OBSTACLE DETECTION IN MOBILE OUTDOOR ROBOTS
A Short-term Memory for the Mobile Outdoor Platform RAVON

H. Schäfer, M. Proetzsch and K. Berns
Robotics Research Lab, University of Kaiserslautern, P.O. Box 3049, D-67653 Kaiserslautern, Germany

Keywords: Obstacle Detection, Obstacle Avoidance, Behaviour-based Navigation, Mobile Robotics, Short-term Memory, Distributed Minimal World Model, Algorithms, Biologically Inspired Robotics.

Abstract: In this paper a biologically inspired approach for compensating the limited angle of vision in obstacle detection systems of mobile robots is presented.
Most of the time it is not feasible to exhaustively monitor the environment of a mobile robot. In order to nonetheless achieve safe navigation obstacle detection mechanisms need to keep in mind certain aspects of the environment. In mammals this task is carried out by the creature's short-term memory. Inspired by this concept an absolute local map storing obstacles in terms of representatives has been introduced in the obstacle detection and avoidance system of the outdoor robot RAVON. That way the gap between the fields of vision of two laser range finders can be monitored which prevents the vehicle from colliding with obstacles seen some time ago.

1 INTRODUCTION

In recent years complex mobile robotic systems have been developed for autonomous navigation in various environments. The more complex a robot's working place gets the more sophisticated its obstacle detection and avoidance facilities need to designed. First of all, sensor equipment must be appropriate for the requirements of given scenarios. In most cases such sensors have a limited field of vision and therefore cannot cover the complete area around the robot. One possibility to overcome this limitation is mounting several devices of the same kind. However, this leads to higher weight and costs and the evaluation of more data might exceed the computational capacity.

In mammals the short-term memory is used to compensate for the relatively small field of vision. The hindlegs of a cat for example have to avoid obstacles some time after the cat has visually perceived the hindrance (McVea and Pearson, 2006). Clearly the cat has to keep in mind certain information about its local environment in order to safely navigate complex terrain.

In this paper an approach for adopting the principle of a local short-term obstacle memory for mobile outdoor robots is presented. The implemented system is used to gather additional information for the behaviour-based control system of the robot RAVON (Robust Autonomous Vehicle for Outdoor Navigation, see Figure 1). For now data of two laser scanners – mounted at the front and the rear of the robot – is evaluated and inserted into the obstacle memory.

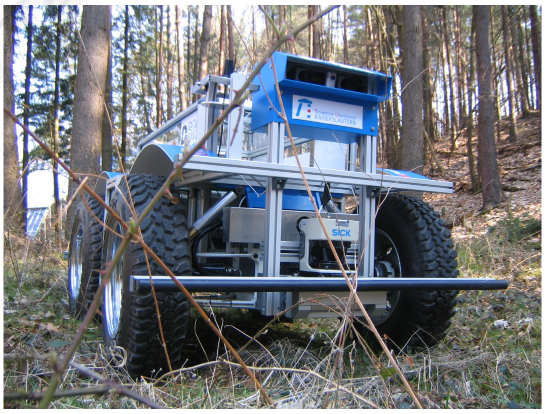
Figure 1: The outdoor robot RAVON in rough terrain.

The approach, however, is kept in a generic way such that additional sensor systems as e.g. stereo camera systems can easily be integrated.

The outline of the paper is as follows: In the next section work related to the topic of this paper is described. Afterward we give some details on the outdoor robot RAVON which is used for validating the presented approach. Section 4 describes the concept of the short-time memory including some details about the obstacle detection methods. Section 5 shows an experiment used for evaluating the presented approach. Finally, we conclude with a summary and directions for future work.

2 STATE OF THE ART

Obstacle avoidance in outdoor terrain is a topic of several publications. Mostly there is the distinction between reactive obstacle avoidance and building up complete geometric maps. Reactive approaches like (Badal et al., 1994) compute steering vectors according to the proximity of obstacles in the current view. However, for vehicles supporting agile steering manoeuvres like sideward motion this neglects the problem of possible collisions outside the field of view. Other similar approaches ((Kamon et al., 1996), (Laubach and Burdick, 1999)) add some kind of state to the obstacle avoidance system but do not explicitly consider positions of hidden obstacles.

