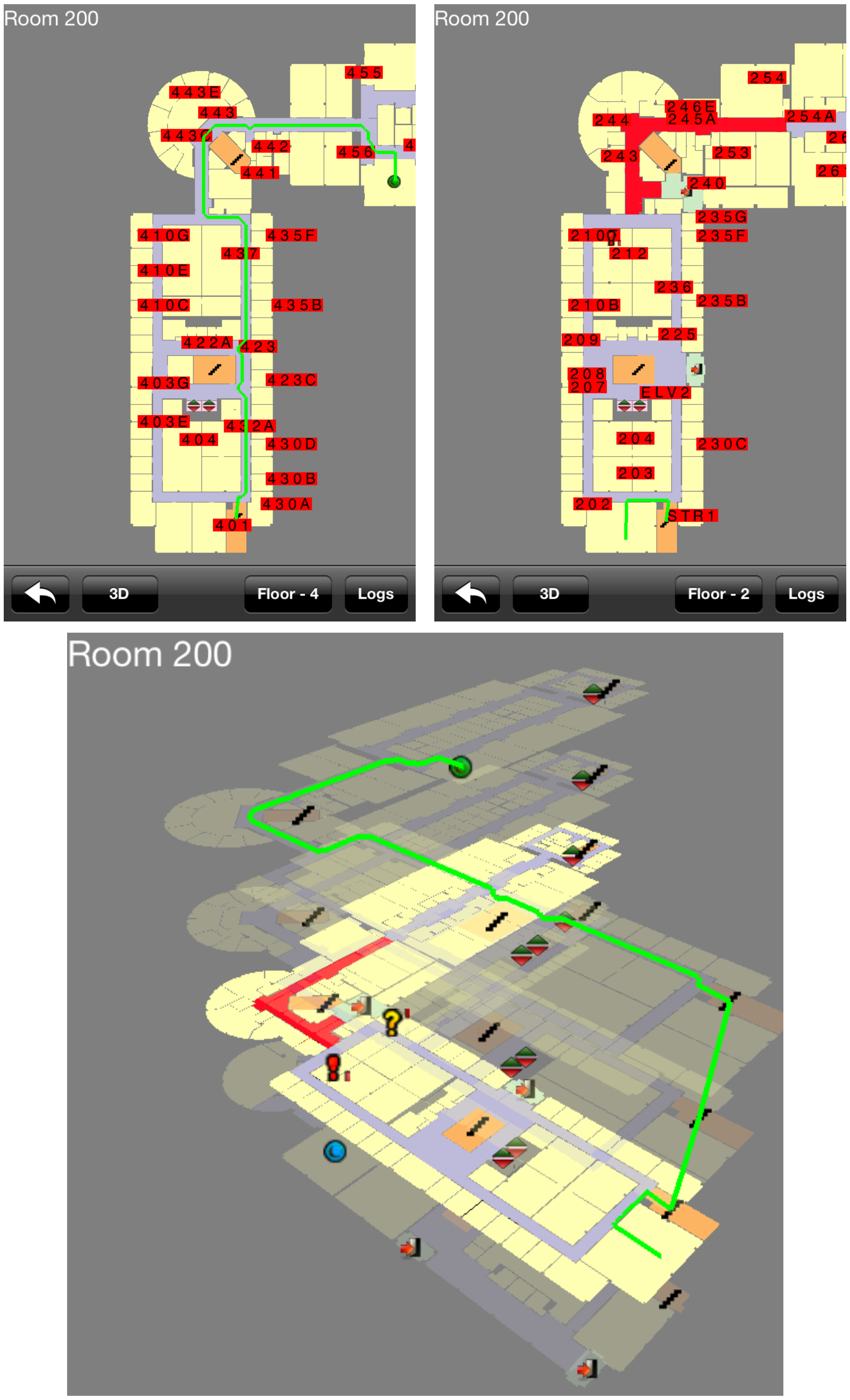


Figure 18: Mobile Application: Routing with Blocked Areas in the Building. The same path (from room 456 to 200) is shown, but now an area(highlighted in red) on the second floor has been marked as inaccessible. The system recalculates the route to avoid these areas. Top images show the fourth and second floor views while the bottom image shows the 3D view.

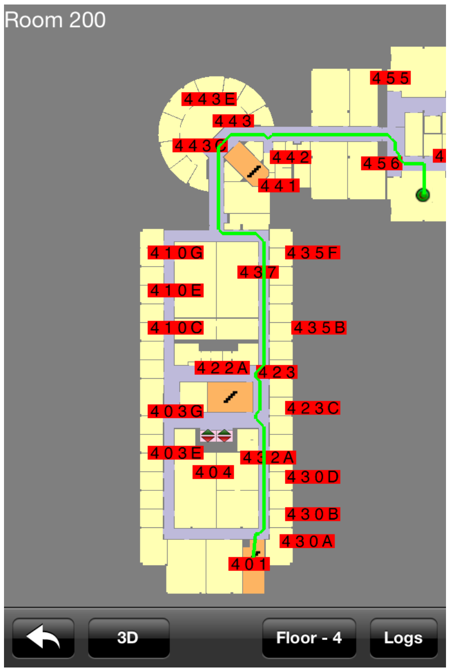


Figure 19: Mobile Application: Reporting Location of A Responder(“Here I am” function).

location that is sent to a command center application via cellular connection or over WIFI. This allows the responder to update their position anywhere inside of a building and sends that location to a central command center. The command center thus has knowledge of the locations of all responders, a critically important feature in emergency management and disaster mitigation. In the long-term, automatic location detection will be the goal, relieving the responders from manual reporting. Fig. 19 illustrates an example of a location specified by a responder (pin illustrated in the 2D floor view) and as a cube icon in the 3D command center view.

7. **Search.** Frequently, the incident commander on the scene might want their responder team to perform a search of a particular area of a building (reports of a shooter seen in particular area, for instance). Our application facilitates this by allowing areas to be marked (interactively drawn over the 2D floor view, as shown in Fig. 13); these are immediately transmitted to the responders who can then focus their search on these areas.

3.3.3 Database Functions

The command center and the mobile application communicate through a network interface to a PostgreSQL/PostGIS database. The communication maintains synchronous knowledge between command center and responders of the building status in the form of blockages, responder locations, search orders, search completion, shortest path to and from a source to a destination, a list of responders with device identification, and location and type of other individuals, such as assailants and victims.

Timers are maintained on the command center and mobile devices for data synchronization.

The following is a list of entities and their interaction with the system:

1. **Building Blockages.** Maintains information required for communication to the mobile devices relating to blocked areas of building. The user receives information from all available sources and informs the command center of a blocked area. The application queries and updates this table. The information is then displayed as red areas on the user interfaces of the mobile and desktop devices.

2. **Responders.** Maintains responder id, building name, room number, and targets for each responder. Targets are ordered destination for a responder set. Room number represents the start location at the time of a command center order. This table is maintained by the command center. The mobile device can register these in the system by adding entries here.

3. **Routine Table.** Maintains source, destination, and path information in a text string. The command center maintains this table as instructions for responders and information for command personnel.

4. **Responder Location Reporting.** Maintains ID, and x, y, z, coordinates of the building location of a responder. This table is updated by the "Here I am" function of

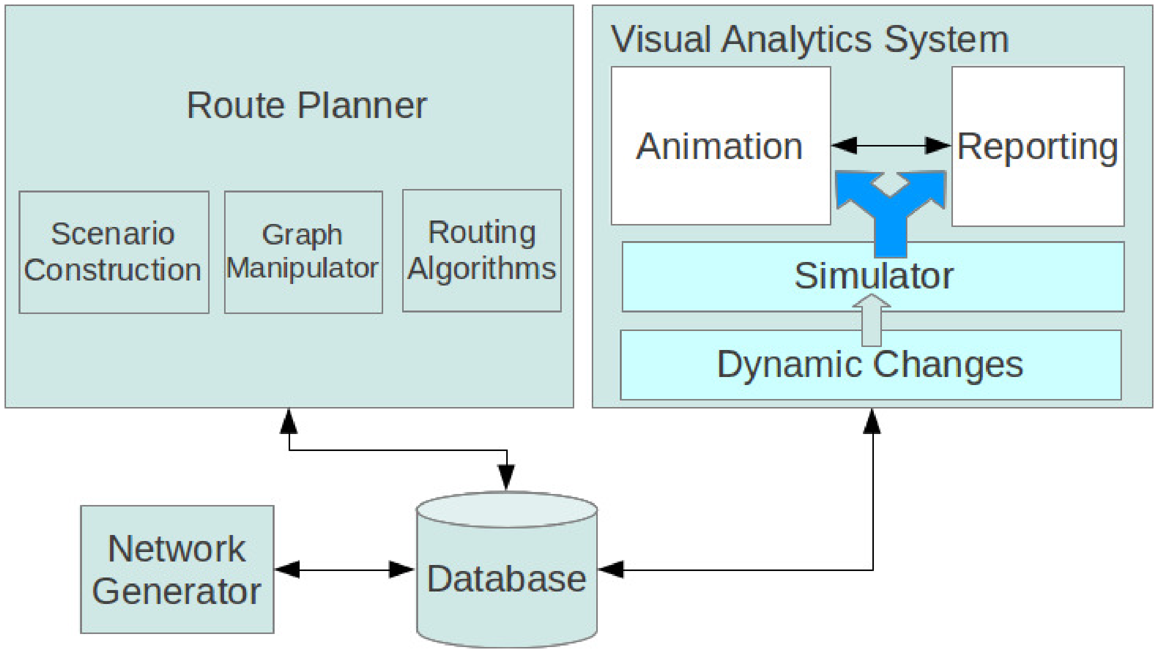


Figure 20: Visual Analytic System Architecture. Route planner involves preprocessing and routing calculations that serve to initialize the system, with processed results stored in a database. The visual analytic system uses visual abstraction and scalable representations that permit real-time interaction, injection of dynamic situational changes and visual analysis.

the mobile devices.

3.4 A Visual Analytics System for Situationally Aware Evacuations

Fig. 20 illustrates the major components of our system for situationally aware evacuations. There are two major components to the system. Functions in the Route Planner involve a significant amount of a priori processing and initialization. The visual analytic system is a highly interactive system that is user driven and can inject and respond to dynamic changes during evacuations. It also consists of reporting and visual analysis functions that can assist an emergency planner on exploring different scenarios, as to the use and deployment of resources, dispatch responders, effect of rerouting occupants, etc.

In section 3.2 and [38] we described a semi-automatic system that constructed a *building* *graph,* incorporating key elements of a georeferenced urban structure critical to evacuations, such as hallways, stairways, elevators and entrances/exits. This building graph generator is used to process urban structures and is stored in a PostgreSQL database. We have successfully processed over 70 buildings of our campus using the graph generator.

We next describe the main components of our visual analytic system.

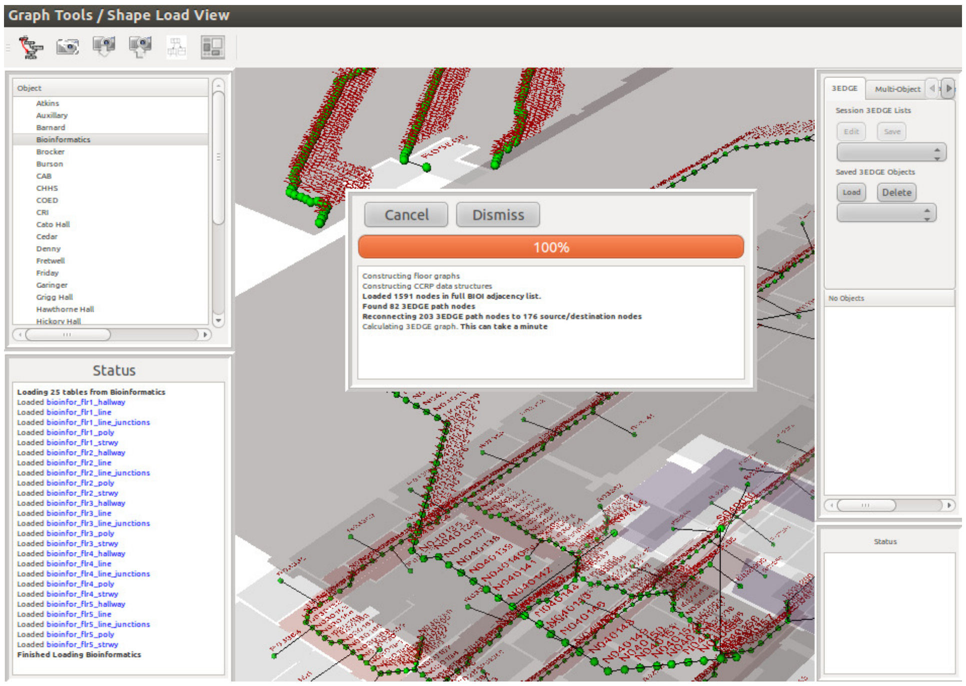
3.4.1 Route Planner

Our route planner provides facilities for loading evacuation objects, that are combinations of urban structures, pathways, streets etc. Evacuations are built as combinations of structural objects and route planner objects and saved in the database. The Route Planner consists of the following components:

LOD Graph Construction and Manipulation. All computed egress paths are saved in the database as node to node connections. Nodes are defined based on their function, and can be rooms, hallways, stairways, elevators, and exits. The evacuation application also creates muster points, that represent locations where evacuees are ordered to congregate during an emergency. During the process of extracting centerline points the geometry of hallways is sampled at approximately two foot intervals. This process is necessary to insure accuracy, particularly at corridor elbows. However, this raises computational issues with extremely large buildings or multi-building graphs, and hinders real-time performance(expensive routing calculations) when dynamic changes need to be accommodated during an event. We address this problem by simplifying the graph with two different levels of detail.

- **Level 1. Remove Redundant Nodes.** In this step, we simplify the original building graph by removing nodes that do not impact routing, for instance, shortest path calculations. In particular, the larger sampling rate used for accurate centerline calculations results in nodes that can be removed for use in building routes. Nodes and edges are collapsed in the process. Beginning with any node with three or more edges (or any arbitrary node), edges are followed until a node with 3 or more edges is encountered. This becomes a node of the simplified graph and a new edge is created connecting to the previous node, as can be seen in Figure 23. The process is continued until all nodes have been visited. Our route planner is executed on these simplified graphs for scenarios in which all original egress paths are available. In our experiments, we see a factor of 5-8 reduction in the number of nodes in the simplified graph.

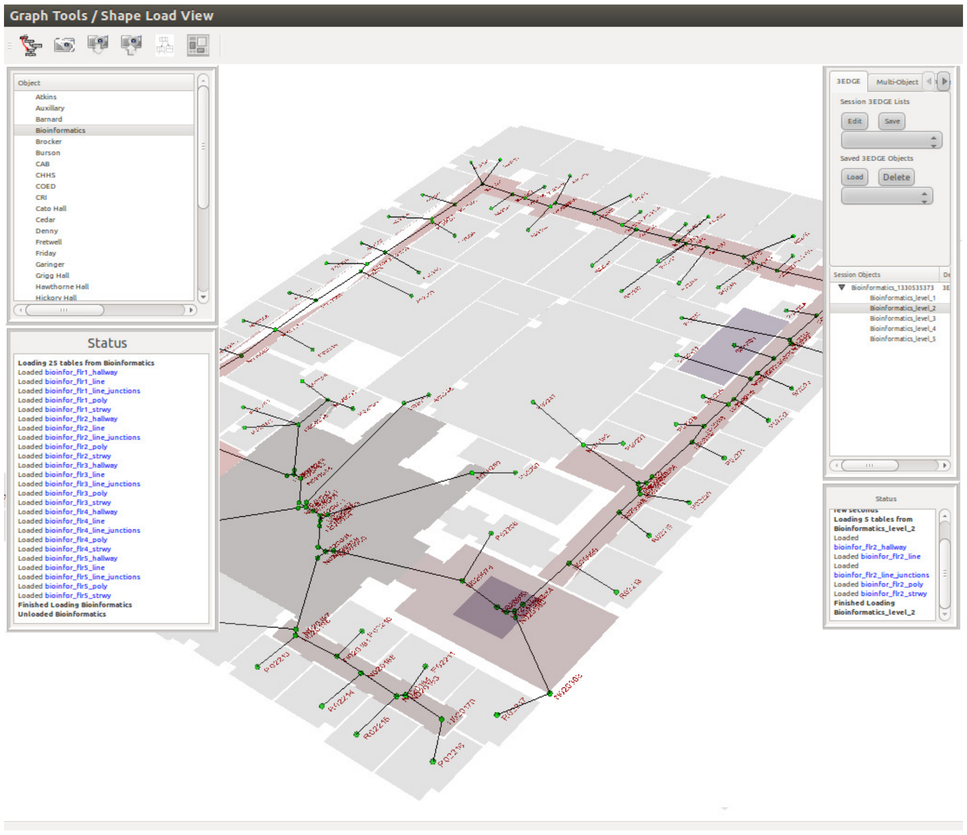
The Level 1 graph reduction process attempts to create a bi-partite graph where all inner building nodes have 3 or more incident edges. The final graph produced may have nodes with 2 incidents edges because unconnected nodes remaining after algorithm execution are simply connected to their closest neighbor with no further concern for reduction. Results of processing dozens of building graphs indicates approximately 95% reduction of the nodes of concern to 3 edge nodes. The following algorithms detail the process.



```
Loading 25 tables from Bioinformatics
Loaded bioinfor_flr1_hallway
Loaded bioinfor_flr1_line
Loaded bioinfor_flr1_line_junctions
Loaded bioinfor_flr1_poly
Loaded bioinfor_flr1_strwy
Loaded bioinfor_flr2_hallway
Loaded bioinfor_flr2_line
Loaded bioinfor_flr2_line_junctions
Loaded bioinfor_flr2_poly
Loaded bioinfor_flr2_strwy
Loaded bioinfor_flr3_hallway
Loaded bioinfor_flr3_line
Loaded bioinfor_flr3_line_junctions
Loaded bioinfor_flr3_poly
Loaded bioinfor_flr3_strwy
Loaded bioinfor_flr4_hallway
Loaded bioinfor_flr4_line
Loaded bioinfor_flr4_line_junctions
Loaded bioinfor_flr4_poly
Loaded bioinfor_flr4_strwy
Loaded bioinfor_flr5_hallway
Loaded bioinfor_flr5_line
Loaded bioinfor_flr5_line_junctions
Loaded bioinfor_flr5_poly
Loaded bioinfor_flr5_strwy
Finished Loading Bioinformatics
```

```
Constructing floor graphs
Constructing CCRP data structures
Loaded 1591 nodes in full BIOI adjacency list.
Found 82 3EDGE path nodes
Reconnecting 203 3EDGE path nodes to 176 source/destination nodes
Calculating 3EDGE graph. This can take a minute
```

Figure 21: User interface for controlling and monitoring the graph reduction algorithm. In this example 1591 nodes are being reduced to 203.



```
Loading 25 tables from Bioinformatics
Loaded bioinfor_flr1_hallway
Loaded bioinfor_flr1_line
Loaded bioinfor_flr1_line_junctions
Loaded bioinfor_flr1_poly
Loaded bioinfor_flr1_strwy
Loaded bioinfor_flr2_hallway
Loaded bioinfor_flr2_line
Loaded bioinfor_flr2_line_junctions
Loaded bioinfor_flr2_poly
Loaded bioinfor_flr2_strwy
Loaded bioinfor_flr3_hallway
Loaded bioinfor_flr3_line
Loaded bioinfor_flr3_line_junctions
Loaded bioinfor_flr3_poly
Loaded bioinfor_flr3_strwy
Loaded bioinfor_flr4_hallway
Loaded bioinfor_flr4_line
Loaded bioinfor_flr4_line_junctions
Loaded bioinfor_flr4_poly
Loaded bioinfor_flr4_strwy
Loaded bioinfor_flr5_hallway
Loaded bioinfor_flr5_line
Loaded bioinfor_flr5_line_junctions
Loaded bioinfor_flr5_poly
Loaded bioinfor_flr5_strwy
Finished Loading Bioinformatics
Unloaded Bioinformatics
```

```
few seconds
Loading 5 tables from
Bioinformatics_level_2
Loaded
bioinfor_flr2_hallway
Loaded bioinfor_flr2_line
Loaded
bioinfor_flr2_line_junctions
Loaded bioinfor_flr2_poly
Loaded bioinfor_flr2_strwy
Finished Loading
Bioinformatics_level_2
```

Figure 22: View of 2nd floor results from operation illustrated in Figure21.

The output of the network generator produces 2D arrays stored in the PostGIS database in the form Node1, Node2, Distance in feet. For efficient manipulation we store these in adjacency lists implemented as associative arrays.

