Wearable Computing meets Multiagent Systems: A real-word interface for the RoboCupRescue simulation platform

Alexander Kleiner
Institut für Informatik
Universität Freiburg
79110 Freiburg, Germany
kleiner@informatik.uni-freiburg.de

Nils Behrens
Technologie-Zentrum Informatik
Universität Bremen
28359 Bremen, Germany
psi@tzi.de

Holger Kenn
Technologie-Zentrum Informatik
Universität Bremen
28359 Bremen, Germany
kenn@tzi.de

ABSTRACT

One big challenge in disaster response is to get an overview over the degree of damage and to provide this information, together with optimized plans for rescue missions, back to teams in the field. Collapsing infrastructure, limited visibility due to smoke and dust, and overloaded communication lines make it nearly impossible for rescue teams to report the total situation consistently. This problem can only be solved by efficiently integrating data of *many* observers into a single consistent view. A Global Positioning System (GPS) device in conjunction with a communication device, and sensors or simple input methods for reporting observations, offer a realistic chance to solve the data integration problem.

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. 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).

Keywords

Wearable Computing, GPS, Multi Agent Systems, MAS, USAR, GIS, RoboCupRescue

1. INTRODUCTION

One big challenge in disaster response is to get an overview over the degree of damage and to provide this information, together with optimized plans for rescue missions, back to teams in the field. Collapsing infrastructure, limited visibility due to smoke and dust, and overloaded communication lines make it nearly impossible for rescue teams to report the total situation consistently. Furthermore, they might be affected psychologically or physically by the situation itself and hence report unreliable information.

This problem can only be solved by efficiently integrating data of *many* observers into a single consistent view. A Global Positioning System (GPS) device in conjunction with a communication device, and sensors or simple input methods for reporting observations, offer a realistic chance to solve the data integration problem. Furthermore, an integrated world model of the disaster allows to apply solutions from the rich set of AI methods developed by the Multi-Agent Systems (MAS) community.

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 [21] 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).

RoboCupRescue simulation aims at simulating large-scale disasters and exploring new ways for the autonomous coordination of rescue teams [8] (see Figure 1). These goals 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. 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 solutions presented in this paper are based on the OpenSource agent software [1], which was developed by the *ResQ Freiburg 2004* team [9], the winner of RoboCup 2004. The advantage of interfacing RoboCupRescue simulation with wearable computing is twofold: First, data collected from a real interface allows to improve the disaster simulation towards disaster reality. Second, agent software developed within RoboCupRescue might be advantageous in real disasters, since it can be tested in many sim-

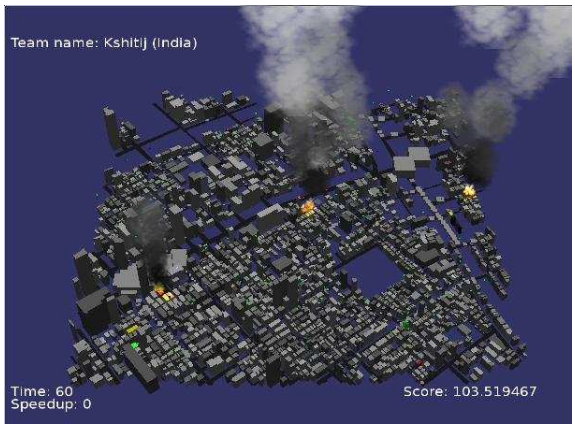


Figure 1: A 3D visualization of the RoboCupRescue model for the City of Kobe, Japan.

ulated disaster situations and can also directly be compared to other approaches.

Nourbakhsh and colleagues utilized the MAS *Retsina* for mixing real-world and simulation-based testing in the context of Urban Search and Rescue [15]. Schurr and colleagues [17] introduced the *DEFACTO* system, which enables agent-human cooperation and has been evaluated in the fire-fighting domain with the RoboCupRescue simulation package. Liao and colleagues presented a system that is capable of recognizing the mode of transportation, i.e., by bus or by car, and predicting common travel destinations, such as the office location or home location, from data sampled by a GPS device [12].