Work describing building up maps which contain information about terrain elevation ((Shiller, 2000), (Bonnafous et al., 2001)) also neglect the need to enlarge the virtual field of vision. A high degree of computation is used to evaluate the traversability of the detected terrain region and to calculate a feasible trajectory. However, in outdoor terrain given paths cannot be followed precisely due to disturbances. Therefore, an evaluation of possibly dangerous obstacles needs to be undertaken during the driving manoeuvre.

The approach presented here can be seen in between the completely reactive and the mapping approach. A short-term memory keeping only the relevant information deals as the source for a behaviour-based system keeping the robot away from currently relevant obstacles.

3 VEHICLE DESCRIPTION

The platform used to examine obstacle avoidance is the four wheeled off-road vehicle RAVON (see Figure 1). It measures 2.35 m in length and 1.4 m in width and weighs 400 kg. The vehicle features a four wheel drive with independent motors yielding maximal velocities of 3 m/s. In combination with its off-road tires, the vehicle can climb slopes of 100% inclination predestining it for the challenges in rough terrain. Front and rear axis can be steered independently which supports agile advanced driving manoeuvres like double Ackerman and parallel steering.

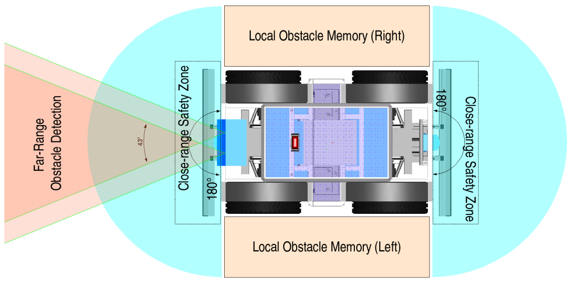
Local Obstacle Memory (Right)
Far-Range Obstacle Detection
Close-range Safety Zone
180°
180°
Close-range Safety Zone
Local Obstacle Memory (Left)

Figure 2: Regions monitored by the obstacle avoidance facilities.

In order to navigate in a self-dependent fashion, RAVON has been equipped with several sensors. For self localisation purposes, the robot uses its odometry, a custom design inertial measurement unit, a magnetic field sensor, and a DGPS receiver. The sensor data fusion is performed by a Kalman filter (Schmitz et al., 2006) which calculates an estimated pose in three dimensions.

In order to protect the vehicle in respect to obstacles, several safety regions are observed by different sensor systems (Schäfer and Berns, 2006) (see Fig. 2). First of all, hindrances can be detected using the stereo camera system mounted at the front of the vehicle. This obstacle detection facility is complemented with two laser range finders (field of vision: 180 degrees, angular resolution: 0.5 degrees, distance resolution: about 0.5 cm) monitoring the environment nearby the vehicle. Data from both sources of proximity data is used for obstacle avoidance by appropriate behaviours, the fusion of which is performed inside the behaviour network (Schäfer et al., 2005). In case of emergency, the system is stopped on collision by the safety bumpers which are directly connected to the emergency stop to ensure maximal safety. In the future, the compression of the spring system shall be used to detect occluded obstacles in situations where geometric obstacle detection cannot be used.

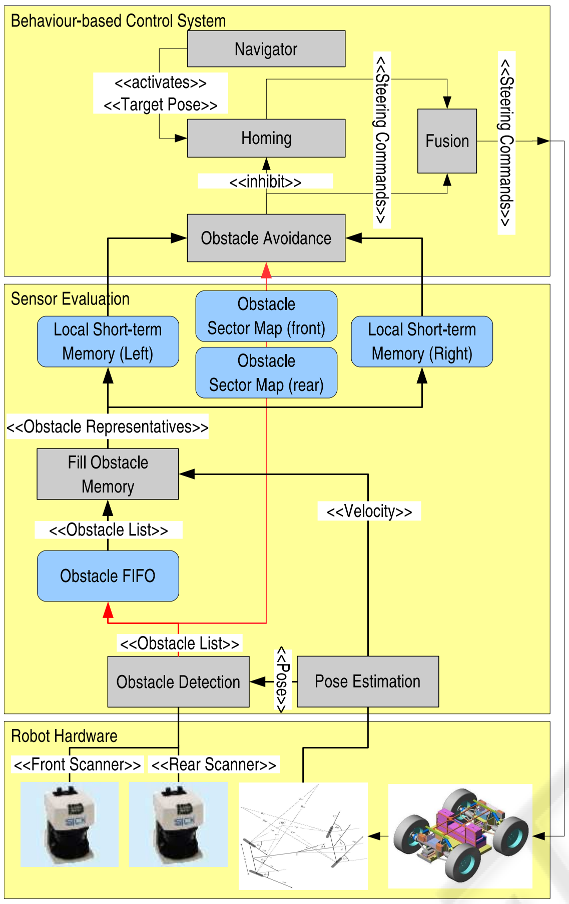

Figure 3: Top-level concept of RAVON's Obstacle Avoidance Facility.