The spG structure produced by algorithm 1 serves as input to a 3 stage divide and conquer algorithm, as listed in Algorithms 2, 3, 4). The procedures perform the following general tasks:

1. Find all nodes with 3 or more incident edges in original graph and add them to the minimal graph.
2. Build a heap of all hallway nodes with less than 3 incident edges.
3. Build a heap of all non hallway nodes with less than 3 incident edges.
4. Iterate hallway nodes combining edges between nodes until leaf node of a "3EDGE"

Algorithm 1: Build an Adjacency List Graph of the building containing all Network Generator Nodes and produce Incident Edge Count Associative Array.

```
Input: (1) Node Properties: nIE, Count of Incident edges per node;
(2) Edge Properties: Length in Feet(LF);
(3) EAr: Array; [$Node1_{i...n}, Node2_{j...n}, LF$]]. Array list of all nodes in spG.
Result: (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident Edges(IE),
          E edges. Note: This structure is an Adjacency List
          $\{N_i : \{N_j : LF, ...N_k : LF\}\}\}$;
(2) Node Properties: nIE, Count of Incident Edges per node;
foreach Node1; Node2; LF in EAr do
    Initialize empty associative array Temp;
    if Node1 not in Adjacency List spG Add It then
        Add edge Node1-Node2-LF to Adjacency List Temp;
        for k = IndexCounter + 1 to LENGTH(EAr) - 1) do
            Add edge Node1-Node2-LF at EAr[k] to Temp if Node1 and Node2 are
            HALLWAY-NODE then
                nIE[Node1] += 1;
                            /* Increment edge count for Node1 */;
            end
        end
        spG[Node1] = Temp;
                    /* Assign associative array Temp to Node1 */;
    end
end
```

node is found, then connect them as one edge.

5. Iterate NON hallway nodes combining edges between nodes until leaf node of a "3EDGE" node is found, then connect them as one edge.

Table 1: Campus Adjacency List

ID(Int)	NodeID(Char)	AdjacencyList(VarChar)
1294290388	N010011_BIOI	"N010007_BIOI 4 N010024_BIOI 13"
1294290388	N010024_BIOI	"N010011_BIOI 13 N010025_BIOI 1"
1294290388	N010025_BIOI	"N010026_BIOI 1 N010189_BIOI 1 N010024_BIOI 1"
1294290388	N010026_BIOI	"N010043_BIOI 17 N010025_BIOI 1"

After the entire series of operations is complete the Associative Array spMinG, which is finalized in Algorithm 4, is used by the Route Planner to improve the efficiency of Single Source Shortest Path Dijkstra searches. It is also used in the Simulator for reconstruction of a real geometric path when a Zone graph is input to a Dijkstra Search. Using Zone calculated routes for visualization causes evacuees to appear to take impossible routes. Unless a building is modified, the structure is permanent through the life of the application; therefore it is saved to the database in the relation table's campus adjacency lists. The table allows multiple lists per campus object but there should only be a need for one unless other algorithms are implemented that require a modified or different underlying structure for searches. Table 1 is an example representation of the campus_adjacency_list entity:

Algorithm 2 adds nodes with 3 or more incident edges to the reduced graph adjacency list spMinG. This initializes the reduced graph for processing through the later functions. At this point it contains 3 or more edge nodes that are disconnected and is not yet a complete graph. Notice that the a subordinate adjacency lists nodes are copied into heaps. There are 2 heaps utilized to accomplish the divide and conquer methods. A no connection heap built from hallway nodes and a no connection heap built from non hallway nodes. The 2 categories of node heaps are created because we always

Algorithm 2: Build a Partial Reduced Adjacency List Graph of unconnected node heaps of “HALLWAY” type nodes and “OTHER” type nodes.

```
Input: (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident Edges(IE), E
       edges. Note: This structure is an Adjacency List $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(2) Node Properties: nIE, Count of Incident edges per nodes;
Result: (1) spMinG(N,E): Directed Graph, N nodes with 3 or more Incident
        Edges(IE), E edges as an Adjacency List $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(2) NoConnHeapPATH: Heap of path nodes with < 3 Incident Edges;
(3) NoConnHeapOTHER: Heap of path nodes with < 3 Incident Edges connected to
Stairwells or Exits, etc;
for Node in original Adjacency List spG do
    if Node has 3 or more incident edges then
        $spMinG[node]$ = spG[$node$];
                /* Work done...add it to reduced graph */;
        /* Node2's refer to 2nd node in Node1-Node2-LF edge definition */;
        for Hallway nodes assume all Node2's are disconnected and build array list of
        these nodes do
            Add Node2's to $NoConnHeapPATH$ Array;
        end
    end
    else if NOT HALLWAY-NODE then
        $spMinG[node]$ = spG[$node$];
            /* add EXIT, STAIRWELL etc, nodes to reduced graph */;
        for NON hallway nodes assume all Node2's are disconnected and build array
        list of these nodes do
            Add Node2's to $NoConnHeapOTHER$ Array;
        end
    end
end
```

know we want to connect hallway nodes adjacent to leaf nodes of 3 or more edge nodes. Nodes that are categorized as stairwells or exits for example are connected after the hallway iterations are complete. This allows us to separately handle central hallway nodes and other nodes, and simplifies the logic of each segment of our procedure.

Algorithm 3: Build partial reduced adjacency List graph by walking original graph adjacency list until node is found in No Connection Heap of "HALLWAY" nodes.

```
Input: (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident Edges(IE), E
       edges. Note: This structure is an Adjacency List $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(2) Node Properties: nIE, Count of Incedent edges per nodes;
(3) spMinG(N,E): Directed Graph, N nodes with 3 or more Incedent Edges(IE), E
edges as an Adjacency List $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(4) NoConnHeapPATH: Heap of path nodes with $<$ 3 Incident Edges;
(5) NoConnHeapOTHER: Heap of path nodes with $<$ 3 Incident Edges connected to
Stairwells or Exits;
Result: spMinG(N,E): Directed Graph, N nodes with Maximum Incident Edges(IE)
        and Minimum Nodes, E edges as an Adjacency List
        $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
foreach Node in NoConnHeapPATH do
    Get an adjacent node next of Node in original Graph spG;
    while keeplooking do
        if next is in No Connection Heaps then
            /* Add edge between Node in Heap and Leaf Node in original
            Graph spG */;
            Add Node to Reduced Graph Adjacency List;
            keeplooking = false;
            /* Found a connection move to next node in No Connection Heap
            of HALLWAY Nodes */;
        end
        else
            /* Keep walking through NON reduced Adjacency Graph spG */;
            pathlength = pathlength + Length between Node and Next;
            next = Get an adjacent node to next in spG;
        end
    end
end
```

Algorithm 3 iterates the hallway nodes no connection heap. The start point is arbitrary. This is an important because due to the complexities of real world building geometry there is no standard definition of a start point for this iteration. The procedure steps to this point to only guarantee that the origin will be a leaf node of a 3 or more incident edge node, and we are looking for the next one that is connected by multiple edges to another node in spG(the original graph), built in Alg. 1. Here we select a node from the heap which is known to be a leaf node of an edge in spMinG then walk through adjacency list spG until we find another node in the both heaps that we know are in spMinG and then connect them and add them to the spMinG adjacency list graph.

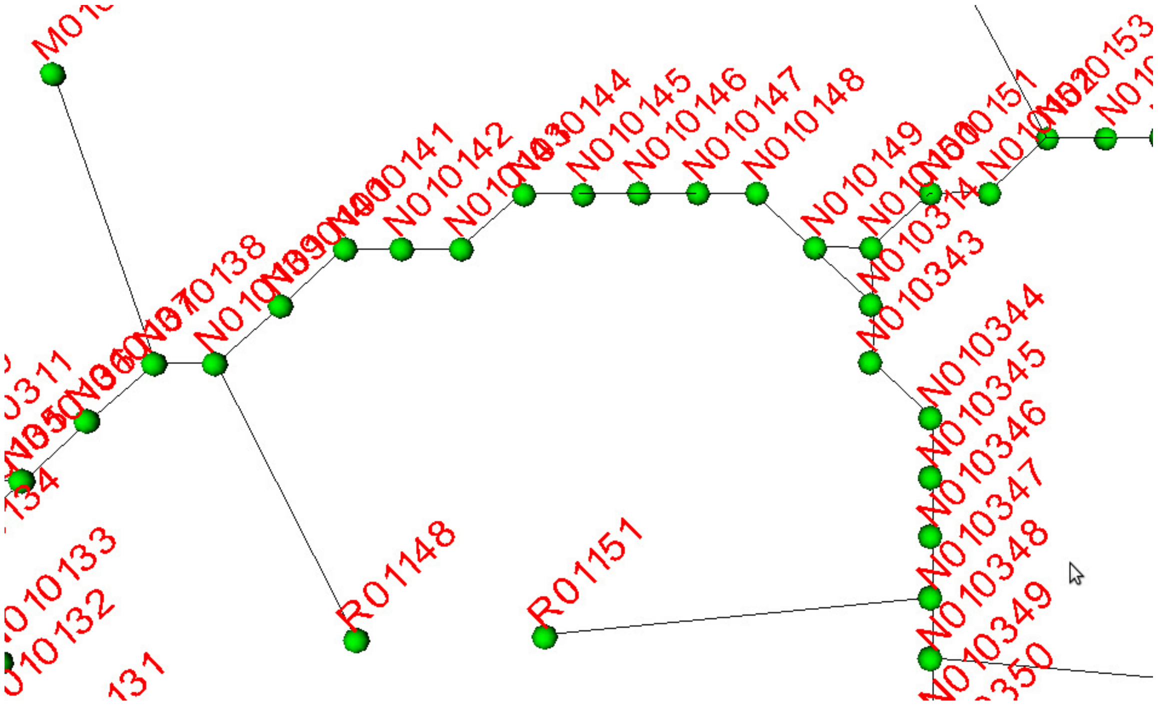
Algorithm 4: Build **Final** Reduced Adjacency List Graph by iterating No Connection heap built from NON HALLWAY Nodes.

```
Input:  (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident Edges(IE), E
        edges. Note: This structure is an Adjacency List $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(2) Node Properties: nIE, Count of Incident edges per nodes;
(3) spMinG(N,E): Directed Graph, N nodes with 3 or more Incident Edges(IE), E
edges as an Adjacency List $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(4) NoConnHeapPATH: Heap of path nodes with $<$ 3 Incident Edges;
(5) NoConnHeapOTHER: Heap of path nodes with $<$ 3 Incident Edges connected to
Stairwells or Exits;
Result:  (1) spMinG(N,E): Directed Graph, N nodes with Maximum Incident
         Edges(IE)and Minimum Nodes, E edges as an Adjacency List
         $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
foreach Node in NoConnHeapOTHER do
    Get an adjacent node next of Node in original Graph spG;
    while keeplooking do
        if next is in No Connection Heaps then
                /* Add edge between Node in Heap and Leaf Node in original
            Graph spG */;
            Add Node to Reduced Graph Adjacency List;
            keeplooking = false;
              /* Found a connection move to next node in No Connection Heap
            of HALLWAY Nodes */;
        end
        else
              /* Keep walking through NON reduced Adjacency Graph spG */;
            pathlength = pathlength + Length between Node and Next;
            next = Get an adjacent node to next in spG;
        end
    end
end
```

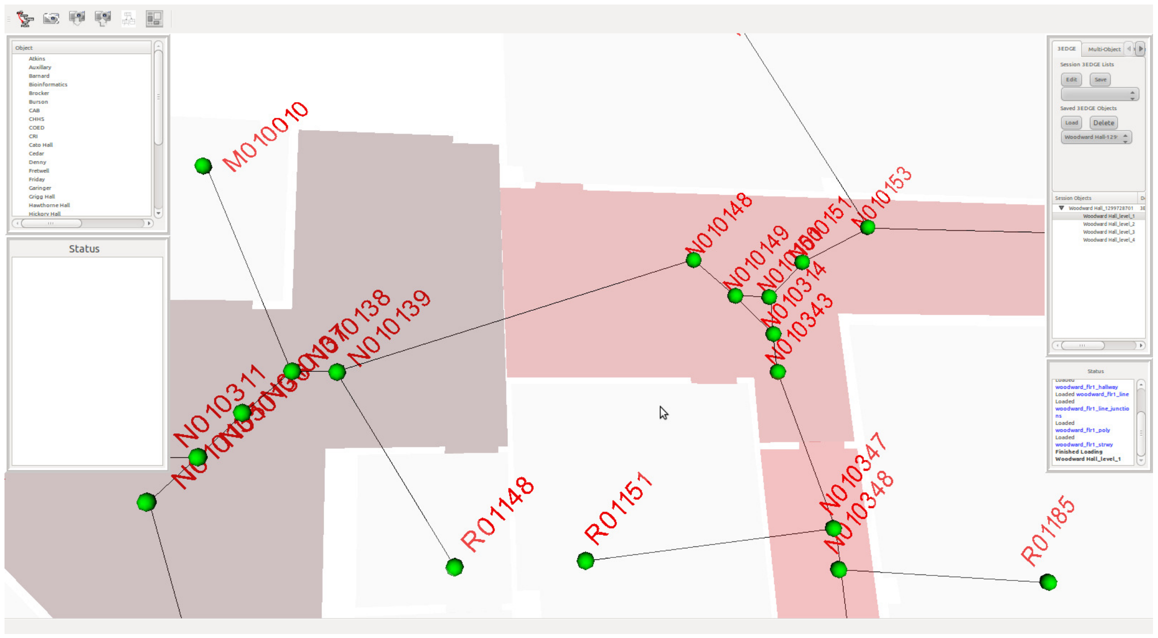
Algorithm 4 iterates on the non hallway nodes no connection heap. The same procedures are used as in Algorithm 3, the only difference being that if an origin has not yet been added to the spMinG, it is a dead end and must be added to insure no dangling nodes. This happens when a node is at the end of a building section or hallway and is connected to a source node. Source nodes are simply connected to their associated hallway node before database insertion. It is not necessary to process them through the graph reduction procedures. Note that a default path length equal to the maximum occupancy for an edge is used. This insures the path from a source to an egress can contain the maximum set by the application.

- **Level 2. Zone Graphs.** For rapid computation of paths when dynamic changes are injected during an event, the simplified graphs can still be large, especially in multi-building evacuations. In these circumstances, we further simplify the building graphs into *zone graphs*, by segmenting the building into evacuation sensitive zones: for instance, stairwells, elevators and exits form the critical elements of any egress path. An example zone graph is illustrated in Figure 25, involving stairwells, hallways and exits(in red)

  To compute the zone graph, we use the precomputed evacuee paths to first associate each node with a zone that is closest to it, using an iterative procedure. In the second pass, the graph connectivity is established by keeping track of the zone of related objects that are encountered in these paths (adjacency lists are maintained). Intermediate nodes(the yellow spheres in Figure 25(b)) are also identified by paths that cross multiple zones and are further used to complete the graph construction. Zone objects span floors in multi-story structures, with appropriate floor identification for proper path determination during routing calculations. The number of nodes in the resulting zone graph depends on the number of zones and the number of floors in the building. In our experiments, a further factor of 5-7 reduction in the number of graph nodes was seen. Table 2 compares reroute times based on these reductions.



(a)

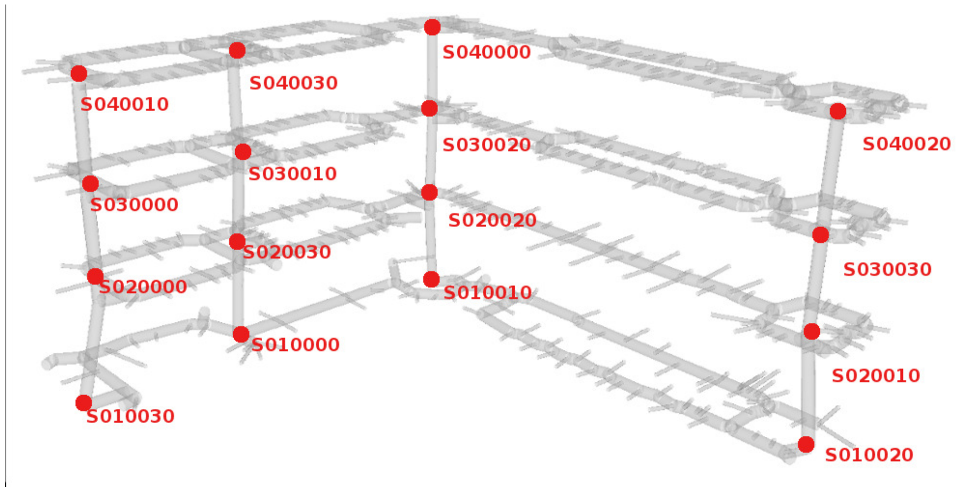


(b)

Figure 23: Building graph Simplification. Removing redundant nodes. (a) a section of a building floor with labeled building elements, (b) simplified graph.

Table 2: Rerouting Costs (full,reduced,zone) in seconds.

Evacuees	Full	Reduced	Zone
1350	98	4	.18
2885	120	6	.22
5130	330	11	.43
7264	460	14	.6

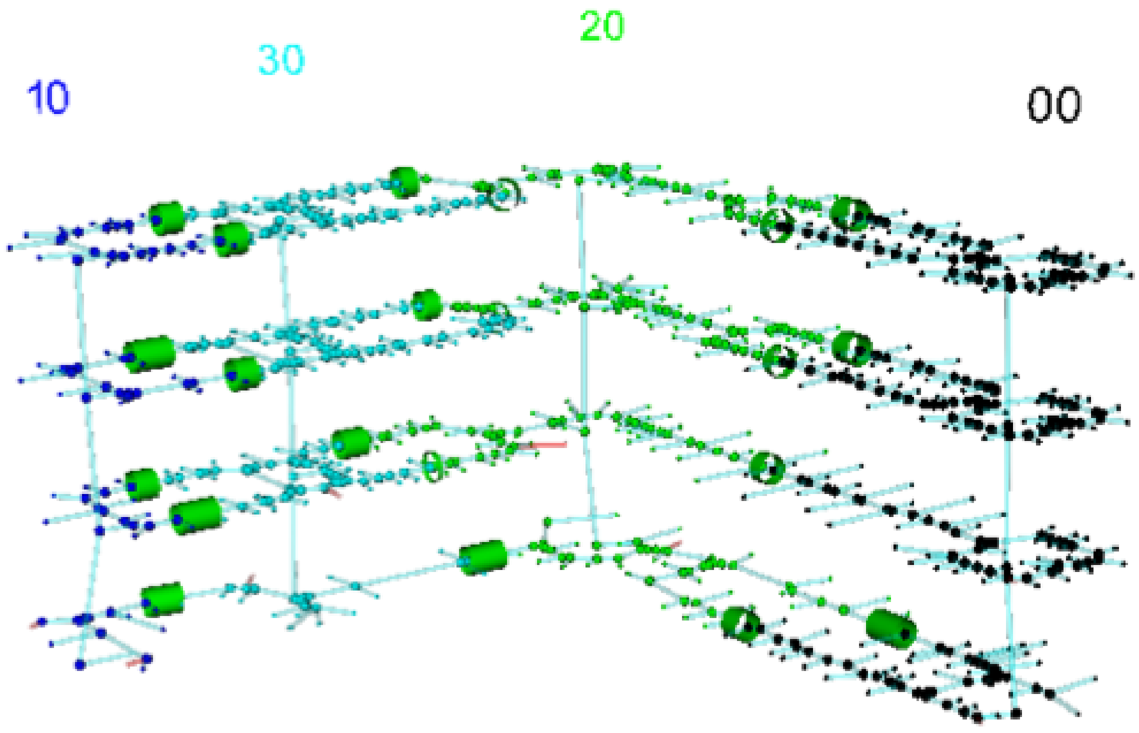


```
WOOD-0030 ['S010030_WOOD', 'S020000_WOOD', 'S030000_WOOD', 'S040010_WOOD']
WOOD-0000 ['S010000_WOOD', 'S020030_WOOD', 'S030010_WOOD', 'S040030_WOOD']
WOOD-0010 ['S010010_WOOD', 'S020020_WOOD', 'S030020_WOOD', 'S040000_WOOD']
WOOD-0020 ['S010020_WOOD', 'S020010_WOOD', 'S030030_WOOD', 'S040020_WOOD']
```

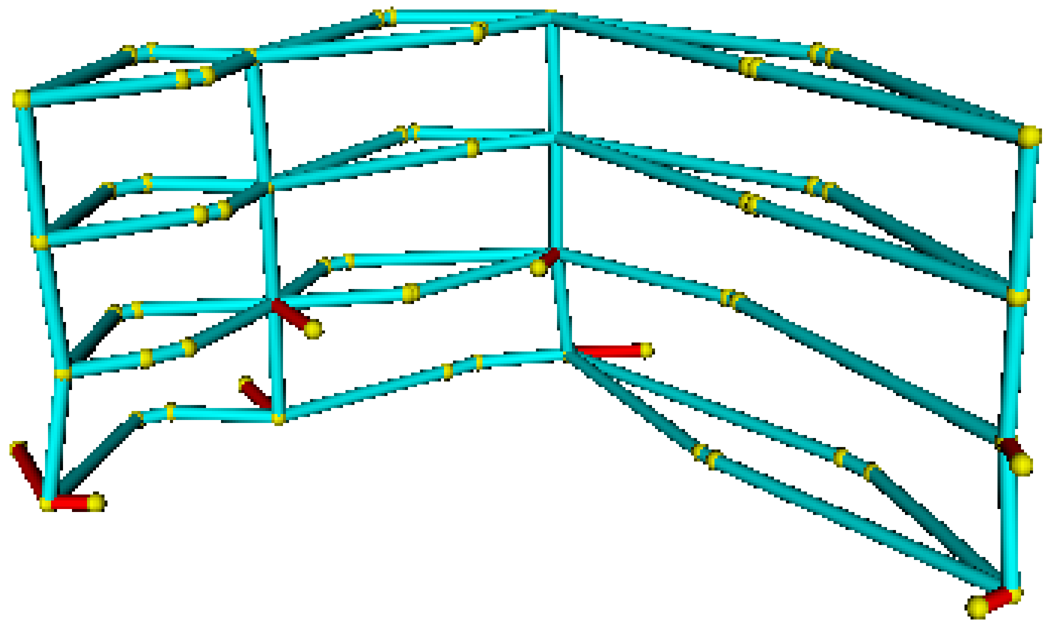
Figure 24: Backbone with associative data structure to form key nodes in the building Zone Graph.

