RoboCup 2009 - RoboCup Rescue
Team resko@UniKoblenz (Germany)

Johannes Pellenz, David Beckmann, Denis Dillenberger, Christian Fuchs, David Gossow, Susanne Maur, Kevin Read, Bernhard Reinert, Dietrich Paulus

Active Vision Group
University of Koblenz-Landau
Universitätsstr. 1
56070 Koblenz, Germany
`pellenz@uni-koblenz.de`
`http://robots.uni-koblenz.de`

Abstract. This paper describes the approach of the team resko@UniKoblenz for the RoboCup Rescue competition 2009. Our mobile system *Robbie* is based on a MobileRobots Pioneer 3 AT. It is equipped with a four-wheel drive and sonar sensors in the front and in the back. On this platform, an aluminum rack is installed where additional sensors are attached: three color cameras (including two high resolution Sony FireWire cameras), an actively contolled Hokuyo URG-04LX laser range finder (LRF), a thermal camera and an inclination sensor. The robot can be operated in autonomous and in teleoperated mode. The map building is done by the computer automatically by merging the collected LRF data with the odometry information using a *Hyper Particle Filter* (a particle filter of particle filters). The automatic victim detection is based on the obtained thermal and color camera images.
Robbie was developed and improved at the University of Koblenz-Landau (Germany) during the last four years as a PhD projekct and in six practical courses. The robot was used by the team resko@UniKoblenz at the RoboCup German Open 2007, 2008 and 2009 in Hannover and at the RoboCup World Championship 2007 in Atlanta (GA, USA) and 2008 in Suzhou (China). The team achieved the "Best in Class Autonomy Award" in all five competitions.

Introduction

The team *resko@UniKoblenz* is a group of researchers and students from the University of Koblenz-Landau, Germany. In 2009, the robot *Robbie* competes in two RoboCup leagues: In the RoboCup Rescue and in the RoboCup@Home league.

1 Team Members and their Contributions

Our team consists of two groups: team *resko@UniKoblenz* and team *homer@UniKoblenz*:

- The team *resko@UniKoblenz* (with focus on the RoboCup Rescue league)
  - Johannes Pellenz: team leader, scientific advisor
  - Kevin Read: technical design, programming
  - Bernhard Reinert: head of project team rescue, programming
  - Christian Fuchs: quality assurance, programming
  - David Beckmann: infrastructure, programming
  - Susanne Maur: scan matching, SLAM
  - Denis Dillenberger: 3D terrain classification
- The team *homer@UniKoblenz* (with focus on the RoboCup@Home league) supports the team *resko@UniKoblenz*:
  - David Gossow: team leader
  - Peter Decker: scientific advisor
  - Marc Ahrends: head of project team @home, programming
  - Susanne Thierfelder: public relations, programming
  - Sönke Greve: hardware assistant, programming
  - Christian Winkens: social events, programming
  - Viktor Seib: multi-robot coordination

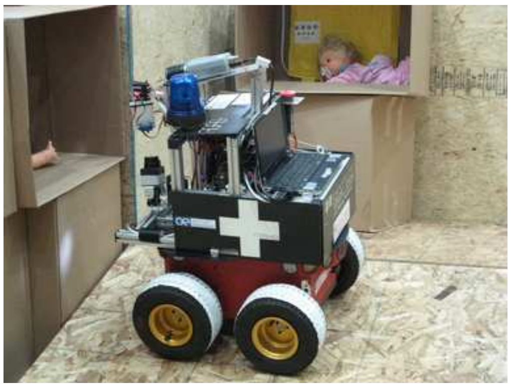
2 Control method and Human-Robot Interface (HRI)

Our robot navigates through the yellow part of the RoboCup Rescue Arena autonomously. The robot is started by the operator from the operator station; afterwards the operator can monitor the robot and the sensor data, which is sent to the operator station via WLAN. In case of an emergency, the operator can stop the autonomous mode and teleoperate the robot. If the connection to the operator station gets lost, the robot continues its mission and the victim search. As soon as the connection is re-established, the victim verification on the operator station is triggered.

