6th IEEE International Conference on Intelligent Engineering Systems (INES),
May 2001, Opatija, Croatia

A Robust Certainty Grid Algorithm for Robotic Vision

Wilfried Elmenreich Lukas Schneider Raimund Kirner
Institut für Technische Informatik
Technische Universität Wien
Karlsplatz 13, Vienna, Austria
{wil,lukas,raimund}@vmars.tuwien.ac.at

Abstract — **In this paper we describe an algorithm for fault tolerant sensor mapping for robotic vision. Basically we use a certainty grid algorithm to map distance measurements into a two-dimensional grid. The well-know certainty grid algorithm can tolerate occasional transient sensor errors and crash failures, but will fail when a sensor provides permanently faulty measurements.**

Therefore we extended the certainty grid algorithm by a sensor validation method that detects abnormal sensor measurements and adjusts a confidence value for each sensor. This robust certainty grid approach works with at least three sensors with an overlapping sensing range and needs fewer sensor inputs and less memory than other approaches. Our method supports also reintegration of recovered sensors and sensor maintenance by providing a measurement for the operability of a sensor.

We present also a case study with an autonomous mobile robot that features the robust certainty grid algorithm in a time-triggered architecture.

1 Introduction

A mobile robot must be able to notice surrounding objects in order to be able to interact with its environment. Sensors come in a great variety of types and each sensor is able to contribute to the task of environmental perception.

However, given multiple sensory inputs, the task of modelling these data into a simple, comprehensible image of the environment can be arduous when problems of temporal accuracy [1], imprecise and faulty measurements, and sensor deprivation are considered.

This paper describes an algorithm for mapping sensor information based on the certainty grid approach. The first certainty grid method has been developed at Carnegie-Mellon University in the 1980ies [2]. However, the certainty grid suffers from faulty sensor measurements when they are not detected at the sensor level.

It is the objective of this paper to propose an extension of the certainty grid algorithm by a sensor validation method at the sensor integration level.

Comparing sensor measurements directly is difficult when their measurements are made at different instants. As a result we used an approval method for calculating confidence values for each sensor. Our algorithm is able to implicitly detect sensors with malfunctions followed by a reduction of the sensor's input contribution to the certainty grid. Our approach supports inherently automatic integration of recovered sensors. Furthermore our approach facilitates sensor maintenance by assigning each sensor dynamically a confidence value which can be a measurement for the reliability of the sensor.

The robust certainty grid algorithm has been tested with simulated sensor faults. We have also implemented a demonstrator with an autonomous mobile robot that features the robust certainty grid algorithm.

The remainder of the paper is organized as follows: Section 2 gives an overview on the original certainty grid algorithm. The following section discusses the influence of sensor faults on the certainty grid. Section 4 describes the robust certainty grid algorithm. Section 5 describes the functionality of our demonstrator. The paper is concluded in Section 6.

2 Certainty Grid Algorithm

This section provides a brief overview of existing certainty grid algorithms.

A *certainty* or *occupancy grid* is a multidimensional (typically 2D or 3D) representation of the robot's environment. The observed space is subdivided into cells, where each cell stores information about the corresponding environment and an estimated probability for the correctness of this information. Typically, a cell state can be "free", if the place appears to be void, or "occupied" if an object has been detected for that cell. Cells not reached by sensors reflect an "uncertain" state. The cell state and the probabilistic estimate of its correctness can be mapped in a single number reflecting the confidence of a cell to be free.

Basically, it is assumed, that the certainty grid application has no a priori knowledge of the geometry of its environment and the objects in this environment are mostly static. The effect of occasional sensor errors will be neglected, because these will have little effect on the grid [3].

The calculation of new grid values is usually done by Bayesian inference. The English clergyman Thomas Bayes stated in a paper (published after his death in the Philosophical Transactions of the Royal Society of London [4]) the rule known today as Bayes' theorem:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{1}$$

Bayes' theorem quantifies the probability of hypothesis H, given that event E has occurred. $P(H)$ is the *a priori* probability of hypothesis H, $P(H|E)$ states the *a posteriori* probability of hypothesis H. $P(E|H)$ is the probability that event E is observed given that H is true. If multiple events have to be considered using Bayes' rule, the order of processing does not influence the result.

