Interacting with a Virtual Destroyed Environment Constructed from Real Disaster Data

Alexander Ferworn, Scott Herman, Christopher Kong, Alex Ufkes, Jimmy Tran
Department of Computer Science
Ryerson University
Toronto, Canada
{ aferworn, scott.herman, c5kong, aufkes, q2tran }@ ryerson.ca

***Abstract*— A Destroyed Environment (DE) is created by disastrous events in the built environment. DEs typically consist of the structures created from the rubble of collapsed buildings--forming a chaotic, unplanned and unmapped environment in which emergency first responders must find the surviving occupants who may now be trapped and hidden within. The more knowledge that search teams have concerning the resulting DE, the better they are equipped to plan and rescue survivors. We present the Disaster Scene Representation system that is able to employ scanned spatial data from a DE and creates a Simulated Environment (SE) that is spatially accurate, allows actions to be preplanned in the DE and is safe to work with. The system provides functionality that allows first responders to perform virtual structural inspections, allowing them the ability to plan and, to a certain extent, test actions in the simulated environment. The goal of this research is to demonstrate that the functionality within our SE can be used to preplan actions in the DE.**

Keywords— USAR; Rubble; Simulation; Physics; Disaster;

I. INTRODUCTION

Increasing urbanization results in what has come to be known as the built environment (BE)[1]. When disasters strike BEs they can become a destroyed environment (DE)—an environment where the buildings and other structures created to shelter humans suffer structural damage resulting in their collapse and providing the material that can kill, injure, trap and entomb them. There are numerous examples of people becoming trapped and entombed in and under rubble [1-4]. As such, this is a real risk to public safety in a DE.

In response to this risk most nations have developed organizations to respond to large-scale urban disasters requiring specialized skills and equipment. Urban Search and Rescue (USAR) teams or task forces (TF) are deployed to locate, medically stabilize and rescue trapped "patients."[2]

One of the most challenging aspects of finding patients trapped under rubble is the inherent danger associated with the DE. Search specialists and structural engineers are often unable to survey the affected area without putting themselves at risk. The instability or inaccessibility of the rubble can lead to injury and traversing the structures within the DE can trigger secondary collapses. In order for searching to begin, the areas of the DE that will be searched must be inspected and declared safe. This requires careful planning and additional shoring and bracing to allow access to the locations where patients may be found hidden in the debris.

Fig. 1 The T-Spot and Window/Door shores constructed within DSR. The output materials are shown to the right.

Increased understanding—or "situational awareness"—of the DE allows USAR leaders to better decide how to proceed with searching for and extricating patients.

Our goal is to increase the situational awareness of those responsible for USAR operations in a DE by providing a model of the DE using actual data extracted from it in a timely manner to create a virtual destroyed environment (VDE) model that is physically representative of the DE and maintains an accurate spatial relationship between all its components. Previously, we have used data collected from an Unmanned Aerial Vehicle (UAV) fitted with an RGB-D sensor to create a 3D point cloud model of scanned terrain [5].

As a continuation of this work we created a proof of concept system presented in [6]. The virtual system utilized a colourized point cloud within a game engine as a meshed model and allowed a user to interact with the model in a virtual environment. In this paper we present an extension of this work.

[1] Roof, K; Oleru N. (2008). "Public Health: Seattle and King County's Push for the Built Environment.". *J Environ Health* **71**: 24–27.

[2] In this work we use the term "patient" in the USAR context—meaning any person trapped in rubble, in need of rescue.

A. Contributions

The proposed system, which we call Disaster Scene Representation (DSR), is comprised of two parts.

The first part is the framework that converts point clouds into meshed models required by game engines—called the Model Creation Pipeline (MCP). In contrast to our previous work in [6], we present a better-defined, well-structured, and modular framework allowing individual components to be swapped out or modified as new techniques and algorithms become available to improve the performance of the system.

The second part of the DSR system is a set of virtual USAR tools designed based on input from USAR professionals, the Federal Emergency Management Agency (FEMA) [7] and Department of Homeland Security (DHS) [8]. Essentially, our virtual tools allow first responders to perform the same functions that would be ordinarily performed in the real-world DE. The system is discussed further in section IV.