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 propose preliminary experiments from integrating data into RoboCupRescue from a real device and conclude in Section 5.

2. INTERFACING REAL RESCUE

2.1 Requirement analysis

In wearable computing, one main goal is to build devices that support a user in the primary task with little or no obstruction. Apart from the usual challenges of wearable computing [20, 19], 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 perceptions and needs feedback from the system [1]. Furthermore, the user needs to receive 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 GPS-based localization device. 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 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 based on additional techniques such as the ones stated in [11], its reliability and accuracy can be sufficiently improved. Particularly the coexistence of a GPS device with an Internet connection allows to utilize Internet-based Differential GPS, which leads to a positioning accuracy of decimeters [2].

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 spillings etc. The device has to remain operable under such conditions, and moreover has to provide alternative means of input and output under conditions that affect human sensing and action abilities. As these requirements are quite complex, we decided to design and implement a preliminary test system and a final system. The components of the two systems and their interconnections can be found in Figure 4.

2.2 A preliminary test system

In order to analyze the properties of the communication and localization systems, a preliminary test system has been implemented, for which two requirements have been dropped, the design for harsh envionmental conditions and the ability to use 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

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

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 [21] standard for GPS locations. Among other things, the protocol defines data structures for *tracks* and *waypoints*. A track is a sequence of locations with time stamps that has been visited with the GPS device. A waypoint describes a single location of interest, e.g., the peak of a mountain. We extended the protocol in order to augment waypoint descriptions with information specific to disaster situations. These extensions allow rescue teams to report the waypoint-relative locations of road blockades, building fires, and victims. Currently, the wearable device automatically sends the user's trajectory to the command center, whereas perceptions can manually be entered. A detailed description of the protocol extension can be found in Appendix A.

2.3 Designing the full emergency response wearable system

In order to fulfill the additional requirements for robustness and user interface, the full system will be based on additional hard- and software. The system uses a wearable CPU core, the so-called *qbic belt-worn computer* [4] (see Figure 3 (a)). It is based on a ARM CPU running the Linux operating system, has a bluetooth interface, and can be extended via USB and RS232 interfaces. The wearable CPU core runs the main application program. For localization, the same mobile GPS receiver as in the test system is used, but can be replaced by a non-bluetooth serial device for increased reliability. For communication, the system can use multiple communication channels whose already used GSM cell phone can be one of those [2].

As already stated, the design of the user interface is a crucial one for this application. Therefore, we envision a user input device integrated in the clothing of the user, e.g., an arm-mounted textile keyboard [13] and a wireless link of the keyboard to the belt computer. Such an interface has already been designed for other applications such as aircraft cabin operation [14] (see Figure 2). Due to the harsh environmental conditions, we plan two independent output devices for information output and user feedback. A bluetooth headset device provides audible feedback for user input, and a text-to-speech engine provides audible text output.

Figure 2: A textile keyboard for aircraft cabin operation.

The second output device is a head-mounted display that can be integrated into existing emergency response gear such as firefighter helmets and masks (see Figure 3(b)). In applications where headgear is not commonly used, the output can also be provided through a body-worn display device.

[2]As we assumed IP-based connectivity, flexible infrastructure-independent transport mechanisms such as MobileIP [16] can be used to improve reliability over multiple independent and redundant communication links.

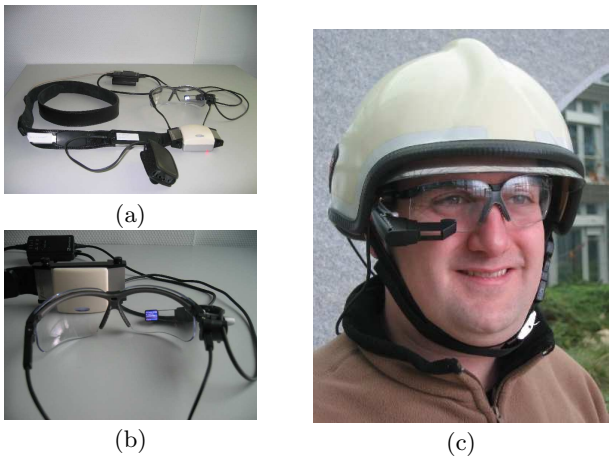


Figure 3: The *qbic belt-worn computer*: (a) The belt with CPU. (b) The head-mounted display. (c) Both worn by the test person.

