Towards the Integration of Real-Time Real-World Data in Urban Search and Rescue Simulation

Holger Kenn[1] and Alexander Kleiner[2]

[1] TZI
Universität Bremen
kenn@tzi.de,
http://www.wearable-computing.de/
[2] Institut für Informatik
Universität Freiburg
kleiner@informatik.uni-freiburg.de,
http://www.informatik.uni-freiburg.de/

Abstract. The coordinated reaction to a large-scale disaster is a challenging research problem. The Robocup rescue simulation league addresses this research problem but is currently lacking an interface to real-world real-time data to test the validity of both simulation and simulated reactions. In this paper, we describe a wearable-computing-based real world interface to the Robocup Rescue simulation software and provide some updated results of preliminary evaluations.

1 Introduction

Large scale urban disaster situations such as earthquakes, floods or terrorist attacks pose an important threat to modern urban civilization centers. Examples such as the Kobe earthquake, the pacific tsunami, the hurricane Kathrina or the tragic events of September 11th 2001 demonstrate that although preventive measures and disaster response plans were in place, these events pushed existing countermeasures over their limits, resulting in loss of human lives, chaotic situations and long term adversary effects on the affected regions.

After the Kobe earthquake, the japanese government decided to promote researchers to work on problems related to large-scale urban disasters. One of the outcomes of this initiative was the Robocup Rescue competition. Using the same successful method of competition-based benchmarks that the Robot Soccer competition was applying, the Robocup Federation added two new competitions to the Robocup, Robocup Rescue Robot League and Robocup Rescue Simulation League.

In the Rescue Robot leagues, physical robots are designed and tested in simulated disaster situation. The aim of the robot and its operator is to map an unknown environment and provide information about simulated disaster victims such as their location and situation, their simulated medical condition and other helpful indications such as ID tags. Robot and operator work under strict time constraints, the operator can only perceive the situation in the disaster arena through the sensors of the robot. In order to support the development of autonomous robots, there are parts of the arena that are covered by a simulated communication blackout, i.e., cannot be explored by a tele-operated robot.

The Rescue simulation league aims at simulating large-scale disasters and exploring new ways for the autonomous coordination of rescue teams [8]. In the Rescue Simulation league, the goal of a team participating in the competition is to provide a software system that reacts to a simulated disaster situation by coordinating a group of simulated agents such as police, ambulance and fire brigade agents. This goal lead to challenges like the coordination of heterogeneous teams with more than 30 agents, the exploration of a large-scale environment in order to localize victims, as well as the scheduling of time-critical rescue missions. Each of these agents only has a limited amount of communication bandwidth they can use to coordinate with each other, so the problem cannot be addressed by a central coordination entity but has to be solved by a true multi-agent system. Moreover, the simulated environment is highly dynamic and only partially observable by a single agent. Agents have to plan and decide their actions asynchronously in real-time. Core problems are *path planning*, *coordinated fire fighting*, and *coordinated search and rescue* of victims.

The performance of a team in a scenario is scored by the simulation software that provides a number of measures derived from the simulation, e.g., the survival rate of civilians, the overall health of the simulated responders and the percentage of buildings burned down. These measures are then used to calculate the final score of a simulation run. Thus, in order to reach a high score, a team has to optimize their multi-agent system towards maximizing beneficiary elements of the scoring function while minimizing the adversary elements.

Both leagues change elements of the competition and scoring functions from competition to competition to foster the development of new capabilities and sustain a continuous progress towards obtaining real-world usable systems which is the long-term goal of the leagues.

Comparing the results of the two leagues over the last years, we find that a rapid progress has been made in specific areas such as robot self-localization and mapping[4] and autonomy[11], but that little progress has been made so far towards the application of the technology developed in real-world disaster and training situations.

Applying the simulation system in the real world currently lacks the interface to the real world information, i.e., currently, the simulation system relies on carefully designed special-purpose map data and observations generated by the simulation itself, e.g., agent motion is computed by a traffic simulator and cannot be observed from real-world motion of responders. However, by using wearable computing technology, we can provide such observations to the simulation system. This has three uses. First, it can be used to record real-world data from real-world intervention scenarios and by this, assess the validity of the simulation. Second, it can be used to observe the reaction of the team multi-agent systems to real-world data. Third, it is a step towards using both the simulation system and the team multi-agent systems to support incident commanders and responders in training and real interventions by providing autonomous decision support and faster-than-realtime simulation for strategy decisions.

