Analyzing Large-Scale Crowd Simulations for Building Evacuation

Carlos García-Cabrera and Pedro Morillo and Juan M. Orduña

Abstract Animated virtual crowds have been used last years for analyzing human factors in scenarios where masses of people gather, such as sporting events, transportation centers, and concerts. A typical example is building evacuation in case of fire. Scalability still remains as an open issue for these multi-agent systems applications. In this paper, we use a scalable architecture to simulate a large-scale version of a virtual crowd in a building evacuation. From the social point of view, the results provided by the large-scale version of the crowd add new and crucial information about the agents behavior, emphasizing the need for a small amount of trained leaders in order to save lives. From the system point of view, the results show that the trend of avatars towards crowding in some areas highly increases the computation time for the agents hosted in some client computers. Therefore, this trend should be taken into account when designing large-scale evacuation simulations.

1 Introduction

The simulation of virtual crowds have been used last years for analyzing human factors in scenarios where masses of people gather, such as sporting events, transportation centers, buildings and concerts[13, 17, 4, 1, 10]. Usually, the motion of crowds and other flock-like groups is modeled as interacting particles that display different behaviors in 2D/3D virtual scenes [16, 3]. Beyond physically based simulations, agent-based crowd models aim to capture the nature of a crowd as a collection of individuals, each of which can have their own goals, knowledge and behaviors [14]. When virtual crowds are used for analyzing human factors in certain scenarios,

Carlos García-Cabrera, Pedro Morillo, and Juan M. Orduña
Departamento de Informática, Universidad de Valencia
Avda. Vicente Andrés Estellés, s/n
Burjassot (Valencia - SPAIN
e-mail: {pedro.morillo, juan.orduna}@uv.es

local motion driven by Helbing's model [5] should be used. The requirements due to 3D graphics, motion and behavioral models add a huge workload to the computer supporting the crowd simulation.

In order to decrease the computational workload of simulations, crowds have been divided in different levels. For example, the ViCrowd system [12] divides the simulation at the level of crowds, groups, and individuals. Modern variants of these simulations use continuum dynamics to reach interactive simulation speeds for thousands of characters [17]. Although these approaches can display very populated and interactive scenes, their usability for analyzing human factors is questionable, since the higher-level behaviors are not based on individual behavior.

Also, there have been efforts to provide efficient and autonomous behaviors to crowd simulations [15, 9, 7, 11]. However, they are based on a centralized system architecture, and they can only control tens of autonomous agents with different skills (pedestrians with navigation and/or social behaviors for urban/evacuation contexts). The scalability of the provided results still remains as an open issue, due to the complexity of the relationships among different agents.

In this paper, we propose the use of a previously proposed system architecture [8] for simulating a large-scale version of the virtual crowd that analyzes human factors in building evacuation [11]. The purpose is on the one hand to test if the results are similar when the crowd size increases. In this sense, the results show a small probability of survival of those agents not achieving to evacuate the building in a short term, emphasizing the need for a small amount of trained leaders. On the other hand, the purpose is to analyze the performance of the underlying computer system, in order to determine the system requirements for large-scale simulations. In this sense, the results show that the trend of avatars towards crowding in some areas highly increases the computation time for the agents hosted in some client computers. As a result, the synchronous scheme followed in the simulation of building evacuation [11] is not suitable for large-scale simulations.

The rest of the paper is organized as follows: Section 2 briefly describes the proposed simulation architecture and the implementation of the crowd simulation. Section 3 shows the the crowd behavior shown in the building evacuation, and also the performance evaluation of the simulation system. Finally, Section 4 shows some conclusions and future work to be done.

