Parallel Computation Using GPGPU to Simulate Crowd Evacuation Behaviors: Planning Effective Evacuation Guidance at Emergencies

Toshinori Niwa, Masaru Okaya, and Tomoichi Takahashi

Meijo University, Aichi, Japan
{e0930064,m0930007}@ccalumni.meijo-u.ac.jp, ttaka@meijo-u.ac.jp

Abstract. We propose parallel computing to simulate crowd evacuation behavior. It allows evacuation of ten thousands of agents to be simulated faster than does the existing system. Our prototype system consists of a new traffic simulator and scenario generator. The traffic simulation system uses a general purpose graphics processing unit (GPGPU) and simulates the agents' movements in a three-dimensional map. Our proposal enables realistic evacuation simulations and provides a platform that widens the applications of RoboCup Rescue Simulation to, for example, crowd evacuation from buildings. The evacuation simulations help security offices to prepare manuals for emergencies.

1 Introduction

The RoboCup Rescue Simulation (RCRS) system was designed in 1998 and the Rescue Simulation league started using its version 0 system [1]. The objectives of RoboCup Rescue Project are

- to apply agent technology to social problems and contribute human welfare,
- to provide a practical problem for development of research fields,
- to promote international research collaboration via the project.

RCRS aims to simulate rescue operations and human behaviors in disaster situations over an area of a few km^2 considering the number of people and facilities that exist in the area. The simulation results will be put to practical use during disasters and even before a disasters occurs.

It has been recognized that there are issues related to achieving the objectives that were set at the beginning of the RoboCup Rescue project [2]; that is it cannot simulate the behavior of a large number of agents or behaviors inside buildings. They are key issues for rescue simulation systems to model realistic disasters. Daidaitoku project tried to solve these issues [3]. The project aimed to simulate the evacuation of 10,000 people and operations to rescue them from a real 4km^2 area. Their system divided the large area into small areas. The size of the divided area was the order of RCRS. The project revealed that communication between agents in different areas impeded the efficient simulation of 4km^2.

In the RoboCup Rescue community, various approaches have been proposed and implemented to increase the scale of simulation: the number of agents, the size of area, etc. [4][5]. In 2009, at the Infrastructure competition of the Rescue Simulation League, a Java based kernel and a new traffic simulator were proposed. They were adopted as the RCRS version 1 system. However, the basic architecture of the RCRS version 1 remained the same as that of version 0. The number of agents remains at the order of a few hundred and it takes a lot of computing resources, time and memory to simulate disaster situations in which a large number of agents and evacuations from three dimensional (3D) buildings were involved.

We propose a system that can simulate situations where more than thousand agents start their evacuations by announcements from emergency broadcast system and share information on the emergency by communications. The system uses a general purpose graphics processing unit (GPGPU) and allows rescue and evacuation simulation to be performed faster than does using only CPU when the number of agents is large. In addition, it represents the agents' simulated movements on a 3D rather than a two-dimensional (2D) map. Section 2 overviews RCRS's architecture and lessons from past cases. Our GPGPU-based system, and parallel computations of the agents' behaviors are described in Section 3. In Section 4, the results of two simulations are shown. The possibility that the evacuation simulations can help security offices prepare manuals for emergencies is discuss in Section 5.

2 Lessons from Past Evacuation Cases and Overview of RCRS's Architecture

2.1 Lessons from Past Evacuation Cases

At the beginning, the RCRS was designed to simulate rescue operations at disaster situations that were modeled after the Hanshi-Awaji earthquakes at 1995 [2]. The importance of evacuations has been reaffirmed by reports on following disasters: the September 11, 2001 attacks and the Great East Japan Earthquake that occurred on March 11, 2011, along with the ensuing tsunami, took many lives and caused serious injuries [6][7].