4 A SHORT-TIME MEMORY FOR RAVON

In order to compensate for the blind angles of the sensor systems local short-term memories shall successively be introduced. Any of these can be regarded as a virtual sensor system which can seamlessly be integrated into RAVON's control system as described in (Schäfer et al., 2005). That way step by step blind regions can be covered with increasing accuracy as overlapping virtual sensors yield more and more additional information. In this paper the validation of the general approach and connected technical facilities are presented by the example of closing the gap between the two laser range finders (See Figure 2).

Listing 1: Declarations.

```
# Raw sensor information from the
# laser range finders
scanner_data

# FIFO containing the candidate obstacles
obstacle_fifo

# The maximal length of the queue
# assures that the obstacle lists
# have a certain age when retrieved.
# Furthermore the scans are discarded
# if the vehicle is not moving to prevent
# the accumulation of outdated information.
max_number_of_scans

# Stores the current robot pose in the
# robot's absolute working coord. system
current_robot_pose

# Stores the current velocity of the robot
current_velocity

# Stores representatives in the blind
# angle to the left of the robot.
short_term_memory_left

# Stores representatives in the blind
# angle to the right of the robot.
short_term_memory_right

# Threshold of the maximal euclidean distance
# between the current vehicle pose and the
# one at which a scan was taken.
progress_threshold
```

Figure 3 illustrates the structure of the laser-scanner-based part of the obstacle avoidance system. The sensor flow[1] of the main loop which is dealing with the short-term memory is outlined in Listing 2 (For explanation of the particular variables used see Listing 1). The first step of the main loop is the detection of obstacles from the laser range data which is carried out in the Obstacle Detection facility. The current vehicle pose in the robot's absolute working coordinate system is applied to the resulting obstacle lists in order to yield an absolute reference map through which the robot can virtually be navigated. After that the lists are casted into sector maps which are used by the `Obstacle Avoidance` behaviours to compute evasive steering commands if necessary.

For the extension described in this paper a FIFO queue was introduced which holds in stock the obstacle lists of several subsequent scans. In every sense cycle older obstacle lists are examined for ob-

[1]The control concept underlying all robot projects at the Robotics Research Lab strictly separates the main loop into a buttom up sense and a top down control cycle which are alternately invoked.

stacle representatives which may have migrated into the blind angles. Older in this sense can be interpreted in two ways: a) the scan was taken a certain time ago (*age criterion*[2]) or b) the scan was taken a certain distance ago (*progress criterion*[3]).

Listing 2: Main Loop.

```
Sense ()
  # Detect obstacles from range information
  DetectObstacles (scanner_data ,
                   obstacle_list_fifo)

  if (current_velocity != 0) do
    # remove representatives which are no
    # longer in the scanner's blind angle
    Trim (short_term_memory_left , current_robot_pose)
    Trim (short_term_memory_right , current_robot_pose)

    # check age criterion (a)
    while (Size (obstacle_list_fifo) >
            max_number_of_scans) do
      obstacle_list = First (obstacle_list_fifo)
      FillObstacleMemory (short_term_memory_left ,
                          short_term_memory_right ,
                          obstacle_list ,
                          current_robot_pose)
      RemoveFirst (obstacle_list_fifo)
    od

    # check progress criterion (b)
    obstacle_list = First (obstacle_list_fifo)
    while (Distance (Pose (obstacle_list),
                           current_robot_pose)
          > progress_threshold) do
      FillObstacleMemory (short_term_memory_left ,
                          short_term_memory_right ,
                          obstacle_list ,
                          current_robot_pose)
      RemoveFirst (obstacle_list_fifo)
      obstacle_list = First (obstacle_list_fifo)
    od
  else
    # discard older scans if not moving
    while (Size (obstacle_list_fifo) >
            max_number_of_scans) do
      RemoveFirst (obstacle_list_fifo)
    od
  esle
esneS
```

If one of the two criteria applies to an obstacle list `FillObstacleMemory` checks the obstacle representatives and places them into the corresponding short-term memory. In order to prevent the short-term memory from overfilling `Trim` (See Listing 4) removes obstacle representatives which are no longer in the robot's blind angle. Furthermore the obstacle list FIFO has a limited maximal size. That way only a limited number of scans is keept in mind and is examined which minimises memory consumption and computational expenses.