The solutions presented in this paper are based on the open source agent software [2], which was developed by the *ResQ Freiburg 2004* team [10], the winner of RoboCup 2004. The software prototype designed for linking responders to the simulation has also been released [1].

We propose preliminary results from a wearable computing device, acquiring disaster relevant data, such as locations of victims and blockades, and show the data integration into the *RoboCupRescue Simulation* [8] platform, which is a benchmark for MAS within the RoboCup competitions. Communication between wearable computing devices and the server is carried out based on the open *GPX* protocol [20] for GPS data exchange, which has been extended for additional information relevant to the rescue task. We show exemplarily how the data can consistently be integrated and how rescue missions can be optimized by solutions developed on the RoboCupRescue simulation platform. The preliminary results indicate that nowadays wearable computing technology combined with MAS technology can serve as a powerful tool for Urban Search and Rescue (USAR).

The remainder of this paper is structured as follows. We present an interface between human rescue teams and the rescue simulator in Section 2. In Section 3 we give some examples how approaches taken from MAS can be utilized for data integration and rescue mission optimization. In Section 4 we show the results of experiments integrating data into RoboCupRescue and infrastructureless indoor tracking of responders and conclude in Section 5.

2 Interfacing Human Responders

In wearable computing, one main goal is to build devices that support a user in his primary task with little or no obstruction. Apart from the usual challenges of wearable computing [19, 18], in the case of emergency response, the situation of the responder is a stressful one. In order to achieve primary task support and user acceptance, special attention has to be given to user interface design. For this application, the user needs the possibility to enter information about his observations and needs feedback from the system which recorded and transmitted the information [3]. Furthermore, the user needs to receive information from the system that provides task-related instructions from the command center.

The implementation has to cope with multiple unreliable communication systems such as existing cell phone networks, special-purpose ad-hoc communication and existing emergency-response communication systems. As the analysis of the different properties of these communication systems is beyond the scope of this article, we will therefore abstract from them and assume an unreliable IP-based connectivity between the mobile device and a central command post. This assumption is motivated by the fact that both infrastructure-based mobile communication networks and current ad-hoc communication systems can transport IP-based user traffic.

For mobile devices, a number of localization techniques are available today, for an overview see [6]. Although some infrastructure-based communication networks are also capable of providing localization information of their mobile terminals, we assume the presence of a localization device with a GPS-like position accuracy.

The rationale behind this is that the localization information provided by communication systems is not very precise (e.g. sometimes limited to the identification of the

[3] Technically, this feedback is actually not required by the application, but we envision that it will improve user acceptance.

current cell, which may span several square kilometers) and therefore not usable for our application. The GPS system also has well-known problems in urban areas and in buildings. But by applying techniques such as the ones stated in [13], we have improved its reliability and accuracy for indoor localization.

The situation of the device and its user is also characterized by harsh environmental conditions related to the emergency response, such as fire, smoke, floods, wind, chemical spilling etc. The device has to remain operable under such conditions, and moreover to provide alternative means of input and output under conditions that affect human sensing and action abilities. Moreover, the system has to be integrated into the user processes of emergency response, e.g. it must have no impact on response times of units and therefore should be integrated into the normal gear of responders.

As these requirements are quite complex, we decided to design and implement a preliminary first test system without these requirements and later a wearable emergency response system which is supporting the requirements in its system design.

2.1 A First Test System

In order to analyze the properties of the communication and localization systems and test the software interface to the simulation system, a preliminary test system has been implemented, for which three requirements have been dropped, the design for harsh environmental conditions and emergency response processes, indoor localization capability and the ability to use multiple alternative input and output.