Evacuation announcements greatly influence human behavior when they start evacuation and during their egress from buildings. Human behavior at emergencies are different. According to the reports on the September 11 and the tsunami 2011; some people evacuated immediately when the disasters occurred, while others did not evacuate, even when they heard emergency alarms from authorities. In the case of the two World Trade Center (WTC) buildings, WTC1 and WTC2 had similar layouts and size. Similar numbers of people were in the buildings during the attacks. However, the evacuation behaviors in each building were different. People in both buildings started evacuation when WTC1 was

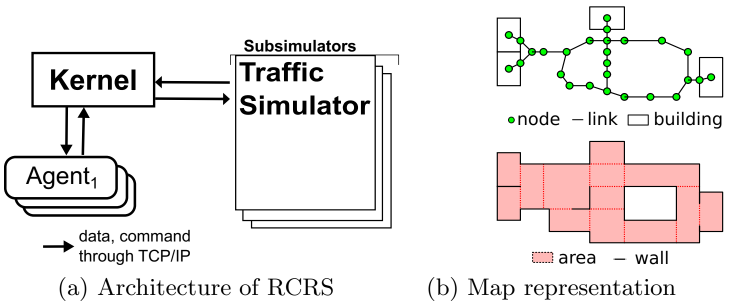


Fig. 1. Architecture and map representation of RoboCup Rescue simulation. (In version 0, network representation was used; in version1, open area representation was adopted.)

attacked. After 17 minutes, when WTC2 was attacked, about 83% of survivors from WTC1 remained inside, while 60% of survivors remained inside WTC2.

It is important to egress safely from buildings and to do rescue operations quickly during emergencies. Evacuation drills have been conducted at schools and shopping malls with intent to smooth evacuations and to conduct rescue operations properly. The evacuation drills are used to estimate the required safe egress time and to improve prevention plans for predictable emergent situations

2.2 Overview of RCRS's Architecure

While the RCRS was expected to simulate situations where 10,000 agents were involved, the number of agents remains a few hundred. Figure 1 (a) shows the basic architecture of RCRS, which consists of a kernel, sub-simulators and agents. The kernel, sub-simulator and agents are connected using the TCP/IP protocol, and therefore they can be executed on one computer or on several computers that are connected to via a network. This architecture allows RCRS to be run on distributed computers when simulations require significantly large computer resources. In fact, at the beginning of the Rescue Simulation league, several computers were used to run a simulation of RCRS. However, the execution of one sub-simulator is confined in one computer, which makes it difficult to simulate disasters at bigger area and with a lot of agents involved.

Figure 1 (b) shows two kinds of map representations. The upper one is a network representation and the lower one is an open area map representation. The network representation was used in RCRS version 0 to simulate the movements of agents from a building to refuges. The nodes of the net representation are intersections and buildings. The links represents roads which has the number of traffic lanes as its property. In the open area representation, roads and floors inside buildings are represented as polygons. The edges of the polygons represent walls or boundaries to adjoining areas. The traffic simulator of version 1 is one of sub-simulators and calculate the positions of the agent in one area and the translation from one are to the other area.

The kernel exchanges data and commands between agents and sub-simulators via kernel at every simulation step ($\mathit{\Delta} t$). The simulation step is the time of an agent's sense-decision-action cycle and its default values is one minute in the RCRS. The traffic simulator simulates the agents' movements in one simulation step by calculating their locations in the area at $\mathit{\Delta} \tau$ step, which is finer than a $\mathit{\Delta} t$ step, $\mathit{\Delta} t = N_{step} \times \mathit{\Delta} \tau$. In the default setting, N_{step} is 6,000.

3 GPGPU Based Simulation System

3.1 Parallel Computation in Calculating Agents' Positions

Our idea is to use GPGPU in calculation of agents' positions, $\mathbf{r}_i$, in a $\mathit{\Delta} t$. The changes of agent's position are calculated at evey micro step $\mathit{\Delta} \tau$ according to the Helbing social force model [8]. A social force model is a particle model in which human movements are controlled with Newton dynamics:

$$m_i \frac{d\mathbf{v}_i}{dt} = m_i \frac{v_i^0(t)\mathbf{e}_i^0(t) - \mathbf{v}_i(t)}{\tau_i} + \sum_{j(\neq i)} \mathbf{f}_{ij} + \sum_{W} \mathbf{f}_{iW} \tag{1}$$

where the first term is the intention-driven force that moves the agent to their desired positions. The agent's intentions determine positions that they want to go and $\mathbf{e}_i^0$ is a unit vector to the target places. m_i and v_i^0 are the weight and the speed of walking of agents i, respectively. The speed is set according to the age and sex of the agent and becomes faster when the agent is afraid and zero when it arrives at its destination. $\mathbf{v}_i(t)$ is a walking vector at t. $\mathbf{f}_{ij}$ and $\mathbf{f}_{iW}$ are repulsion forces to avoid collision with other agents or walls, respectively. The resultant force is the change of velocity, $\mathbf{v}_i(t) = \frac{d\mathbf{r}_i}{dt}$, and the change is used to calculate the next position of the agent after $\mathit{\Delta} t$.