Hoover and Olsen present an application of a certainty grid where a set of video cameras is used to detect free space in the vicinity of a robot [5]. They use the multiple views to overcome the problem of occlusion and to increase performance, however they do not discuss the subject of sensors delivering faulty measurements.

Sensor information usually is imperfect with respect to restricted temporal and spatial coverage, limited precision, and possible sensor malfunctions or ambiguous measurements. To maximize the capabilities and performance it is often necessary to use a variety of sensor devices that complement each other. Modelling such sensor measurements into the grid is an estimation problem [6].

Matthies and Elfes [7] propose a uniform method for integration of various sensor types. Each sensor is assigned a spatial interpretation model, developed for each kind of sensor, that maps the sensor measurement into corresponding cells. When sensor uncertainties are taken into account, we arrive at a probabilistic sensor model.

Figure 1 depicts the data flow of a certainty grid implementation with three sensors. The sensor in the right position delivers faulty measurements which results in a deviation of the certainty grid from the real object positions.

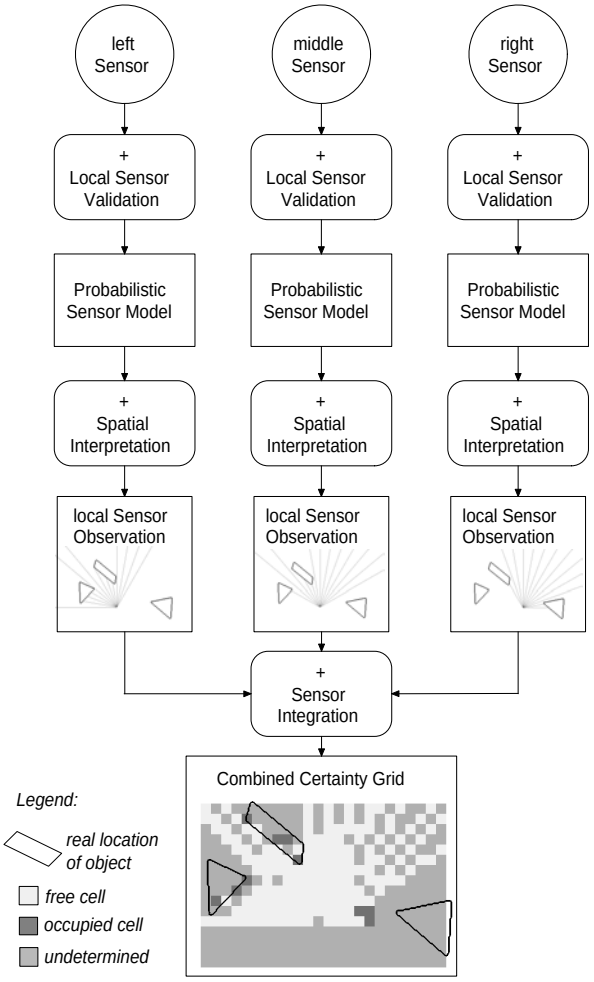


Figure 1: Certainty Grid Data Flow

3 Dealing With Sensor Faults

Martin and Moravec [3] concluded that the effects of occasional sensor faults on the grid can be neglected.

Furthermore, if a sensor input has crash failure semantics, i. e. is either a correct value or no value at all, the existing methods are sufficient to handle this situation if each important grid cell is served by more than one sensor.

However one problem of the certainty grid algorithm as found in the literature [2, 3, 6, 7] arrives when a sensor permanently provides faulty measurements. For example a distance sensor could refuse to detect any object and always report “no object nearby”. Such a fault would result in a significant deviation of the representation of the environment in the grid from the actual environment.