2 Architecture and Implementation of Building Evacuation

In order to implement a large-scale version of the behavioral model for building evacuations [11], different modifications have been made to the distributed architecture for crowd simulations [8]. This architecture controls how the existing agents can share information about the 3D virtual scene. We briefly describe this architecture here, in order to make this paper self-contained. It consists of a distributed computer system for supporting the software architecture shown in Figure 1. This software architecture is composed by two elements: the action server (AS) and the client processes (CP). The AS is devoted to execute the crowd actions, and it is hosted on a single computer, while each CP handles a subset of the existing agents and it is hosted on a single computer. Agents are implemented as threads of the CP for reducing the communication cost. Each thread manages the perception of the environment and the reasoning about the next action. Since reasoning formalisms can involve a high computational cost, each client process is hosted on a different computer, in such a way that the system can have a different number of client processes, depending on the number of agents in the system. In this way, this organization takes advantage of the underlying distributed hardware. This scheme allows the correct simulation of tens of thousands of autonomous agents at interactive rates.

The first modification consists of decoupling the different system functionalities in different layers. Concretely, we have implemented the low-level communication protocols, the mechanisms of inter-agent communications, and the functionality of generating new behaviors as different and independent software layers. The first layer, consisting of the low-level communication protocols, is based on the same model (based on BSD sockets) shown in the previous architecture for crowd simulations [8]. However, we have added a new kind of agents with new communication mechanisms, in order to extend the functionality of generating new behaviors.

In the previous architecture, the computing of the agent movements was based on a grid obtained from the logical division of the virtual scene in a regular grid. For each grid cell, an A^* algorithm was computed in order to provide that cell with a pointer indicating which adjacent cell was the closest one (the most logical one) to the system exit (the virtual scene was a building and the purpose of the simulation was the building evacuation). In the modified architecture, the cells exclusively contain information about which room they belong to, and agents only use the grid for knowing in which room they are currently located. With this information, they compute their own path to the exit. In this way, the system can dynamically react to both the creation and removal of exits during the evacuation. Additionally, each

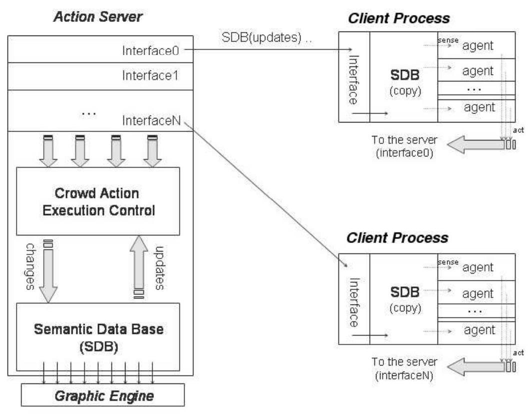

Fig. 1 Software architecture for large-scale building evacuation

agent has a small memory that allows it to save which rooms have been already visited. These changes allow agents to make movements that obey a pre-defined behavior, depending on the kind of agents (trained leaders, non-trained leaders, normal agents, etc.). Once the path to be followed is computed, a force system based on the Helbing's model is used for moving.

Another modification of the previous architecture consists of the kind of information exchanged. In the previous architecture, each agent (a thread of a CP hosted in a given computer) exclusively sent to the server the new location (the point of the virtual world where it wanted to move). In the modified architecture, each agent not only sends its new desired location, but it also sends messages to other agents. These messages can be either destined to another agents (asking them to perform some actions) or messages describing the visited rooms.

Additionally, in order to provide this system with the same working scheme used for building evacuations [11], normal agents should send their requests to the server, completing the following cycle in each iteration: first, the agent should compute the path to follow, the Helbing forces, and the graphic updates for the graphical interface. If these computations are performed in less than 250 milliseconds (the agent cycle), then the agent (the thread) should sleep until this time has expired. At that moment, it wakes up and sends a request to the server for checking the computed movement. The server computes the answer in real-time, and it sends back this answer to the agent as soon as it is ready. When the answer arrives to the agent, then the cycle starts again. It should be noticed that, unlike the previous architecture for crowd simulations [8], there is no a server cycle (it computes the answer to each request as soon as it arrives), but an agent cycle. The reason for establishing an agent cycle is to provide all the agents with the same speed of movement. The reason for choosing 250 milliseconds as the cycle period is that it is the longer response time that human users perceive as interactive in DVE systems [6]. This period is the same used for building evacuations [11]. For illustration purposes, Figure 2 shows the flow diagram for the cycle period of a normal agent.