The communication and localization system is independent of the user requirements with the exception of the fact that the system has to be portable. Therefore we chose a mobile GPS receiver device and a GSM cell phone device as our test implementation platform. The GPS receiver uses the bluetooth [3] personal area network standard to connect to the cell phone. The cell phone firmware includes a Java VM based on the J2ME standard with JSR82 extensions, i.e., a Java application running on the VM can present its user interface on the phone but can also directly communicate with bluetooth devices in the local vicinity and with Internet hosts via the GSM networks GPRS standard.

The implementation of the test application is straightforward: It regularly decodes the current geographic position from the NMEA data stream provided by the GPS receiver and sends this information to the (a priori configured) server IP address of the central command center. The utilized protocol between the cell phone and the command center is based on the widely used GPX [20] standard for GPS locations.

A detailed description of the protocol extension can be found in [9].

2.2 A Wearable Emergency-Response System

In order to fulfill more requirements, another system has been designed based on additional hard- and software. The system uses a miniature PC system, the so-called *OQO*. It is based on a transmeta CPU running the Linux operating system, has bluetooth and WiFi wireless interfaces, and can be extended via USB interfaces. The wearable CPU

core runs the main application program. For localization, a bluetooth GPS receiver and a XSENS 6DOF motion sensor are used.

As already stated, the design of the user interface is a crucial one for this application. Therefore, we use a glove as wearable user input device [14] and a wireless link between the user interface device and the wearable computer. Such an interface has already been used in other applications such as aircraft maintenance [16] (see Figure 1(d)).

(a) (b) (c) (d)

Fig. 1. The *OQO-based wearable computer*: (a) CPU unit with GPS and mobile phone. (b) Sensor for Pedestrian Dead Reckoning. (c) HMD worn by the test person. (d) A glove-based wireless wearable interaction device

The primary output device is a head-mounted display that can be integrated into existing emergency-response gear such as firefighter helmets and masks (see Figure 1(b)). In applications where headgear is not commonly used, the output can also be provided through a body-worn display device or audio output.

3 Multi Agent Systems (MAS) for Urban Search And Rescue (USAR)

3.1 Data Integration

Generally, we assume that, if communication is possible and new GPS fixes are available, the wearable device of a rescue team continuously reports the team's trajectory as a *track* message to the command center. Additionally, the rescue team might provide information for specific locations, as for example, indicating the successful exploration of a building, the detection of a victim, and the detection of a blocked road, by sending a *waypoint* message.

Based on an initial road map and on the information on road blockage and the autonomously collected data on trajectories traveled by the agents, the current system builds up a connectivity graph indicating the connectivity of locations. The connectivity graph between a single location and all other locations is constructed by the Dijkstra algorithm. The connectivity between two neighboring locations, i.e. the weight of the corresponding edge in the graph, depends on the true distance, the amount of blockage, the number of crossings, and the number of other agents known to travel on the same route. In the worst case, the graph can be calculated in $O\left(m + nlog\left(n\right)\right)$, where n is the number of locations and m the number of connections between them. The knowledge of the connectivity between locations allows the system to recommend "safe" routes to rescue teams and to optimize their target selection.

The search for victims of many rescue teams can only be coordinated efficiently if the rescue teams share information on exploration. We assume that rescue teams report when they have finished to explore a building and when they have found a victim, by transmitting the according message to the command center. The command center utilizes this information for distributing rescue teams efficiently among unexplored and reachable locations.

3.2 Rescue Sequence Optimization

Time is a critical issue during a real rescue operation. If ambulance teams arrive at an accident site, such as a car accident on a highway, it is common practice to optimize the rescue sequence heuristically, i.e. to estimate the chance of survival for each victim and to rescue urgent cases earliest. During a large-scale disaster, such as an earthquake, the efficient distribution of rescue teams is even more important since there are many more victims and usually an insufficient number of rescue teams. Furthermore, the time needed for rescuing a group of victims might significantly vary, depending on the collapsed building structures trapping the victims.