II. Related Work

USAR operations normally begin by ensuring that a structure within the DE is safe for additional operations. Usually this involves inspecting the various structural components found in situ and providing additional structural support where necessary. Shoring is the process of temporarily supporting an area that is either damaged or partially collapsed in order to conduct operations at a reduced risk. A structural specialist inspects debris fields to determine which areas require support. In most cases, shores are used as a last resort if unstable areas cannot be removed or avoided altogether. Different shoring techniques using wood, hydraulic or pneumatic jacks or struts, and high-pressure air lifting bags exist, but wood lumber is by far the most commonly used material for shore construction, and is the focus of DSR.

If search areas are deemed too unsafe for humans, ground-robots can be used to circumvent hazards and access areas hard to reach. For example, 17 robots were deployed to search the wreckage of the World Trade Center collapse in New York [9]. However, a defining challenge for ground-robots is traversing the difficult terrain presented by the rubble of any collapsed structure.

Having mobility advantages over ground robots, UAVs have seen an increasing use in emergency situations due to their ability to fly over and survey terrain quickly. For example, The Royal Canadian Mountie Police (RCMP) located an injured person involved in a car crash in a remote forest using a UAV and heat-sensor [10]. In our previous work, we have shown that a UAV equipped with an RGB-D sensor can be used to collect disaster terrain data in a short time period and create a 3D point cloud model [5].

A point cloud data is a set of vertices with XYZ coordinates that can have associated colour attributes. Point sets can be derived using a variety of sensors including Laser Range Finders (LRFs), Infrared (IR) sensors, and inexpensive RGB-D sensors like the Microsoft Kinect [11]. Various applications exist to view or alter point cloud data such as Meshlab [12]. Unfortunately, point clouds lack the ability to interact with the data in a simulated physical manner as the points are simply recorded individually. Processing these point clouds allows developers to alter various aspects of the data, either by reducing the overall size, fixing anomalies such as stray points/redundant points, and constructing a 3D mesh model. Unlike point clouds, a 3D mesh model can be manipulated with physics-based rules applied to simulate real-world phenomena like gravity and force.

Physics engines are the backbone of any system providing realistic interactions. The engine handles the mathematical functions related to the physical interactions such as collision detection or motion dynamics. Reviews of available physics engines such as Open Dynamics Engine (ODE) or Newton have been conducted and each engine excels in different areas. The current industry standards are Nvidia PhysX [13] and Havok [14].

A computer simulation is defined as representing real-world systems and their internal processes within a computer program. Simulations provide many benefits such as testing many configurations without substantial costs, materials, and effect. Simulating urban disasters has been conducted both in the academic and private sectors to train users in scenarios that are either too expensive or too dangerous to replicate. USARSim [15] is a robotics simulation built within a game engine and used for training. The Hanshin-Awaji earthquake in Japan led to a simulation to better understand the implications of wooden houses collapsing [16]. In the private sector, ETC Solutions [17] provides an array of emergency training simulations. The limitation of all these simulations is that they cannot be used at a real-world DE using real-world data, but rather are used to learn about DEs in general and prepare for future occurrences.

Game engines are a development platform used to create virtual environments for interactive video games. Examples include the Unreal Development Kit (UDK) [18] and Unity [19]. Game engines provide support for physics (through physics engines), graphics rendering, audio, user interface, and networking. Applications built using a game engine can be rapidly prototyped to ease the development process.

III. Technical Approach

There are two aspects to this research. First, the point cloud data must be converted into a polygon mesh model. Point cloud data is not an accepted format for game engines. The added advantage of mesh models is that they form closed surfaces, which closely represents the environment. We call this conversion process the Model Creation Pipeline (MCP). This is discussed in Section IV part A.

The second part of this research is the creation of a series of virtual tools that allow first responders to interact with the disaster model in meaningful ways. These tools are developed using the Unity game engine and include functionality for measuring distances between surfaces, placing points of interest, adding custom illumination effects, and placing virtual shoring that accurately represents what responders would construct when on site. These tools are discussed in detail in Section IV part B.