Zone graphs are saved in the database as adjacency lists and imported into responder processes for fast situationally aware rerouting. They are also used in building the reporting tools.

The most significant factor here is the problem set itself. We are manipulating connectivity graphs for building egress. If we are to construct subgraphs of the overall graph of a building we need to define what constitutes a node and an edge in the subgraph. We chose the vertical edges(stairwells) as the backbone of our zone network. Figure 26 shows this concept visually. The building has 4 vertical stairwells which create 16 backbone nodes that will serve as the basis for construction of zone graphs. In single floor buildings the backbone will be formed by exits only. Exits are part of the multiple



(a)



(b)

Figure 25: Zone Graph Simplication. (a) building graph of a campus building, (b) transformation to zone graph representation. Yellow spheres are nodes and cyan tubes represent edges. The red tubes represent paths to exits

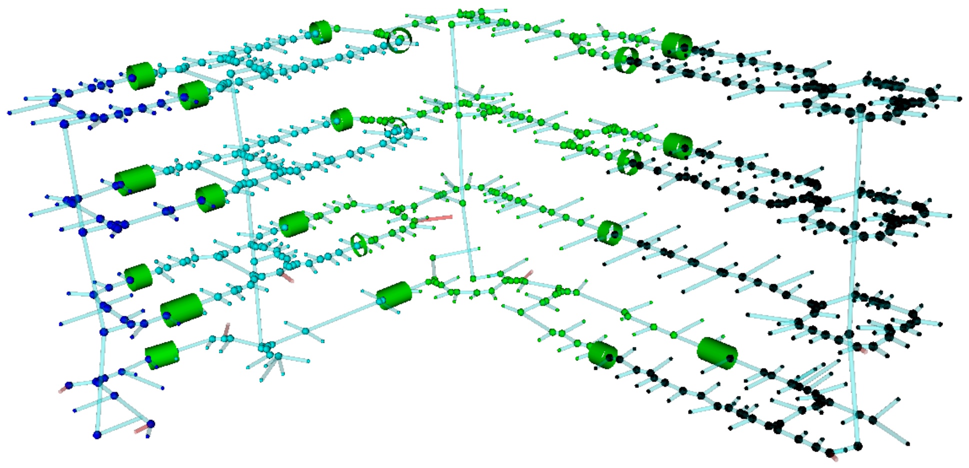


Figure 26: Backbone with associated nodes colorized by stairwell. The large green tubes show the location of extents.

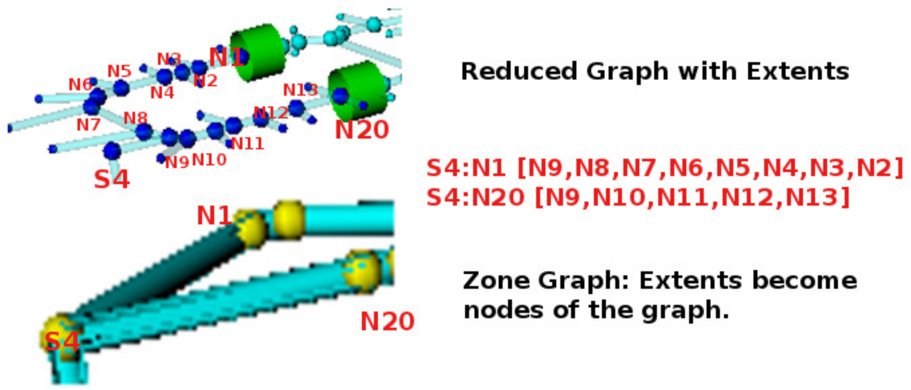

Reduced Graph with Extents
N1
N2
N3
N4
N5
N6
N7
N8
N9
N10
N11
N12
N13
N20
S4
S4:N1 [N9,N8,N7,N6,N5,N4,N3,N2]
S4:N20 [N9,N10,N11,N12,N13]
N1
N20
S4
Zone Graph: Extents become nodes of the graph.

Figure 27: When a zone graph is used for rerouting in the Simulator the results of the shortest path search return a geometric path that does not resemble that of the building. The evacuees are assigned the zone nodes and the nodes between them for a more realistic visualization.

floor building graphs also but will be added later.

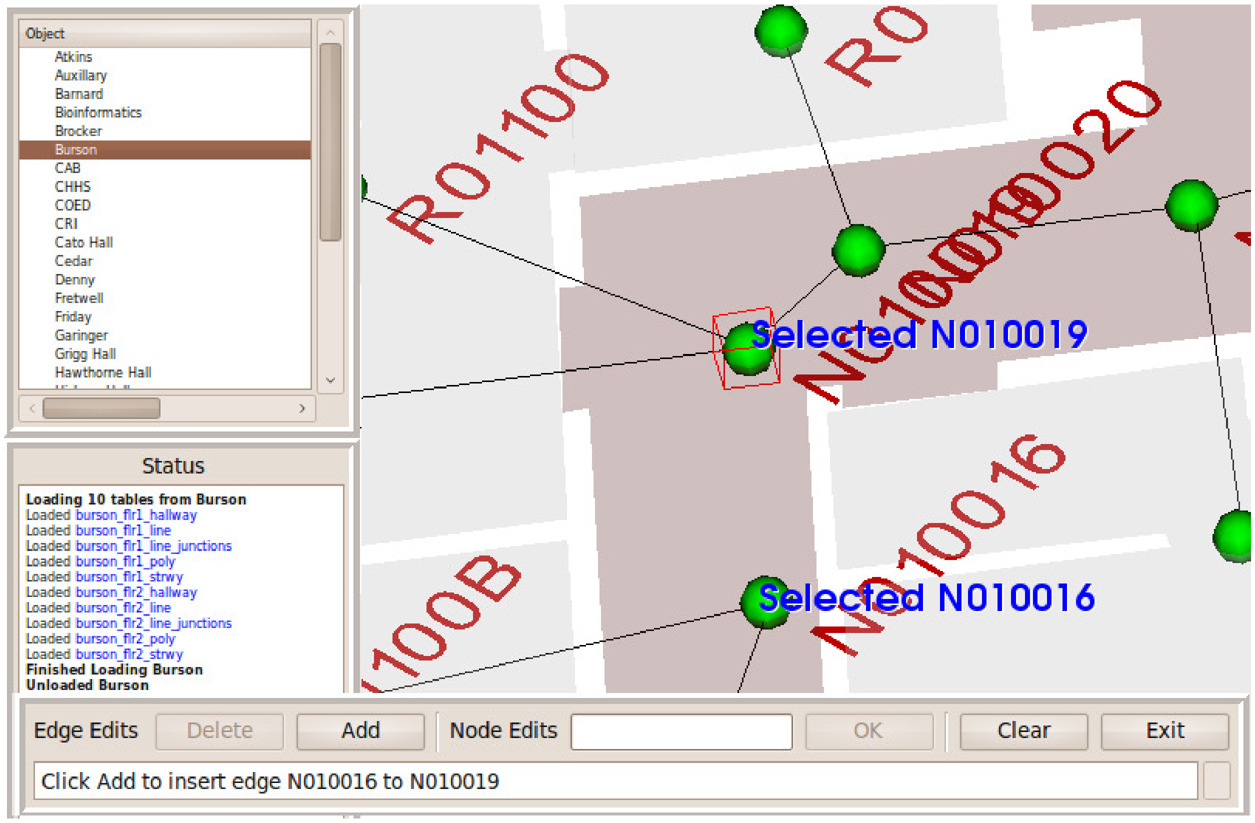
The associative array structure in Figure 24 is built as edges are loaded from the campus_adjacency_list table before a route plan is built. The next step is to identify the nodes in the overall building graph that are associated with each stairwell node and the extents(outer most nodes) for each zone. This is done in the Route Planner after a plan has been constructed. Algorithm 5 builds 2 associative arrays that define all the nodes in each zone and the points where 1 zone ends and another zone begins.

Figure 26 expands on Figure 24 by showing these zone subset structures colorized with all their associated nodes. Extents are important to building an adjacency list Zone Graph. They define where zones will be connected by edges and they serve as pointers to the nodes between them in real-time reroute mode. The final step is to add exits to the zone graphs and connect them directly to stairwells if available, or otherwise to the closest hallway node.

Algorithm 5: Associate Overall Graph Hallway Nodes with a Stairwell Zone and find Extents for each Zone.

```
Input: (1) ZoneBackbone: Array See Figure 24;
(2) EvacueeRoute: Array [EvacueeID, Path[0...n]];
(3) AllEdgeArray: Array of all edges in Reduced Building Graph: [Node1, Node2, LF]
Result: (1) ZoneNodes: Associative Array: [ZoneID, Nodes[0...n]];
(2) ZoneExtents: Associative Array [ZoneID, Nodes[0...x]];
for EvacueeID j in EvacueeRoute do
    Path = EvacueeRoute[j];
    for i = LENGTH(Path) - 1 to 1 do
                                        /* Start at exit */;
        if Path[i] is STAIRWELL_NODE then
            ZoneID = FindZoneID(ZoneBackbone);
            Add Node Path[i] to ZoneNodes[ZoneID]; Move to next EvacueeID;
        end
    end
end
                                /* Find Zone Extents */;
for Node1 N1, Node2 N2 in AllEdgeArray do
    Z1 = GetZoneID(N1);
    Z2 = GetZoneID(N2);
    if Z1 not equal Z2 then
                  /* Nodes in different zones.  Found Extents */;
        Add Node N1 to ZoneExtents[Z1];
        Add Node N2 to ZoneExtents[Z2];
    end
end
```

Manual Graph Manipulation. There are several steps required to create an efficient graph for evacuation evaluations. Our process, from initial data acquisition to the final graph includes, CAD conversion, geo-referencing, centerline geometry calculations, PostGIS SQL conversion,



```
Loading 10 tables from Burson
Loaded burson_flr1_hallway
Loaded burson_flr1_line
Loaded burson_flr1_line_junctions
Loaded burson_flr1_poly
Loaded burson_flr1_strwy
Loaded burson_flr2_hallway
Loaded burson_flr2_line
Loaded burson_flr2_line_junctions
Loaded burson_flr2_poly
Loaded burson_flr2_strwy
Finished Loading Burson
Unloaded Burson
```

Click Add to insert edge N010016 to N010019

Figure 28: User adds an edge. Note that there are also functions for deleting edges and adding and deleting nodes.

and then the processes described above. The process is robust and quality control is done at every step, however in producing the final graphs here approximately 2-3 nodes per building are left disconnected or missing.

Rather than assuming that this is always produced in the graph manipulation steps here we have included a manual function to visually inspect graphs and provide another step in quality control provided by the user. Figure 28 shows an inspection of the output of the route manipulation algorithms above. An edge is missing between a "3EDGE" node N010019 and hallway node N010016. In this particular case this is a disconnection artifact detected by Algorithm 4. A user clicks the "Add" button and an edge is inserted. Another manual function of the graph manipulation process is to build a default set of outdoor muster points associated with each exit. Currently mustering is a matter of campus procedure but locations are not identified for geo-referencing. We added these manually in our application.

Scenario Construction. Scenario construction includes loading single or multi-building evacuation objects, selecting a route planning algorithm (currently limited to our modified capacity constrained route planner[40]), and setting visualization modes and evacuation parameters, such as building capacity, egress width, evacuee speed and density, and stairway up resistance.

When an evacuation scenario is built the objects in the evacuation are combined into a structure that includes all the attributes of each character in the event. Evacuees are loaded and placed in commonly occupied areas. Pointers are assigned to the egress graph to be used for routing. The graphs are saved as “3EDGE” or Zone graphs in the database. Note however, a zone graph is not saved until at least one a priori execution of the Routing Algorithm for an evacuation object has been executed.

Routing Algorithms. We have implemented a modified version of the Capacity Constrained Route Planner(CCRP)[40], which is illustrated in Algorithm 6. Given a directed graph with node and edge capacities, the algorithm repeatedly computes the shortest path for each evacuee with available capacity. If a path is found, then it assigns as many evacuees as possible through that path, i.e., until the capacity of any of node or edge along the path is exceeded. This is followed by moving all the evacuees at that time step. The process repeats until all evacuees have found paths to exit the structure. The final step is to evacuate the remaining evacuees in the building(who already have paths, but not exited the building).

We have augmented the CCRP algorithm by specifying the movement of the evacuees(in addition to finding the paths) at each iteration. Secondly, the algorithm is modified to work with our simplified graphs; in the simplified graphs, weights of the collapsed edges are accumulated and assigned to the new simplified edges. Running the routing algorithm on the simplified graphs makes it more scalable to larger urban structures as well as facilitating dynamic changes to the graph that will require rerouting occupants around blockages or other hazards caused by the emergency event. Finally, although each evacuee has a set average speed (3 ft. per second), evacuees cannot exceed the set density threshold. Thus, as congestion builds up, evacuee movements are naturally slowed down. Additional data structures are maintained to make these computations efficient.

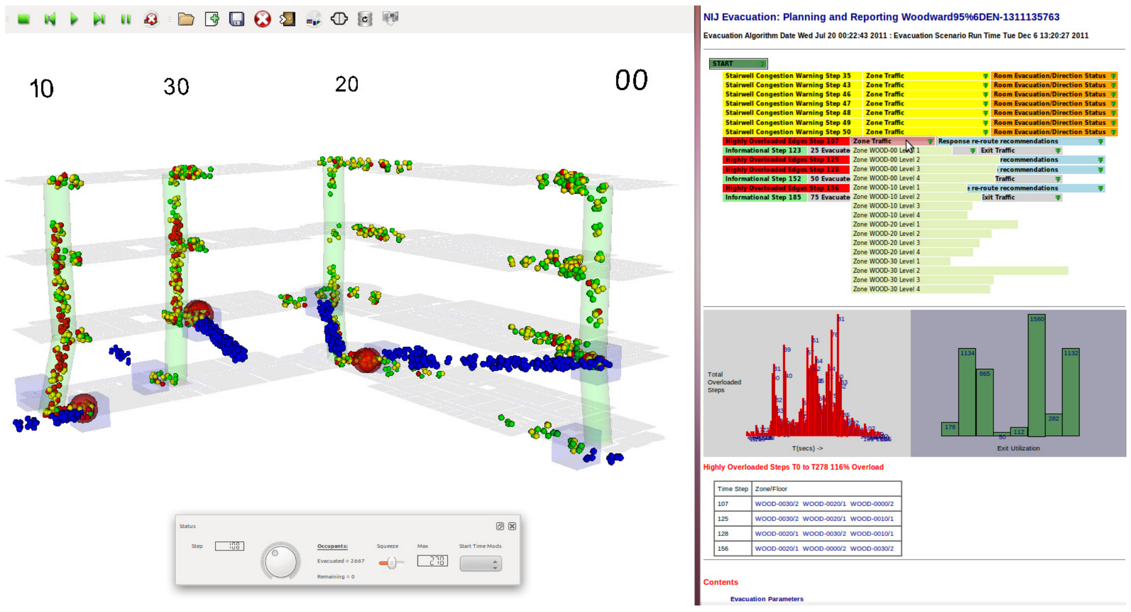
Algorithm 6: Modified Capacity Based Route Planner(CCRP) Algorithm

```
Input: ;
(1) G(N,E): Directed Graph, N nodes, E edges;
(2) Node Properties: capacity, occupancy;
(3) Edge Properties: capacity, length in feet;
(4) Set of Source Nodes;
(5) Set of Destination Nodes;
(6) Set of evacuee objects;
Result: Evacuation Plan : Routes with schedules of evacuees on each route
foreach evacuee i at each source node s do
    path_found = find shortest path p from s to all destinations      with available
    capacity;
    if path_found then
        while p < max_capacity do
            /* can route evacuee via p */ evacuees[i].path = p;
        end
    end
    move all evacuees;
end
while evacuees not at destination nodes do
    move all evacuees;
end
```

3.4.2 Visual Analytics System

The visual analytic system (Fig. 20) consists of a simulator that accepts user input during an emergency, a 3D interactive animated display of the ongoing evacuation, and reporting and analytic modules. All these views accept direct input and the views are linked to update automatically, resulting in presenting the user with the most current information.

Simulator. The simulator loads evacuation plans that were saved for scenarios under nor-



NIJ Evacuation: Planning and Reporting Woodward95%6DEN-1311135763

Evacuation Algorithm Date Wed Jul 20 00:22:43 2011 : Evacuation Scenario Run Time Tue Dec 6 13:20:27 2011

START

Stairwell Congestion Warning Step 35 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 43 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 46 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 47 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 48 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 49 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 50 | Zone Traffic | Room Evacuation/Direction Status
Highly Overloaded Edges Step 107 | Zone Traffic | Response re-route recommendations
Informational Step 123 | 25 Evacuate | Exit Traffic
Highly Overloaded Edges Step 125 | recommendations
Highly Overloaded Edges Step 128 | recommendations
Informational Step 152 | 50 Evacuate | Traffic
Highly Overloaded Edges Step 156 | re-route recommendations
Informational Step 185 | 75 Evacuate | Exit Traffic

Zone WOOD-00 Level 1
Zone WOOD-00 Level 2
Zone WOOD-00 Level 3
Zone WOOD-00 Level 4
Zone WOOD-10 Level 1
Zone WOOD-10 Level 2
Zone WOOD-10 Level 3
Zone WOOD-10 Level 4
Zone WOOD-20 Level 1
Zone WOOD-20 Level 2
Zone WOOD-20 Level 3
Zone WOOD-20 Level 4
Zone WOOD-30 Level 1
Zone WOOD-30 Level 2
Zone WOOD-30 Level 3
Zone WOOD-30 Level 4

Highly Overloaded Steps T0 to T278 116% Overload

Time Step	Zone/Floor
107	WOOD-0030/2 WOOD-0020/1 WOOD-0000/2
125	WOOD-0030/2 WOOD-0020/1 WOOD-0010/1
128	WOOD-0020/1 WOOD-0030/2 WOOD-0010/1
156	WOOD-0020/1 WOOD-0000/2 WOOD-0030/2

Contents

Evacuation Parameters

Figure 29: Visualization Design. The upper left panel is the 3D view of the building undergoing an evacuation. Spheres encapsulate evacuee population densities, permitting easy identification of congestion points in the building. Vertical tubes (light green) represent stairways/elevators. Cubes on the first and second floor indicate exits. Lower left indicates the status bar for animation control. The upper right panel is for displaying reports of significant events, that can be drilled down. Lower right panels(bar graphs) show aggregate information on exit occupancy as well as events arranged on a timeline. The report views and the 3D views are linked for immediate updates.

mal static conditions. Since dynamic changes affect only a small part of the structure, a *large amount of preprocessed data can be reused, contributing to our real-time performance.* The simulator accepts user defined dynamic changes (specification of blockages or casualty reports through the 3D display or the report modules) and modifies computed paths, and reroutes occupants. In addition it updates the generated reports and responder recommendations or action plans.

Visualization Design. Fig. 29 presents our interactive visualization system. Almost all of the interaction are via *direct manipulation.* There are three components that make up the design, (1) a 3D animated view of the urban structure, where the user can load and play evacuation simulations. Operations to manipulate the evacuation include play, pause, step forward and back, and restart the simulation. Evacuees are represented as spheres and colored green, yellow, or red based on low to high congestion. Partially transparent light green tubes represent stairwells, while. purple cubes are exits. Blue polygons represent areas that can be occupied. Large red spheres of varying opacity represent edge (capacity) congestion. On the bottom left is the *status tool widget*, that allows moving around in the animation with a slider, indicating current step, evacuee counts and total simulation time. The top right panel is the *significant event* window. The rectangular bars are menus with varying levels of detail of the simulation report. The bottom right panel is a scrolling widget with interactive charts and graphs for interacting with the simulation and visual analysis.

Following are notable features of our visualization design:

- **Visual Cues.** Congestion is the primary concern of each evacuee and predictions of future congestion and mitigation is of concern to the emergency response teams. In our system, congestion levels range from green to red (low to high). In earlier work[25] it was more common to represent evacuees as 3D human models with detailed walking and turning patterns. We believe this is unnecessary in emergency evacuation scenarios, where the evacuee densities and their locations are of prime concern to first responders.

- **Temporal Cues.** The color and size of spheres is modified at significant event times. For example sphere size is enlarged when the simulator starts moving evacuees from a source location. The resultant pulsing in the animation yields important spatial and temporal information to the user about where a new source of traffic will originate and likely areas of future congestion.

- **Details on Demand Display** The 'Significant Event' window in the reporting tool uses a colorized layered menu so that the visualization of significant information is presented as needed by the user. This allows a maximum amount of reporting while allowing for quick event scanning in the event report. A user sees a limited top level distribution of data unless there is a reason to drill down deeper into the event for details. In the top right of Fig. 29 the user has selected a *Heavy Zone Congestion* item (in red) with its time step. The user can explore further via a mouseover operation. to reveal a bar graph that shows the relative congestion of each zone in the building.