As figure 2 shows, the first action to be performed by the agent is to check if it has arrived to its destination (the exit). If not, then the next step is to check if it follows the correct path. Each agent knows the path to the exit where it entered the

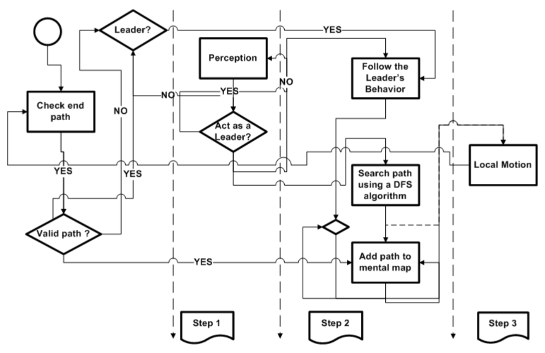

Fig. 2 Cycle period for normal agents

building. If that path is not blocked (valid path), then agents simply follow that path. In case the path is not valid, the agent checks if there is a leader nearby, in order to follow him. If there is not a leader, then the agents in that area vote for a leader (after all, in real world someone ends acting as a leader in these situations). When the leader has been elected, the agent that becomes the leader searches the path using its own mental map. Concretely, a deep-first search algorithm is computed, and the new rooms visited are added to its mental map. Additionally, it sends messages to the other agents around with its next location, for guiding purposes. Finally, all the agents use the Helbing's model to compute the local motion.

The cycle followed by the non-trained leaders is illustrated by Figure 3, and it consists of the following steps: first the agent checks if it has arrived to its destination (the exit). If not, then (perception) each agent sends messages to all the agents surrounding him, indicating the visited rooms. Additionally, if there are no other leaders, he will send messages to other leaders indicating them to follow him. After that, the agent will receive all the messages from the surrounding agents and will update his map. The rest of the steps are the same than the ones in the normal agent.

The goal for of all the agents is to leave the virtual scene. Regarding the agent functionality, we have implemented the same kind of agents shown in the behavioral model: normal agents, untrained leaders and trained leaders [11]. When the normal agents and untrained leaders see a hazard, they explore the maze (following a depth-first scheme in order to avoid cycles). The trained leaders know the correct path to the exits. Finally, the untrained leaders can exchange information with other agents about the state of the maze. Whenever two or more agents meet in a room, they share two pieces of information: locations of some of the hazards that are blocking paths, and parts of the building that have been fully explored by other agents and found to have no accessible exit (passed along by previous communications). The communication is local to a room, so agents exchange only relevant information about neighboring rooms, as for example, do not go through that door (there is fire), do not go in that direction (there is no exit), or follow me.

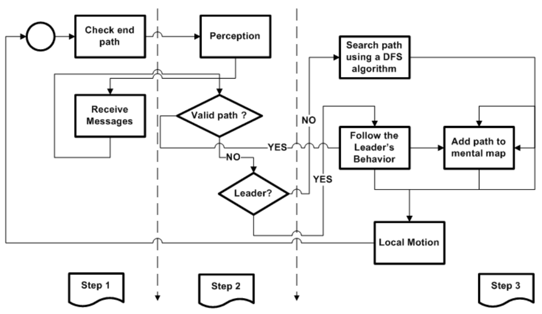

Fig. 3 Cycle period for non-trained leaders

3 Performance Evaluation

In this section, we evaluate a large-scale version of the behavioral model for building evacuation [11]. Instead of a population ranging from 20 to 200 agents, we use a minimum of 400 agents and a maximum of 800 agents. However, due to space limitations we only show some representative results, corresponding to a population size of 400 agents. The results for other sizes were very similar. The computer system used for evaluation purposes consists of eight interconnected personal computers. All of them were composed of a Dual-Core Intel Core 2 Duo processor at 2666 MHz with 1024 MB of DDR2 RAM, NVIDIA GeForce 8500 GT (512 MB) graphic card, and Microsoft Windows XP Professional operating system (with service pack 2).