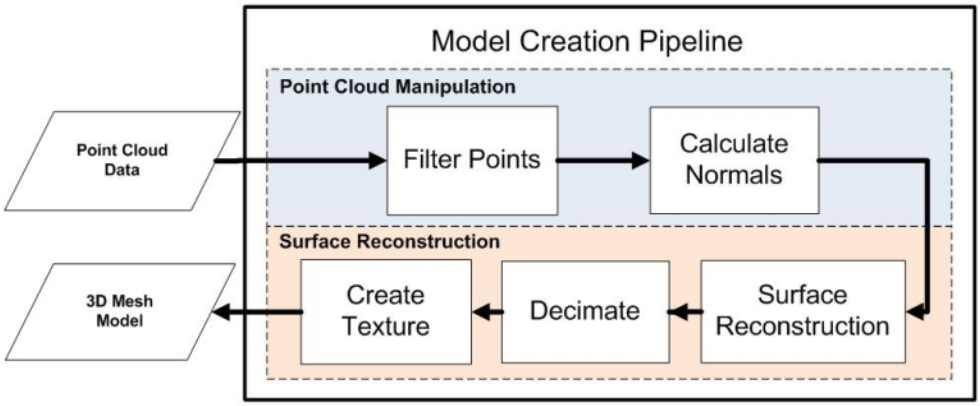

Fig. 2 The Model Creation Pipeline

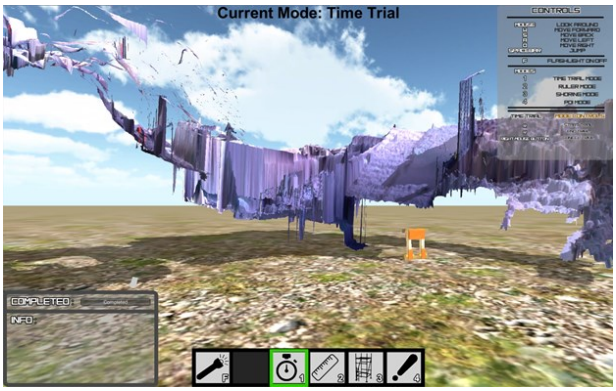

Fig. 3 A sample screenshot of the DSR system being used.

A. Model Creation Pipeline

The Model Creation Pipeline is a multi-step process to convert point cloud data into a 3D mesh model. It includes two phases, point cloud processing and surface reconstruction (Fig. 2). The algorithms used at each stage of the pipeline were chosen empirically through tests conducted on complex disaster data. The pipeline is modular, meaning that the algorithm used at any given step can be swapped out or modified as more effective techniques or implementations become available. Once the initial point cloud input has been provided, the steps are as follows:

1) Point Density Filtering

As the point clouds we collect are normally quite large, we employ Poisson-Disk Filtering [20] to spatially distribute the points equally and reduce the overall point count by 85-90%. For point clouds containing less than one million points, a filter distance of 10mm is used. For clouds larger than this, a distance of 15mm is used.

2) Point Normal Calculation

Next, vertex normals are computed. Normals determine how the final model is shaded and rendered in the game engine and are extremely important for a visually accurate user experience. We use functionality available in Meshlab to calculate the normals, using the viewpoint of the sensor as a constraint (i.e. the model will likely be viewed from the same direction as the sensor used to gather the point cloud data—from above). Once they are computed at each vertex, local smoothing is carried out on the resulting normals.

3) Surface Reconstruction

The points can now be meshed using surface reconstruction. Many techniques exist. Empirically we have found that using either Marching Cubes [21] or the Ball-Pivoting Algorithm [22] can be quite effective. Marching cubes is a voxel-based approach that can be used to create quick proxy models, but may result in a large number of polygons. Ball Pivoting is used when time permits to create 3D models with lower vertex counts and less surface noise (making the resulting simulation look more real).

4) Decimation

Polygon and vertex counts are further reduced through decimation [23]. Decimation removes polygons in a mesh through automated classification and deletion of vertices and approximating new surfaces from the resulting holes. The reduction is percentage-based and we found that 50% was sufficient to remove superfluous polygons without greatly affecting the accuracy of the data.