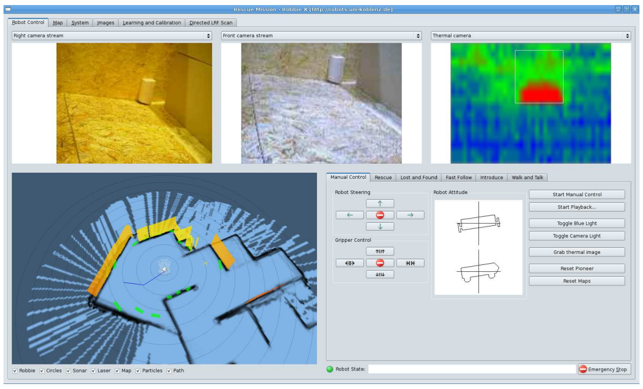
The graphical user interface (GUI) of the operator laptop is shown in Fig. 1(b). The human-robot interface is implemented in Qt4, with OpenGL widgets for visualizing easy-to-understand 3D views of the sensor data. The operator sees all sensor measurements, the color images and the last thermal image on a single screen. The current laser and sonar range readings are merged into the already learned map in the lower left part of the windows. In the upper part of the window, the images of the color cameras and the thermal image are shown. The operator can mark the position of victims in the map manually. In the autonomous mode, a dialog pops up when the robot has found a victim. This (potential) victim can then be verified or declined; the position is automatically marked in the map. Additionally, the robot has a blue flashing light that indicates the detection of a victim. This shows the audience that the victim search was successful.

3 Map generation/printing

While exploring the environment, the robot automatically generates a map of the building. The map is based on the laser scans and the odometry information. Fig. 3 shows the result of the mapping process.

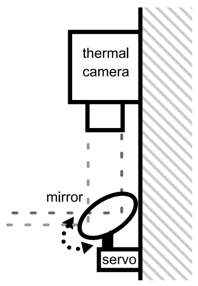


(a) Our Pioneer 3 AT-based robot *Robbie*.



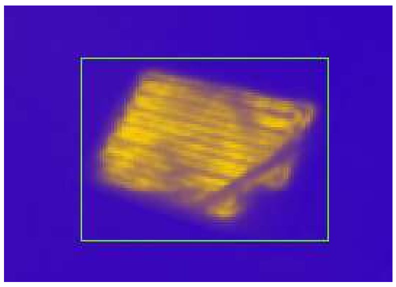
(b) User interface of the operator station.

Fig. 1. The mobile system *Robbie* and the user interface.



(a) Thermal camera and mirror

(b) ThermoVision Micron/A10



(c) Thermal image

Fig. 2. Thermal camera mounting (a) and image of the thermal camera (b).

The data structure to store the map is a single occupancy map. In the context of RoboCup Rescue, the grid usually has a size of 800 × 800 cells, which represents a map of an area of 40 × 40 meters with a grid cell size of 50 × 50 mm. The grid is stored in two planes: one plane counts how often a cell was "seen" by a laser beam. This value is increased either if the laser beam measured the cell as free or as occupied. A second plane stores the information how often a cell was seen as occupied. By dividing these two planes, the occupancy probability for a cell c_i is calculated as the following ratio:

$$p_{\mathrm{occ}}(c_i) = \frac{\mathrm{count_{occ}(c_i)}}{\mathrm{count_{seen}(c_i)}} \tag{1}$$

To solve the SLAM problem, we use a particle filter [IB98] with about 1,000 particles [Pel08]. Each particle represents a hypothesis for the pose of the robot in 2D-space: $(x, y, \Theta)^T$. Fig. 4 illustrates the robot pose hypotheses,