The building evacuation consists of a structured square virtual world (a kind of maze) with a uniform initial distributions of the avatars (agents). That is, the ratio number of agents/room is constant. We have considered three different scenarios (maps). The first map contains two exits (in two opposite sides of the square) and two hazards that are quite close to the existing exits. The second map contains a random distribution of hazards within the maze. Finally, the third map is identical to the first map, but changing the exits to the opposite sides of the square with respect to the first map. Also, the two hazards are located close to the exits.

The first part of the performance evaluation has been the behavioral model. Figure 4 shows the performance (in terms of number of evacuated agents) of the different options for communication strategies and for a population of 400 agents. This figure shows that when communication among agents exists, then the performance is improved. Thus, when ninety simulations steps have been performed, around 360 agents (90% of the population) have been already evacuated if communication among agents exists, and only 270 agents (67.5% of the population) have been evacuated if no communications are exchanged. However, the rest of the population takes a similar number of steps to be completely evacuated, regardless of both the communication strategy and the size of the remaining population.

These results differ from the ones shown for 200 agents [11], where the simulation with communication finished (the population evacuated the world) in about half of the time that it took the non-communication case to finish. Similar results were obtained for greater populations and different maps. These results indicate that for

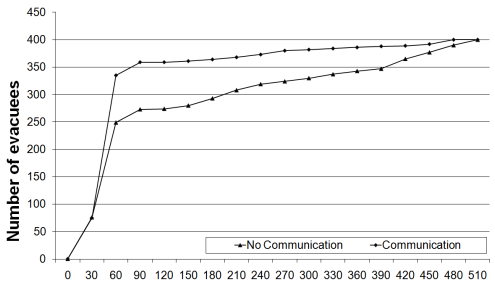

Fig. 4 Performance for different communication strategies

large-scale evacuations, the communication strategy determines the portion of the population that can be evacuated on a short-term period, but it has not a noticeable effect on the long term performance. The reason is that the remaining population after the first simulation steps cannot exploit the knowledge provided by the communications. That is, the remaining agents are normal agents that do not communicate each other. The leaders manage to evacuate the building in the first steps of the simulation, due to their ability to share information.

Figure 5 shows the performance (in terms of the number of simulation steps in order to evacuate the building) for different simulations with different percentages of trained leaders. This figure shows that the only simulation where the population is fully evacuated in 120 simulation steps is the one with a 100% of trained leaders. The rest of simulations do not finish the evacuation after 510 simulation steps. In this sense, the performance is similar to the one shown in the previous figure. The performance provided by the different options considered greatly differs in the short term, but they tend to converge in the long term. Thus, for 120 simulation steps the biggest differences are shown between 0% and 25 % of trained leaders, with 270 and 350 agents evacuated, respectively.

If we compare these results with the ones provided by the behavioral model for building evacuation [11], we can see that they are proportional. However, the large-scale results (Figure 5) show that those agents that could not evacuate the building in a short term require a long time to reach their goal, with a low probability of survival. These results emphasize the need for a small percentage of trained leaders in building evacuations in order to save lives.

Additionally, we have analyzed the system performance, in order to characterize the system requirements for large-scale simulations. Since the underlying platform is a distributed computer system, the most important performance measures are latency and throughput [2]. Figures 6 a) and b) show the system performance (in terms of average system responses and computation times, respectively) measured in the simulation of building evacuation for 400 agents. This figures contain three plots for the different maps considered. They show on the X-axis the simulation step, and they show on the Y-axis the average system response time or average computation time, respectively. The former one includes from the starting of the agent cycle to the instant when the server response arrives. The latter one measures the average

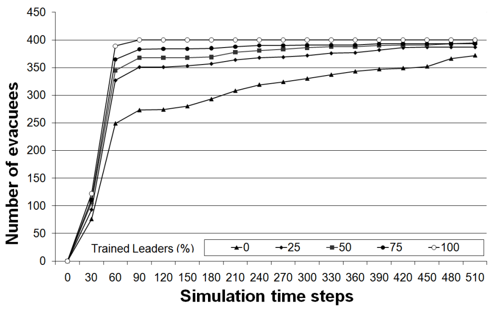

Fig. 5 Evacuation time for different numbers of trained leaders

computation time that agents require to complete the first part of the agent cycle (computing the path to follow, the Helbing forces, and the graphic updates for the graphical interface). Both Figures show the average values for one of the computers hosting either normal agents or non-trained leaders.