Algorithm 1 shows a typical of traffic simulator code. The computations of the agents' position are performed for every agents and are iterated N_{step} times in one $\mathit{\Delta} t$. Simulations involving a large number of agents requires computations of $O(n^2)$ where n is the number of agents, because there are interaction terms between one agent and other agents that are around. More than 1,000 people are involved in real situations, and therefore many computer resources are required: CPU power, memory, and communications among the agent. Algorithm 2 shows a code that uses parallel computation. Codes from line 5 to 12 are computed in parallel on multi-threads. Line 1, 4 and 13 are data conversion and read/write to parallel computations.

3.2 Prototype System of GPGPU-Based Traffic Simulator

Figure 2 shows the architecture of our system, which is a subset of RCRS and specializes in crowd evacuation in emergencies. Our prototype system consists of three components: scenario generator, 3D traffic simulator, and agents:

Algorithm 1. Computation with one thread:

```
 1: call micro-step function $N_{step}$ times.
 2: function MICROSTEP
 3:     $n$ is the number of agents.
 4:     for $i = 0, \cdots, n$ do
 5:         calculate own intention-driven force and list walls near agent$_i$
 6:         for agent$_j$ near agent$_i$ do
 7:             calculate $\mathbf{f}_{ij}$
 8:         end for
 9:         calculate $\mathbf{f}_{iW}$
10:         calculate next candidate position of agent$_i$.
11:     end for
12:     calculate next position of all agents and return.
13: end function
```

Algorithm 2. Parallel computation with multiple threads:

```
 1: call initialization data conversion function for OpenCL.
 2: call micro-step function $N_{step}$ times.
 3: function MICROSTEP
 4:     transfer data to device and calculate following on parallel $M$ threads.
 5:     for $k(0, \cdots, M-1)$ thread,
 6:     for $i = k \times n/M, k \times n/M + 1, \cdots, k \times n/M + n/M - 1$ do
 7:         calculate own intention-driven force and list walls near agent$_i$
 8:         for agent$_j$ near agent$_i$ do
 9:             calculate $\mathbf{f}_{ij}$
10:         end for
11:         calculate $\mathbf{f}_{iW}$ and next position of agent$_i$.
12:     end for
13:     read data from device and calculate next position of all agents
14:     wait termination of all thread and return
15: end function
```

scenario generator: It plays the same role as the kernel of RCRS and controls the related 3D data and commands; for example, oral communication and vision are limited on the same story of the building. The initial positions of agents are also set as three dimensional coordinates by this scenario generator.

3D traffic simulator: It is based on the traffic simulator of version 1 which we developed further. The new features of the traffic simulator are (1) it can handle agents' movement in three dimensional buildings, and (2) it can use GPGPU to compute the calculations of the social forces of agents and the behaviors of mass agents in parallel at micro-steps.

agent: It is the same as the one used in RCRS version 1. It receives sensor data and decides its action based on their Belief-Desire-Intention (BDI) model at every simulation step (Δt). The BDI model features the evacuation behaviors of the agents. [9].