5) Texture Application

Finally, we add the colour component from the point cloud vertices to the 3D mesh model. For each vertex in the mesh, the colour data of the nearest vertex in the original point cloud data is found and interpolated into the mesh.

The resulting 3D mesh is stored as an *obj* file and can be imported directly into the Unity engine.

B. Virtual Toolset

The purpose of the virtual toolset is to allow a first responder who is on site or en-route to inspect and learn from the VDE in a USAR context. Thus, the functionality we implement in these virtual tools is based heavily on information found in the literature [7-8] and from discussions with USAR professionals.

The core set of functionality consists of several “modes” that can be triggered by a user to assist in inspecting and interacting with the virtual terrain as shown in Fig. 3. These modes are as follows:

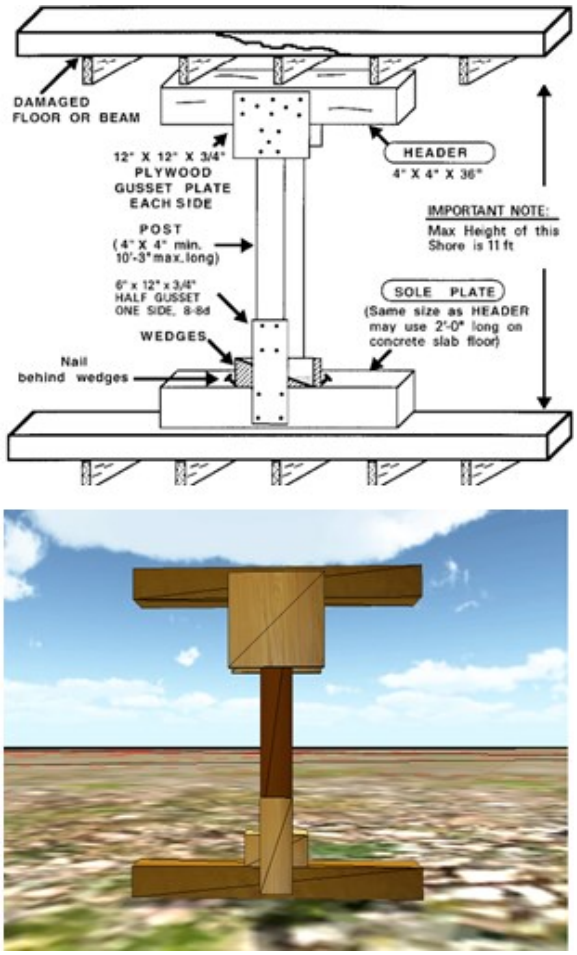

Fig. 4 The T-Spot schematic (top) and virtual shore (bottom).

Fig. 5 A partially collapsed building and bus at the OPP RRP in Bolton, Ontario.

Flashlight: This is a simple virtual flashlight that can be used to illuminate local areas within the VDE.

Time Trial: This tool can be used to measure traversal time across specific sections of rubble for actors within the simulation. In the current version we have implemented simulated human actors (making assumptions about height and weight). It is also possible to represent search dogs and robots as well.

Point of Interest: This allows users to place Point of Interest (PoI) markers in the VDE. These can be used to mark areas of instability, indicate voids or insertion points, etc.

Ruler: The ruler tool measures the metric distance between two user-defined points in the VDE. This is useful for tasks such as determining if a hole is large enough to potentially hide a trapped patient, or if a narrow opening will permit the passage of a robot.

Shoring: The purpose of shoring mode is to provide the ability to create virtual supports within the 3D VDE in order to determine if a shore will work and to estimate the needed quantities and the measurements of the lumber to construct these shores in the real DE. Multiple types of shores can be simulated. Three shoring configurations are currently implemented including T-Spot, Double T-Spot, and Window/Door for entry. An example virtual T-Spot shore and its schematic representation are shown in Fig. 4. Building a virtual shore consists of three steps: positioning, rotating, and growing. The user positions and rotates the shore through mouse clicks. During the "growing phase" the shore's height is increased automatically and halts when a section of the shore model intersects the terrain model. The final height of the shore is used to calculate the material lengths needed to build it. If a shore exceeds a maximum height, the shore model will be coloured red to indicate an ineligible shore—meaning the shore could not be used in the real DE.

