2008 IEEE/RSJ International Conference on Intelligent Robots and Systems
Acropolis Convention Center
Nice, France, Sept, 22-26, 2008

Designing of online simulation environment for development control algorithms for robots operating in rough terrains

Kensuke Kurose, Satoshi Saga, Shogo Okamoto, Kazunori Ohno, and Satoshi Tadokoro

Abstract— **In the search and rescue situation at the disaster area, the searching task should be conducted by robots in order to avoid the second disaster. We study crawler robots that can traverse rough terrains and are used in such search operations. This paper describes the designing of an online simulation environment using which we can develop control algorithms for crawler robots, especially for the robot named "Kenaf." We use USARSim that can simulate three-dimensional space. By employing realtime simulation, we connect a real robot controller and the simulator directly and realize online simulation using the real robot controller. The tests show that the created simulation system can be used for the development of control algorithms.**

I. Introduction

In the search and rescue situation at the disaster area, the searching task should be conducted by robots in order to avoid the second disaster. Robots are expected to aid rescue workers in dangerous rescue operations. However, the realization of these robots requires significant scientific advances to address some fundamental challenges. The RoboCupRescue competition aims at evaluating the new rescue robot technologies in order to speed up the development of rescue and exploration systems. The competition has two leagues, the Rescue Robot League and the Rescue Simulation League. The Rescue Robot League fosters the development of high-mobility platforms with adequate sensing capabilities, and the Rescue Simulation League promotes research in planning, learning, and information exchange in an inherently distributed rescue effort [4]. In the recent years, rescue robots have increasingly attracted the attention of researchers.

We study crawler robots that can traverse rough terrains and are used in search operations. Because these crawler robots tend to have high degrees of freedom in order to change its shape for traversing rough terrains, their control is complex. Therefore, we are developing control algorithms that are aimed at attaching the autonomy toward crawler robots and making their control easier. These control algorithms have to be validated on real robots. However, the failure of the control during the validation may cause the breakdown of the robot, and it is difficult carry out repetitive validation.

Then, we create an online simulation environment that can connect a simulator to a real robot controller and simulate several types of movements, which are difficult to realize using real robots. By employing the online simulation environment system, we can validate the control algorithms when robots execute dangerous movements or repetitive movements. Additionally, using this system we can develop and evaluate the controller interface of robot, algorithms of controlling many robots simultaneously, and conduct a training of search and rescue operations using robot.

Various robot software development projects using simulators have been undertaken, for example, OROCOS/Orca [1], Player/Stage [2], CARMEN, MOAST [3], etc. However, in these researches, two-dimensional movements are considered. They don't consider the two-and-half-dimensional movements that crawler robots perform for traversing rough terrains.

On the other hand, there exist some simulators that can be used to simulate three-dimensional space. For example, OpenHRP3, which has its own physics simulator, can carry out distributed processing owing to RT-Middleware. This simulation system provides a seamless development environment comprising robot controlling algorithms and connects real robots and simulators. Using this system, we can run the robot controlling programs on the simulator, and also on real robots. Since many of the simulators are targetted to be used in humanoid robots, so the physics simulation mainly emphasize the kinematics of robot itself. Additionally the simulators were on the limited distribution stage on February 22, 2008.

USARSim is a widely known and used simulator and is used in the RoboCup Rescue Simulation League. Researches have been carried out to connect the USARSim simulator and some robot frameworks [1], [3], and some robot controlling algorithms have been developed. However, in these researches, two-dimensional movements are considered. They don't consider the two-and-half-dimensional movements that crawler robots perform for traversing rough terrains. For the

Fig. 1. The real Kenaf and the simulated Kenaf

Graduate School of Information Sciences, Tohoku University, Japan, (e-mail: {kurose, saga, okamoto, kazunori, tadokoro}@rm.is.tohoku.ac.jp)

crawler robots that travarse rough terrains at disaster sites, the simulator have to calculate and validate two-and-half-dimensional movements of the robots and interaction with many unknown environment.