As the `Obstacle Sector Maps` the short-term memories are evaluated by the `Obstacle Avoidance` component which computes evasive steering commands and inhibits the `Homing` behaviours if applicable. The steering commands

[2]The age of a scan is modelled implicitly by the maximal number of scans in the queue.

[3]The *progress_threshold* assures that no large gaps occur in the obstacle memory at higher speeds as scans are examined earlier.

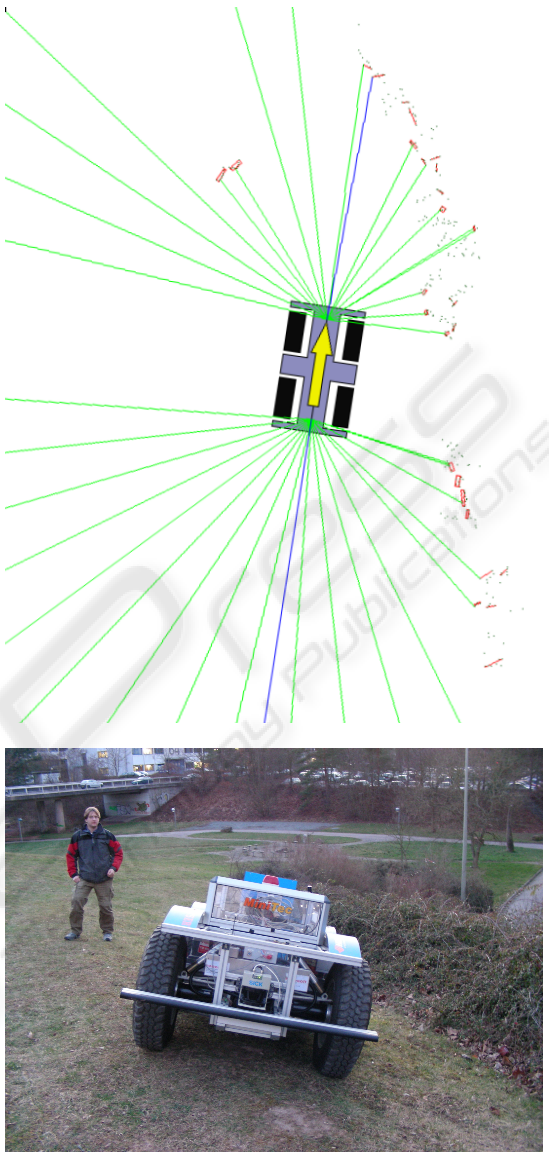

Figure 4: Screenshot showing clustering of detected range values (top) and photo showing the scenario (bottom).

from the `Homing` facility and the `Obstacle Avoidance` are fused according to the behaviour architecture described in (Proetzsch et al., 2005). If no obstacles are present `Homing` successively navigates along a trajectory given in subsequent target poses provided by the `Navigator` component.

4.1 Obstacle Detection

The range data from the laser scanners is first of all clustered according to point-wise adjacency in order to filter artefacts and entities which are likely to represent flexible vegetation (See Figure 4). In this figure the current laser scanner data (dark points) are interpreted as clusters (red boxes) according to their distance. Each of the clusters is then evaluated concerning its dimension and its number of cluster representatives. If both criteria are true clusters are interpreted as obstacles.

In order to have a uniform interpretation of near obstacles which can be used by obstacle avoidance behaviours the sensor coverage area is divided into sectors. In Figure 4 the closest obstacle in each of these sectors is indicated by a green line. In Figure 4 the human to the left and the bushes to the right of the robot have clearly been detected as obstacles. Note that the bushes are subject to severe noise and that they are therefore detected as a large number of small fluctuating objects. For that reason obstacles cannot be correlated over time which is a problem for local map building as representatives accumulate over time if the robot is moving slowly. Using an occupancy grid would be a solution to this problem but it lacks accuracy because of rasterisation. Furthermore the consideration of objects moving independently from the vehicle – which shall be introduced in the future – would be complicated a lot.