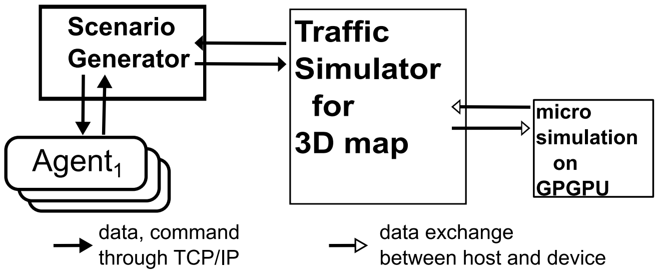
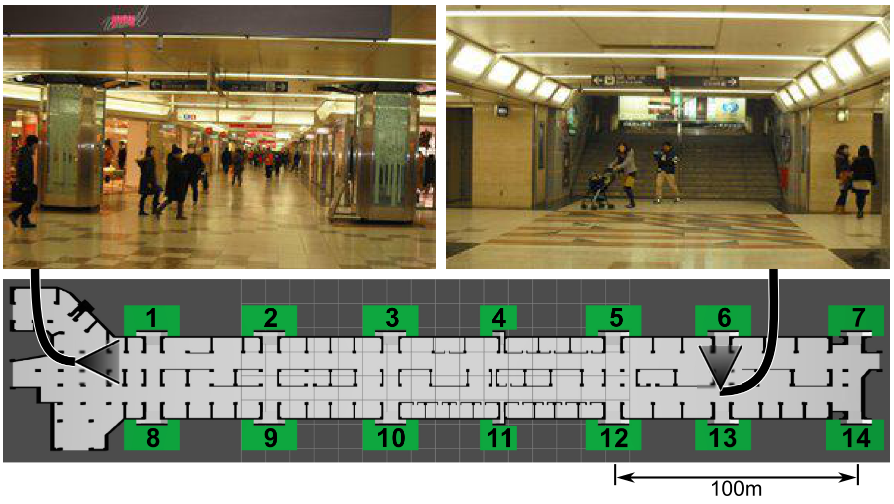
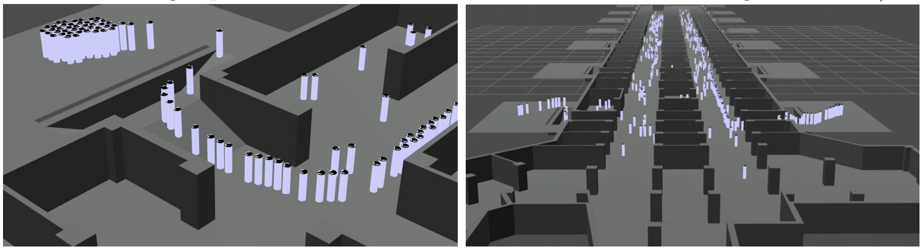

Scenario Generator
Traffic Simulator for 3D map
micro simulation on GPGPU
Agent$_1$
data, command through TCP/IP
data exchange between host and device

Fig. 2. Architecture of prototype evacuation system



(a) Layout of subterranean shopping mall: (Left photo is taken at the left end of one street and right photo is at exit 6. The numbers indicate exits to ground level.)



(b) Snapshot of evacuation simulation from a subterranean shopping mall. (left: a stair way to ground level from street in the mall. right: a birds-eye view of the mall.)

Fig. 3. Simulations of evacuations at subterranean shopping mall

4 Evacuation Simulation Experiments

4.1 Evacuation Scenario

Simulations of crowd's evacuations are useful to make a good prevention plan for emergencies. Figure 3 (a) shows the layout of the subterranean shopping mall in our city, Nagoya. The length and width are about 300m and 30m respectively, and there are about 90 shops are in three rows; two main streets pass between the rows. Exits to the ground are located at intervals of 50m.

At big earthquakes, the city office assumes that many pepole stay at this area. For security offices, it is important to guide people in the mall promptly to the ground at emergencies. Figure 3 (b) is a snapshot of the simulation of evacuating 1,000 agents to ground level from a subterranean shopping mall. We show two simulation experiments: one demonstrates an usage of simulation and the other is the effect of parallel computation using GPGPU.

4.2 Experiment 1: Effect of Guidance to Evacuation

This simulation shows how announcements from the security office effect the time of evacuations. The simulation scenario was as follows. A thousand agents are randomly located in the mall. The behaviors of the agents are modeled according to the report of the Great East Japan Earthquake that shows three types of human reactions when alarms are given.

instant evacuation: People who feel anxious after experiencing accidents that have involved extreme shaking initiate their own evacuation.

evacuation after tasks: People who do not feel anxious after accidents evacuate after completing their current activity. They do, however, feel anxious when they hear the guidance information announcement.

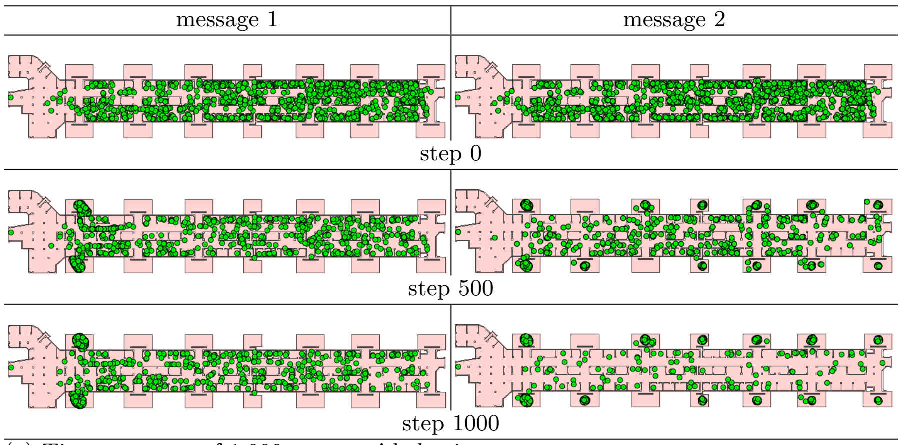
emergent evacuation: People who do not feel anxious and do not evacuate after completing their current activity and after hearing the evacuation guidance information initiate evacuation when they become extremely anxious after receiving new information from others.