The application software driving the user interface is based on the so-called *WUI toolkit* [22], which uses an abstract description to define user interface semantics independent of the input and output devices used. The application code is therefore independent of the devices available in a particular instance of an implementation, i.e., with or without head-mounted display. The WUI toolkit can also take context information into account, such as the user's current situation, in order to decide on which device and in what form output and input are provided.

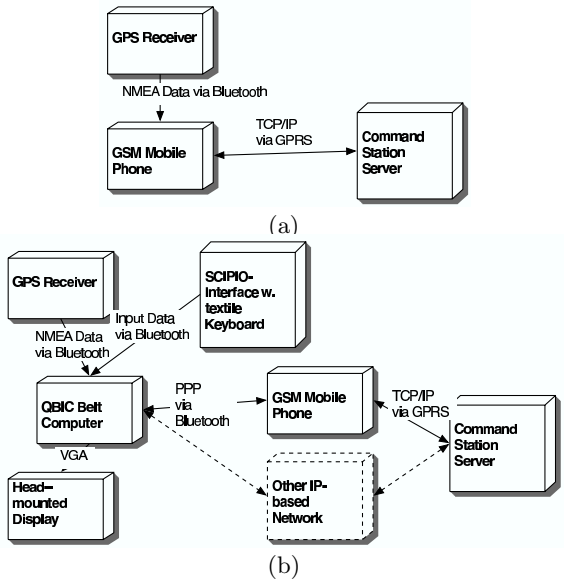


Figure 4: System diagrams: (a) test system based on a GSM phone (b) full system design based on a belt-worn wearable computer

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.

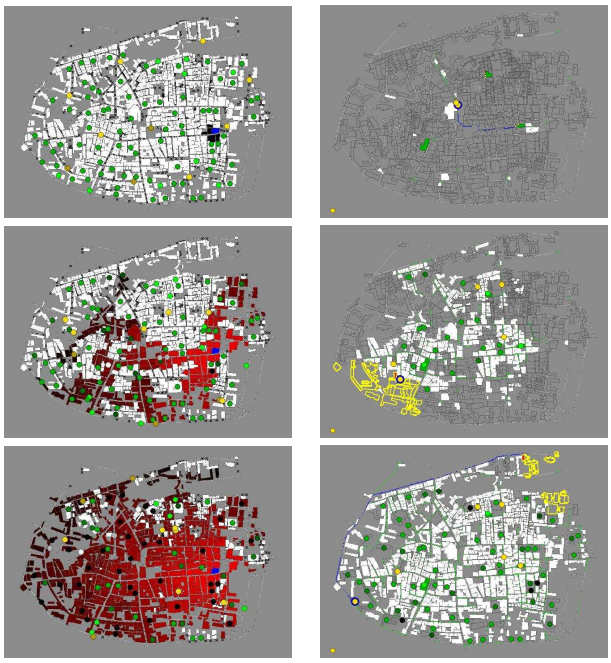
Based on an initial road map and on the information on road blockage and the autonomously collected data on trajectories traveled by the agents, the current system builds up a connectivity graph indicating the connectivity of locations. The connectivity graph between a single location and all other locations 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 sequence in Figure 5(a) shows the continuous update of the connectivity graph for a building within the simulated City of Foligno. Note that the graph has to be revised if new information on the connectivity between two locations is available, e.g if a new blockage has been detected or an old blockage has been removed.

The search for victims of many rescue teams can only be coordinated efficiently if the rescue teams share information on the 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 to distribute rescue teams efficiently among unexplored and reachable locations. The sequence in Figure 5(b) shows an agent's increasing knowledge on the exploration status of the map over time. Victims (indicated by green dots) and explored buildings (indicated by white color) are jointly reported by all agents. Regions that are marked by a yellow border indicate exploration targets recommended by the command center to the agent.