4.2 Short-term Memory

As already indicated above obstacles detected in laser rage data consist of points which represent the boundary of a rigid object. Thus in every sensor cycle a list of such point clusters[4] is generated and stored in a FIFO queue for further processing. To every obstacle list the obstacle detection facility attaches the current robot pose in an absolute working coordinate system.

In later sense cycles older obstacle lists are retrieved from the FIFO queue in order to be examined for obstacle representatives which might now be in the blind angle of the laser scanners. Note that in this context the algorithm drops the abstraction of obstacles in favour of simplicity. This does not result in a loss of information as all the robot can actually do if it detects a remembered obstacle at its flank is try to keep away from it by adding a translational component to the steering command. Which representative belongs to what obstacle is in that sense irrelevant.

To fill the obstacle memory routine `FillObstacleMemory` (See Listing 3) examines all obstacles in a provided obstacle list. In order to determine the relation between the vehicle at its current pose and the scan points registered earlier, all representatives are first of all transferred into the absolute working coordinate system of the robot using the

[4]In the following this list shall be referred to as *obstacle list*.

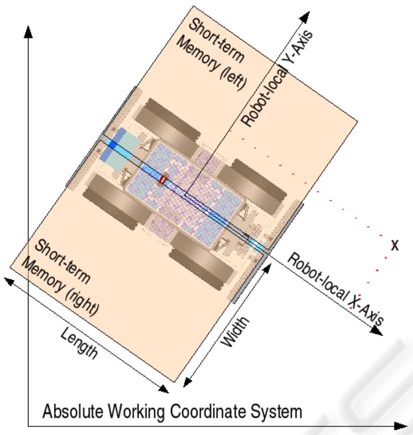

Figure 5: Schema of how to determine whether an obstacle representative resides in the blind region of the vehicle.

pose previously attached to the obstacle list (Subroutine `ToAbsoluteContext`). Intuitively explained this is as if you would virtually drive a robot model through an absolute landscape while checking for points in the blind regions of the robot. That way it is not necessary to transform the representatives whenever the robot changes its pose.

Listing 3: Fill the Obstacle Memory.

```
FillObstacleMemory (short_term_memory_left,
                    short_term_memory_right,
                    obstacle_list,
                    current_robot_pose)
  forall (obstacle in obstacle_list) do
    forall (representative in obstacle) do
      absolute_representative =
        ToAbsoluteContext (Pose (obstacle_list),
                           representative)

      y = PointOnLine (XAxis (current_robot_pose),
                       absolute_representative)
      x = PointOnLine (YAxis (current_robot_pose),
                       absolute_representative)

      if (Abs (y) < Width (short_term_memory)/2 AND
          Abs (x) < Length (short_term_memory)/2) {
        if (ToRightOfRobot (current_robot_pose,
                            absolute_representative))
          Append (short_term_memory_right,
                  absolute_representative)
        else
          Append (short_term_memory_left,
                  absolute_representative)
        fi
      od
    od
  od
yromeMelcatsbOlliF
```

Once in an absolute context the coordinates can reside where they are. With the subroutine

`PointOnLine` the distance of the representative to the X-axis and the Y-axis of the robot-local coordinate system is computed (See Figure 5). This yields the robot-local coordinate of the representative which offers a straightforward check whether the representative resides in a blind area of the robot. Depending whether the representative is rather to the left or the right of the robot it is added to the corresponding short-term memory. Routine `Trim` (See Listing 4) works in analogy just the other way around to sweep representatives from the short-term memories which are no longer in the blind angle of the scanners.

Listing 4: Trim the Short-term Memory.

```
Trim (short_term_memory, current_robot_pose)
  forall (representative in short_term_memory) do

    y = PointOnLine (XAxis (current_robot_pose),
                            representative)
    x = PointOnLine (YAxis (current_robot_pose),
                            representative)

    # Check whether the representative is
    # still in the blind angle
    if (Abs (y) > Width (short_term_memory)/2 OR
        Abs (x) > Length (short_term_memory)/2)
        Remove (representative, short_term_memory)
    fi
  od
mirT
```

Note that the robot pose used during the procedures outlined above currently relies on odometry only. This is due to the fact that odometry is locally coherent in comparison to data fused from odometry, GPS, the inertial measurement unit and the magnetic field sensor. Eventhough globally more accurate, the latter three introduce certain local jumps into the pose, which are a problem for precise local navigation. The pose drift immanent in odometry on the other hand is not an issue for the approach presented in this paper as it is enough to be rather precise over the distance of a vehicle length. When the robot has passed the obstacle it is erased from the short-term memory anyway. In the future visual odometry shall additionally be called into service in order to gain more accurate pose information. In particular on slippery surfaces the authors expect this combination to be a lot more robust than banking on odometry only.