IV. Experiments and Results

We validated the simulation framework by comparing movement and interaction with the models in simulation to reality. Any inaccuracies in the final model could lead to poor response planning by users of the system. This applies to simulated movement, measurements and model dimensions. We evaluate the accuracy of our simulation using data captured from two real-world environments.

Our first model was generated using data collected at the Ontario Provincial Police (OPP) Reference Rubble Pile (RRP) located in Bolton, Ontario. The RRP contains types of debris used for USAR training, including a partially collapsed building, several vehicles and shipping containers, and a passenger bus partially buried in rubble. Fig. 5 depicts a sample section of the RRP.

Our second model was generated using readily available materials found within the N-CART research lab at Ryerson University. Mock rubble was constructed similar to that which would require shoring supports for the safety of search teams working in an actual structural collapse. This micro-environment allowed us to gather data for the shoring experiments.

A. Modeling Tests

Raw data was collected data using a Microsoft Kinect RGB-D sensor, which served as input to a registration pipeline [5] to produce a point cloud model and used as input to the MCP.

We evaluated a variety of filtering resolutions to determine what was appropriate for noisy data based on our UAV rapid collection technique. Our goal was to determine which

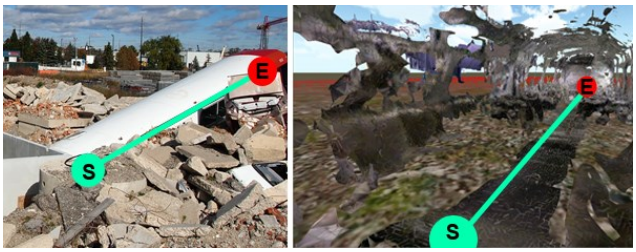

Fig. 6 The path taken during the passenger bus time trial. An exterior view of the bus located at the OPP RRP (left). The interior of the bus as viewed in the DSR simulation (right).

distance parameter is most effective at reducing the vertex count without oversimplifying the implied surface.

Three surface reconstruction techniques were applied to determine which approach was best at utilizing the points within the set to create a surface that preserves the geometric properties of the terrain.

1) Filtering

We applied the Poisson-Disk filter to the model with distance parameter values varying between 2.5mm-20mm. A filter distance of 10mm displayed an optimal reduction in point count of 85-90% for models under a million points. Similar results were achieved for models greater than one million points using a filter distance of 15mm. Distances greater than this threshold provided only minor improvements to point cloud reduction and smaller filter distances insignificantly reduced the point set. All parameters and thresholds were experimentally determined and used in our other experiments.

2) Surface Reconstruction

After reducing the total size of our point cloud model we applied a surface reconstruction algorithm to obtain a meshed model.

Marching Cubes produced a good model that was accurate and required the least amount of processing time, but led to the highest amount of surface polygons out of the tested methods. While a low polygon count model is ideal, Marching Cubes can be used to create quick proxy models.

The Ball Pivoting Algorithm performed the best out of the tested methods. This method required the most amount of processing time, but created a low polygon count model while holding the acuity of the terrain. We selected the Ball Pivoting Algorithm to create a low memory model that is efficiently rendered where time permitted.

B. Time Trials

In our previous work [6], we performed a time trial to validate that our simulation produced similar results to reality while traversing over the RRP. Here, we extend our experiments to include a second trial, recorded within the interior of a passenger bus. The bus interior was intact aside from broken windows and minor debris being present. In our experiment, an adult male traversed through the centre aisle of the bus which was positioned at a slope of approximately 45° as seen in Fig. 6. We recorded the distance of the path as 8.84m. The results of the trial are shown in Table I.