Figure 6 a) shows that the plots for the three maps show similar shapes and values. They range from 252 milliseconds to 260 milliseconds, and they alternate periods of peak values with periods of low latencies. The reason for these behaviors is the simultaneous waking-up of the agents (threads) in the same computer when the agent cycle finishes. Figure 6 a) also shows different simulation length for the different maps considered. While the simulation steps required for map 1 and map 2 are similar (both of them require around 350 simulation steps), map 3 requires around 650 simulation steps. It is due to the fact that there is a longer distance to the exits in map 3.

On other hand, figure 6 b) shows that, regardless of the considered map, the average computation time required for the first part of the agent cycle has a parabolic shape, continuously increasing from the start to the end of the simulation. Thus, the average computation time reaches the highest values for the map whose simulation lasts more time (map 3). The reason for this behavior is that the hazards are located close to the two existing exits in all the maps. Thus, agents tend to crowd in the rooms near the exits along time. As a result, the time required for computing the Helbing's model (where the forces produced by the surrounding agents should be taken into account) increases as so does the agent density. Although the computation times shown in Figure 6 a) are still far from the agent cycle, if the simulation lasts more time then computation times could reach the agent cycle, providing an unacceptable system performance.

Additionally, we have measured the same performance metrics in the computers hosting trained leaders, in order to analyze the system behavior for all the kinds of agents. Trained leaders are different from the rest of the agents, since they should guide them to the exits. Therefore, their mental map contain all the exits, and they should also broadcast messages to all the agents in the same room in order to guide them to the next room. Figure 7 shows the average computation time for all for the trained leaders hosted on the same computer. This figure shows a completely differ-

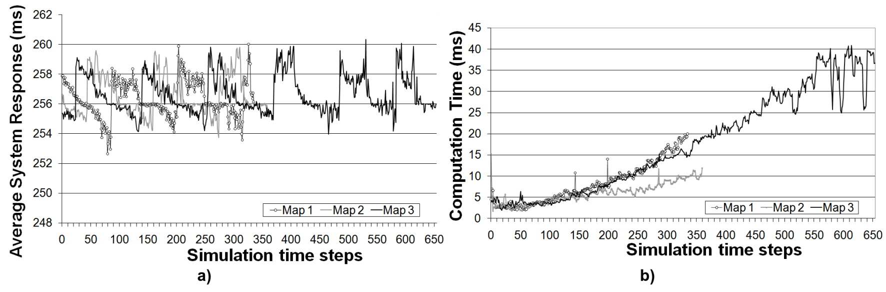

Fig. 6 a) average latencies and b) computation times for normal agents

ent behavior for all the considered maps with respect to the ones shown in Figure 6 b). In this case, for all the considered maps the computation time linearly decreases as the simulation proceeds. The maximum values reached in this case slightly exceeds 5 milliseconds, far away from the 40 milliseconds shown in Figure 6 b). These results show that the computers hosting trained leaders do not tend to saturation in large scale simulations.

4 Conclusions

In this paper, we have used a previously proposed system architecture to simulate a large-scale version of a virtual crowd that analyzes human factors in building evacuation. From the social pont of view, the results show that those agents that cannot be evacuated in a short term need a very long time to be evacuated (thus with a low probability of survival), regardless of the percentage of trained leaders. These results emphasize the need for a small amount of trained leaders for managing building evacuation.

Additionally, we have analyzed the system performance, in order to establish the system requirements for large-scale simulations. The results show that due to the trend of avatars to crowd in some areas, the computing time highly increases for some client computers as so does the population size. Since this trend is not bounded and the cycle period is constant, the computation times can exceed the cycle period for large-scale population sizes. The reason for this behavior is the saturation of the client computers. This trend should be taken into account when designing larger-scale evacuation simulations, and less normal agents or non-trained leaders should be hosted in each computer, in order to avoid system saturation and performance degradation.