We focus mainly on the mobile robot named "Kenaf," which is being developed by our group, and construct an online simulation environment for the development of controlling algorithms for the robot. We use USARSim as a three-dimensional physics interaction simulator. By employing realtime simulation, we connect the real robot controller and the simulator and realize an online simulation environment. Our goal is to improve the development environment of the control algorithms of these mobile robots.

II. Creation of online simulation enviromnent

Our aim is to create a seamless development environment comprising robot controlling algorithms to connect real robot controller and the simulator. We use the USARSim simulator, which is widely used mainly in the RoboCup Rescue Simulation League. Many researches have been carried out using USARSim ([4], [5], [6]), and the simulator will be updated for precise calculations.

A. USARSim

USARSim is a high-fidelity simulator for robots and environments and is based on the Unreal Engine. It is intended to be used as a research tool and is the basis of the RoboCup USAR simulation competition.

USARSim is incorporated with many robot models and some types disaster sites in its simulation environment. Additionally, with many types of sensors and actuators available, creating new robot models is not as difficult as scratch building. We can test these robot models in many created fields, such as NIST reference arenas and the fixed NIKE Silo environment. Further, we can create a new environment model by using Unreal Editor.

Because this simulator is primarily based on realtime simulation, kinematic simulations are not exactly precise. Hence, control algorithms can be tested in these environments in realtime with approximate precision. Further, the simulator is based on the client-server model; the characteristics of the simulator is well matched the control process on remote-controled robots.

USARSim is still under development and is expected to be a popular simulator for mobile robots.

B. Creation of model

USARSim has many sensors and actuators, which have been created for wheel-type robots, and we can reuse these modules in new robots. However, Kenaf has 4 flipper arms on its side and 2 tracks below its body. The flippers have their own tracks and can be moved like arms [7]. Kenaf has won the RoboCup Rescue Mobility Challenge 2007, which shows that the mobility of Kenaf is better than that of other robots. Each track of the robot holds the ground firmly, which aids the robot to stand firmly on the ground and traverse unknown rough terrains.

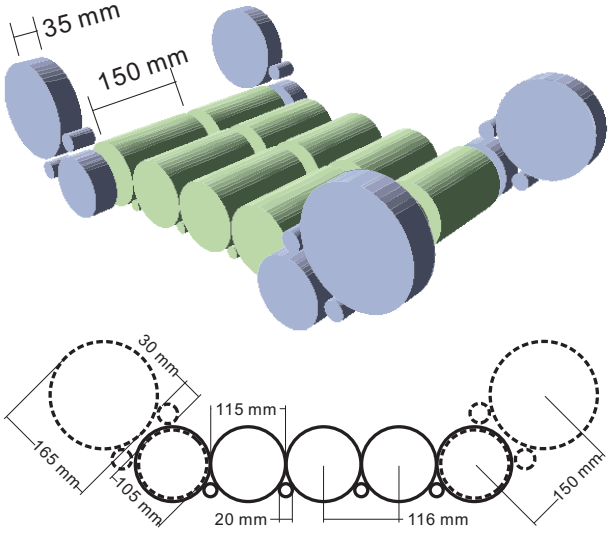

Fig. 2. Collision boxes of simulated Kenaf

The calculation cost for the simulation of movement of crawler robots is expected to behigher than that of wheel-type robots, which have only one contact point. For the online realtime simulation, we adopt a wheel-type robot that mimics a flipper-type crawler robot. This method is used in the snake-type robot model, "Soryu," which has 3 linked track bases [8].

In USARSim, the "Track class" option considers many wheels. This class realizes unified control and unified tangential velocity of many wheels, each having a different diameter. However, the use of this class increases the calculation cost. In order to decrease the calculation cost as far as possible, we use this class for the four flippers, which have different lengths. The main tracks mounted as normal wheels below the body of the robot.

In the robot model, the actuators are mounted in its 2 main tracks and in its 4 flipper arms for flipping the arms up and down. The Encoder sensors of the robot are placed along the motor axis in order to determine each motor rotation. By employing the "GroundTruth class," another sensor is attached to the body of the robot to measure the position, (x, y, z), of the robot's center of gravity, the robot's posture, $(roll, pitch, yaw)$, and the precise time.