- **Direct Interaction On Linked Views.** The simulation is manipulated by direct (mouse and keyboard) interaction over the 3D animation and report views. For instance, a blockage can be introduced via the 3D view, and simulation rerun to generate new (rerouted) paths for impacted evacuees; the report view is updated to reflect the situational change. Similarly the interaction with the reports menus, charts, etc. temporally updates the 3D animation view. All such operations are performed in near real-time as both the visual representation (spheres representing collections of evacuees) and the underlying simplified graph structure allows these operations to be rapidly performed, facilitating interactive visual analysis.

4 Results

4.1 Implementation

We have implemented our system in C++. We use the open source tools, including PostgreSQL[2] and PostGIS[1] as the main database server. ArcGIS was used for initial preprocessing. The desktop application uses the OpenGL for all the graphics and FLTK[45] for the graphical interface. Future work on the mobile application will use more low-level tools, to scale better with the limited resources available on hand-held devices.

4.2 Building Network Generator

We have successfully applied our building network generator on over 70 campus buildings and residence halls of UNC Charlotte. After processing, these are stored in the PostgreSQL database for use in our emergency response application. As described earlier, the database is the main means of communication and acts as the bridge between the command center and the mobile responder applications.

Tables 3 and 4 show a comparison of performance using our automation tools for 3D network construction on two of our campus office/classroom buildings. Processing time ranges from 10-20 min. using our tools. Majority of the processing time is due to interactive processing. Doing the entire processing manually takes anywhere from 2-4 hours. For more complex buildings (the buildings in our experiments had 3 and 4 floors respectively) manual processing will be impractical.

4.3 Example Evacuation Scenarios

Here we describe three experimental scenarios to illustrate the use of our situationally aware evacuation system. In this example scenario, there are 3500 evacuees. The maximum egress capacity is set to 1 evacuee per cubic foot and navigation speed is set at 3ft/sec (congestion can slow down or halt evacuees during a simulation). The building is loaded to 95 percent capacity.

Building	Adj. Graph Constr.	Centerline Extraction	Bldg. n/w Constr	Manual Proc.
Cameron	4.0s	0.22s	7.1 s	540s
Woodward	5.82s	1.04s	0.34s	1080s

Table 3: Performance: Semi-Automatic Method

Building	Centerline drawing	Network creation	Attributes assignment
Cameron	60 minutes	5 minutes	60 minutes
Woodward	90 minutes	5 minutes	120 minutes

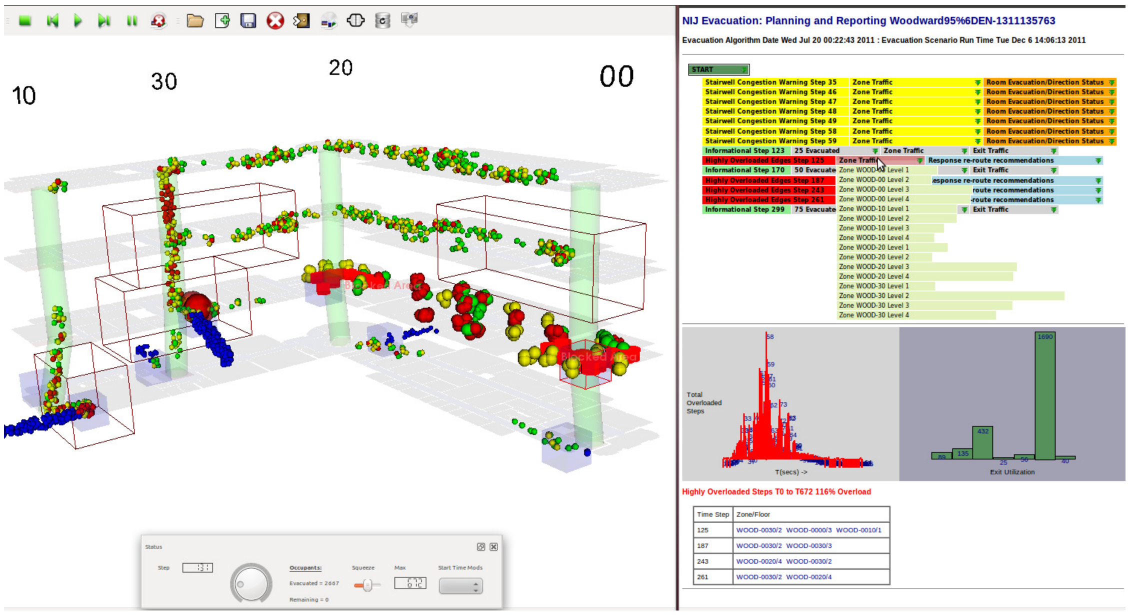
Table 4: Time usage:Manual Processing

4.3.1 Situation 1: No Blockages

Figure 29 is a screen shot of our visual analytic system loaded with a campus building with no inaccessible areas. The 3D animation view is a direct representation of the evacuation scenario which was built in the scenario construction phase of the process. Evacuees are represented by spheres, clustered visually using an algorithm which arranges them based on egress width. A small red sphere indicates an evacuee cluster on a congested step. Large red spheres of varying opacity indicate congestion greater than 116 percent of rated capacity(goal was to test an overloaded situation to assess system performance). In the timestep shown in Fig. 29, the user has clicked on the reporting panel(upper right), representing a highly loaded event at timestep 108 sec. The user has also rolled over the zone congestion bar to reveal the detailed zone congestion graph. As indicated by the bar at Zone 30 Level 2 this is the most traveled and congested route. The application suggests that a responder be dispatched to this area. As the user rolls over the associated 'Response Reroute Recommendations' menu bar in the list the recommended action can be made visible.

Easy access for responders and special evacuation needs can be found by looking at the green exit utilization bars(lower right of Fig. 29). If an area has become too congested for a responder, he or she can be dispatched to exits with lower utilization, and then can direct evacuees toward less congested areas. The zone exits in this scenario that are not utilized are on the 1st level. This type of information can be a powerful dispatch tool. The emergency commander has what he/she needs to make a decision: the amount and location of the congestion in the context of the overall evacuation and where related less congested paths are located.

Each bullet in the significant event list (upper right panel in Fig. 29) serves as a visual clue to the overall execution of the scenario. The list covers the entire evacuation. The yellow stairway warnings can be drilled deeper to see which areas are becoming congested. These cues are important for responders to quickly react during the beginning of an event or if a campus lock down has been released. The room evacuation, direction status options can be rolled over to indicate the direction from which the traffic is proceeding, which in turn could result in congestion at a later point.



NIJ Evacuation: Planning and Reporting Woodward95%6DEN-1311135763

Evacuation Algorithm Date Wed Jul 20 00:22:43 2011 : Evacuation Scenario Run Time Tue Dec 6 14:06:13 2011

```
START
  Stairwell Congestion Warning Step 35   Zone Traffic   Room Evacuation/Direction Status
  Stairwell Congestion Warning Step 46   Zone Traffic   Room Evacuation/Direction Status
  Stairwell Congestion Warning Step 47   Zone Traffic   Room Evacuation/Direction Status
  Stairwell Congestion Warning Step 48   Zone Traffic   Room Evacuation/Direction Status
  Stairwell Congestion Warning Step 49   Zone Traffic   Room Evacuation/Direction Status
  Stairwell Congestion Warning Step 58   Zone Traffic   Room Evacuation/Direction Status
  Stairwell Congestion Warning Step 59   Zone Traffic   Room Evacuation/Direction Status
  Informational Step 123   25 Evacuated   Zone Traffic   Exit Traffic
  Highly Overloaded Edges Step 125   Zone Traffic   Response re-route recommendations
  Informational Step 170   50 Evacuate   Zone WOOD-00 Level 1   Exit Traffic
  Highly Overloaded Edges Step 187   Zone WOOD-00 Level 2   esponse re-route recommendations
  Highly Overloaded Edges Step 243   Zone WOOD-00 Level 3   route recommendations
  Highly Overloaded Edges Step 261   Zone WOOD-00 Level 4   -route recommendations
  Informational Step 299   75 Evacuate   Zone WOOD-10 Level 1   Exit Traffic
                                        Zone WOOD-10 Level 2
                                        Zone WOOD-10 Level 3
                                        Zone WOOD-10 Level 4
                                        Zone WOOD-20 Level 1
                                        Zone WOOD-20 Level 2
                                        Zone WOOD-20 Level 3
                                        Zone WOOD-20 Level 4
                                        Zone WOOD-30 Level 1
                                        Zone WOOD-30 Level 2
                                        Zone WOOD-30 Level 3
                                        Zone WOOD-30 Level 4
```

Highly Overloaded Steps T0 to T672 116% Overload

Time Step	Zone/Floor
125	WOOD-0030/2 WOOD-0000/3 WOOD-0010/1
187	WOOD-0030/2 WOOD-0030/3
243	WOOD-0020/4 WOOD-0030/2
261	WOOD-0030/2 WOOD-0020/4

Figure 30: Two blockages have been placed in the building at 45 seconds into an emergency evacuation. (1) Floor 2 at zone 20 stairwell and (2) floor 2 at zone 00 stairwell. Individuals are shown trapped between them by enlarged spheres. Wireframe cubes indicate traffic flow areas, obtained by rolling over the bar chart at the bottom of the report window.

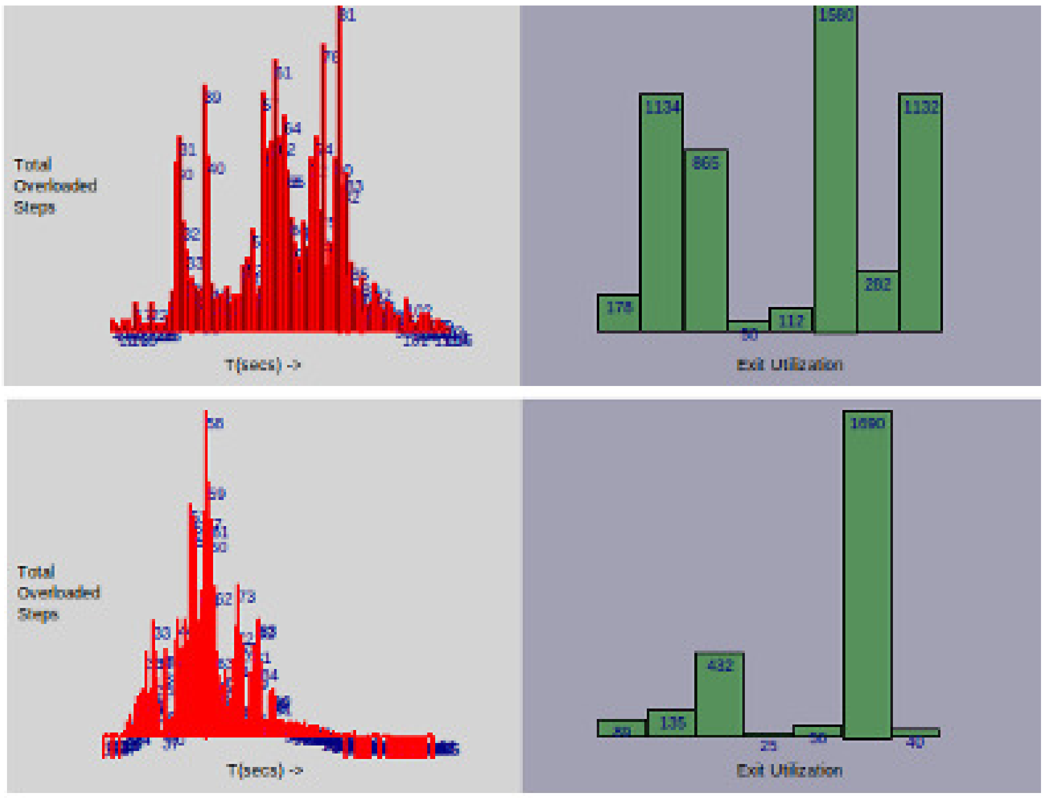
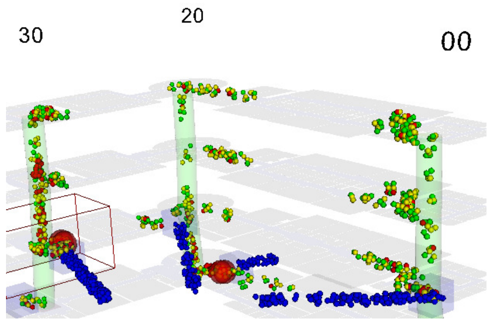


Figure 31: Before and after congestion and exit charts.

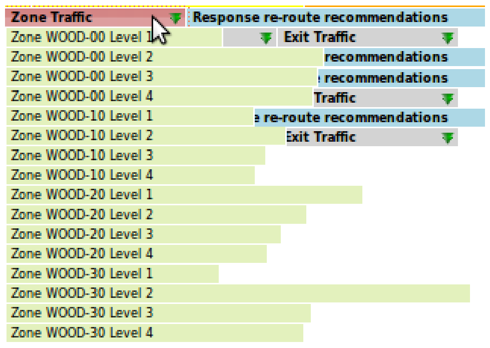
4.3.2 Situation 2: Induced Blockages

As illustrated in Fig. 30, two blockages have been introduced into the scenario of Fig. 29. The blocked areas represented by red squares spread out over several square feet on the second floor at zone 00 and zone 20. In this example, a total of 3100 evacuees were rerouted and all reporting re-calculated in 2.8sec. Note that in the control bar at the bottom of the 3D animation view the maximum evacuation time has increased to nearly 7 minutes from less than 3 minutes.

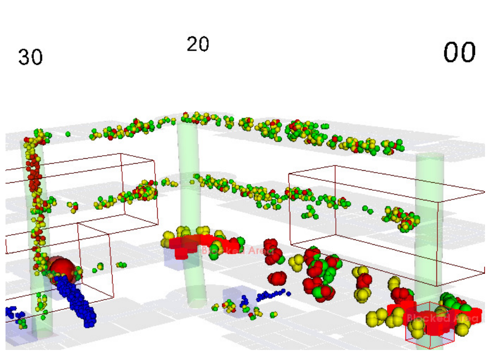
Fig. 31 shows the drastic shifts in the movement of people from a standard evacuation of the building. This example serves to show that evacuation modeling of normal (non-blocked) scenarios is considerably different than when blockages are introduced. In particular, notice the difference in the utilization of the exits (right panel). The left panel shows the congestion amounts as a function of time. In the blocked case, the congestion occurs earlier and is



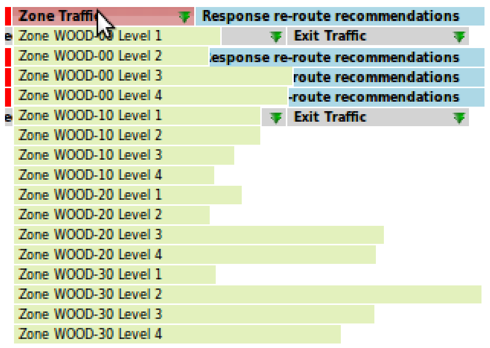
(a)



(b)



(c)



(d)

Figure 32: Contrast between normal(unblocked) vs. blocked scenarios. Two blockages have been introduced. (a,b) 3D view at 108 seconds into the evacuation, with top floors mostly evacuated, (c,d) 3D view at 139 sec. Floors 3 and 4 are still heavily occupied. Zone traffic (right panels) confirm and illustrate the aggregate picture of the traffic across the entire evacuation.

steeper, with a longer tail, resulting in longer times for all evacuees to exit the building. Probing these congestion points can immediately bring up the 3D view at that instant of time to see the evacuee locations, thus providing a powerful means to quickly understand the situation.

Mouse rollover on the significant event list view shows that the third and fourth floors are getting backed up above the exit at zone 20 floor 2 which is loaded heavily even during a normal scenario. Because the event occurred early there were a number of evacuees occupying the upper floors.

Fig. 32 further contrasts the blocked and non-blocked cases. Here the top row shows the normal(unblocked) case and the bottom row shows the blocked case. We compare the zone traffic via the light green bar charts. The bar charts show total zone traffic from the beginning to the end of the evacuation scenario. The snapshot of the building in Fig. 32a,b is at 108 seconds and Fig. 32c,d at 131 seconds. Even though the bar chart for the blocked case is the total picture it is still clear from both the building and zone traffic charts that the traffic on the top two floors is heavier in the blocked case (Fig. 32c,d), and in particular, is shown by the size of the bars labeled Zone WOOD-20 Level 3, Zone WOOD-20 Level 4, Zone WOOD-30 Level 3, and Zone WOOD-30 Level 4. The total picture of the bar chart and the single step picture of the building reinforce each other. Each can stand on its own but more insight is gained if they are viewed together. Comparison of non blocked and blocked evacuation visualizations based on these factors can give previously unavailable clues on critical areas of the building across multiple scenarios.

4.3.3 Situation 3: Rerouting Evacuees

When certain parts of a building are blocked, our system will automatically reroute all evacuees in that area to other nearby exits. Additionally, our system provides rerouting functions that allow a selected number of evacuees to be rerouted; for instance, in congested areas it might be desirable to move evacuees to other exits. This operation is performed in real-time and the simulation played through to evaluate the traffic or congestion patterns resulting from such an intervention. This could then let a commander make a more informed decision and dispatch responders to the affected areas in the building.

Figure 33 illustrates an example evacuation from a cluster of 4 academic buildings. The original evacuee densities are illustrated in panel 2. Exits 8 and 16 are heavily used, as indicated by the area of their exit circles. The red cubes in panel 1 have been interactively selected(panel 3 shows a zoomed-in view of these areas) for rerouting occupants within those areas. This is followed by specifying the number of evacuees to be rerouted to specific exits (here exits 2 and 23 were chosen). Panel 4 shows the results of these actions, leading to reduced densities at exits 8 and 23.

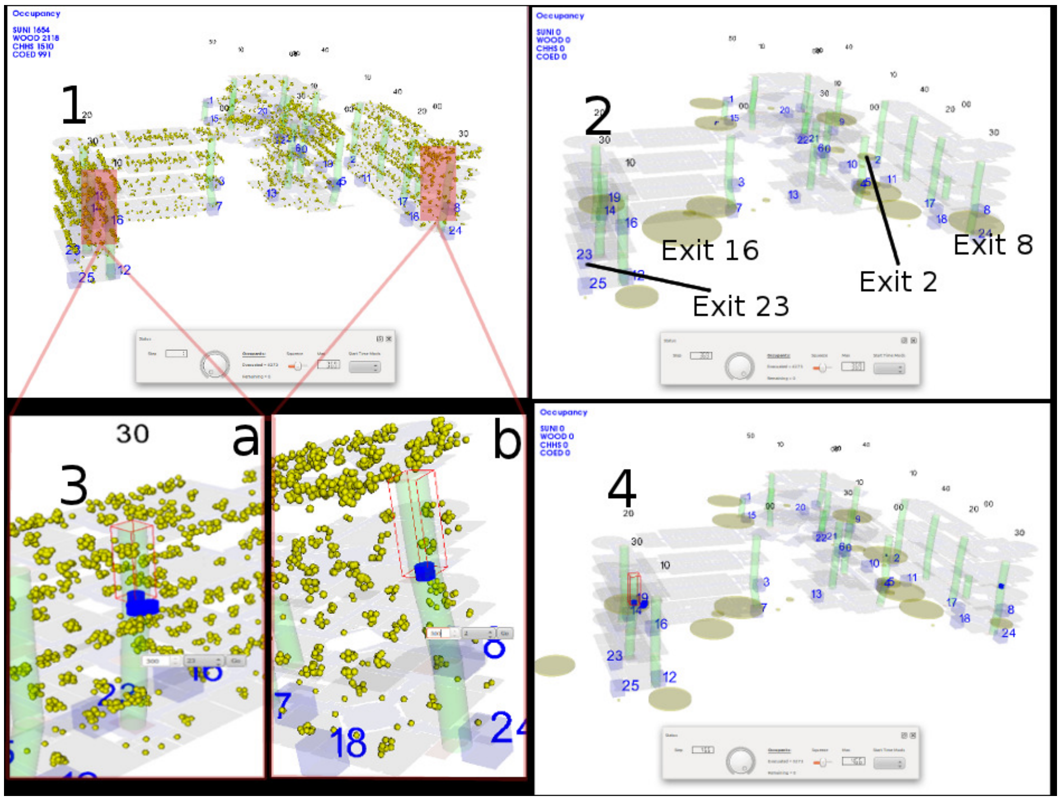


Figure 33: Rerouting evacuees from congested areas in a 4 building evacuation simulation. **1.** Four building scenario. **2.** Resulting evacuee density circles after simulation. **3a and 3b.** User added rerouting flags as indicated by the blue discs and associated with the opaque red rectangles in 1. **4.** Resulting evacuee density circles after modified simulation, changing the routes of evacuees to exit 2 and exit 23 in their respective buildings.

This function has value for planning and training. Building lockdown and release operations can benefit from such 'what-if' style scenarios that brings together rich spatio-temporal information into the hands of first responders. This is facilitated by the responsiveness of the system and thus its effectiveness in real-time decision support. In this example, running the entire scenario from initiation to results and analysis took approximately 2 minutes. When large collections of buildings are involved with traffic routed to the adjacent street networks, such tools can be invaluable for effective and timely evacuation as well as optimal asset deployment decisions.

5 Evaluation

The majority of our evaluations have been in collaboration with the UNC Charlotte Police department, and the Risk Management, Safety and Security (RMSS) office. We have conducted two training events using our system and third event in July of 2012 was completed using our Atkins Libray (high rise building). This last even had extensive participation and media coverage.