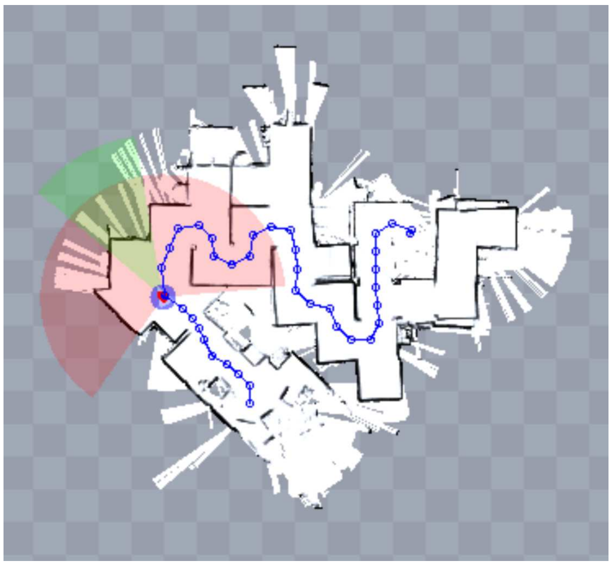
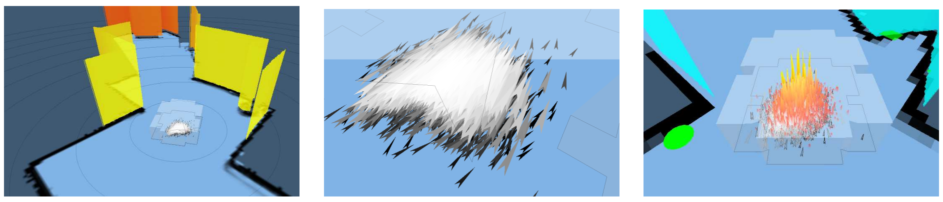


Fig. 3. Map of the Yellow Arena. The blue line shows a planned path back to an unexplored room in the upper right corner (generated at the RoboCup German Open 2008 during the Mapping Mission).

represented by particles.



(a) Particles (in the center of the image). (b) Detailled view of the particles. (c) Weights of the particles.

Fig. 4. Robot poses, represented by particles. The poses are visualized in real time in the user interface.

The algorithm of the particle filter includes the following steps: *resample*, *drift*, *measure*, and *normalize*. The result is the most likely pose of the robot when the laser scan was taken. This pose is then used for the *map update*. **NEW in 2009:** An ICP (Iterative Closest Point) scan matcher with different metrics improves the pose estimation (which is the input for the particle filter) that is based on the odometry of the robot.
Resample: Depending on the weight of the particles (generated by the last mea-

surement step), the probability distribution is adjusted. The resampling step is only done if the robot moved a certain distance (at least 20 mm) or turned (at least 5°) since the last mapping step.

Drift: During the drift step, the particles are moved depending on the odometry data and the result of the scan matching. It also models noise due to sensor inaccuracies and wheel slippage (see [HBFT03]).

Measure: During the measurement step, new weights for each particle are calculated using the current laser scan. To weighten a particular particle, the endpoints of the laser beams are calculated, using the robot pose stored in this particle. **NEW in 2009:** We use a subsampling of the laser data: Only laser beam endpoints, that are at least 10 cm away from the last considered endpoint are used for the weighting function. This speeds up the measurement function significantly.

Normalize: The assigned weight of a particle is a sum of occupancy probability values. During the normalization step, the weight of each particle is divided by the sum of all particle weights.

Map update: The average pose of the top 5% particles with the highest weight is assumed to be the location of the robot. Using this pose, the current laser range scan is added to the global occupancy map by constructing a local map and "stamping" it into the global map, incrementing the counts for *seen* and *occupied* cells (cf. equation 1).

NEW in 2009: We use a Hyper Particle Filter (HPF) [PP09] – a Particle Filter of Particle Filters – for solving the SLAM problem. Each particle of the HPF contains a standard localization Particle Filter (with a map and a set of particles, that model the belief of the robot pose in this particular map). To measure the weight of a particle in the HPF, we developed two map quality measures that can be calculated automatically and do not rely on a ground truth map: The first map quality measure determines the contrast of the occupancy map. If the map has a high contrast, it is likely that the pose of the robot was always determined correctly before the map was updated, which finally leads to an overall consistent map. The second map quality measure determines the distribution of the orientation of wall pixels calculated by the Sobel operator. Using the model of a rectangular overall structure, slight but systematic errors in the map can be detected. Using the two measures, broken maps can automatically be detected. The corresponding particle is then more likely to be replaced by a particle with a better map within the HPF. Using this technique, broken maps are very unlikely. Also, the problem of loop closing is solved (see Fig. 6).

The resulting map is displayed in realtime on the operator station. After the mission, it can be printed out and handed to the responders. To make a step further to standardization, we can also export our map as a GeoTIFF file, which e. g. can be copied on a USB stick.