The percentages of the types are 57%, 31%, and 12%, respectively. When evacuation guidances are announced, the agents start to evacuate according to their behavior types.

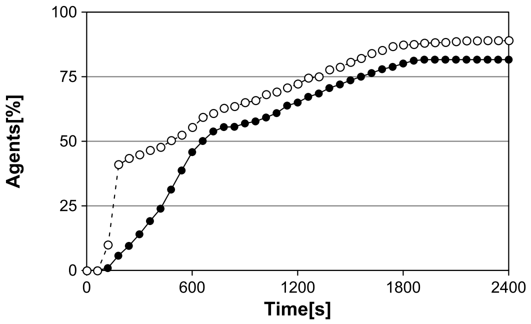
Following two different evacuation messages are announced.

- Message 1 gives instructions to evacuate by using two exits, numbers 1 and 8.
- Message 2 gives instructions to evacuate using the nearest exit except two exits, number 2 and 10.

Figure 4 (a) shows snapshots of the evacuation. The snapshots in the left and right column show evacuation guided by message 1 and message 2, respectively. In the snapshots, it can be seen that message 2 results in a faster evacuation than with guidance 1. Figure 4 (b) shows time sequence of evacuations and the rate of



(a) Time sequence of 1,000 evacuees' behaviors.



(b) Rates of evacuation from shopping mall (Black and white points correspond to evacuation guided with messages 1 and 2.)

Fig. 4. Simulations of evacuations at subterranean shopping mall

evacuees over simulation step. The evacuation rate of white points corresponding to message 2 is faster at start time, and the evacuation rates for both messages are similar for both guidances after congestion started at exits, The evacuation rate did not reach 100%, because all the agents are not instant evacuation type.

This experiment shows that the guidances given to evacuees influences their behaviors and, furthermore, that evacuation simulations can facilitate planning manuals for emergencies.

4.3 Experiment 2: Parallel Computation in on Simulation Step

The mall is connected to other subterranean malls and buildings. At emergencies, people pass through the mall and evacuate to safe places in the ground level. The number of people at emergencies become larger than at normal times. Table 1 shows the computation time of one evacuation simulation (Δt). The first row

Table 1. Execution time of one simulation step (ms)

	Algorithm & coding	number of agents										
		1,000	2,000	3,000	4,000	5,000	6,000	7,000	8,000	9,000	10,000	11,000
1	A	4.7	8.9	13.7	19.7	26.5	36.9	43.9	55.5	65.0	100.5	116.6
2	B_{cpu}	11.1	16.1	21.5	29.0	34.8	52.1	67.0	79.2	96.4	111.4	149.6
	device	(2.5)	(5.3)	(7.5)	(11.8)	(13.9)	(19.2)	(26.2)	(33.2)	(41.4)	(53.6)	(74.1)
	host	(0.7)	(1.4)	(2.2)	(3.6)	(4.7)	(5.7)	(7.0)	(9.1)	(14.6)	(12.0)	(16.5)
	B_{gpgpu}	10.8	13.6	16.9	21.4	24.3	38.8	42.7	52.7	62.6	80.7	90.3
	device	(1.4)	(2.9)	(3.4)	(4.3)	(5.3)	(7.5)	(7.8)	(10.3)	(12.7)	(14.2)	(22.9)
	host	(0.7)	(1.1)	(1.7)	(3.4)	(3.4)	(5.2)	(5.6)	(9.7)	(12.1)	(16.3)	(16.8)
	C_{cpu}	3.4	8.3	12.1	20.8	28.8	40.5	55.4	59.3	83.3	114.0	117.4
	device	(1.9)	(5.6)	(8.1)	(15.3)	(22.2)	(31.9)	(46.4)	(48.9)	(69.8)	(98.3)	(97.9)
	host	(1.1)	(2.1)	(3.0)	(4.3)	(5.1)	(6.8)	(7.0)	(8.2)	(10.7)	(12.6)	(16.1)
	C_{gpgpu}	6.1	10.8	15.6	17.7	22.2	29.1	35.4	42.9	48.5	58.8	80.6
	device	(4.6)	(8.1)	(11.8)	(12.4)	(16.0)	(21.2)	(26.0)	(31.8)	(37.1)	(46.0)	(60.3)
	host	(0.9)	(1.7)	(2.5)	(3.6)	(4.3)	(5.6)	(6.8)	(8.1)	(8.3)	(9.2)	(15.2)