Kenaf has a pan-tilt-zoom camera fixed at the front of its body and bird's-eye view camera at the top of its body. Our model also simulates these cameras. By adjusting the position and angle of view of the cameras to be same as those of the real camera, we can obtain the similar images of the simulated environment.

In order to match the physical properties of the real robot, we have to adjust many parameters. However, we introduce only the main parameters that are important for traversing two-and-half-dimensional rough terrains. The mass is set to be equal to that of the real robot (20 kg, $ChassisMass = 20.0$). The position of the center of gravity is 50 mm below

the center of the robot ($KCOMOffset = (x = 0.0, z = -0.05)$). In the following experiments, since the batteries are removed from the robot, we shift the center of gravity 15 mm behind the set position ($KCOMOffset = (x = -0.015, z = -0.05)$). Then, we consider the coefficient of friction. Since the real robot has some tracks, the physical interaction with environment is complex. Additionally, the tracks have some lugs. These features lead to a higher tangential driving force in the tracks as compared to ordinary wheels. In order to simulate the driving force using ordinary wheels, we set the coefficient of friction to a large value in the rolling direction ($TireRollFriction = 1.3$).

C. Simulator library

In order to create a seamless development environment to connect the real robot and the simulator, we create a simulator library that corresponds to the program library of the real robot. The user program can drive the real robot and the simulator by changing only the linking library without changing any source code of the user program.

In the real robot, the output motor driving command from the robot controller and the input information from the sensors such as encoders are sent throuth a CAN communication bus. The instructions for controlling of these devices are provided as functions in the development robot library of the user programs. On the other hand, in the simulator, the motor driving command output and input information from the sensors are all realized by TCP/IP. In order to connect the two systems seamlessly, we create the simulator library, which corresponds to the program library of the real robot.

The library for the robot controller is shown at the center of figs. 3 and 4. By creating this simulator library, we connect the real robot controller and the simulator by intercepting the communication information between real robot controller and the motors encoders in the real robot.

Using this library, we can share the header file of the development library of the real robot. Further, the function name and the parameters are same as these in the program library of the real robot. Using these techniques, we can use the real robot and the simulator effortlessly by changing only the linking library without changing any source code of the user program.

D. Intermediate process between simulator and real robot controller

There are two I/O ports, one each in the real motors/sensors and the simulated motors/sensors in the simulator system (fig. 4). The two ports are different from each other in the following aspects:

- protocols
- send/receive timing
- command structure

Therefore, an intermediate process required to resolve the difference between the simulator and the development library of real robot.

In order to realize the connectivity by providing all the information to the sensors/actuators/cameras and the robot

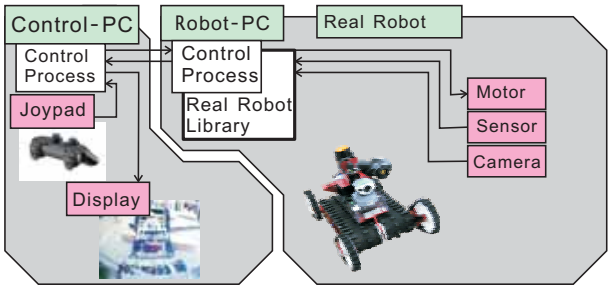

Fig. 3. Connections in real Kenaf

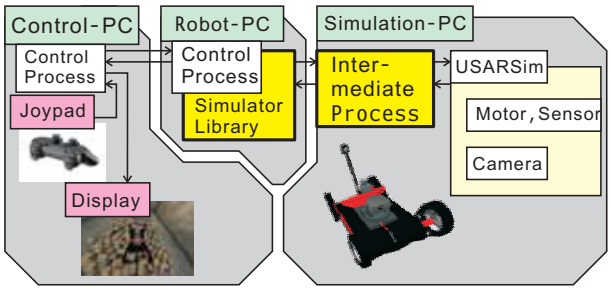

Fig. 4. Connections in simulated Kenaf

controller on the Ethernet and intercepting these information, we create the online simulation environment. Kenaf is designed to be compatible with this connectivity. Therefore, we can connect the real robot controller and the simulator directly without changing any hardware configuration.