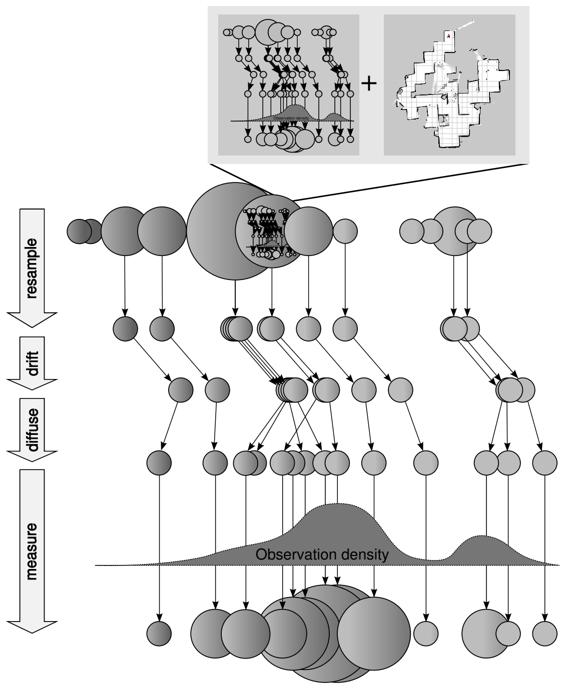
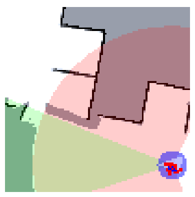
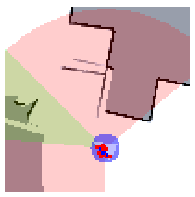


Fig. 5. The principle of the Hyper Particle Filter: The particles of the Hyper Particle Filter are standard SLAM Particle Filters (with a map and particles that represent robot poses). For the measurement step, the quality of each map is evaluated.


(a) Situation right before the loop closing.


(b) The robot scans areas that it has visited before; first errors in the map are visible.

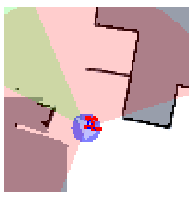

(c) The HPF closes the loop by switching to a consistent map.

Fig. 6. Successful loop closing using a Hyper Particle Filter (HPF) with 30 particles.

4 Sensors for Navigation and Localization

4.1 Odometry data

The Pioneer robot has a built-in system to report the robot pose $(x, y, \Theta)^T$, based on the readings from the wheel encoders. However, our experiments have shown that the data is quite unreliable, and the errors sum up over time. Therefore this data can only be used to make a rough guess about the robot location and orientation.

4.2 Sonar sensors

Our Pioneer 3 AT robot platform has two sonar rings (one scanning the front and one scanning the back) with eight sonar sensors each. In total they cover 360°, but with larger gaps between the sensors. The sonar sensors are used for collision avoidance and their data is displayed in the GUI to make remote controlling easier. **NEW in 2009:** The data is used to generate a sonar echo map in addition to the laser map.

4.3 Laser scanner

The Hokuyo URG-04LX laser scanner generates 240° scans that measure the distance (**NEW in 2009:** up to 5.60 meters) of the closest objects. The resolution of the laser scanner is 0.36°. The operator interface shows these scans with respect to the robot's position. With this information the operator is able to estimate distances to obstacles better than with video images only. The scanner is attached to the front of the robot as shown in Fig. 8(a). Detected obstacles (such as walls) are continuously integrated into the map.

5 Sensors for Victim Identification

5.1 Vision System

Robbie has a vision system with 3 cameras mounted at the front. Two Sony cameras (DFW-X700 and DFW-X710) are adjusted along the robot's moving direction. In teleoperated mode, the operator uses the images of these cameras to detect victims visually. A ring of LEDs allows to search also in dark areas (such as the interior of boxes). In addition to the high resolution vision system we have a single webcam mounted at the robot's front at a very low level. We use this camera to recognize small obstacles that are right in front of the robot. This camera is also useful to detect victims that are placed in boxes on floor level.