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 are directly taken from the connectivity graph. Travel costs between two reachable targets are estimated by continuously averaging costs experienced by the agents [3].



(a) (b)

Figure 5: Online data integration of information reported by simulated agents: (a) The connectivity between the blue building and other locations increases over time due to removed blockades. White colored locations are unreachable, red colored locations are reachable. The brighter the red color, the better the location is reachable. (b) The agent's information on the explored roads and buildings (green roads are known to be passable, green and white buildings are known as explored). Regions marked with a yellow border are exploration targets recommended by the command center.

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 approximate travel time between any two locations by analyzing the GPS trajectories received from rescue teams in the field. Moreover, the sys-

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

tem can provide for different means of transport, e.g., 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 [12].

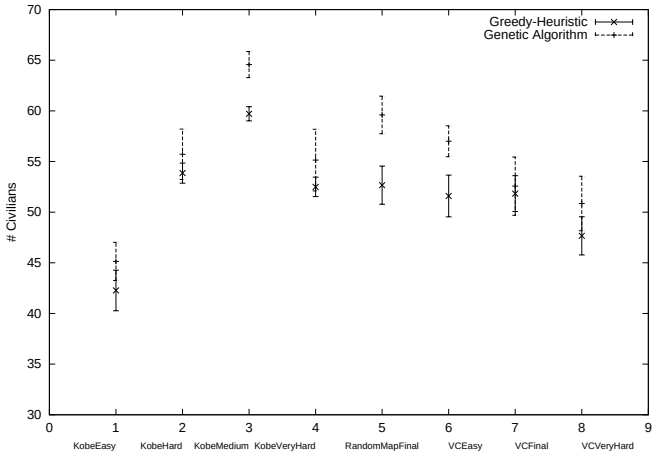


Figure 6: The number of civilian suvivors if applying a greedy rescue strategy and a GA optimized rescue strategy within simulated cities

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 $U(S)$ 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, as shown in Figure 6, this might lead to poor solutions. A better method could be the so-called *Hungarian Method* [10], 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. As can be seen in Figure 6, in each of the tested environments, sequence optimization improved the performance of the rescue team. One important property of our implementation is that it can be considered as an *anytime algorithm*: The method provides at least a solution that is as good as the greedy solution, but also a better one, depending on the given amount of time.