First, the simulator library intercepts and decomposes the command from the robot controller and sends the recomposed command to the simulator, thus, it controls the robot model in the simulator. Then, the library stores the robot status and sensor data continuously and sends the feedback information to the robot controller when a request command is received.

This communication between the simulator and the robot controller requires the following functions:

Packet intermediate function (TCP/UDP send/receive)
: The communication between USARSim and the intermediate process is based on TCP/IP, and the communication between the intermediate process and the real robot controller is based on UDP. This function transforms these two protocols.

Coordinate transformation function
: A coordinate transformation takes place between the three-dimensional coordinate of USARSim and the three-dimensional coordinate of the real robot/sensors.

Reconstruction of commands functions
: The difference between the commands of the simulator and the real robots are resolved.

To realize these functions, we create three threads. One receiving thread is created for analyzing the command streams that are sent from the robot controller irregularly and for

fast communication using UDP. Another receiving thread is created for analyzing the sensing and robot status data streams that are sent from the simulator by using TCP/IP. The latter thread transforms the coordinate systems and units between the real robot controller and the simulator by using the received data. Simultaneously, required commands, which are based on the command systems, are created. The created commands are sent to each system by using UDP and TCP/IP connections. Fig. 5 shows these structures.

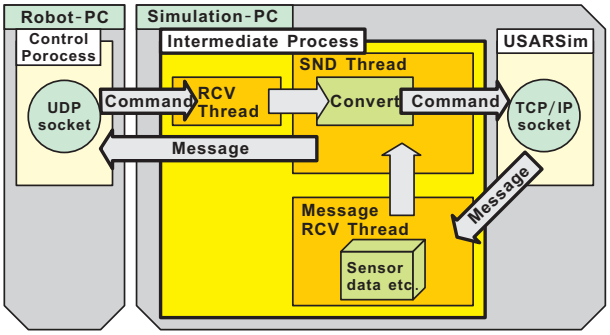

Fig. 5. Threads of intermediate process

III. Evaluation of Simulation System

We evaluate the constructed simulation system by comparing the movements of the simulated robot and the movements of the real robot. Afterwards we introduce programming examples to confirm the online simulation environment connecting the simulator model and the real robot controller.

A. Evaluation of simulator and the robot model

First, we evaluate the simulator by comparing the movements of the simulated robot and the real robot. Following experiments are carried out:

- climbing a step (50, 100, and 150 mm in height) without using flippers
- climbing a step (100, 200, 300, and 350 mm in height) using flippers
- crossing a trench (100, 200, 300, 350, and 400 mm in width)

The step and trench are shown in figs. 6 and 7. Fig. 8 shows an example of a mobility experiment carried out using the simulator and the real robot (climbing a step (200 mm in height) using flippers).

Three tests were conducted to confirm that the simulated robot had similar traversing ability as that of the real robot. In the present study, step climbing and trench crossing were conducted to evaluate the basic performance of the robots.

The result (successful ($\surd$) or failed ($\times$)) of the tests are shown in Table I. As shown in the table, both the simulated and real robots can climb a step up to 100 mm in height without using flippers. Both the simulated and the real robots can climb a step up to 300 mm in height using flippers. The real and simulated robots can cross a trench of width up to 300 mm and 350 mm,respectively.

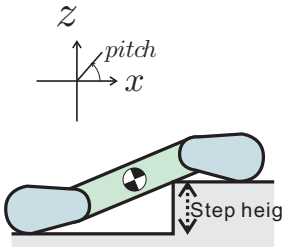

Fig. 6. Climbing step

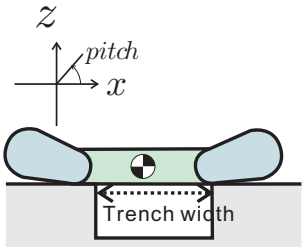

Fig. 7. Crossing trench

Fig. 8. Simulator evaluation. Top: simulated robot and bottom: real robot.