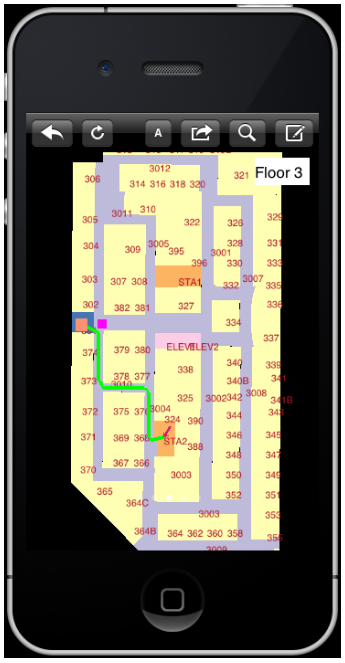
Early versions of the system were used in two smaller tests, (1) digitization of part of an office building at Pacific Northwest National Laboratory(PNNL), as part of the First Annual Enterprise Resiliency Experiment[18] and a search/routing experiment, and (2) digitization of a hotel in Washington DC hosting the NIJ Technical Working Group meeting.

5.1 PNNL Experiment

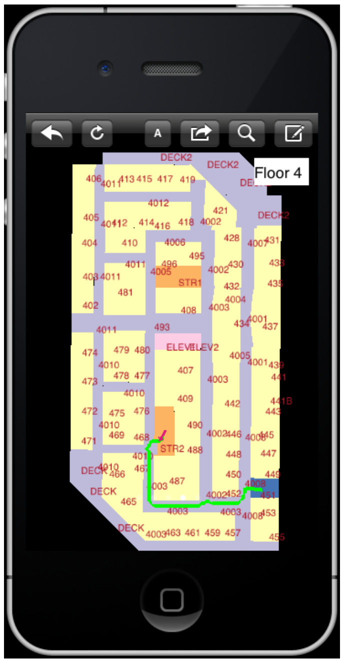
An early version of our system was demonstrated at PNNL(summer 2011) as part of the *First Annual Enterprise Resiliency Experiment*, sponsored by the Defense Research and Development Canada (DRDC) Centre for Security Sciences (CSS) and U.S. Department of Homeland Security (DHS) Science and Technology Directorate.

This experiment involved digitizing three floors of a PNNL office building and a live demonstration of our search and routing functions. Digitizing the building was performed ahead of the experiment using our custom tools, as described in Section 3.2) The live demonstration was performed on site using the command center and responders navigating the building. The responders used an Apple iPad. Internet voice and video were used for live communication (simulating VOIP and radio) over separate communication devices. The command center performed all responder location and routing functions. The responders used the device for informational purposes only: an output only device. To simplify responder actions, all responder locations and search criteria were computed on the command center.

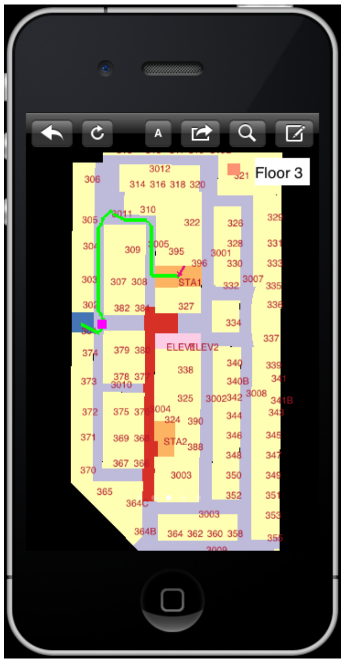
The experiment scenario consisted of the following sequence of actions:



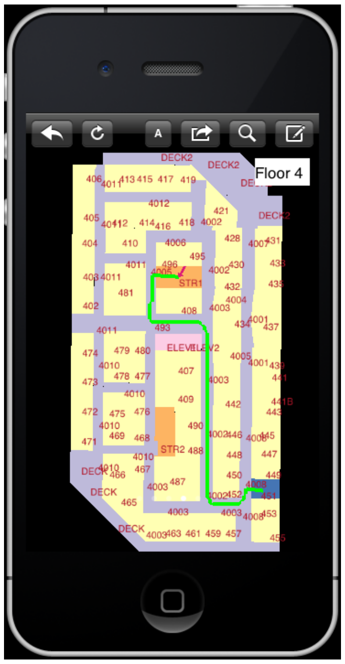
(a) 3rd floor(pre-blockage)



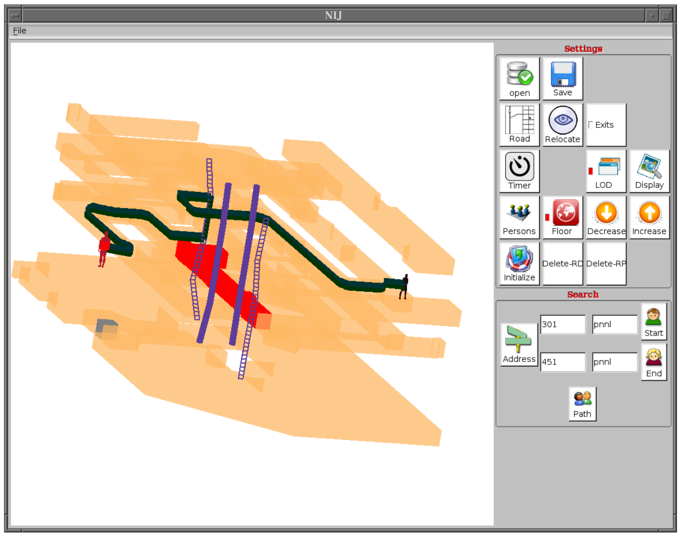
(b) 4th floor(pre-blockage)



(c) 3rd floor(after blockage)



(d) 4th floor(after blockage)



(e) Command Center view

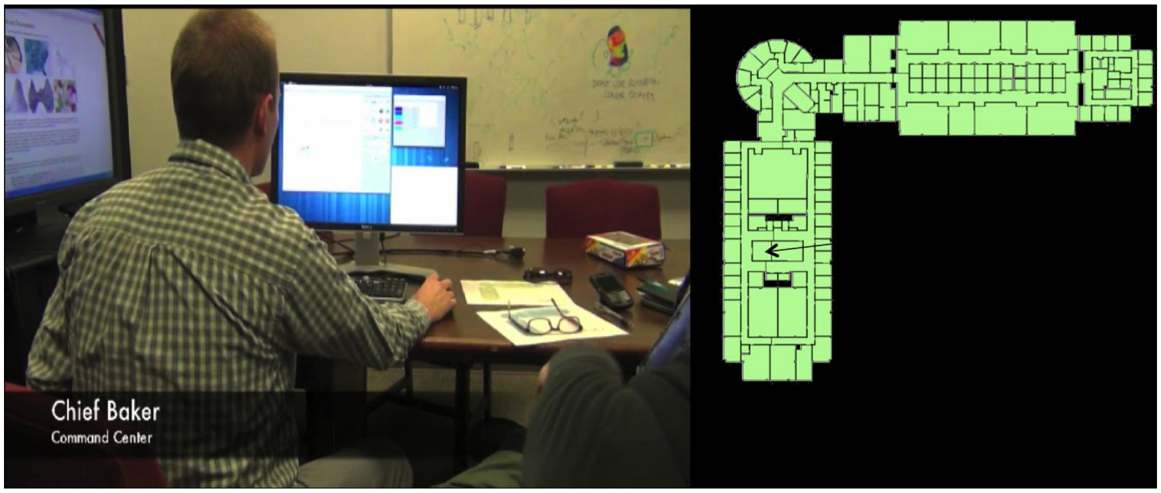
Figure 34: PNNL Experiment - Searching/Routing for a trapped victim in a PNNL Office Building. When the command center was notified of the blockage, the route to the victim was re-computed; the responders' devices automatically fetched command center changes.

1. Incoming call from a trapped woman on the fourth floor.
2. Command center asks responder,“where are you?”.
3. Responder replies with location.
4. Command center computes route to the victim.
5. Responder follows the path, with live video displayed at the command center.
6. Responder encounters a blockage.
7. Command center reroutes responder.
8. Responder follows new path to victim.
9. Victim is found.
10. Command center operator leads the responders out of the building, visible as newly computed routes on their devices.

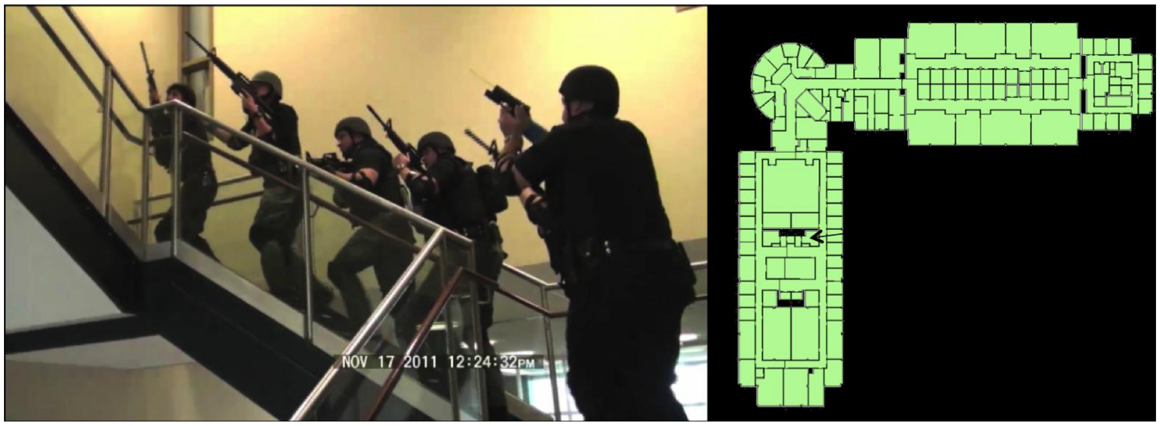
Fig. 34 illustrates snapshots of this experiment. The search route was from room 301 to room 451. Paths on the mobile are shown on both floors before and after the blockage is encountered. Also shown in 3D is the view from the command center. The blockage(in red, on the mobile and command center) on the third floor was discovered after the responders began their search. When the command center was notified of the blockage, the route to the victim was computed; the responders' device polls and downloads any command center changes automatically.

5.2 Training Event 1: Office Building

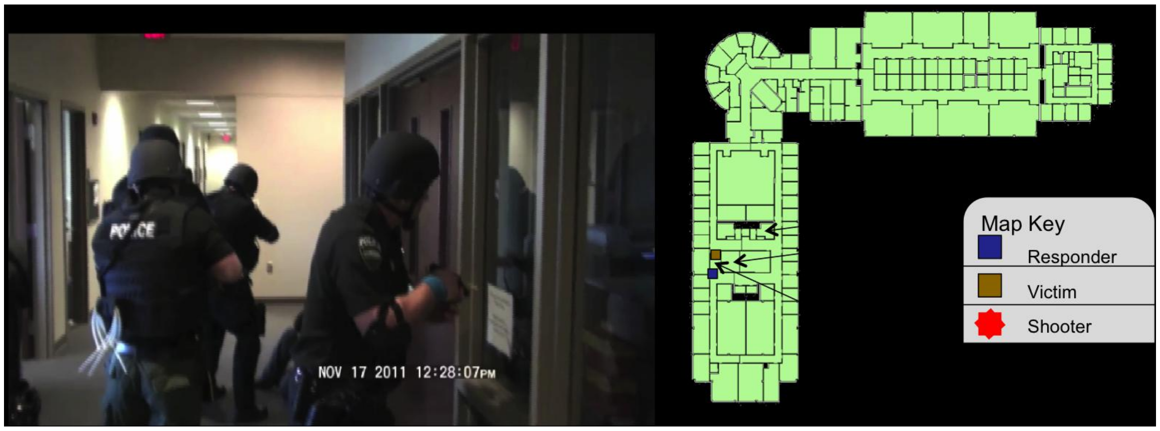
This event was conducted on November 18, 2011 and simulated a *Shooter in a Building* scenario. We used a part of Woodward Hall, a campus office and classroom building to perform the exercise. The event involved about 11 police officers (SWAT team officers),



(a) Command Center

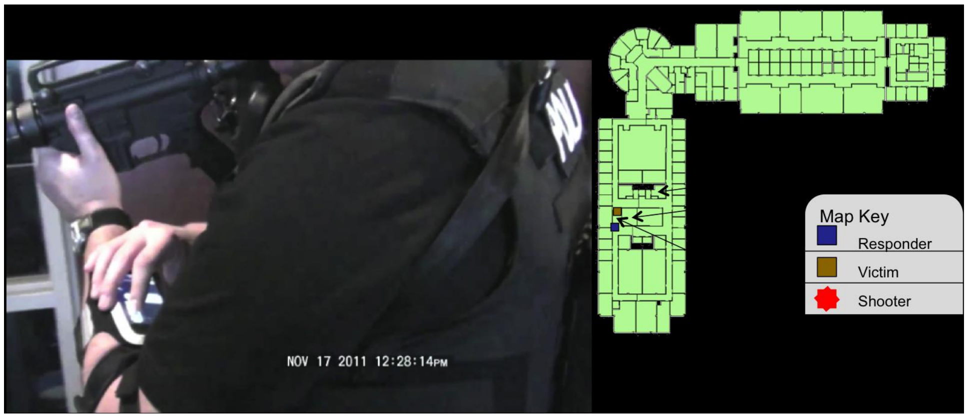


(b) SWAT team moving to the third floor.

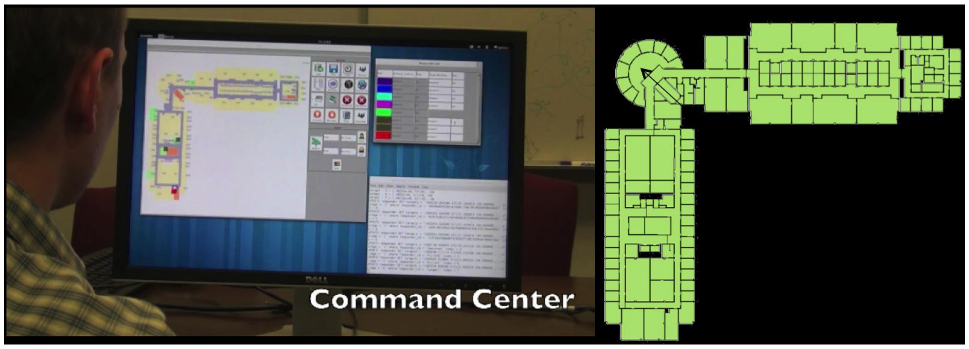


(c)

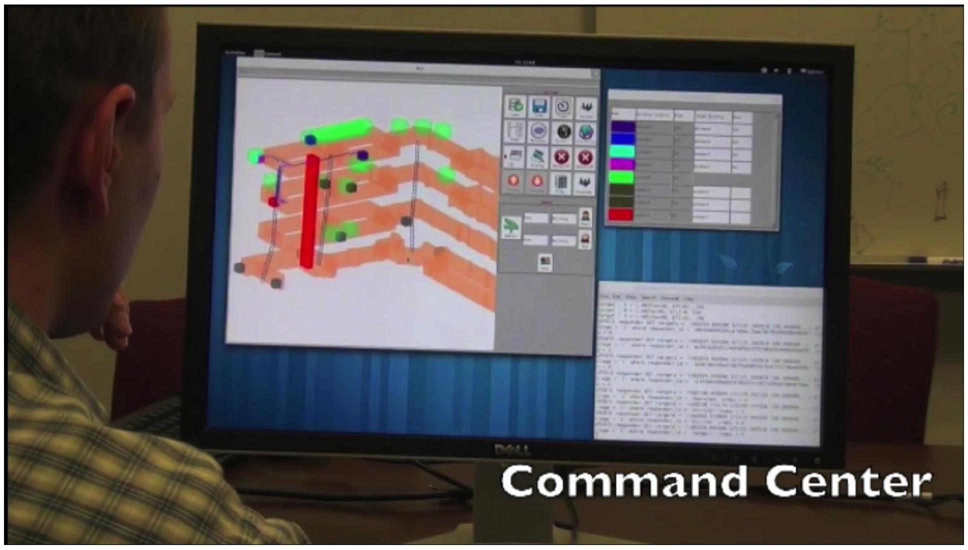
Figure 35: Training Event: Shooter in an Office Building - Sequence of Events.



(a)

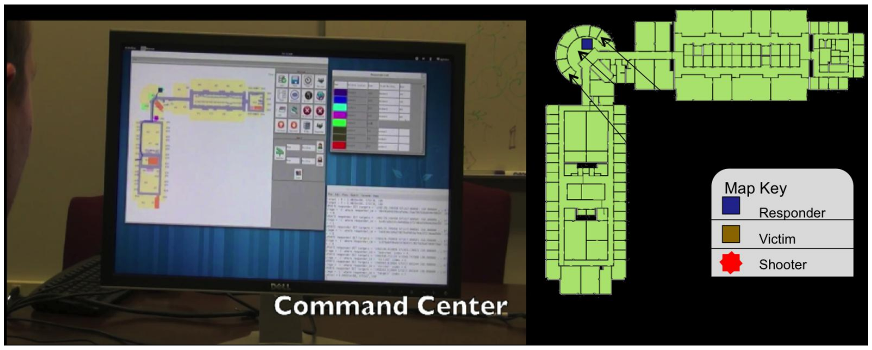


(b)

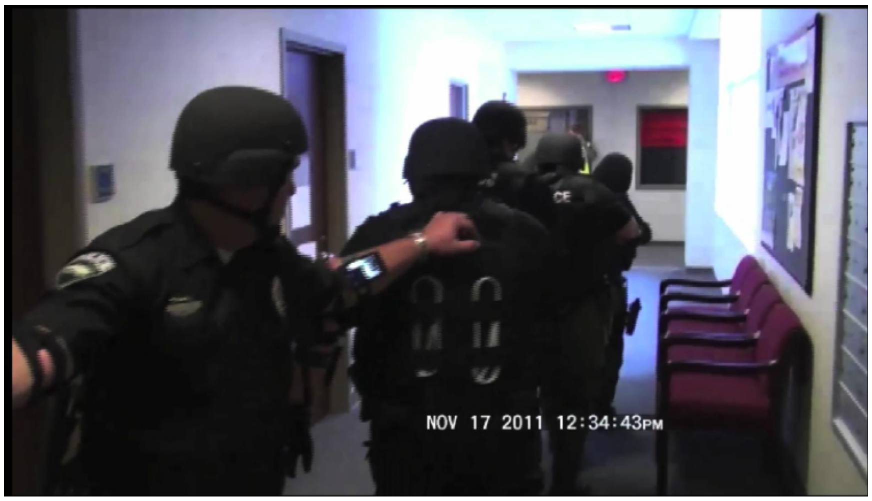


(c)

Figure 36: Training Event: Shooter in an Office Building - Sequence of Events.



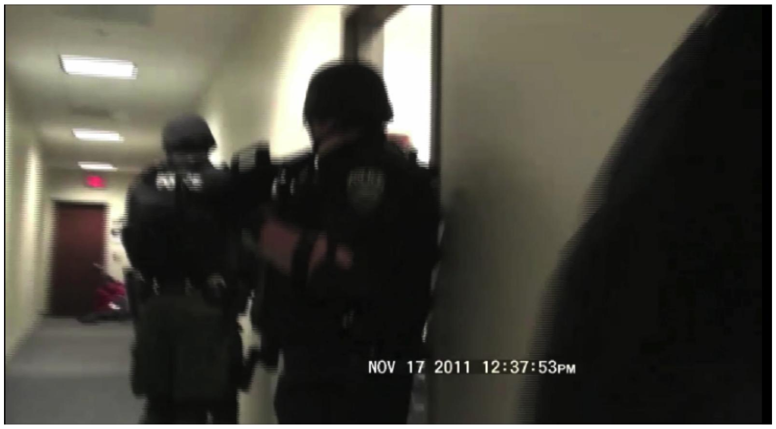
(a)



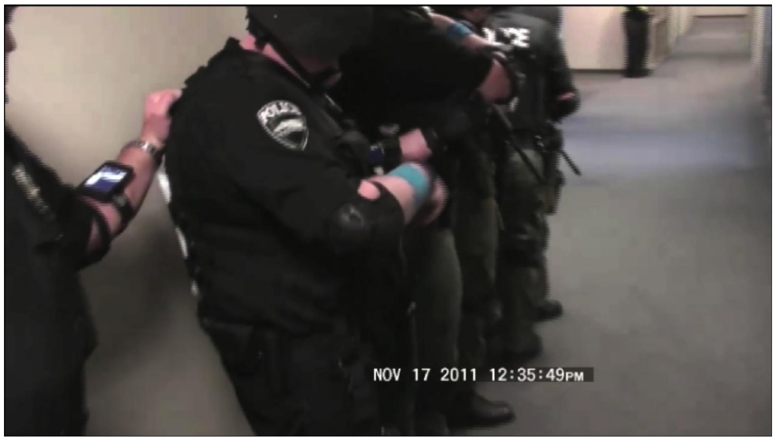
(b)

(c)

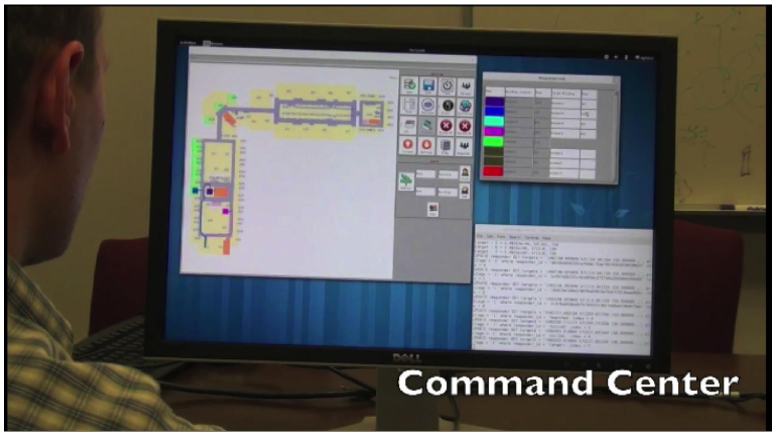
Figure 37: Training Event: Shooter in an Office Building - Sequence of Events.