In RoboCupRescue, victims are simulated by the three variables *damage*, *health* and *buridness*, expressing an individual's damage due to fire or debris, the current health that continuously decreases depending on damage, and the difficulty of rescuing the victim, respectively. The challenge here is to predict an upper bound on the time necessary to rescue a victim and a lower bound on the time the victim will survive. In the simulation environment these predictions are carried out based on classifiers which were induced by machine learning techniques from a large amount of simulation runs. The time for rescuing civilians is approximated by a linear regression based on the buridness of a civilian and the number of ambulance teams that are dispatched to the rescue. Travel costs towards a target can directly be taken from the connectivity graph. Travel costs between two reachable targets are estimated by continuously averaging costs experienced by the agents [4].

We assume that in a real scenario expert knowledge can be acquired for giving rough estimates on these predictions, i.e. rescue teams estimate whether the removal of debris needs minutes or hours. Note that in a real disaster situation the system can sample the

[4] Note that the consideration of specific travel costs between targets would make the problem unnecessarily complex.

approximate travel time between any two locations by analyzing the GPS trajectories received from rescue teams in the field. Moreover, the system can provide for different means of transport, i.e. car or by feet, the expected travel time between two locations. The successful recognition of the means of transport from GPS trajectories was already shown by Liao and colleagues [15].

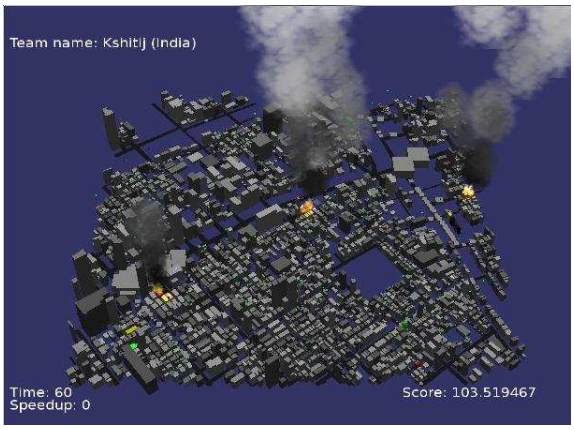


Fig. 2. A 3D visualization of the RoboCupRescue model for the City of Kobe, Japan.

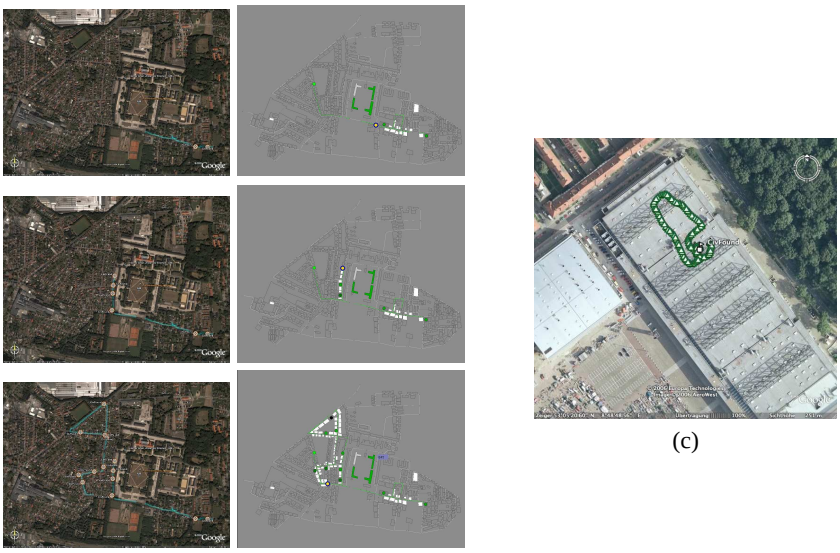
If the time needed for rescuing civilians and the chance of survival of civilians is roughly predictable, one can estimate the overall number of survivors by summing up the necessary time for each single rescue and by determining the overall number of survivors within the total time. For each rescue sequence $S = \langle t_1, t_2, ..., t_n \rangle$ of n rescue targets, a utility value $U(S)$ that is equal to the number of civilians that are expected to survive is calculated. Unfortunately, an exhaustive search over all $n!$ possible rescue sequences is intractable. A good heuristic solution is to sort the list of targets according to the time necessary to reach and rescue them and to subsequently rescue targets from the top of the list. However, this might lead to inferior solutions. A better method could be the so-called *Hungarian Method* [12], which optimizes the costs for assigning n workers to m tasks in $O\left(mn^2\right)$. The method requires that the time needed until a task is finished does not influence the overall outcome. However, this is not the case for a rescue task, since a victim will die if rescued too late. Hence, we decided to utilize a Genetic Algorithm [7] (GA) for the optimization of sequences and to utilize it for continuously improving the rescue sequence executed by the ambulance teams.