4. PRELIMINARY EXPERIMENTS

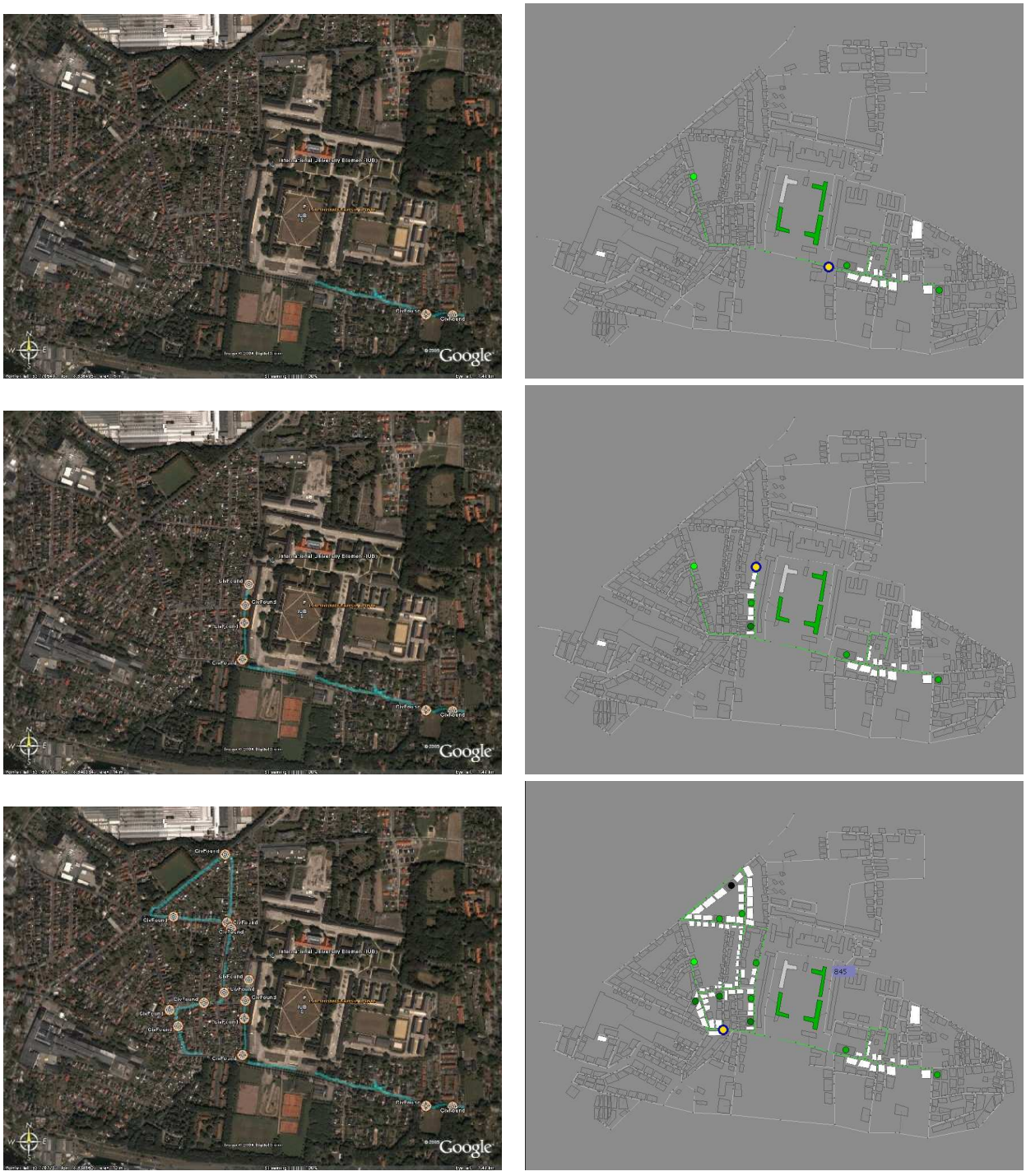
The system has preliminary been tested by successively integrating data received from a test person. The test person equipped with the test device described in Section 2 walked several tracks within a district of the City of Bremen (see Figure 7). During the experiment, the mobile device continuously 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 selected arbitrarily, since fortunately no victims were 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) [18] 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. In order to cope with erroneous data, we decided to simply ignore outliers, i.e. locations far from the track, that were detected based on assumptions made on the test person's maximal velocity. In the next version of the system it is planned to detect outliers based on the *mahanalobis distance* estimated by a Kalman Filter, likewise as dead reckoning methods used in the context of autonomous mobile robots. Figure 7(b) shows the successive integration of the received data into the rescue system and Figure 7(a) displays the same data plotted by *GoogleEarth*. Note that GPX data can be directly processed by GoogleEarth without any conversion.

5. CONCLUSION

We introduced the preliminary design of a wearable device which can be utilized for USAR. Furthermore we have demonstrated a system which is generally capable of integrating trajectories and observations from many of these wearable 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, and to optimize the sequence of rescuing victims. The application of this coordination also in real scenarios, 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 future work.

As we can see from our experiments, the accuracy of the GPS locations suffices for mapping trajectories on a given road graph. However, during a real disaster, a city's infrastructure might change completely, i.e., former roads might



(a) (b)

Figure 7: 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).

be impassable or disappear at all, and people search for new connections between places (e.g., off-road or even through buildings). Therefore, it is necessary that the system is capable of learning new connections between places and to modify the existing graph accordingly. Brüntrup and colleagues already studied the problem of map generation from GPS traces [5]. Our future work will particularly deal with the problem of learning from multiple noisy routes. We will extend the existing rescue system with the capability of adding new connections to the road graph and to augment these connections with the estimated travel time, sampled from the observed trajectories.