Thus, except for one test, all tests show consistent result for the simulated and real robots. Next, we discuss the inconsistency observed in one test. In this trench (350-mm width) test, the real robot loses its balance for a moment, and its body tilts and the back flippers of the robot touch the ground. Thus the real robot fails this test. On the other hand, the simulated robot also loses its balance for a moment; however, it regains its balance by go forward before the back flippers touch the ground. This result shows that the simulated inertia of pitch rotation of the simulated robot is lower than that of the real robot. There exist limitations on the adjustment of inertia tensor parameters, though. Further improvement is difficult at this stage.

TABLE I
(Successful ($\surd$) / failed ($\times$)) climbing steps and crossing trenches tests

	Step height [mm]						
	Without flippers			With flippers			
	50	100	150	100	200	300	350
Real	$\surd$	$\surd$	$\times$	$\surd$	$\surd$	$\surd$	$\times$
Sim.	$\surd$	$\surd$	$\times$	$\surd$	$\surd$	$\surd$	$\times$

	Trench width [mm]				
	100	200	300	350	400
Real	$\surd$	$\surd$	$\surd$	$\times$	$\times$
Sim.	$\surd$	$\surd$	$\surd$	$\surd$	$\times$

In order to analyze the test results, we compare the trajectories of position and posture as a excluding dynamic effect of pitch rotation inertia. We apply DP matching by using the norm ($norm = \sqrt{x^2 + z^2}$) of the position as an index and compare the trajectories of the simulated and real robots. The following figure shows some examples of trajectories before and after the DP matching. Each component of position is shown in figs. 9-12. The pitch component of posture is shown in figs. 13 and 14. These are examples of climbing a step

of height 200 mm using flippers. These figures show the DP matching using the norm index minimizes the elements of position, (x, z), and pitch component of posture. From fig. 15, the maximum errors in $x, zandpitch$ per frame are 83.8 mm, 38.4 mm and 11°, respectively. Thus, the movements of the simulated and real robots are more or less similar.

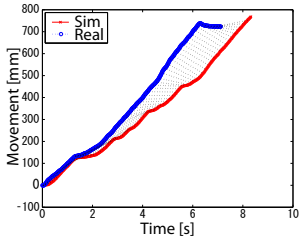

Fig. 9. Comparison of real and simulated movements (x-axis) in climbing 20-cm-high step

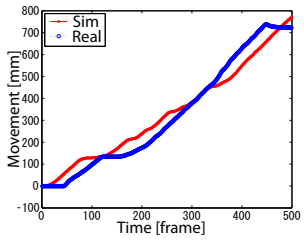

Fig. 10. Comparison of real and simulated movements (x-axis) in climbing 20-cm-high step

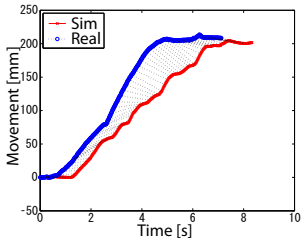

Fig. 11. Comparison of real and simulated movements (z-axis) in climbing 20-cm-high step

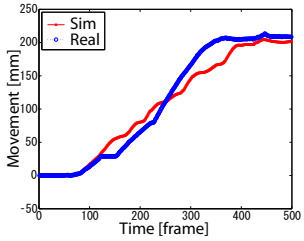

Fig. 12. Comparison of real and simulated movements (z-axis) in climbing 20-cm-high step

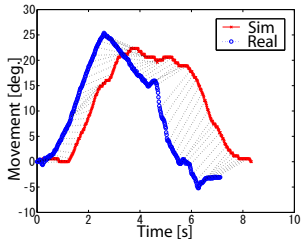

Fig. 13. Comparison of real and simulated movements (pitch) in climbing 20-cm-high step