TABLE I. REAL TIME AND SIMULATED TIME TRIALS

Trail Number	Real Life Time (in seconds)	Simulated Time (in seconds)
1	28.5	24.36
2	32.8	29.68
3	29.9	37.09
4	28.6	23.54
5	32.6	27.51
Average:	30.48	28.44

The trials within the simulated environment closely matched the times measured in the real-world trials. We conclude from these results that accurate walking times can be estimated within a game simulation. Rescue personnel could traverse the rubble virtually and estimate times to cross the terrain accurately and safely in order to make estimates of how long particular tasks might take.

C. Shoring Calculator

Before a shore can be constructed, a TF member(s) is deployed into a DE to measure an area requiring shoring to determine the cut lengths of lumber required to construct the shore.

One of the goals in of our VDE is the virtual preparation of shoring measurements (cut tables) without endangering a human responder.

For the purpose of our experiments we had selected a T-Spot shore and Window/Door shore configuration for simulation and to build. These shores are utilized frequently in USAR operations and could be constructed with resources readily available to TF teams and to us.

To simulate the type of area that would require a T-Spot shore, three desks were stacked to represent an area requiring reinforcement to support weight from above.

The entry doorway to the N-CART lab was selected to simulate an area needing a Window/Door shore.

These areas were modeled using the Model Creation Pipeline and imported into the VDE. Using the shoring tool, virtual shores were created to support the simulated collapses and measurements were calculated to cut lumber and create a support for the real-world structures. Fig. 7 shows the final T-Spot shore obtained from this experiment.

Shore construction normally incorporates “shimming”—the use of wedges to ensure shores are closely fitted to the surfaces requiring support—as a technique to ensure close fits in an uncertain environment reliant on rough measurements and cuts.

We constructed actual shores based on the measurements we made in our VDE. The results of our experiments produced a T-Spot shore compliant with USAR guidelines with a tight fit between the ground and load without shimming.

The Window/Door shore required an extra two inches of shimming to ensure a tight fit between the posts and sole pieces. The posts were not cut to the proper length because the

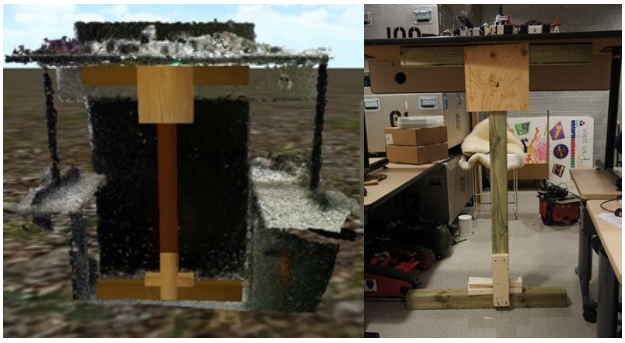

Fig. 7 Virtual shore created in DSR (left). Actual shore built from measurements supplied by DSR system (right).

virtual shore header intersected with a polygon from the model slightly lower than the top of the frame during the growing phase however the shimming process corrected for this error in the VDE when constructed in reality.

Our results from this experiment demonstrate that it is feasible to construct physical working shores from simulated data using our VDE.

V. Conclusion and Future Work

The motivation for our work has been to create a methodology and tool set for the capture, accurate reproduction and interaction with models of DEs. This was achieved using DSR.

It is our claim that employing our framework, it is possible to generate spatially accurate 3D models and employ them within a game-based simulation. The resulting VDE can be used to remotely inspect virtual versions of the DE and provide useful data to simulate real-world tasks. As a result, direct interaction with rubble can be delayed or mitigated entirely under some circumstances.

Future development will focus on adding functionality based on common tasks that would normally be performed as part of USAR operations.

Currently our system supports only a single user. Our vision of this system involves having concurrent user instances to allow information sharing and multiple viewpoints. We envision that DSR can be employed in many scenarios including those where an inbound Task Force may be many hours away from an incident. The disaster scene could be modeled by local personnel through DSR and the resulting model transmitted to the Task Force for planning purposes.