5.2 Thermal Sensors

NEW in 2009: We replaced our self-made thermal camera with a ThermoVision Micron/A10 thermal camera. It has a 164×128 pixel thermal sensor to measure temperature values in a range from 0°C to 40°C. The sensor provides continuous thermal images, so that the robot no longer has to stop in order to generate a thermal scan. The camera is attached to the robot facing downward (see Fig. 2(a)), but the thermal radiation is reflected to the camera by a mirror. This mirror is mounted on a servo motor, such that the pan angle can be adjusted. This construction allows to generate a 200° thermal scan of the robots environment without the need to move the robot. We use the camera to detect victims autonomously by their body heat. For the visualization in the GUI the image is colored depending on the temperature values. Values that are considerably above the room temperature indicate a victim.

6 Robot Locomotion

The Pioneer 3 AT is a robot equipped with four air-filled tires. They are driven by 4 DC motors (12 volts). We control them by using the programming library (called ARIA) provided by the manufacturer.[1] Apart from the autonomous operation, we implemented a gamepad and a keyboard control.

7 Other Mechanisms

7.1 Active Adjustment of the Laser Range Finder

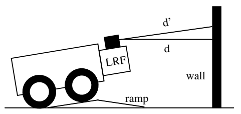
Rescue robots are usually used in unstructured environments with uneven floors and piles of debris. Therefore the sensor readings might be spurious if the sensors are rigidly coupled to the robot. The influence of uneven floor are simulated at the RoboCup since the German Open 2007 using 10° pitch/roll ramps. These ramps influenced the LRF data in three different ways:

1. The measured distance to an obstacle slightly changes by the factor of $\frac{1}{\cos\alpha}$ compared to the real distance, where α is the pitch angle of the robot relative to the ground.
2. The obstacle is not detected at all, since the sensor pointed above the object.
3. "Phantom walls" were detected, since the sensor detected the next pitch ramp as an obstacle.

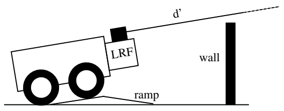
These problems are illustrated in Fig. 7: While 7(a) usually adds only a small error (about 1.5% at 10°), 7(b) and 7(c) result in inaccurate occupancy maps. Case 7(c) can cause the robot to avoid a free path, because the map blocks its way with a phantom obstacle.

Therefore, we adjust the orientation of the LRF based on the readings from a gravity sensor [Pel07]. The sensor measures the current orientation of the robot, and the orientation of the LRF is corrected so that it always scans horizontally (see Fig. 8(a)). On our mobile system, we use a cheap dual-axis (2D), up to 2 g accelerometer. The correction information is sent to a microcontroller which controls two servo motors that adjust the orientation of the LRF. In Fig. 8, a map based on data of a fixed LRF (Fig. 8(b)) and a map based on data of an adjusted LRF (Fig. 8(c)) are shown. The latter map shows a higher accuracy and much less artefacts.

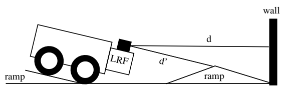
[1] `http://www.activrobots.com/SOFTWARE/aria.html`



(a) LRF measures the wrong distance

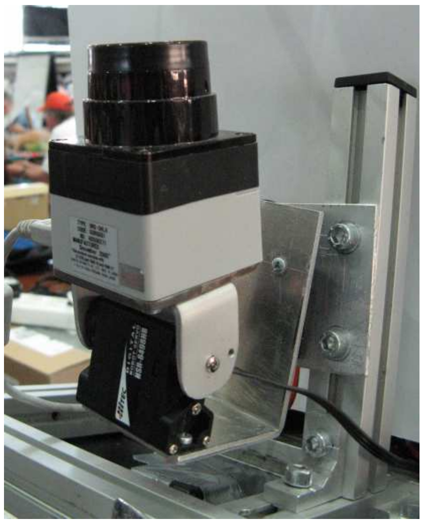


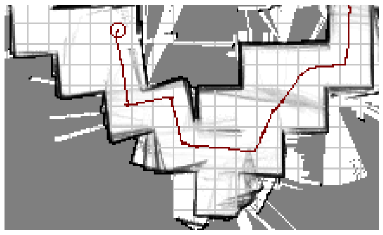
(b) LRF misses the wall



(c) LRF detects a phantom wall

Fig. 7. Correct (d) and measured (d') distances to walls due to ramps. (a): slightly incorrect measurement. (b) and (c): incorrect measurements that cause serious problems for the localization and the mapping.