Furthermore we are investigating methods of visual odometry for estimating the trajectories of humans walking within buildings, or more general, in situations where no GPS localization is possible. We are confident that this odometry data together with partial GPS localization will suffice to integrate an accurate map of the disaster area, including routes leading through buildings and debris.

Finally, it would be interesting to compare the system with conventional methods that are used in emergency response nowadays. This could be achieved by comparing the efficiency of two groups of rescue teams exploring buildings within an unknown area, whereas one group is coordinated by conventional radio communication and the other group by our system via wearable devices.

6. REFERENCES

[1] Resq freiburg 2004 source code. Available on: `http://gkiweb.informatik.uni-freiburg.de/~rescue/sim04/source/resq.tgz`. release September, 2004.

[2] Satellitenpositionierungsdienst der deutschen landesvermessung sapos. Available on: `http://www.sapos.de/`.

[3] The ieee standard 802.15.1 : Wireless personal area network standard based on the bluetooth v1.1 foundation specifications, 2002.

[4] O. Amft, M. Lauffer, S. Ossevoort, F. Macaluso, P. Lukowicz, and G. Tröster. Design of the QBIC wearable computing platform. In *15th International Conference on Application-Specific Systems, Architectures and Processors (ASAP '04)*, Galveston, Texas, September 2004.

[5] R. Bruentrup, S. Edelkamp, S. Jabbar, and B. Scholz. Incremental map generation with gps traces. In *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Vienna, Austria, 2005.

[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, 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.

[10] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quaterly*, 2:83–97, 1955.

[11] 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.

[12] L. Liao, D. Fox, and H. A. Kautz. Learning and inferring transportation routines. In *AAAI*, pages 348–353, 2004.

[13] U. Möhring, S. Gimpel, A. Neudeck, W. Scheibner, and D. Zschenderlein. Conductive, sensorial and luminiscent features in textile structures. In *H. Kenn, U. Glotzbach, O. Herzog (eds.) : The Smart Glove Workshop*, TZI Report, 2005.

[14] 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.

[15] I. Nourbakhsh, K. Sycara, M. Koes, M. Yong, M. Lewis, and S. Burion. Human-robot teaming for search and rescue. *IEEE Pervasive Computing: Mobile and Ubiquitous Systems*, pages 72–78, January 2005.

[16] C. Perkins. Ip mobility support for ipv4. RFC, August 2002.

[17] N. Schurr, J. Marecki, P. Scerri, J. P. Lewi, and M. Tambe. The defacto system: Coordinating human-agent teams for the future of disaster response. *Programming Multiagent Systems*, 2005.

[18] J. P. Snyder. *Map Projections - A Working Manual. U.S. Geological Survey Professional Paper 1395.* United States Government Printing Office, Washington, D.C., 1987.

[19] T. Starner. The challenges of wearable computing: Part 1. *IEEE Micro*, 21(4):44–52, 2001.

[20] T. Starner. The challenges of wearable computing: Part 2. *IEEE Micro*, 21(4):54–67, 2001.

[21] TopoGrafix. Gpx - the gps exchange format. Available on: `http://www.topografix.com/gpx.asp`. release August, 9th 2004.

[22] H. Witt, T. Nicolai, and H. Kenn. Designing a wearable user interface for hands-free interaction ind maintenance applications. In *PerCom 2006 - Fourth Annual IEEE International Conference on Pervasive Computer and Communication*, 2006.