References

[1] Wikipedia. (2014, Jun. 28). *Algo Centre Mall* [Online]. Available: http://en.wikipedia.org/wiki/Algo_Centre_Mall

[2] D. J. Van Hoving, W.P. Smith, E.B. Kramer, S. De Vries, F. Docrat, L.A. Wallis, “Haiti: The South African Perspective.” *SAMJ: South African Medical Journal.* vol. 100, pp. 513-515, Jul. 2010.

[3] I. Takewaki, S. Murakami, K. Fujita, S. Yoshitomi, M. Tsuji, "The 2011 off the Pacific coast of Tohoku earthquake and response of high-rise buildings under long-period ground motions." *Soil Dynamics and Earthquake Engineering,* vol. 31, pp. 1511-1528, Nov. 2011.

[4] J. Yardley. (2013, Mar. 28). *Report on Deadly Factory Collapse in Bangladesh Finds Widespread Blame* [Online]. Available: http://www.nytimes.com/2013/05/23/world/asia/report-on-bangladesh-building-collapse-finds-widespread-blame.html

[5] A. Ferworn, J. Tran, A. Ufkes, A. D’Souza, "Initial experiments on 3D modeling of complex disaster environments using unmanned aerial vehicles," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR).*, pp. 167-171, 2011.

[6] A. Ferworn, S. Herman, J. Tran, A. Ufkes, R. McDonald, "Disaster Scene Reconstruction: Modeling and Simulating Urban Building Collapse Rubble within a Game Engine," in *Summer Computer Simulation Conference (SCSC).*, Article No. 18, 2013

[7] FEMA, "National Urban Search & Rescue (USAR) Response System: Rescue Field Operations Guide," *US Department of Homeland Security,* 2006.

[8] DHS, “Field Guide for Building Stabilization and Shoring Techniques”, Online: https://www.dhs.gov/xlibrary/assets/st/st-120108-final-shoring-guidebook.pdf

[9] J. Casper and R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center," in *IEEE Transactions on Systems, Man, and Cybernetics,* 33(3), pp. 367-385, 2003.

[10] C. Frazen. (2014, Jul. 5). *Canadian Mounties Claim First Person's Life Saved by a Police Drone*. [Online]. Available: http://www.theverge.com/2013/5/10/4318770/canada-draganflyer-drone-claims-first-life-saved-search-rescue

[11] Microsoft. (2014, Jul. 5). *Microsoft Kinect* [Online]. Available: http://www.microsoft.com/en-us/kinectforwindows/

[12] Meshlab. (2014, Jul. 5). *Meshlab* [Online]. Available: http://sourceforge.net/projects/meshlab/

[13] Nvidia. (2014, Jul. 5). *PhysX info* [Online]. Available: http://physxinfo.com/

[14] A. Bond, "Havok FX: GPU-accelerated physics for PC games," presented at the Game Developers Conference, San Jose., CA, 2006.

[15] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "USARSim: a robot simulator for research and education," in *IEEE International Conference on Robotics and Automation*, pp. 1400-1405, 2007.

[16] M. Onosato, S. Yamamoto, M. Kawajiri, and F. Tanaka, "Digital gareki archives: An approach to know more about collapsed houses for supporting search and rescue activities," in *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR).*, pp. 1-6, 2012.

[17] ETC Solutions. (2014, March. 25). *Advanced Disaster Management Simulator* [Online]. Available: http://www.trainingfordisastermanagement.com/

[18] Epic Games. (2014, Jul. 18). *Unreal Development Kit (UDK)* [Online]. Available: https://www.unrealengine.com/products/udk/

[19] Unity Technologies. (2014, Jul. 18). *Unity Game Engine-Official Site* [Online]. Available: http://unity3d. com

[20] R. L. Cook, "Stochastic sampling in computer graphics," in *ACM Transactions on Graphics (TOG),* vol. 5, pp. 51-72, 1986.

[21] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *ACM Siggraph Computer Graphics*, vol. 21, pp. 163-169, 1987.

[22] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," in *IEEE Transactions on Visualization and Computer Graphics,* vol. 5, pp. 349-359, 1999.

[23] L. Kobbelt, S. Campagna, and H.-P. Seidel, "A general framework for mesh decimation," in *Graphics Interface*, pp. 43-50, 1998.