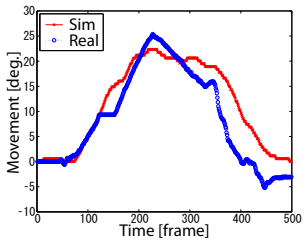

Fig. 14. Comparison of real and simulated movements (pitch) in climbing 20-cm-high step

Fig. 16 shows all the results of the conducted tests. The norm error indicates the position error after the DP matching, and the pitch error indicates the posture error after the DP matching. In tests in which the robots successfully traversed 750 mm, the maximum mean errors are 132.2 mm (norm) and 11.4° (pitch). In the failed tests (150-mm-high step, 350-mm-high step using flippers, 400-mm-width trench), the maximum mean errors are 88.1 mm (norm) and 18.2° (pitch). In the successful tests, the pitch errors are limited to small values. On the other hand, the variances in the failed tests are larger than those in the successful tests.

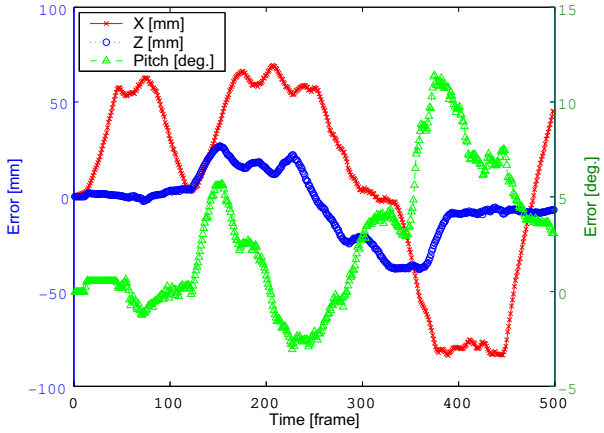

Fig. 15. Change in error while climbing 20-cm-high step

This result shows that the error was produced because the simulated inertia of the pitch rotation of the simulated robot was lower than that of the real robot.

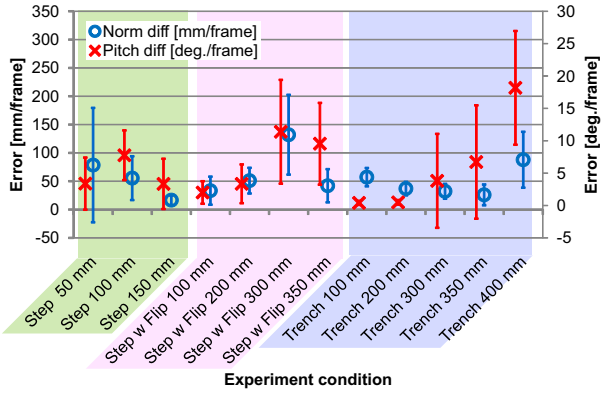

Fig. 16. Mean error per frame for real and simulated robots

B. Programming sample

To evaluate the performance of the created library and the intermediate process, we make the robots execute simple movements.

1) Autonomous square-shaped movement validation: In this test, we program the robots to execute the simple movement shown in fig. 17, "autonomous square-shaped movement." Fig. 18 shows similar results for the simulated and real robots.

The difference of initial value of each robot, fig. 18 shows the difference in movement distance. However, we can realize similar movement in both the robots by using the same source code.

2) Operational control program validation: In the real robot, there already exists a basic user program for the control PC that sends the operation command from the operator directly to the robot. We evaluate the performance

```
int main(int argc, char *argv[]){
   int i;
   hrt_kenaf_init();
   for(i=0;i<4;i++) {
     hrt_show_status();
     hrt_go_forward(100.0); // move 100.0 mm
     hrt_turning(90.0);     // turn  90.0 deg.
   }
   hrt_stop();
   return (0);
 }
```

Fig. 17. Source code for square movement

Fig. 18. Real and simulated square movements

of the simulator library and the movement of the simulated robot without changing the simple operation program.

We simultaneously run the existing operation program on the control PC and the control process constructed using the simulator library on the robot PC. Thus, without changing the existing operation program, we confirm that the movement of the simulated robot is similar to that of the real robot.