(a) The Hokuyo URG-04LX laser range finder (LRF), mounted on a rotation/tilt-unit.


(b) Map generated from data of a rigidly coupled LRF.

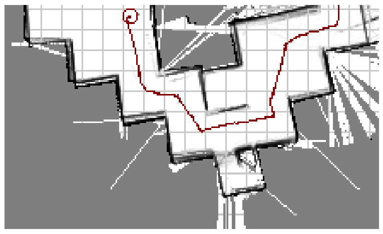

(c) Map generated from data of an adjusted LRF.

Fig. 8. The gimbal-mounted Hokuyo URG-04LX laser range finder (a) and the effect on the mapping: Map without (b) and with (c) adjustment. The map based on the data acquired by the adjusted LRF has more aligned walls and less artefacts.

7.2 Terrain classification

Due to its limited mobility the robot must stay out of rough terrain as simulated with the stepfields in the Orange Arena. Therefore, the robot performs 3D scanning every 2 meters, to locate boundaries to impassable areas. The 3D scans are acquired with same (adjusted) LRF that is used for the 2D localization and mapping. The scan is analyzed as follows:

1. Each point of the local scan is assigned to a grid cell (of size $100\times$ 100 mm) on the x/y-plane (the x/y-plane is parallel to the ground).

2. For each grid cell c_i, the variance in z (the height), as well as the the lowest and the highest z value is computed ($\sigma^2_{z,i}$, $\min_{z,i}$ and $\max_{z,i}$). Also, the number of points in each cell is counted ($\#_i$).
3. A cell is assigned one of three classes, depending on the precalculated parameters:

$$\text{class}(c_i) = \begin{cases} \textit{unknown}, & \text{if } \#_i < \text{minCnt}, \\ \textit{occupied}, & \text{if } (\sigma_i^2 > \text{maxVar}) \wedge ((\max_{z,i} - \min_{z,i}) > \text{maxStep}), \\ \textit{free}, & \text{else.} \end{cases} \tag{2}$$

   The values for the thresholds were determined experimentally and are set to: minCnt $= 10$, maxVar $= 18^2$ and maxStep $= 80$ (all distances in mm).
4. Finally, using the current pose of the robot, the world coordinates of all cells of class *occupied* is calculated and stored in a 2D grid, the "inaccessible grid": inaccessible(localToWorld(c_i)) $=$ *true*

An example of classified terrain is given in Fig. 9.

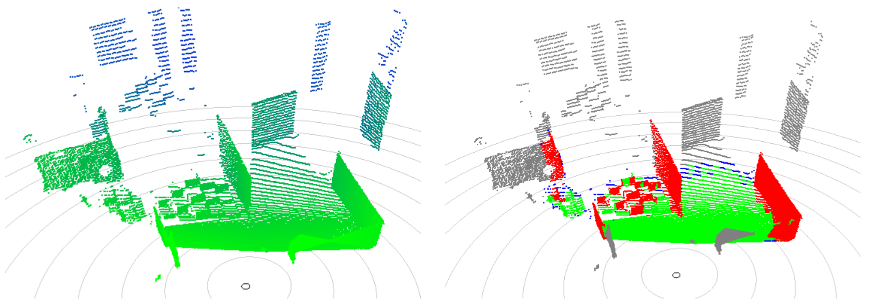

(a) 3D scan of a stepfield (b) Same scan; sectors classified as drivable or not

Fig. 9. Example of a 3D scan of the laser range finder.

7.3 Path planing: The Exploration Transform

By combining Yamauchi's frontier based exploration [Yam97] with Zelinsky's path transform (see [Zel88,Zel91]) an elegant solution for the exploration problem can be achieved: the path transform is extended in a way that not the cost of a path to a certain target cell is calculated, but the cost of a path that goes to a close frontier (the boundary between known and unknown areas). The path is not necessarily the shortest and the frontier not necessarily the closest, since the cost is determined by the travel distance *and* the safety of the path. The overall formula of the Exploration Transform [WP07] is given in (3); it yields for a given cell c the frontier cell that is close and save to reach:

$$\Psi(c) = \min_{c_g \in F} \left(\min_{C \in \chi_c^{c_g}} \left(l(C) + \alpha \sum_{c_i \in C} c_{\text{danger}}(c_i) \right) \right) \tag{3}$$

with F the set of all frontier cells, $\chi_c^{c_g}$ the set of all paths from c to c_g, $l(C)$ the length of the path C, $c_{\text{danger}}(c_i)$ the cost function for the "discomfort" of entering cell c_i, and α a weighting factor ≥ 0. The Exploration Transform has the favorable property that by construction no local minima can occur. Therefore, from each cell a path to a close frontier can directly be extracted. Compared to the path transform, the Exploration Transform performs a search over all possible frontier cells. More details can be found in [WP07].