The GA is initialized with heuristic solutions, for example, solutions that *greedily* prefer targets that can be rescued within a short time or urgent targets that have only little chance of survival. The fitness function of solutions is set equal to the sequence utility $U(S)$. In order to guarantee that solutions in the genetic pool are at least as good as the heuristic solutions, the so-called *elitism* mechanism, which forces the permanent existence of the best found solution in the pool, has been used. Furthermore, we utilized a simple one-point-crossover strategy, a uniform mutation probability of $p \approx 1/n$, and

a population size of 10. Within each minute, approximately $300,000$ solutions can be calculated on a *1.0 GHz Pentium4 computer*.

We tested the GA-based sequence optimization on different city maps in the simulation and compared the result with a greedy strategy. In each of the tested environments, sequence optimization improved the performance of the rescue team. More information on the results can be found in [10]. One important property of our implementation is that it can be considered as an *anytime algorithm*: The method provides at least a solution as good as the greedy solution, but also a better one, depending on the given amount of time.

4 Experiments



(a) (b) (c)

Fig. 3. Successive integration of data reported by a test person equipped with a wearable device. (a) The real trajectory and observations of victims plotted with GoogleEarth (victims are labeled with "civFound"). (b) The same data integrated into the rescue system (green roads are known to be passable, white buildings are known as explored, and green dots indicate observed victims). (c) The result of an indoor tracking experiment performed at Robocup 2006, visualized in Google Earth. The event took place in the exhibition center Messe Bremen which can be seen in this aerial photography.

The system has preliminary been tested by successively integrating data received from a test person. The test person equipped with the first test device described in Section 2 walked several tracks within a district of the City of Bremen (see Figure 3). During the experiment, the mobile device continuously transmitted the trajectory of the

test person. Additionally, the test person reported *victim found* waypoints after having visual contact with a victim. Note that victim waypoints were arbitrarily selected, since fortunately there were no real victims found in Bremen.

In order to integrate the data into the rescue system, the received data, encoded by the extended GPX protocol that represents location by latitude and longitude, has to be converted into a grid-based representation. We utilized the Universal Transverse Mercator (UTM) [17] projection system, which provides a zone for any location on the surface of the Earth, whereas coordinates are described relatively to this zone. By calibrating maps from the rescue system to the point of origin of the UTM coordinate system, locations from the GPS device can directly be mapped. Figure 3(b) shows the successive integration of the received data into the rescue system and Figure 3(a) displays the same data plotted by *GoogleEarth*. Note that GPX data can without any conversion be directly processed by GoogleEarth.

In a further test, a person wearing the emergency-response wearable system was tracked while walking indoors and the trajectory data and victim identification information was visualized in realtime. The test took place at Robocup 2006 in front of a scientific audience. Indoor localization was performed by using an implementation of pedestrian dead-reckoning and GPS data fusion [5].

5 Conclusion

We have demonstrated a system which is generally capable of integrating trajectories and observations of many mobile devices into a consistent world model. As shown by the results of the simulation, the consistent world model allows the system to coordinate exploration by directing teams to globally unexplored regions as well as to optimize their plans based on the sampled connectivity of roads. To apply this global coordination also outside the simulation, i.e. to send the road graph and mission commands back to the wearable devices of real rescue teams in the field, will be a part of the future work.

Through augmentation with additional sensing techniques such as pedestrian dead-reckoning, we can obtain sufficient GPS-like accuracy even in poor signal reception conditions such an in buildings or urban corridors. However, when used for an extended period, pure PDR accumulates position errors that lead to unreliable positioning. We are therefore studying techniques to augment PDR-based indoor localization to limit error accumulation.