Simultaneously, by installing a camera module on the simulated robot at the same position as that in the real robot, we can acquire the images of the simulated environment (fig. 19). This method realizes a training system without using the real robot. The real robots are primarily manipulated by an operator via a video stream from a camera fixed on the robot. In USARSim, it is possible to capture the scenes during the simulation and use them as video feedback. This tequnique is very close to the working of a real camera. Hence, this tequnique can be used to simulate a teleoperation environment.

Fig. 19. Simulated camera views

IV. Conclusion

This paper describes the designing of an online simulation environment using which we can develop control algorithms for crawler robots, especially for the robot named "Kenaf."

First, we simulate a robot model on the USARSim environment and compare its movements with that of the real robot. The results show that the dynamics of pitch rotation have to be considered. The comparison of the trajectories shows that the pitch component of the maximum mean errors is $11.4°$ in successful traversing a distance of 750 mm. In failed tests, the pitch component of the maximum mean errors is $18.2°$. Thus, we confirm that the movements of the simulated robot model are similar to those of the real robot.

Next, we design an intermediate process that translates and relays the data using the real robot controller and the simulator system and the simulation library that realizes a seamless development environment connecting the real robot and the simulator. Then, in order to evaluate the performance of the intermediate process and simulator library, we test a simple user program for performing square movement and an operational control program.

Finally, we create online simulation development environment comprising control algorithms for crawler robots. The characteristic of the system is that it can connect a real robot controller and a simulated robot.

In the future, by using the developed online simulation environment, we will develop new control algorithms for crawler robots and validate the algorithms for repetitive or dangerous movements. Further, we plan to use this system as a simulation environment for the development of operation interfaces.

Acknowledgement

This research was carried out as a part of the NEDO Project for Strategic Development of Advanced Robotics Elemental Technologies, High-speed Search Robot System in Confined Space.

References

[1] Herman Bruyninckx. Open robot control software: the OROCOS project. In *IEEE International Conference on Robotics and Automation*, pp. 2523–2528, 2001.

[2] B. Gerkey, R.T. Vaughan, and A. Howard. The Player/Stage project: tools for multi-robot and distributed sensor systems. In *Proceedings of 11 th International Conference on Advanced Robotics*, 2003.

[3] Christopher Scrapper, Stephen Balakirsky, and Elena Messina. MOAST and USARSim: a combined framework for the development and testing of autonomous systems. In *Unmanned Systems Technology VIII. Proceedings of SPIE*, Vol. 6230, p. 62301T, 2006.

[4] Stephen Balakirsky, Chris Scrapper, Stefano Carpin, and Mike Lewis. USARSim: a robocup virtual urban search and rescue competition. In *Unmanned Systems Technology IX. Proceedings of SPIE*, Vol. 6561, p. 65611M, 2007.

[5] S. Nourbakhsh, M. Sycara, M. Koes, M. Young, Lewis, and S. Burion. Human-robot teaming for search and rescue. *IEEE Pervasive Computing: Mobile and Ubiquitous Systems*, pp. 72–78, 2005.

[6] S. Carpin, T. Stoyanav, Y. Nevatia, M. Lewis, and J. Wang. Qunatitative assessments of USARSim accuracy. In *Proceedings of the 2006 Performance Metrics for Intelligent Systems*, 2006.

[7] Tomoaki Yoshida, Eiji Koyanagi, Satoshi Tadokoro, Kazuya Yoshida, Keiji Nagatani, Kazunori Ohno, Takashi Tsubouchi, Shoichi Maeyama, Itsuki Noda, Osamu Takizawa, and Yasushi Hada. A high mobility 6-crawler mobile robot "Kenaf". In *Proceedings of 4th International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster*, p. 38, 2007.

[8] Shogo Okamoto, Adam Jacoff, Steven Balakirsky, and Satoshi Tadokoro. Qualitative validation of a serpentine robot in USARim. In *Proceedings of 2007 JSME Conference on Robotics and Mechatronics, Akita, Japan*, pp. 2P1–L02, 2007.