(a)

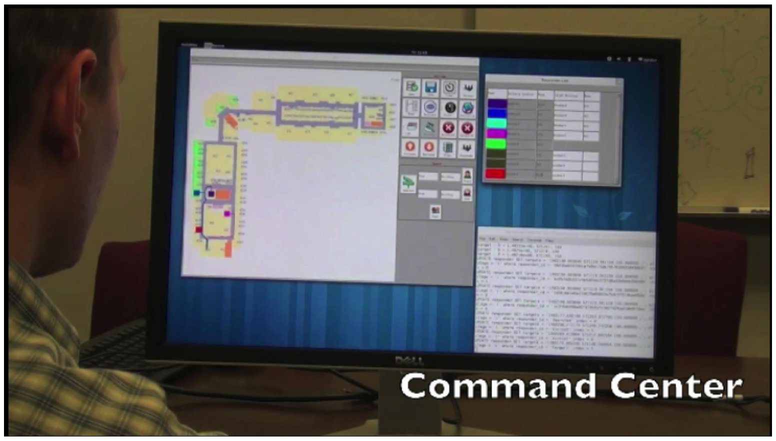


(b)

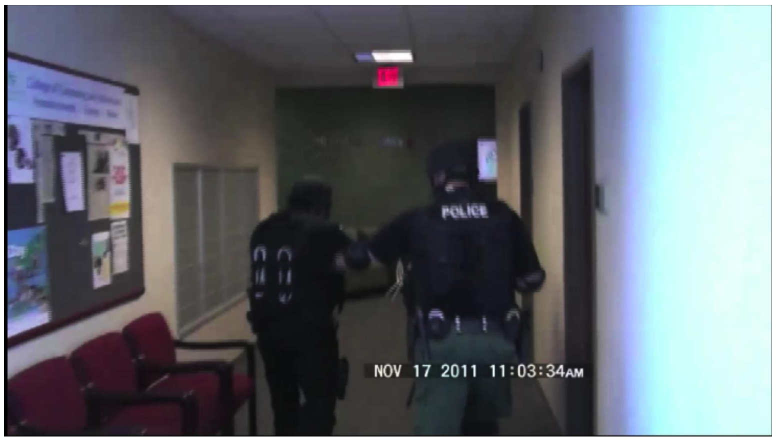


(c)

Figure 38: Training Event: Shooter in an Office Building - Sequence of Events.



(a)



(b)

(c)

Figure 39: Training Event: Shooter in an Office Building - Sequence of Events.

Figure 40: Training Event: Shooter in an Office Building - Target neutralized and building secured.

12-15 student volunteers to perform the roles of victims and observers for data collection tasks. The command center was directed by the chief of Police (Mr. Jeff Baker) and one of our software team members. Extensive notifications via campus communication and posters on building entrances were used to warn building occupants of the upcoming event. The simulation also involved the campus communications and publicity personnel.

Prior to the start of the exercise, the SWAT team members underwent a short training phase (conducted by a software team member) to become familiar with the mobile device(responder) functions. The training involved understanding the graphical layouts, the interface to report current location and basic navigation.

There were two iterations of this exercise. The exercise had been planned for the incident commander to oversee the event from UNC Charlotte's Mobile Command Center. However, due to wireless networking and connectivity issues, it was halted after about 30 minutes. The command center was then shifted to the building (see Fig. 35(a)) to the building's wired network and the exercise restarted. The mobile devices continued to use the campus wireless network for all communications.

5.2.1 Training Event: Sequence of Events

Figures 35-40 show snapshots of the sequence of events of the training exercise. Floor views of the building (in green to the right) are also shown in some of the figures for spatial context.

The exercise began with the SWAT teams (code numbered 4, 7, 9) on the second floor. Command post received their location and team 4 was dispatched to room 308(as seen in Fig. 35(b)) and the path to the target location was also transmitted to team 4. The closestt SWAT team cleared the hallway and located a victim at room 308(Fig. 35(c)). They reported their location via the "Here I am" function (Fig. 36(a)). The location of the victim is reported via radio to the command post, and is added to the command center display(which is immediately transmitted to all responders).

At this point, command center dispatched team 4 to the fourth floor, room 443(Fig. 36(b)), and sent them the path. Orders to clear the fourth floor were sent by radio. Command center updated the rooms cleared or searched by the SWAT team. These rooms are illustrated by the green boxes overlaid on the rooms; Figs. 36(c),37(a) show the elbow of the fourth floor being cleared. The teams are pictured in Fig. 37(b) as they continue to clear the building. The command center was then notified of their progress.

Fig. 37(c) shows the mobile device being updated. Though it is not evident from the figure the team has progressed to the end of the hallway shown in Fig. 37(b) and has cleared the building to the right. Notice the time difference of approximately 23 seconds. At this time the command center is aware of the SWAT team's progress as information is available to him from a spatio-temporal perspective. The police chief, serving as incident commander, appreciated the situational awareness; visually he could track his officers, understand the current situation in real-time and make informed decisions in the midst of the crisis.

The SWAT teams continue around the hallways as ordered by the commander clearing rooms and informing command of their status(Fig. 37(c),38(a),38(b)). The chief was engaged in discussions with the public relations staff who were also part of the exercise. Fig. 39(a) shows the SWAT team progress as green colored polygons along the far left of the screen. The rectangular icons indicate their position. The widgets on the right identifies the (color-coded) teams in the building with names and ids.

At this time a victim is observed at the end of the fourth floor hallway(Fig. 39(c)). The command center is notified by radio and a new victim icon is placed on the screen. Public Relations(PR) is informed for transmission to appropriate campus authorities. The chief noted later that referring to the command center display was a useful tool to keep track of the significant events of the incident.

Fig. 40 illustrates the neutralization of the target; however, this is from the first iteration of the exercise. A similar(but not recorded) neutralization of the target was part of the second iteration of the exercise.

5.2.2 Training Event: Data Collection

Extensive data was collected during the training event. Each SWAT team and the command center was captured on video for postmortem analysis. In addition, an observer accompanied each SWAT team to make notes on the paths followed by the team and note for signs of confusion, disorientation or retraced paths. In addition, feedback was obtained from the observers, the command center (police chief) and the officers who participated in the event. All data was collected and digitized for later analysis.

5.2.3 Training Event: Observations

Feedback from the exercise was collected during and after the exercise. Observations related primarily to the operation and software system on the mobile devices. They are as follows:

SWAT Officers Feedback

1. As the exercise progressed, the officers became more comfortable and proficient with the mobile system interface; there were issues with location reporting, via the "Here

I am" function. Closer analysis revealed that this was due to network connectivity issues, and the underlying software. These have been corrected and improved for the upcoming training event.

2. The SWAT officers suggested the ability to input victim location and using a color-coding scheme to victims, based on the severity of the injuries, viz., red for critical, yellow for minor injury, green for low priority. This would help prioritize assistance to victims, and find good routes through the building on subsequent sweeps.

3. Officers pointed out that the "Here I am" function was not working in conjunction with zoom function. This was a known limitation and has since been corrected. This is needed when rooms in very large buildings can appear to be small and difficult to select, requiring zooming.

Feedback from Team Observers

An observer accompanied each SWAT team and noted any issues with both the mobile device operation as well as the software system.

1. Even with the initial training on the operation of the mobile system, officers had a noticeable learning curve in using the system. Further ease-of-use improvements will be needed.

2. Observers noted that the officers were able to anticipate their actions by looking at the building geometry ahead and planning their strategies. For example a T in hallway created a dangerous situation. They were able to see these on the mobile device ahead of time and alert the officers leading the team, and thus plan for upcoming hazards.

3. One of the SWAT teams got temporarily lost/disoriented after exiting the stairs, prior to sweeping through the building. Inexperience with the device was the likely cause.

4. Officers used the device as an interactive map. They used room numbers, and hallways to correspond with their anticipated location on the device until they were able to update using the "Here I am" function.

5. Location updates were more frequent as the officers became more experienced with the device operation.

6. Command center operated as expected without any major issues. The command center operator was able to keep up the incident commander's requests and view the responder locations throughout the exercise.

Software Issues

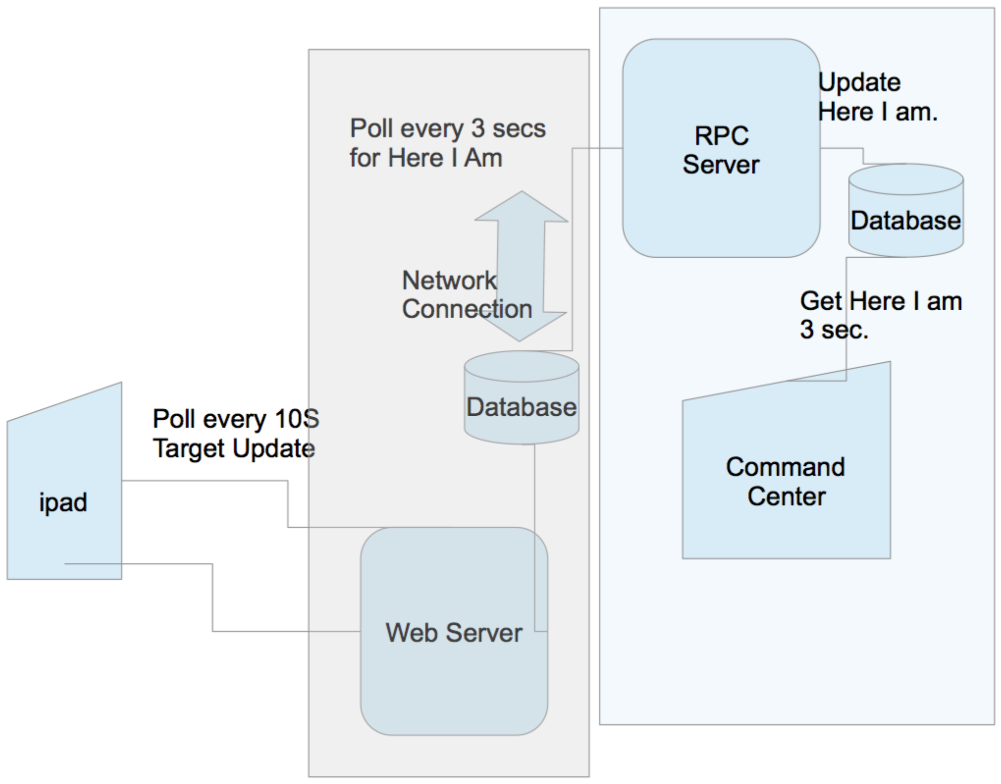


Figure 41: Revised client-Server architecture that implements non-persistent network connections

Beyond the system issues detailed above, the main issue from the software system involved aspects of connecting to the server(database) that related to the architecture of the system. Our implementation of the communication between the command center and the database required a persistant network connection, directly to the Postgresql port on the server. This was satisfactory as long as the connection was implemented through the ethernet network. When we moved our testing environment to the remote Mobile Communication Center(MCC), the secure shell pipe required for database communication became problematic. The network connection was established through the campus wireless LAN. Since the wireless connection was not robust the secure shell pipe dropped intermitantly and communications ceased. If the wireless connection dropped even momentarily the secure shell pipe had to be reset. The primary indications of this from an operations perspective was a frozen command center user interface.

Thus, we needed to establish a non-peristant database connection via the webserver similar to the mobile device connection. We created a local database on the command center system, creating the tables required for communication with the mobile devices. A Remote Procedure server(RPC) was implemented using a trigger mechanism in the command center database to communicate via HTTP to the server. This maintains the command center in sync with the database and therefore the remote devices without having to rewrite our command center code, as illustrated in Fig. 41.

This implementation prevented command center lockup and crashes and also established a robust wireless friendly connection to the command center for remote usability. This setup requires a timer for polling the database to interact with mobile devices similar to the way the mobile devices poll the web server. The implementation resulted in no negative effects on performance in our testing and training events.

There were a few other minor issues, such as more extensive logging (for studying effectiveness of the interface), syncing clocks across all the data gathering devices, that have already been addressed in the system.

5.3 Tabletop Exercise: Evacuation

The development of our evacuation application has included regular feedback and demonstrations with campus emergency and safety personnel, including the chief of police, other safety officers, and campus business continuity staff. This partnership has resulted in continuous improvement of the system based on the knowledge and experience of stakeholders in the area of public safety. This process also led to the scheduling of a table top exercise with these campus officers. We ran the application through three different scenarios to determine our system's usability, effectiveness, and to assess needs for improvements. A business continuity office staff member (an expert in running training exercises) designed the scenarios. The campus police chief, a senior police officer, and the software team participated in the exercise. Well-designed tabletop exercises are essential when dealing with policing and other experts who have little time or for whom physical exercises would be disruptive.

All three scenarios involved a cluster of four campus buildings and a base scenario for the evacuation of approximately 5000 evacuees; preprocessing was performed in accordance with the description in previous sections. The preprocessing step was timed at approximately 8 minutes. All simulations used this base evacuation object. Video of each of the 3 exercises were recorded for analysis, followed by feedback from the emergency personnel (excerpts in the submitted video). The system was operated by a member of the software team while commands were received from the police chief.

5.3.1 Scenario 1. Gas Leak in Building.

Figure 42 shows time sequenced snapshots of a simulated gas leak somewhere in the exercise area. Initially the gas leak was reported as "near Woodward Hall". The police chief requested a simulation start. As seen in Fig. 42(a) the buildings are being evacuated as expected with all exits being utilized. Several seconds into the simulation (sim time: 7:26:38) a report is received that the leak is in the "courtyard", as shown in the red ellipse in the figure. The simulation is halted and reset. The police chief instructed first responders to be dispatched to the building exits facing the courtyard. Also, entrance/exits into the courtyard were to be blocked from further use.

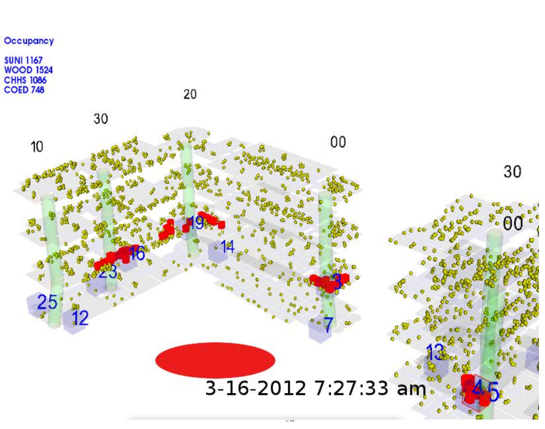
The simulation was restarted based on the new situation. We interacted with the software by placing blocks at the requested areas from 7:27:27 until 7:28:19 (Figs. 42(a), 42(b)). At this point, the software began to recalculate the 5000 evacuee paths. At 7:29:08 calculations were completed and the reporting process rebuilt, including the scenario timeline and the temporal congestion and exit utilization charts. The police chief requested to see the simulation based on the new situation.

Evacuees are confirmed to be exiting the buildings away from the hazard, as seen in Figs. 42(c), 42(d). The simulated time to exit all buildings increased from 318 seconds to 687 seconds. There were large evacuee populations in the areas of Woodward hall opposite the hazard and it was noted that due to the blockages in the second floor, some of the evacuees were trapped.

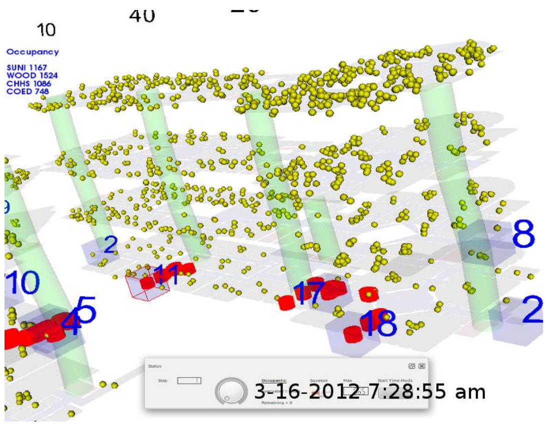
5.3.2 Scenario 2. Active Shooter in Building.

Figure 43 shows time sequenced snapshots of a simulated active shooter exercise in the Woodward hall. First, the police chief ordered a campus lock down and the building to be evacuated. At this point we switched from the base evacuation scenario of the lower quad (building cluster) to a base scenario of Woodward hall. The reason for this is that the scenarios are built as objects and run a priori. We could have placed blockages in the locked down buildings but chose to open a single building scenario for the purposes of the exercise.

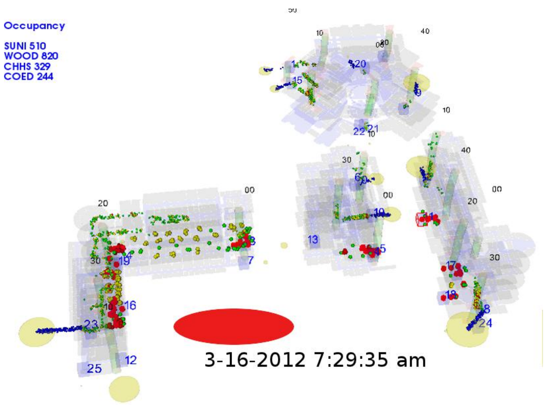
Some highlights in this exercise include: building rerouting and reporting occurs in 49 seconds (3 seconds for evacuee rerouting and 46 seconds for report generation). The total time here is similar to the multi-building evacuation because our base scenario included 3600 evacuees. This simulates a highly overloaded building to exercise the software for testing.



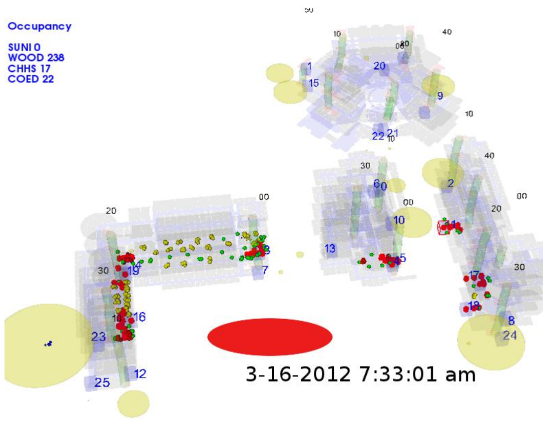
(a)



(b)

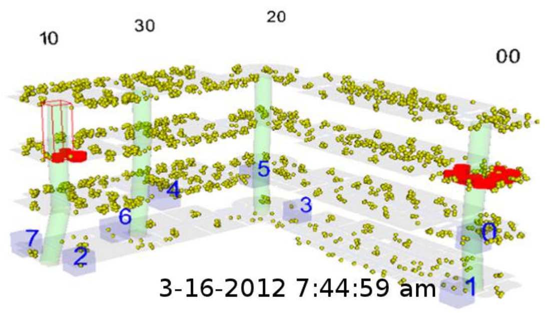


(c)

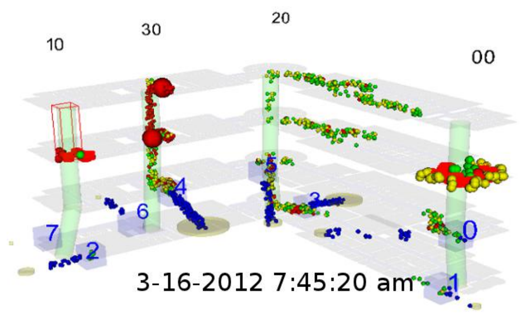


(d)

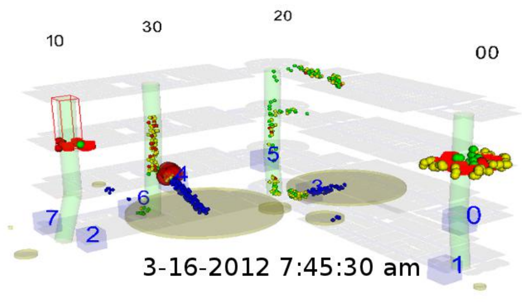
Figure 42: Tabletop Exercise: Gas Leak in Building. At approximately 7:20 a gas leak is reported. Lower quad campus possibly affecting four buildings. Evacuation simulation begins. At **Sim. Time: 7:26:38.**, leak confirmed near red ellipse. Simulation suspended, assets are deployed to prevent evacuation into the hazard. Views modified to simulate asset activities at building exits. (a)**Sim. Time; 7:27:33.** Blocking building exits into quad (b)**(5). Sim. Time; 7:28:55.** Blocking building exits from adjacent buildings into quad complete. Signal sent for application to perform situationally aware rerouting, (c) **Sim Time; 7:29:35.** Visualization of new simulation, evacuation in progress, simulating responders interaction at exits to affected areas, (d) **Sim. Time; 7:33:01.** Evacuees are avoiding hazard and and exiting to safe zones, evacuee densities are indicated by areas of yellow circles.