APPENDIX
A. COMMUNICATION PROTOCOL

```
<xsd:complexType name="RescueWaypoint">
<xsd:annotation><xsd:documentation>
    This type describes an extension of GPX 1.1 waypoints.
    Waypoints within the disaster area can be augmented
    with additional information, such as observations of fires,
    blockades and victims.
</xsd:documentation></xsd:annotation>
    <xsd:sequence>
       <xsd:element name="Agent"
                    type="RescueAgent_t" minOccurs="0" maxOccurs="1" />
       <xsd:element name="Fire"
                    type="RescueFire_t" minOccurs="0" maxOccurs="unbounded" />
       <xsd:element name="Blockade"
                    type="RescueBlockade_t" minOccurs="0" maxOccurs="unbounded" />
       <xsd:element name="VictimSoundEvidence"
                   type="RescueVictimSoundEvidence_t"  minOccurs="0" maxOccurs="unbounded" />
       <xsd:element name="Victim"
                    type="RescueVictim_t" minOccurs="0" maxOccurs="unbounded" />
       <xsd:element name="Exploration"
                    type="RescueExploration_t" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>


<xsd:complexType name="RescueVictim_t">
<xsd:annotation><xsd:documentation>
    This type describes information on a victim
    relatively to the waypoint.
</xsd:documentation></xsd:annotation>
    <xsd:sequence>
        <xsd:element name="VictimDescription"
                    type="xsd:string"   "minOccurs="0" maxOccurs="1"/>
        <xsd:element name="VictimSurvivalTime"
                    type="xsd:integer"   "minOccurs="0" maxOccurs="1"/>
        <xsd:element name="VictimRescueTime"
                    type="xsd:integer"   "minOccurs="0" maxOccurs="1"/>
        <xsd:element name="VictimProximity"
                    type="Meters_t" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="VictimBearing"
                    type="Degree_t" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="VictimDepth"
                    type="Meters_t" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>


<xsd:complexType name="RescueFire_t">
<xsd:annotation><xsd:documentation>
    This type describes the observation of fire
    relatively to the waypoint.
</xsd:documentation></xsd:annotation>
    <xsd:sequence>
        <xsd:element name="FireDescription"
                    type="xsd:string"   "minOccurs="0" maxOccurs="1"/>
        <xsd:element name="FireProximity"
                    type="Meters_t" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="FireBearing"
                    type="Degree_t" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>


<xsd:complexType name="RescueBlockage_t">
<xsd:annotation><xsd:documentation>
    This type describes detected road blockages
    relatively to the waypoint.
</xsd:documentation></xsd:annotation>
    <xsd:sequence>
        <xsd:element name="BlockageDescription"
                    type="xsd:string"   "minOccurs="0" maxOccurs="1"/>
        <xsd:element name="BlockageProximity"
                    type="Meters_t" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="BlockageBearing"
                    type="Degree_t" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>


<xsd:complexType name="RescueVictimSoundEvidence_t">
<xsd:annotation><xsd:documentation>
    This type describes evidence on hearing a victim
    relatively to the waypoint.
</xsd:documentation></xsd:annotation>
    <xsd:sequence>
        <xsd:element name="VictimEvidenceRadius"
                    type="Meters_t" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>


<xsd:complexType name="RescueExploration_t">
<xsd:annotation><xsd:documentation>
    This type describes the area that has been exploration
    around the waypoint.
</xsd:documentation></xsd:annotation>
    <xsd:sequence>
        <xsd:element name="ExploredRadius"
                    type="Meters_t" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="RescueAgent_t">
<xsd:annotation><xsd:documentation>
    This type describes the observant agent.
</xsd:documentation></xsd:annotation>
    <xsd:sequence>
        <xsd:element name="AgentName"
                    type="xsd:string"   "minOccurs="0" maxOccurs="1"/>
        <xsd:element name="AgentTeam"
                    type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>



<xsd:simpleType name="Meters_t">
  <xsd:annotation><xsd:documentation>
    This type contains a distance value measured in meters.
  </xsd:documentation></xsd:annotation>
  <xsd:restriction base="xsd:integer"/>
</xsd:simpleType>

<xsd:simpleType name="Degree_t">
  <xsd:annotation><xsd:documentation>
    This type contains a bearing value measured in degree.
  </xsd:documentation></xsd:annotation>
  <xsd:restriction base="xsd:integer"/>
</xsd:simpleType>
```