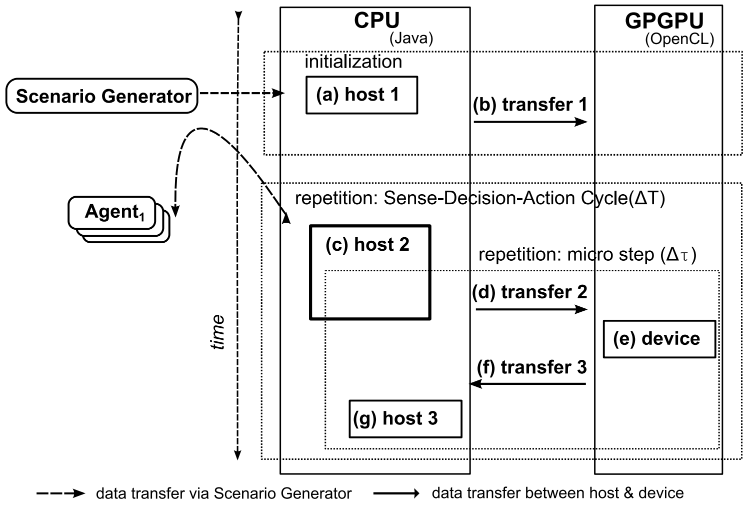


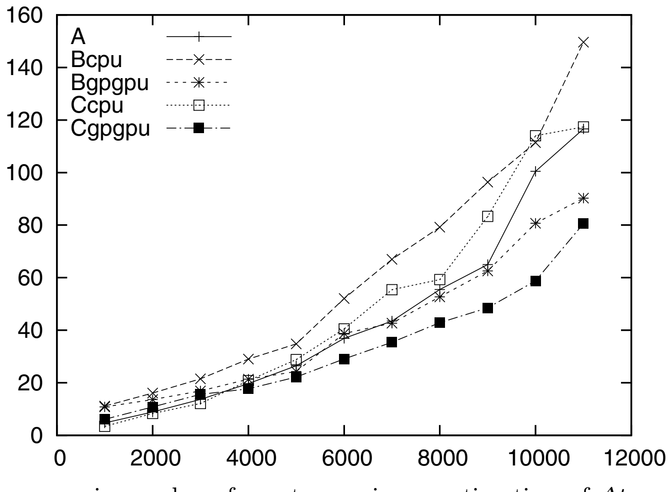
Fig. 5. Task assignments and data flow between CPU and GPGPU

of the table, labeled as algorithm 1 and coding A, shows the time and the time increases as the number of agents varies from 1,000 to 11,000. The simulations are executed on one PC with Fedora 14, and the simulation program is written by Java. Its CPU was Intel Core I7-3960X 3 (30GHz, 6core, 64GB memory (48GB heap)) . The second and third rows of Table 1 are execution times of a simulation based on Algorithm 2. Coding B differs from coding C in data exchange timing between host and device. Coding B exchanges data at every $\Delta\tau$ and coding C at every Δt. The values in parentheses: labeled as device and host of Table 1 are time executed in GPGPU and CPU.

Figure 5 shows task assignments between CPU and GPGPU. Agents send their goals to the traffic simulator every Δt. The traffic simulator is consisted of two modules; one is executed in the host (CPU) and the other is executed in parallel in the device (GPGPU) at every $\Delta\tau$. The codes in the host are written in Java and the codes in the device are written in Open Computing Language (OpenCL). OpenCL is a framework for programming on GPGPU hardware. GPGPU is a utilization of a GPU; it provides a parallel computing platform. We use OpenCL because codes written in OpenCL can be executed on PCs with or without GPGPU hardware.