There are two possible solutions to this problem:

Replicated sensors: Making the sensors fault-tolerant by replication results in costs for extra sensors and voting nodes. For example each sensor could be extended to triple-modular redundant sensors, the basic idea of such fault-tolerant units has already been presented in [8]. However these extra sensors would not contribute to the grid resolution or improve the update frequency.

Replicated certainty grids: The generation of multiple grids and the application of standard fault-tolerant algorithms among these grids does not need extra sensors. Each single certainty grid would represent a fault isolation area, e. g. supported by a single sensor. The final view will then be generated by majority voting among the separate grids. However this approach, although technically possible, has the great disadvantage of increased memory requirements. A system with n sensors would need the $(n+1)$-fold amount of memory to represent the grids.

While the replicated sensors approach deals with the problems at sensor level, the second approach takes effect at the grid level. Because of hardware and wiring costs we decided for a grid level solution as described in the second approach. However RAM memory is one of the most critical resources in embedded systems like a mobile robot - therefore we developed a more sophisticated algorithm described in the following section.

4 Robust Certainty Grid Algorithm

We assume, that a sensor node may have a failure mode where it permanently submits measurements with incorrect values. It is our goal to extend the existing certainty grid to tolerate such sensor faults.

This goal will by achieved by analyzing the redundant parts of the certainty grid. Furthermore, we assume that we have no a priori knowledge about the redundant and none-redundant parts, thus we head for an automatic sensor validation.

It is difficult to validate sensors directly by comparing their inputs, because measurements from different sensors for the certainty grid are often made from different angles and at different instants - a deviation in sensor measurements may be caused by a sensor fault as well as by a change in the environment.

Therefore we use an approval method for maintaining a confidence measurement for each sensor. The confidence value will be a measurement for the correctness of a sensor. This confidence measurement *conf* may be a real value ranging from 0 to 1:

$$conf = \begin{cases} 0 & \text{sensor appears to be wrong} \\ \ldots & \\ 1 & \text{sensor appears to be correct} \end{cases}$$

If we have a priori knowledge about the sensor reliabilities, an initial confidence value that reflects the respective reliability can be chosen at startup. If we have no knowledge about the reliability of sensors the respective confidence values are initialized with 1.

As in the known certainty grid algorithms, each grid cell contains a probabilistic value *occ* ranging from 0 to 1 corresponding to the believe, that this cell is occupied by an object:

$$cell.occ = \begin{cases} 0 & \text{free} \\ \ldots & \\ 0.5 & \text{uncertain} \\ \ldots & \\ 1 & \text{occupied} \end{cases}$$

Additionally we store the main contributor (e. g., the sensor that updated this cell most recently) of the *occ* value with the cell. This property of each cell will be named the *current owner* of the

```
procedure AddToGrid( sensor, cell )
begin
  if (cell.owner = unknown) or (cell.owner = sensor) then
    cell.occ := sensor.measurement;
    cell.owner := sensor;
  else
    comparison := 4∗(cell.occ-0.5)∗(sensor.measurement-0.5);
    weight1 := abs(cell.occ-0.5)∗cell.owner.conf;
    weight2 := abs(sensor.measurement-0.5)∗sensor.conf;
    cell.occ := (cell.occ∗weight1+sensor.measurement∗weight2)
                                    / (weight1 + weight2);
    if comparison > THRESHOLD then
      inc(cell.owner.conf);
      inc(sensor.conf);
    if comparison < –THRESHOLD then
      dec(cell.owner.conf);
      dec(sensor.conf);
    contribution := 4∗(cell.occ-0.5)∗(sensor.measurement-0.5);
    if contribution > THRESHOLD then
      cell.owner := sensor;
    else
      cell.owner := unknown;
end
```

Figure 2: Pseudocode of the AddToGrid algorithm

cell:

$$cell.owner = \begin{cases} 0 & \text{unknown} \\ 1 & \text{sensor 1} \\ \ldots & \\ n_{Sensors} & \text{sensor n} \end{cases}$$

All grid cells are initialized with $cell.occ = 0.5$ and $cell.owner = unknown$.