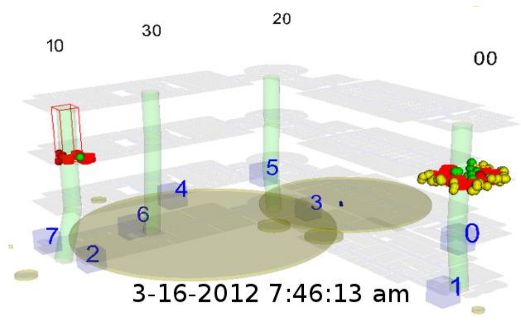
(a)



(b)



(c)



(d)

Figure 43: Tabletop Exercise:Active Shooter. At approximately 7:40 a shooter is reported at Woodward hall. Campus is locked down. Reports are received that 3rd floor stairwells are blocked at each end of the building. At 7:44:10 blockages are placed in Woodward by the operator and the simulation is recalculated. (a)**Sim. Time; 7:44:59.** Processing for new evacuation simulation is complete, commander orders visualization of new simulation, (b) **Sim. Time; 7:45:20.** Trapped evacuees noted at 3rd floor zone 00 and zone 10, (c)**Sim. Time; 7:45:30.** Extreme congestion noted at stairwells in floors 2, 3, and 4 at zone 30. (d)**Sim. Time; 7:46:13.** Simulated evacuation complete. Evacuee populations are indicated by approximate area occupied circles.

5.3.3 Scenario 3. Explosion in Utility Plant

An explosion in the RUP(regional utility plant) building created a scenario where the four building evacuation simulation of Figure 42 was also used. This scenario also found evacuees blocked in the upper floors and the explosion created a hazard in the building courtyard. As reports were received the building floors were blocked and the simulation was started. As more reports were received it became obvious that the personnel would exit toward the hazard in the courtyard. The exit density circles alerted the police chief to this problem and emergency personnel were dispatched to redirect these evacuees. At this point the police chief requested the exits facing the courtyard to be blocked. The simulation was restarted and evacuation times and exit results were evaluated as in previous scenarios.

At 8:04:00AM the explosion was reported. At 8:06:18AM we inserted blocks on three stairways on the fourth floor because of a report from Woodward hall. The simulation was started. As the simulation was running, responders were discussing logistics of fire and other emergency personnel deployment in the exercise. The simulation ran to the end. At 8:09:52 the police chief noted the large exit circles in an area near the explosion in the utility building. He requested we reset the simulation and block the exits toward the utility building. At 8:10:30 all second floor exits are blocked in the simulator. The simulation was reset and restarted at 8:11:02.

5.3.4 Analysis and System Assessment

We detail below both the observations from first responders as well as the important features and current limitations of our evacuation system, as noted from the table top exercise.

First Responder Observations. Overall, the feedback from the chief of police and his officers was positive and consisted of the following observations:

- The ability to see the layout of the buildings and surrounding areas and get a sense of the current situation was considered most valuable.

- The near real-time responsiveness of the system and the ability to see the evacuation under blockages gave the ability to get a quick assessment for taking appropriate action, dispatch first responders, etc.

- Immediate knowledge and visualization of the clearly marked exits that faced a hazard was only possible with the visualization. The number of exits and required assets to deploy was possible due to the accuracy of the building representation.

- In the gas leak exercise, a review of the evacuation helped the commander quickly size up the situation (number of evacuees, exit routes, etc) and order a building evacuation. Once the hazard was located, evacuees were routed away from it by injecting suitable blockages at key points in the building.

- A redistribution of the typical evacuee populations outside the buildings resulting from rerouting of personnel is an important issue for crowd or traffic control. The ability to view the simulation and the evacuee densities (red spheres in Figs. 42,43), is thus important for planning (dispatch officers to these areas, for instance) after the event.

- In the active shooter scenario, the police chief noted that the exit utilization and congestion reports would be an invaluable tool for first responders to analyze the condition of a building and dispatch personnel.

Evacuation System Effectiveness. Overall, the system performed well and was responsive. A number of desirable features and potential improvements will be needed as we move towards real-world exercise training scenarios, and actual deployment.

- Additional work on the user interface will be needed to ensure minimum delay during a dynamically changing situation; for instance, blockages are specified one at a time; a 'lasso' style interface to include multiple exits spanning a region around multiple buildings would be more intuitive and efficient.

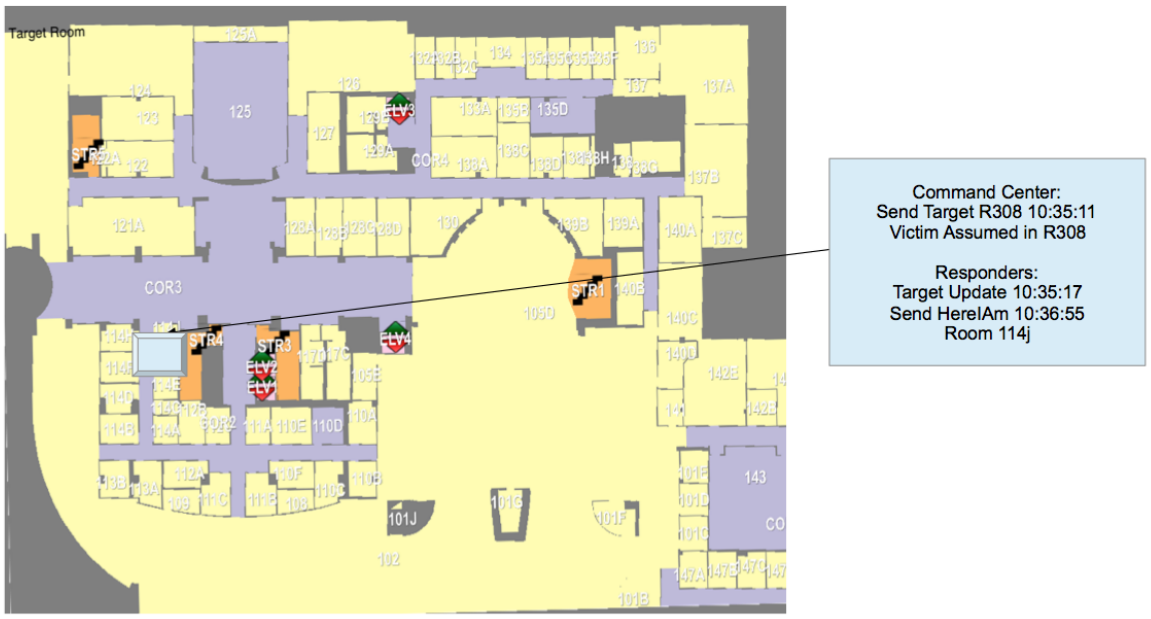
- A limitation of the current system is its inability to localize blockages to the exits, which can trap evacuees in the vicinity. Additionally, blockages need to be flexible enough to be removed when not needed; early reports might be unreliable, for instance, requiring exits to be reopened.

- The current system runs on a quad processor laptop at 2GHz. More optimal implementation and a faster system should further improve the performance with a better response time. It is to be noted that the system's near real-time response exploits the simplified building graph representations: this also makes it scalable to large collections of buildings.

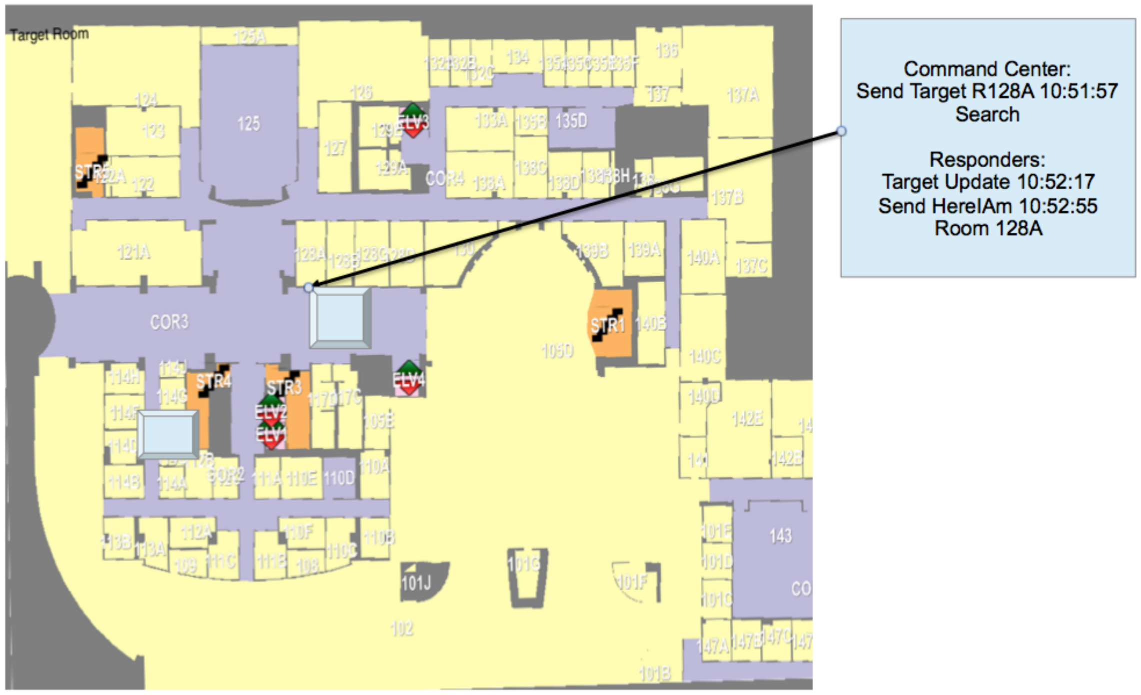
- A visualization issue that is common to visual analytic systems is visual clutter and the ability to unambiguously visualize critical information. As we extend our system to incorporate tens of buildings in evacuation scenario, these issues will require careful design and representation choices. Working with actual responders will be valuable in this design process.

5.4 Training Event 2: Highrise Building

The next major evaluation of our system was conducted on July 27, 2012. It was similar to the first training event, but was located in a 12 story building housing the Atkins campus library. The Atkins library is actually a combination of 4 older buildings with restricted paths (including some one-way paths) between these structures. There are large open areas, inaccessible areas (rare books collections, for example) and certain doorways that are normally closed to the public. All these features were reflected in the geographical maps and 3D routing graphs in our system. UNC Charlotte police were particularly interested in housing a training event in such a structure, as there were possible situations where officers might need to exit the building via windows from the top of the building using ropes (this scenario was not simulated in this event, but was mentioned among possible scenarios).

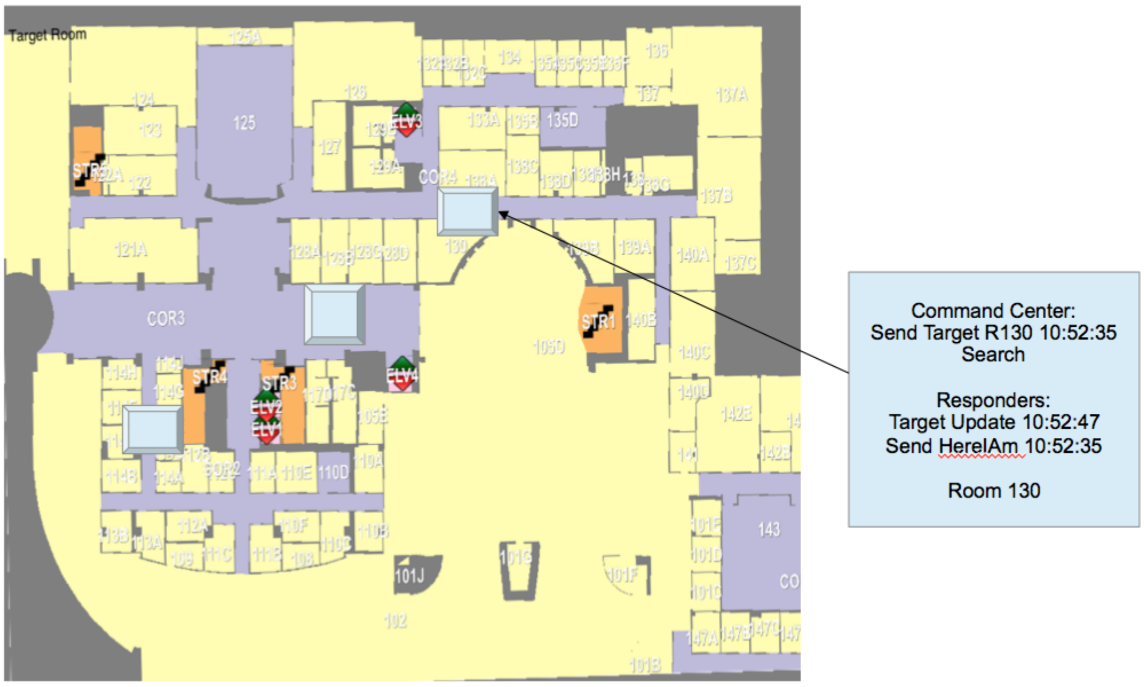


(a)

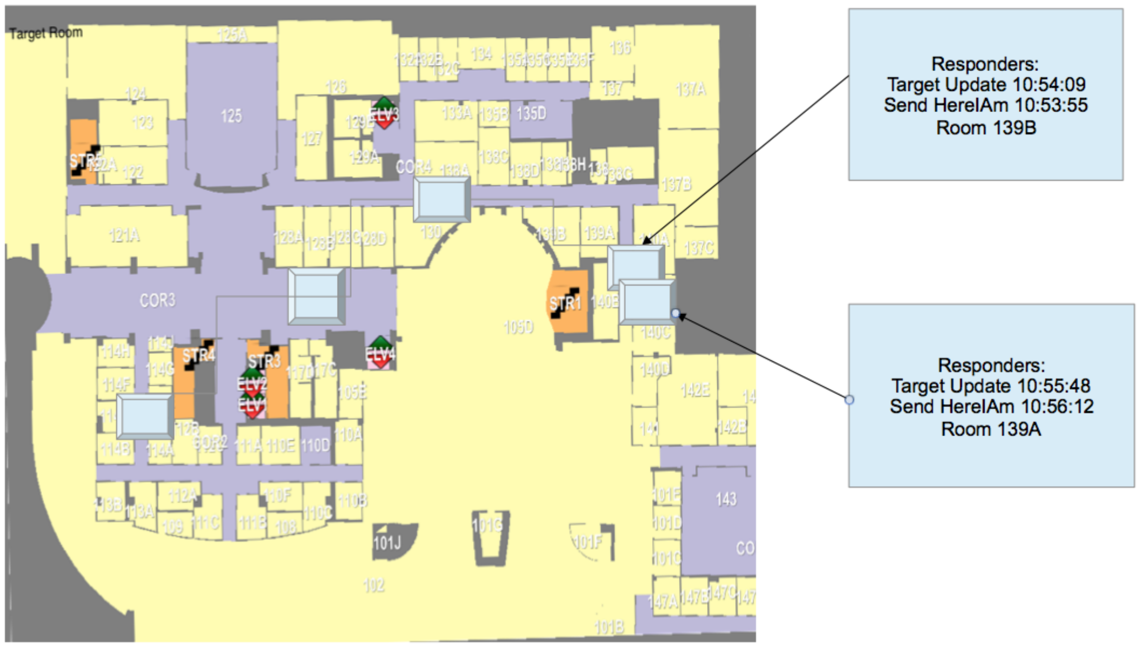


(b)

Figure 44: Training Event: Shooter in an Highrise Building - Sequence of Events.

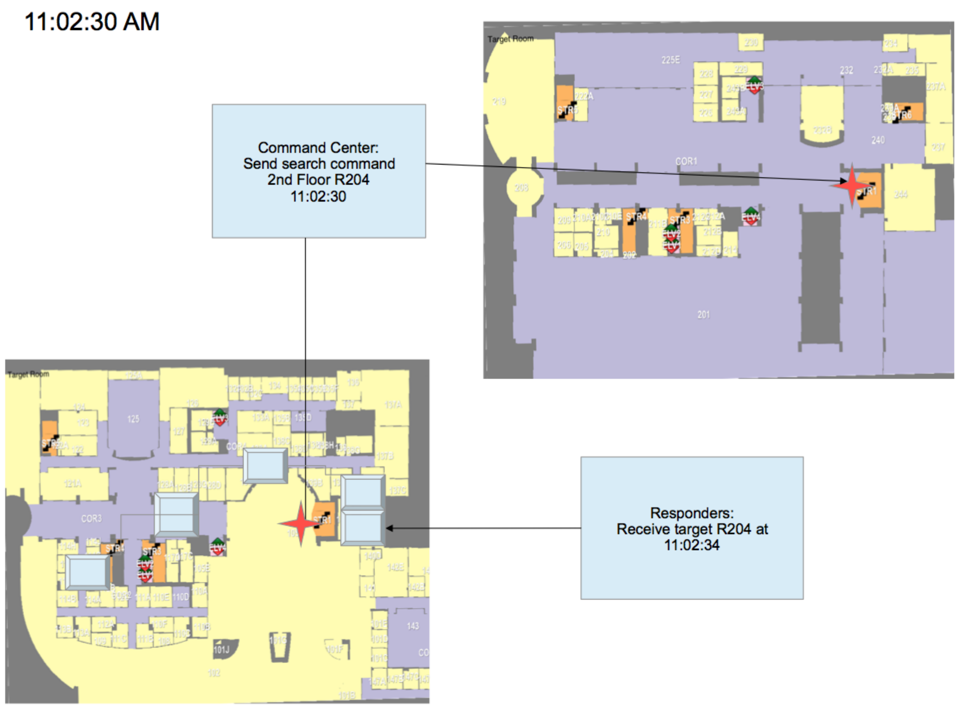


(a)

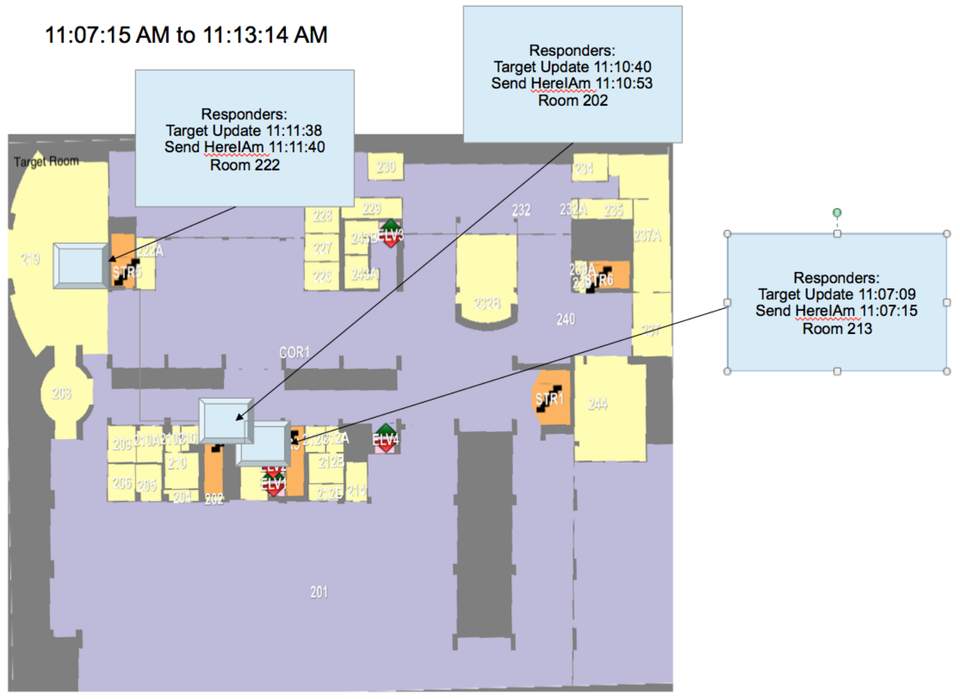


(b)

Figure 45: Training Event: Shooter in an Highrise Building - Sequence of Events.