Acknowledgements This work has been jointly supported by the Spanish MEC and the European Commission FEDER funds under grants Consolider-Ingenio 2010 CSD2006-00046 and TIN2009-14475-C04-04.

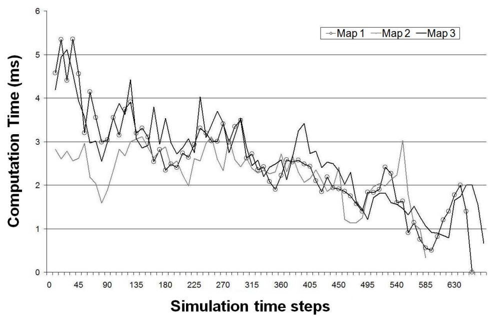

Fig. 7 Average computation times for trained leaders.

References

1. Exodus, the evacuation model for the safety industry. homepage, http://fseg.gre.ac.uk/exodus/
2. Duato, J., Yalamanchili, S., Ni, L.: Interconnection Networks: An Engineering Approach. IEEE Computer Society Press (1997)
3. Frank, T., Bernert, K., Pachler, K.: Dynamic load balancing for lagrangian particle tracking algorithms on mimd cluster computers. In: PARCO - International Conference on Parallel Computing (2001)
4. Galea, E.R., Sharp, G., Lawrence, P.J.: Investigating the representation of merging behavior at the floor–stair interface in computer simulations of multi-floor building evacuations. Journal of Fire Protection Engineering **18**, 291–316 (2008)
5. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic. Nature **407**(6803), 487–490 (2000). URL http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0009448
6. Henderson, T., Bhatti, S.: Networked games: a qos-sensitive application for qos-insensitive users? In: Proceedings of the ACM SIGCOMM 2003, pp. 141–147. ACM Press / ACM SIGCOMM (2003)
7. Iglesias, A., Luengo, F.: New goal selection scheme for behavioral animation of intelligent virtual agents. IEICE Transactions on Information and Systems **E88-D**(5), 865–871 (2005)
8. Lozano, M., Morillo, P., Orduña, J.M., Cavero, V., Vigueras, G.: A new system architecture for crowd simulation. J. Netw. Comput. Appl. **32**(2), 474–482 (2009). DOI http://dx.doi.org/10.1016/j.jnca.2008.02.011
9. Nakanishi, H., Ishida, T.: Freewalk/q: social interaction platform in virtual space. In: VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology, pp. 97–104. ACM Press, New York, NY, USA (2004). DOI http://doi.acm.org/10.1145/1077534.1077552
10. Pelechano, N., Allbeck, J.M., Badler, N.I.: Controlling individual agents in high-density crowd simulation. In: SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 99–108 (2007)
11. Pelechano, N., Badler, N.I.: Modeling crowd and trained leader behavior during building evacuation. IEEE Comput. Graph. Appl. **26**(6), 80–86 (2006). DOI http://dx.doi.org/10.1109/MCG.2006.133
12. Raupp, S., Thalmann, D.: Hierarchical model for real time simulation of virtual human crowds. IEEE Transactions oon Visualization and Computer Graphics **7**(2), 152–164 (2001)
13. Reynolds, C.: Big fast crowds on ps3. In: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames, pp. 113–121 (2006). DOI http://doi.acm.org/10.1145/1183316.1183333
14. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques, pp. 25–34. ACM, New York, NY, USA (1987). DOI http://doi.acm.org/10.1145/37401.37406
15. Shao, W., Terzopoulos, D.: Autonomous pedestrians. In: SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 19–28. ACM Press, New York, NY, USA (2005). DOI http://doi.acm.org/10.1145/1073368.1073371
16. Sims, K.: Particle animation and rendering using data parallel computation. In: SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques, pp. 405–413. ACM, New York, NY, USA (1990). DOI http://doi.acm.org/10.1145/97879.97923
17. Treuille, A., Cooper, S., Popovic, Z.: Continuum crowds. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, pp. 1160–1168. ACM (2006)