- Routine (a) of Figure 5 corresponds to lines 1 and 2 in Algorithm 2 and is called once. It converts the structure of data and transfers them between codes written by Java and OpenCL in routine (b).
- Routines from (c) to (g) are executed every RCRS step. At the first step, the destinations where agents want to go are sent from the modules of agents in routine (c).

In code B, routines (c) to (g) are repeated N_{step} times at $\Delta\tau$ cycle. The destinations, and the present polygon and position in the polygon are transferred to GPU. The next positions of agents are calculated according to equation in routine (e). The positions are returned to host. Codes in routine (g) check overlaps between the positions and whether the agents move to other polygons. And it is repeated from routine (c). In code C, the repetition of N_{step} times are calculated in routine (e) on GPGPU. The same programs are executed on GPGPU, so the checks of overlapping and transfer to other polygons are executed for all agents



x-axis: number of agents, y-axis: execution time of Δt

Fig. 6. Execution time of vacuation simulation at shopping mall per step

whether it is necessary for not. Routine (c), (d), (f) and (g) are executed once in Δt.

The performance is trade-off cost of data transfer between CPU and GPCPU and the parallel programming in GPGPU. The subscriptions, $_{cpu}$ and $_{gpgpu}$, represents the computations with and without GPGPU. B_{cpu} stands for the code B running only CPU and C_{gpgpu} for the code C running on CPU and GPGPU. The GPU was NVIDIA Tesla C2075 (448core, 1.15GHz, 6GB memory).

Figure 6 shows the execution time in Table 1. At first, the overhead of data transfer negates the benefit of parallel computation using GPGPU. Over 4,000, the simulation coded using OpenCL with GPGPU (C_{gpgpu}) outperforms the others and the execution time remains quite low at ten thousands of agents.

5 Discussion and Summary

For achieving the objectives of the RCRS, the scale of simulation and the behaviors of agents in disaster situations are of key issues. While proposals and implementations have been made to apply distributed computing or parallel computing to RCRS, the interactions among agents over the area: communication among them and their movements impede the distributed computations. RCRS has not been sufficiently scaled up to simulate situations at the level of that the simulation results are put to practical use.

In this paper, we proposed a parallel computation in a traffic simulator. The traffic simulation system uses a GPGPU, which allows us to simulate evacuation of thousands of agents faster than the present system. Further, it can simulate the movements of agents inside buildings and guide the agents at their sense-decision-action cycle. These makes evacuation simulation realistic one, and the results show that evacuation behaviors of a large number evacuees can be simulated without an excessive increase in the required computation resources.

Our proposed system provides a platform that simulates ten thousands of crowd evacuation inside buildings and the evacuation simulations help security offices to prepare manuals for emergencies. This widens the applications of RCRS to make plans for decrease dameges from disasters.

References

1. Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjou, A., Shimada, S.: Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In: IEEE International Conference on System, Man, and Cybernetics (1999)
2. Takahashi, T., Tadokoro, S., Ohta, M., Ito, N.: Agent based approach in disaster rescue simulation - from test-bed of multiagent system to practical application. In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, pp. 102–111. Springer, Heidelberg (2002)
3. Takeuchi, I., Kakumoto, S., Goto, Y.: Towards an integrated earthquake disaster simulation system. In: First International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster (2003)

4. Skinner, C., Barley, M.: Robocup rescue simulation competition: Status report. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 632–639. Springer, Heidelberg (2006)
5. Takahashi, T.: `http://sakura.meijo-u.ac.jp/ttakaHP/MyPresentation/agentCompetition09.pdf`
6. Averill, J.D., Mileti, D.S., Peacock, R.D., Kuligowski, E.D., Groner, N.E.: Occupant behavior, egress, and emergency communications (NIST NCSTAR 1-7). Technical report. National Institute of Standards and Technology, Gaitherburg (2005)
7. C. O. G. of Japan. Prevention Disaster Conference, the Great West Japan Earthquake and Tsunami. Report on evacuation behavior of people (in Japanese), `http://www.bousai.go.jp/jishin/chubou/higashinihon/7/1.pdf` (date: February 9, 2012)
8. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic. NATURE 407, 487–490 (2000)
9. Okaya, M., Takahashi, T.: Evacuation simulation with guidance for anti-disaster planning. In: Chen, X., Stone, P., Sucar, L.E., van der Zant, T. (eds.) RoboCup 2012. LNCS (LNAI), vol. 7500, pp. 202–212. Springer, Heidelberg (2013)