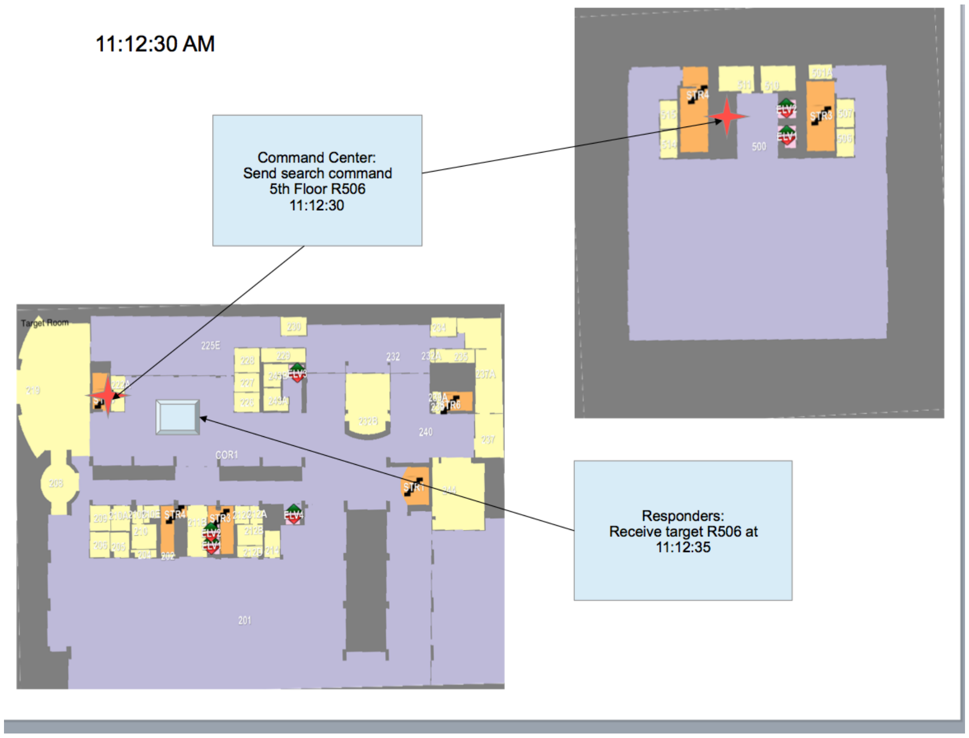


(a)

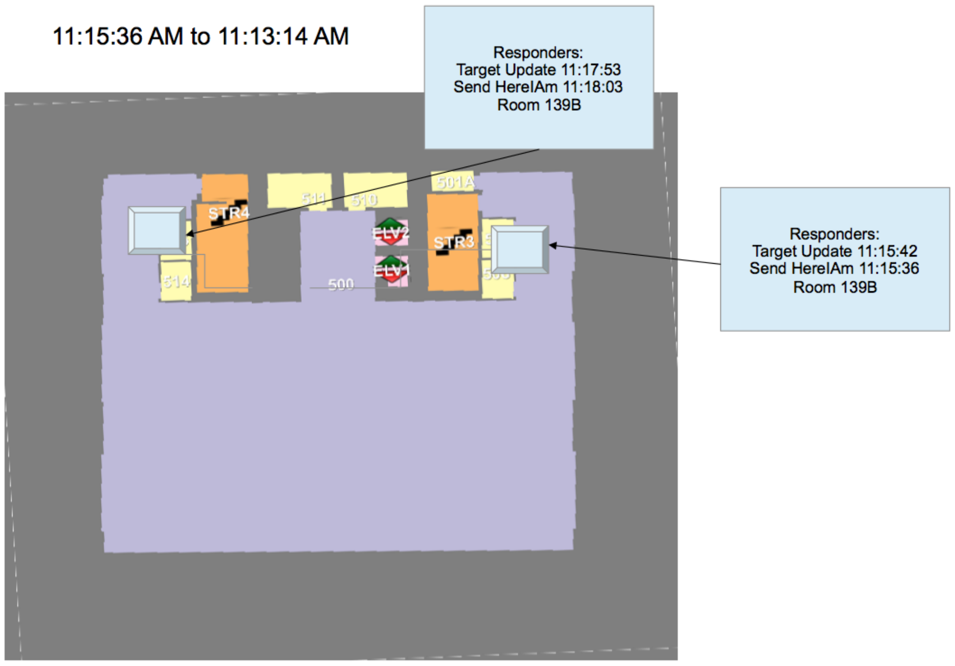


(b)

Figure 46: Training Event: Shooter in an Highrise Building - Sequence of Events.



(a)



(b)

Figure 47: Training Event: Shooter in an Highrise Building - Sequence of Events.

The event began at 10:35:11am with a victim reported in Rm 308(note that early reports are uncertain and maybe incorrect). Responders entered the building and acknowledged their location from Rm. 114 (using the “Here I am” function) at 10:35:17am. The command center ordered the responders to proceed to Rm. 128A and start a search; responders reached the location and acknowledged at 10:52:55am, as can be seen in the panels of Fig. 44.

The command center then ordered the responders to continue the search and sent them to Rm 130; responders acknowledged from this location at 10:52:35am. Additional updates were sent by responders from Romms 139A and 139B, as shown in Fig. 45

At 11:02:30am, command center sent a search command directing the responders to search the area around Rm. 204(Fig. 46(a)) and responders acknowledged the command. Fig. 46(b) illustrates the responders searching and reporting their location from 3 different points.

At 11:12:30am, command center orders the responder to search the fifth floor and with a target of Rm. 506 (Fig. 47(a). Responders acknowledge their position on the fifth floor at 11:18:03. Responders encounter the shooter in the vicinity and neutralize the shooter, ending the exercise.

5.4.1 Assessment

In contrast to our earlier training event, the feedback from the SWAT teams and police chief were mostly positive. The ‘Here I am’ function worked well throughout the training event, networking connections from the Mobile Communication Center had no issues. There was one instance when the SWAT team members mentioned being confused as to their current location, when they entered one of the upper floors during their search for the target; the orientation issue was related to the symmetric nature of the floor geometry. In the future, additional help or orientation using the mobile devices will be considered, for instance, use of the compass in newer versions of the mobile devices.

6 Conclusions

This project has resulted in EERC, an Effective Emergency Response and Communication system, that provides the basic infrastructure for emergency management and disaster mitigation. We have described in detail the three technical components of the system: the command center, the mobile application and the database server functions, that forms a bridge facilitating the communication between the incident commander and responder teams. We have described collaborative efforts with the UNC Charlotte police department personnel in running multiple training events with the system, as well as early experiments with the system at Pacific Northwest National Labs. The uniqueness of our design relies on the use of visualization and visual analytic tools, particularly 3D tools at the command center and a mobile application in the hands of responders; by direct visual communication through the network, unambiguous directives or information can be passed back and forth. More important, the ability for the incident commander to view the responder teams in a customized 3D view of the urban structure provides situational information that is invaluable, as per the feedback from initial evaluations of the system. The evacuation system also provides situationally aware capabilities to evacuate occupants of a building, that now includes dynamic changes to the building structure during an emergency. The real-time update of the evacuation provides valuable clues to congestion points and other foreseeable problems and allows for informed decision-making.

7 Dissemination of Research Findings

This project was divided into Phase I and Phase II. Phase I focused almost exclusively in developing and implementing the EERC system. Evaluations consisted mostly of ensuring the results were consistent with expected outcomes. Phase II involved development of the evacuation model and its real-time response capabilities to dynamically changing conditions. Phase II also saw the beginning of our collaboration with UNC Charlotte Police and completion of two training exercises. Two additional exercises are planned, one on July 27 and a more extensive campus wide exercise in the fall of 2012.

7.1 Presentations at Meetings

1. Invited speaker, "A Mobile System for Effective Emergency Response and Situation Awareness in Large Indoor Environments, *Precision Indoor Personnel Location and Tracking for Emergency Responders*, Worcester Polytechnic Institute, August, 2009.

2. Situation Aware Mobile Tools for First Response and Emergency Evacuation, Invited presentation, DHS University Programs Summit, March, 2010.

3. William Ribarsky, Invited speaker, Mobile Emergency Evacuation for Urban Area u Security, Workshop for Charlotte Fire Department and Regional Homeland Security Officials, Charlotte, April, 2010.

4. Kalpathi Subramanian, Emergency Response within Indoor Structures: Interactive Visual- ization Tools, Invited presentation, NIJ Conference, June 14-16, Arlington, VA. 2010.

5. W. Ribarsky, Invited participant, National Academy of Sciences workshop to set the research agenda for the National Geospatial Intelligence Agency (NGA), June, 2010.

6. W. Ribarsky, Invited speaker, Mobile Application for First Response and Emergency Evacu- ation in Urban Settings. ACM Com.Geo 2010 (Washington, DC, June 2010).

7. Kalpathi Subramanian, W. Ribarsky, Emergency Response within Urban Structures: Visual Analytic Tools, VACCINE Center (DHS) Meeting, Purdue University, West Lafayette, July 13, 2010.

7.2 EERC System Demonstrations

1. **Richard Harris, W. Lafayette Police**, July 13, 2010. VACCINE Center meeting; discussion and demonstration on deploying our EERC system with the Lafayette police department, procurement of a stadium dataset to test our current tools.

2. **Chancellor Phil Dubois, UNC Charlottte and his cabinet.** Demonstrated our emergency response system to the UNCC chancellors cabinet; lively discussion and strong interest in our system.

3. **Henry James, Associate Vice Chancellor, Risk Management Safety and Security (RMSS), UNC Charlotte.** Demonstrated our system to the RMSS group; favorable feedback and followups on working together on relevant topics of mutual benefit.

4. **Pacific Northwest National Laboratory(PNNL), July 2011.** Used our system as part of the "The First Annual Enterprise Resiliency Experiment(ERE) (http://www.theivac.org/co annual-enterprise-resiliency-experiment) at PNNL. Digitized 3 floors of their office building and successfully ran a navigation/search exercise with our system.

5. **UNC Charlotte Safety Committee, February 2012.** Demonstrations was met with quite positive response. In attendance was Mr. Brent Herron, Associate Vice President of Campus Safety and Emergency Operations for the University System (17 campuses).

6. **University System Police Chiefs Conference, April, 2012.** At the invitation of Brent Herron, we presented our 3D first responder routing and situationally-aware

evacuation capabilities to the University Systems Police Chiefs Conference. Over 50 people, including Police Chiefs from all 17 campuses and some Vice Presidents for Risk Management and Safety, were in attendance. The response and interest were quite positive. In fact, we have had preliminary discussions with Appalachian State University on an evacuation model for their football stadium. More broadly, we are working on a proposed system that would be prepared to ingest all the building information for other campuses in the system. If approved by the university system or the individual campuses, this activity would make UNC Charlotte a state leader in comprehensive campus safety and student flow systems. Such work is already underway for UNC Charlotte.

7.3 Publications

1. Jianfei Liu, Kyle Lyons, Kalpathi Subramanian, William Ribarsky, "Semi-Automated Processing and Routing Within Indoor Structures for Emergency Response Applications", *Proceedings of SPIE Defense Security+Sensing 2010, Visual Analytics for Homeland Defense and Security*

2. Saeed Osmann M., Ram B., Standfield P., Samanlioglu F., Davis L., Bhadury J., "Optimization Model for Distributed Routing for Disaster Area Logistics", *The Fifth IEEE International Conference on Service Operations, Logistics, and Informatics*, July 2009, Chicago, USA., pp 278-283. **Note: This is a related publication; Dr. Bala Ram and his doctoral student, Saeed Osmann spent a summer with us and collaborated with us on this project.**

3. Jackie Guest, Kalpathi Subramanian, William Ribarsky, "Visual Analysis of Situationally Aware Building Evacuations", SPIE Proceedings on Visualization and Data Analysis", Feb. 2013 (**Best Student Paper Award**).

4. Todd Eaglin, Kalpathi Subramanian, Jamie Payton, "3D Modeling by the Masses:

A Mobile App for Modeling Buildings", Pervasive Computing, (Percom) 2013 (Demo Track), March 18-22, 2013.

7.4 Theses

1. Jackie Guest, "Visual analysis of situationally aware building evacuations", Department of Computer Science, June, 2012. *Advisors: Kalpathi Subramanian, William Ribarsky.*

7.5 Videos

- We have made several short videos of our system and training exercises. These will be made available from our project website.

7.6 Website

- All material resulting from the project will be made available from the project website. This is currently under construction (temporarily materials can be found from *http://www.cs.uncc.edu/~krs/research.html*)

7.7 Community Partners and Collaborators

1. Jeff Baker, Chief of Police, UNC Charlotte Police Dept.
2. Officers Brian Thomas, Shawn Smith, UNC Charlotte Police.
3. Fred Brillante, UNC Charlotte Facilities Dept.
4. Josh Allen, Hank James, Rist Management, Safety and Security(RMSS), UNC Charlotte.
5. Michael Bess, Mike Mauldin, Steve Uedel, Research Planning and Analysis, Charlotte Mecklenberg Police.

6. Dr. Tim Collins, Director, VACCINE(DHS) Center, Purdue University.
7. Jeff Dulin, Director, Urban Area Security Initiative(UASI).
8. Dr. Bala Ram, Professor, UNC Greensboro and his student Saeed Osman, spent summer 2010 with us (sponsored by DHS summer programs).

References

[1] Postgis. Refractions Research, http://postgis.refractions.net.

[2] Postgresql. http://www.postgresql.org.

[3] G. Andrienko, N. Andrienko, and U. Bartling. Interactive visual interfaces for evacuation planning. In *Working Conference on Advanced Visual Interfaces(AVI) 2008 Proceedings*, pages 472–473. ACM Press, 2008.

[4] N. Andrienko, G. Andrienko, and P. Gatalsky. Towards exploratory visualization of spatio-temporal data. In *3rd AGILE Conference on Geo-graphic Information Science*, 2000.

[5] M. Berg, O. Cheong, M. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, third edition, 2008.

[6] I. Bitter, A. Kaufman, and M. Sato. Penalized-distance volumetri skeleton algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):195–206, 2001.

[7] I. Bitter, A. Kaufman, and M. Sato. Penalized-distance volumetric skeleton algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):195–206, 2001.

[8] I. Bitter, M. Sato, , M. Bender, K.McDonnel, A. Kaufman, and M. Wan. Ceaser: A smooth, accurate and robust centerline-extraction algorithm. In *Proceedings of IEEE Visualization 2000*, pages 45–52, Oct. 2000.

[9] I. Bitter, M. Sato, M. Bender, K. McDonnel, A. Kaufman, and M. Wan. Ceaser: A smooth, accurate and robust centerline-extraction algorithm. In *Proceedings of IEEE Visualization 2000*, pages 45–52, Oct. 2000.

[10] H. Blum. A Transformation for Extracting New Descriptors of Shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[11] J. Brandt and V. Alazi. Continuous skeleton computation by voronoi diagram. *CVGIP: Image Understanding*, 55:329–338, 1992.

[12] B. Campbell and C. Weaver. Rimsim response hospital evacuation: Improving situation awareness and insight through serious games play and analysis. *Journal of Information Systems for Crisis Response and Management*, 3(3):1–15, Jul-Sept 2011.

[13] C. J. E. Castle and A. T. Crooks. Principles and concepts of agent-based modelling for developing geospatial simulations. In *Centre for Advanced Spatial Analysis. University College London*, volume 110, pages 1–52, 2007.

[14] D. Chen, B.Li, Z. Liang, M. Wan, A. Kaufman, and M. Wax. A tree-branch searching multiresolution approach to skeleteonization for virtual endoscopy. In *Proceedings of the SPIE Medical Imaging*, volume 3979, pages 726–734, 2000.

[15] J. Chuang, C. Tsai, and M. Kuo. Skeletonization of three-dimensional object using generalized potential field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1241–1251, 2000.

[16] N. Cornea, D. Silver, and P. Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007.

[17] N. Cornea, D. Silver, X. Yuan, and R. Balasubramanian. Computing hierarchical curve-skeletons of 3d objects. *Visual Computer*, 21(11):945–955, 2005.

[18] DHS. The first annual enterprise resiliency experiment. *Integrated Visualization and Analytic Community(IVAC)*, July 2011. http://www.theivac.org/content/first-annual-enterprise-resiliency-experiment.

[19] R. D. D.T. Lee. Generalization of voronoi diagrams in the plane. *SIAM Journal on Computing*, 10(1):73–87, 1981.

[20] M. Egenhofer and J. Herring. A mathematical framework for the definition of topological relationships. In *Proceedings of the 4th International Symposium on SDH*, pages 803–813, 1990. Zurich, Switzeland.

[21] EVACNET4. http://www.ise.ufl.edu/kisko/files/evacnet/.

[22] G. Fang. Swarm interaction-based simulation of ocucupant evacuation. *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, 2008.

[23] N. Gagvani and D. Gagvani. Parameter controlled skeletonization of three dimensional objects. Technical Report CAIP-TR-216, Dept. of Electrical and Computer Engineering and CAIPCenter, Rutgers University, 1997.

[24] A. E. Gunes and J. Kovel. Using gis in emergency management operations. *Journal of Urban Planning and Development*, 126(3):136–149, Sept. 2000.

[25] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey. Clearpath: Highly parallel collision avoidance for multi-agent simulation. In E. Grinspun and J. Hodgins, editors, *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2009)*, 2009.

[26] Y. Ivanov, C. Wren, A. Sorokin, and I. Kaur. Visualizing the history of living spaces. *IEEE Transactions on Visualization and Computer Graphics*, 110:1153–1159, November 2007.

[27] N. Kamiyama, N. Katoh, and A. Takizawa. An efficient algorithm for the evacuation problem in a certain class of networks with uniform path-lengths. *Discrete Applied Mathematics*, pages 3665–3677, 2009.

[28] S. Kim, R. Maciejewski, K. Ostmo, E. Delp, T. Collins, and D. Ebert. Mobile analytics for emergency response and training. *Information Visualization*, 7(1):77–88, 2008.

[29] S. Kim, S. Shekhar, and M. Min. Contraflow transportation network reconfiguration for evacuation route planning. *IEEE Trans. on Knowl. and Data Eng.*, 20:1115–1129, August 2008.

[30] S. Kim, Y. Yang, A. Mellama, D. Ebert, and T. Collins. Visual analytics on mobile devices for emergency response. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 35–42, 2007.

[31] M.-P. Kwan and J. Lee. Emergency response after 9/11: the potential of real-time 3d gis for quick emergency response in micro-spatial environments. *Computers, Environment and Urban Systems*, 29(2):93–113, 2005.

[32] I. Lee and K. Lee. A generic triangle-based data structure of the complete set of higher order voronoi diagrams for emergency management. *Computers, Environment and Urban Systems*, 33(2):90–99, 2009.

[33] J. Lee. A spatial access-oriented implementation of a 3-d gis topological data model for urban entities. *GeoInformatica*, 8(3):237–264, 2004.

[34] J. Lee. A spatial access-oriented implementation of a 3-d gis topological data model for urban entities. *GeoInformatica*, 8(3):237–264, 2004.

[35] J. Lee and M.-P. Kwan. A combinatorial data model for representing topological relations among 3d geographical features in micro-spatial environments. *International Journal of Geographical Information Science*, 19(10):1039–1056, 2005.

[36] J. Lee and S. Zlatanova. A 3d data model and topological analyses for emergency response in urban areas. *Geospatial Information Technology for Emergency Response*, pages 143–168, 2008.

[37] J. Liu, K. Lyons, K. Subramanian, and W. Ribarsky. Semi-automated processing and routing within indoor structures for emergency response applications. In *Proceedings of*

the SPIE Cyber Security, Situation Management, and Impact Assessment II; and Visual Analytics for Homeland Defense and Security II, volume 7709, pages 77090Z–77090Z–10, 2010.

[38] J. Liu, K. Lyons, K. Subramanian, and W. Ribarsky. Semi-automated processing and routing within indoor structures for emergency response applications. In *Proceedings of SPIE Defense, Security+Sensing*, Apr. 2010. 5-9 April, Orlando, FL.

[39] S. M. Lo, M. Liu, and R. K. K. Yuen. An artificial neural-network based predictive model for pre-evacuation human response in domestic building fire. *Fire Technology*, pages 431–449, Sept. 2009.

[40] Q. Lu, B. George, and S. Shekhar. Capacity constrained routing algorithms for evacuation planning: A summary of results. *Springer-Verlag Berlin Heidelberg 2005*, pages 291–307, Sept. 2005.

[41] B. Meiguins and A. Meiguins. Multiple coordinated views supporting visual analytics. In *Proceedings of the ACM SIGKDD Workshop on Visual Analytics and Knowledge Discovery: Integrating Automated Analysis with Interactive Exploration*, pages 40–45, New York, NY, USA, 2009. ACM.

[42] R. Ogniewicz and M. Ilg. Voronoi skeletons: theory and applications. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 63–69, 1992.

[43] A. Requicha. Representation of rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, 12(4):437–464, December 1980.

[44] S. Shekhar and J. S. Yoo. Processing in-route nearest neighbor queries: a comparison of alternative approaches. In *Proceedings of the 11th ACM international symposium on Advances in geographic information systems*, 2003.

[45] B. Spitzak. The fast light toolkit. http://www.fltk.org.

[46] J. Stasko, C. Gorg, Z. Liu, and K. Singhal. Jigsaw: Supporting investigative analysis through interactive visualization. In *Proceedings of the 2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 131–138. IEEE Computer Society, 2007.

[47] J. Thomas and K. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Press, 2005.

[48] M. Wan, Z. Liang, I. Bitter, and A. Kaufman. Automatic centerline extraction for virtual colonoscopy. *IEEE Transactions on Medical Imaging*, 21(12):1450–1460, 2002.

[49] M. Wan, Z. Liang, I. Bitter, and A. Kaufman. Automatic centerline extraction for virtual colonoscopy. *IEEE Transactions on Medical Imaging*, 21(12):1450–1460, 2002.

[50] H. Wei, S. Zabuawala, H. Nguyen, S. Zeferjahn, and J. Yadegar. Towards a real-time 3-d situational awareness visualization for emergency response in urban environment. *Multimedia, International Symposium on*, 0:1–8, 2008.

[51] C.-K. Yap. An o (n log n) algorithm for the voronoi diagram of a set of simple curve segments. *Discrete Computational Geometry*, 2:365–393, 1987.

[52] A. Zerger and D. I. Smith. Impediments to using gis for real-time disaster decision support. *Computers, Environment and Urban Systems*, 27(2):123–141, 2003.

[53] Y. Zhou and A. W. Toga. Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):196–209, 1999.

[54] S. Zlatanaova, A. Rahman, and M. Pilouk. Trends in 3d gis development. *Journal of Geospatial Engineering*, 4(2):1–10, 2002.