5 EVALUATION IN EXPERIMENTS

The short-term memory described before was implemented and integrated into the control system of RAVON. In order to evaluate the approach several test runs have been executed. First of all autonomous navigation along predefined tracks has been executed. In this case the vehicle follows the given direction to the target point as long as there is no hindrance. In case of obstacles, however, the straight motion is interrupted by avoidance behaviours. This effect remains active even if the obstacle is in the blind side region. Therefore the robot gets back on track only as soon as all hindrances are left behind.

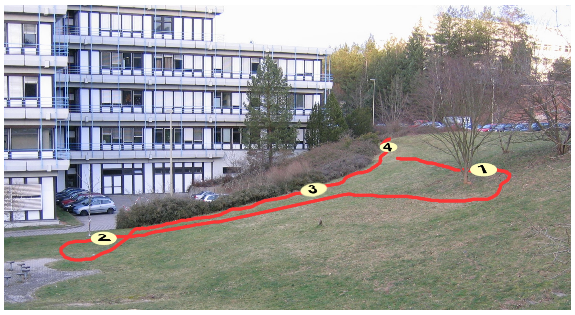

Figure 6: Image of the test scenario with overlaid trace.

For clarification of the properties of the short-term memory in this section a test run is presented where the robot is remote controlled. In this experiment the robot is driven through a locally flat environment. All the time the behaviour-based control system evaluates sensor information and overrides the given commands if necessary in order to avoid collisions.

The scenario for this run is hilly grass land as demonstrated in Figure 6. The image shows some trees to be avoided as well as some bushes with unknown load bearing surface. For clarification the approximate path of the vehicle is overlaid. Four markers indicate positions referred to in the following description.

Using the localisation system the trace of the vehicle can be plotted as presented in Figure 7. Some of the trees and bushes are integrated in order to point out the sensor data and vehicle reaction. Figure 8 shows the scanner data as well as clusters and sector values (see Section 4.1) for the four positions indicated by markers in the trace plot. Additionally data of the obstacle memory at the side of the vehicle is depicted as blue lines.

Position 1 shows a situation where the robot was steered into the direction of a tree. After an evading manoeuvre the robot is now located next to the tree. The sensor data shows that the obstacle is at the edge of the side region and would eventually be hidden. However, the blue lines indicate data stored in the obstacle memory. Therefore the robot continues preserving a minimal distance to the obstacle. At

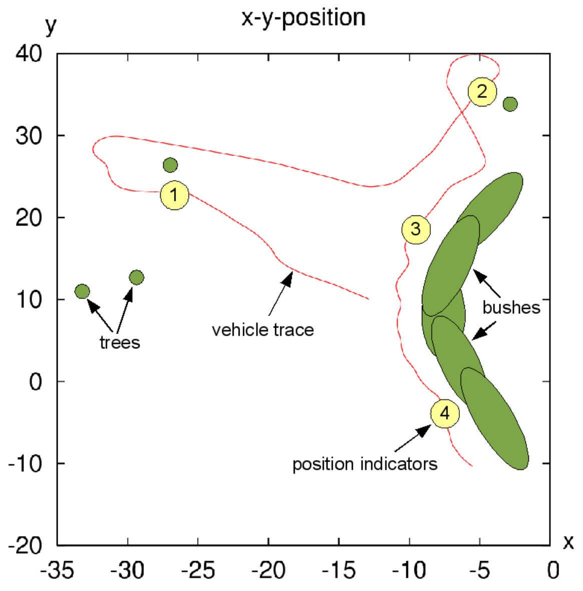

Figure 7: Top view position diagram of the experiment.

position 2 the robot is in a similar situation. A tree which was detected by the laser range finder at the front changed its relative position such that it is located at the right side of the robot.

Positions 3 and 4 show the situation where the robot is manually steered to the left. Due to the obstacles detected and stored in the short-term memory the evading behaviours force the robot away from the hindrances. This interaction results in a smooth following of the bushes.

The experiments showed a major improvement of the reaction to obstacles. Behaviours regarding the short-term memory prevent the robot of driving into the direction of hidden obstacles. For static obstacles this leads to a system reacting on hindrances which can be located anywhere around the vehicle.