Finally, we plan to use our system to record a training event of emergency responders and compare the instructions given by our system with commands given by human incident commanders. This hopefully will lead to a better understanding of advantages and limitations of our system and to new information that can be taken into account for designing the robocup rescue simulation software in order to create more realistic challenges for the robocup community.

References

1. Mobile rescue prototype code. Available on: http://matrix.wearlab.de/MobileRescue/. release October, 2006.

2. Resq freiburg 2004 source code. Available on: http://gkiweb.informatik.uni-freiburg.de/ rescue/sim04/source/resq.tgz. release September, 2004.
3. The ieee standard 802.15.1 : Wireless personal area network standard based on the bluetooth v1.1 foundation specifications, 2002.
4. A. Birk, S. Carpin, and H. Kenn. The IUB 2003 Rescue Robot Team. In *RoboCup 2003: Robot Soccer World Cup VII*, 2003.
5. M. Dipplod. Personal Dead Reckoning with Acceleratometers. In O. Herzog, H. Kenn, M. Lawo, P. Lukowicz, and G. Troester, editors, *IFAWC 3rd international forum on applied wearable computing*, 2006.
6. M. Hazas, J. Scott, and J. Krumm. Location-aware computing comes of age. *IEEE Computer*, 37(2):95–97, February 2004.
7. J. H. Holland. *Adaption in Natural and Artificial Systems*. University of Michigan Press, 1975.
8. H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. RoboCup Rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE Conf. on Man, Systems, and Cybernetics(SMC-99)*, 1999.
9. A. Kleiner, N. Behrens, and H. Kenn. Wearable computing meets multiagent systems: a real-world interface for the RoboCupRescue simulation platform. In N. Jennings, M. Tambe, T. Ishida, and S. Ramchurn, editors, *First International Workshop on Agent Technology for Disaster Management at AAMAS06*, 2006.
10. A. Kleiner, M. Brenner, T. Braeuer, C. Dornhege, M. Goebelbecker, M. Luber, J. Prediger, J. Stueckler, and B. Nebel. Successful search and rescue in simulated disaster areas. In *In Proc. of the International RoboCup Symposium '05*, 2005.
11. A. Kleiner, C. Dornhege, R. Kuemmerle, M. Ruhnke, B. Steder, B. Nebel, P. Doherty, M. Wzorek, P. Rudol, G. Conte, S. Durante, and D. Lundstrom. Robocuprescue - robot league team rescuerobots freiburg (germany). In *RoboCup 2006: Robot Soccer World Cup X*, 2006.
12. H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quaterly*, 2:83–97, 1955.
13. Q. Ladetto, B. Merminod, P. Terrirt, and Y. Schutz. On foot navigation: When gps alone is not enough. *Journal of Navigation*, 53(02):279–285, Mai 2000.
14. M. Lawo, H. Witt, H. Kenn, T. Nicolai, and R. Leibrand. A Glove for Seamless Computer Interaction - Understand the WINSPECT. In H. Kenn, U. Glotzbach, and O. Herzog, editors, *The Smart Glove Workshop*, number 33, 2006.
15. L. Liao, D. Fox, and H. A. Kautz. Learning and inferring transportation routines. In *AAAI*, pages 348–353, 2004.
16. T. Nicolai, T. Sindt, H. Kenn, and H. Witt. Case study of wearable computing for aircraft maintenance. In *Otthein Herzog, Michael Lawo, Paul Lukowicz and Julian Randall (eds.), 2nd International Forum on Applied Wearable Computing (IFAWC)*, pages 97–110,. VDE Verlag, March 2005.
17. J. P. Snyder. *Map Projections - A Working Manual. U.S. Geological Survey Professional Paper 1395*. United States Government Printing Office, Washington, D.C., 1987.
18. T. Starner. The challenges of wearable computing: Part 1. *IEEE Micro*, 21(4):44–52, 2001.
19. T. Starner. The challenges of wearable computing: Part 2. *IEEE Micro*, 21(4):54–67, 2001.
20. TopoGrafix. Gpx - the gps exchange format. Available on: http://www.topografix.com/gpx.asp. release August, 9th 2004.