8 Team Training for Operation (Human Factors)

The training required to use our system consists of knowing how to turn on the robot (and its devices) and how to start the application. Only a minimum amount of training is needed for using the GUI due to the intuitive representation of the data.

We mainly use log files (recorded at various RoboCup events) to test and enhance our software (e. g. the map building algorithms) and also to train people to use the application.

9 Possibility for Practical Application to Real Disaster Site

We don't have any practical experience with real disaster sites yet, except the participation at the *Response Robots Exercise 2008* in College Station, Texas, USA. So far we did not focus on mobility, but on precise mapping, autonomous exploration and a suitable presentation of the data to the operator. Our robot is incapable to drive through difficult terrain, which is covered with large pieces of debris. However, because of the four-wheel drive it is still able to drive on slippery ground as long as the ground is not too steep.

10 System Cost

The total price of the robot including all sensors is **23,552 EUR** (which is approximately 30,400 USD). The detailed prices are listed in Tab. 1.

11 Lessons Learned

– The adjustable laser range finder provides consistent 2D laser scans, also on uneven terrain. This makes the localization and the mapping much easier.
– The 3D scans generated by the tilted laser range finder are very useful for terrain classification.

Article	Type	Price
Notebook	NEXOC Osiris, (T7800, 2 GB RAM)	1,200 EUR
Robot	Pioneer 3-AT	6,050 EUR
FireWire Camera	Sony DFW–X710	1,392 EUR
Lens	Cosmicar	222 EUR
FireWire Camera	Sony DFW–X700	2,038 EUR
Lens	Cosmicar	222 EUR
FireWire Camera	Unibrain Fire-i	80 EUR
Laser Range Finder	Hokuyo URG-04LX	1,555 EUR
Thermal camera	ThermoVision Micron/A10	10,000 EUR
Microcontrollers	Crumb8-USB	60 EUR
USB Hub	4 port USB hub	30 EUR
Access point	Linksys WAP55AG	100 EUR
Aluminum rack	Maschinenbau Kitz	400 EUR
Misc. electronic parts	–	200 EUR
Blue flashing light	–	3 EUR
Total		**23,552 EUR**

(The total price is approximately 30,400 USD)

Table 1. Detailled prices of the components of *Robbie*

References

[HBFT03] Dirk Hähnel, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. A highly efficient fastslam algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[IB98] Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *International Journal for Computer Vision*, 29(1):5–28, 1998.

[Pel07] Johannes Pellenz. Rescue robot sensor design: An active sensing approach. In *SRMED2007: Fourth International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster*, pages 33–37, Atlanta (USA), 2007.

[Pel08] Johannes Pellenz. Mapping and map scoring at the robocuprescue competition. In *Quantitative Performance Evaluation of Navigation Solutions for Mobile Robots (RSS 08, Workshop CD)*, 2008.

[PP09] Johannes Pellenz and Dietrich Paulus. Stable mapping using a hyper particle filter. *RoboCup Symposium 2009*, 2009. (accepted for publication).

[WP07] Stephan Wirth and Johannes Pellenz. Exploration transform: A stable exploring algorithm for robots in rescue environments. *Workshop on Safety, Security, and Rescue Robotics, http://sied.dis.uniroma1.it/ssrr07/*, pages 1–5, 9 2007.

[Yam97] B. Yamauchi. A frontier-based approach for autonomous exploration. *1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, page 146 ff., 1997.

[Zel88] Alexander Zelinsky. Robot navigation with learning. *Australian Computer Journal*, 20(2):85–93, 5 1988.

[Zel91] Alexander Zelinsky. *Environment Exploration and Path Planning Algorithms for a Mobile Robot using Sonar.* PhD thesis, Wollongong University, Australia, 1991.