6 CONCLUSIONS AND FUTURE WORK

In this paper we presented an approach for virtually extending the coverage of obstacle detection sensor systems. The formulated algorithm uses information about the vehicle motion to propagate sensor data into the blind zone of the robot. Due to a massive reduction of stored data the system features low memory usage. Additionally there is no need for calculating the correspondence between consecutive data making the approach applicable for realtime scenarios.

Next steps involve the integration of a stereo cam-

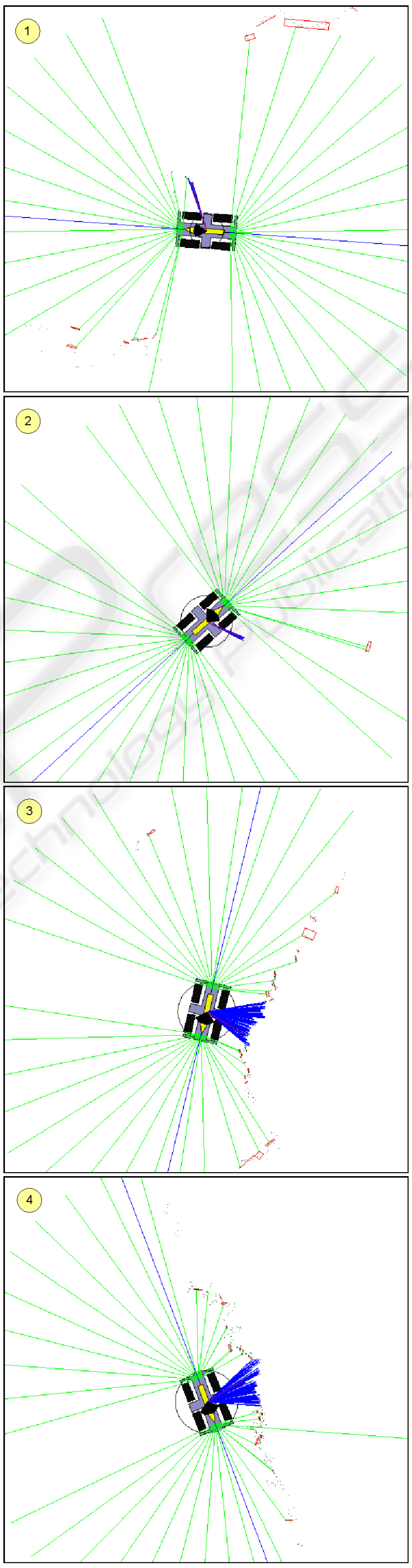

Figure 8: Sensor data visualisation during the experiment.

era system. Due to the device's limited field of view the presented algorithm leads to an eminent extension of the supervised area surrounding the vehicle. Additionally the accuracy of the local positioning system will be enhanced by visual ego motion estimation.

REFERENCES

Badal, S., Ravela, S., Draper, B., and Hanson, A. (1994). A practical obstacle detection and avoidance system. In *IEEE Workshop on Applications of Computer Vision*.

Bonnafous, D., Lacroix, S., and Simon, T. (2001). Motion generation for a rover on rough terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Kamon, I., Rivlin, E., and Rimon, E. (1996). A new range-sensor based globally convergent navigation algorithm for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Laubach, S. L. and Burdick, J. W. (1999). An autonomous sensor-based path-planner for planetary microrovers. In *IEEE Int. Conf. on Robotics and Automation*.

McVea, D. and Pearson, K. (2006). Long-lasting memories of obstacles guide leg movements in the walking cat. *The Journal of Neuroscience*.

Proetzsch, M., Luksch, T., and Berns, K. (2005). Fault-tolerant behavior-based motion control for offroad navigation. In *20th IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain.

Schäfer, H. and Berns, K. (2006). Ravon - an autonomous vehicle for risky intervention and surveillance. In *International Workshop on Robotics for risky intervention and environmental surveillance - RISE*.

Schäfer, H., Proetzsch, M., and Berns, K. (2005). Extension approach for the behaviour-based control system of the outdoor robot ravon. In *Autonome Mobile Systeme*.

Schmitz, N., Proetzsch, M., and Berns, K. (2006). Pose estimation in rough terrain for the outdoor vehicle ravon. In *37th International Symposium on Robotics (ISR)*.

Shiller, Z. (2000). Obstacle traversal for space exploration. In *IEEE International Conference on Robotics and Automation*.