When a new measurement has to be added to the grid, the following *AddToGrid* algorithm is executed: (Fig. 2 lists the algorithm in pseudocode)

If the particular grid cell has no contributor listed in its owner field, the measurement of the sensor is taken as is and the cell stores the index of the sensor as new owner.

If there was a contributor, the measurement is first compared to the cell value $cell.occ$. A value named *comparison* is calculated that means a *confirmation* of old cell value and new measurement, if the value is above a certain threshold, and means a *contradiction* of old cell value and new measurement, if the value is below a certain different threshold. In case of a confirmation, the confidence values of the new sensor and the owner are both increased up to a maximal bound of confidence. In case of a contradiction, the confidence values of the new sensor and the owner are both decreased down to a lower bound of confidence. If *comparison* is not significant, it does neither yield a confirmation nor a contradiction.

The new occupancy value of the cell is calculated as a weighted average between old value and measurement. The weights are derived from the respective confidence values and the significance of the measurement. A measurement is more significant if it has a greater absolute distance to the *uncertain* state (0.5).

Thereafter, a new owner has to be selected. Therefore, a value *contribution* is derived. *contribution* is calculated the same way as the comparison value, but it uses the new $cell.occ$ value.

cell.occ	cell.owner.conf	sensor.measurement	sensor.conf	**comparison**	**new cell.occ**	**confidences**	**contribution**	**new cell.owner**
0.8	1	1	1	0.6	0.925	increased	0.85	$sensor_i$
0.925	1	0	1	-0.85	0.425	decreased	0.15	*unknown*
0.425	1	1	0.8	-0.15	0.909	unchanged	0.818	$sensor_i$
0.909	0.8	1	0.8	0.818	0.959	increased	0.918	$sensor_i$

Table 1: Examples for grid cell updates

The *contribution* is a measurement of the consistency of the sensor measurement with the new *cell.occ* value. If the *contribution* is above a certain threshold, the contributing sensor becomes the new owner of the cell. Otherwise the *cell.owner* value will be reset to *unknown*.

Table 4 gives examples for updating grid cell values by sensor measurements. The threshold values had been chosen to 0.5. In the first case, the sensor measurement and the grid cell value confirm each other. The result is an increased confidence for the sensor that originally contributed to this cell (the *owner*) and the sensor that produced the new measurement. In this example the sensor becomes also the new *owner* of the entry. In the second case, the sensor's measurement does contradict the grid value – the sensor reports free space while the grid cell value is sure about an object. Thus, the confidences of the involved sensors are decreased. Case 3 shows a less severe contradiction, because the grid cell is not quite certain about its content. Hence mainly the new measurement influences the updated grid value. Case 4 shows again a measurement that confirms the grid value and leads to a rise of the sensors' confidences. Thus, the sensors' confidence values are dynamically updated according to the comparison of their measurements to the grid. A bad performing sensor will subsequently loose confidence and eventually drop out of the set of contributing sensors. However if the sensor recovers, it will gain confidence again by repeated confirming measurements.

The approach works with at least three sensors whereof one sensor might be faulty at one time. In comparison to the node level approach discussed in the last section the proposed method gains extra sensor space, because the sensor views must overlap only partially. There must be at least one grid cell, which is served by all three sensors.

The extra amount of memory for the grid representation is the storage for the owner values, thus

$$\frac{\lceil log_2(n_{sensors}+1)\rceil}{8} \cdot gridheight \cdot gridwidth, \tag{2}$$

more bytes of memory, where $n_{sensors}$ is the number of sensor contributing to the grid. The memory requirements for the confidence values can usually be neglected, if the number of sensors is remarkably lower than the total number of cells in the grid. Thus, the memory requirements of the robust certainty grid algorithm are considerable less than the memory consumption of the fault-tolerant approach at grid level discussed in Section 3.

In contrast to Bayes' formula, the *AddToGrid* procedure is not commutative. Thus, when a grid cell is updated by subsequent measurements, the order of updates makes a difference in the result. This can be explained because we change the a priori probabilities for the sensors with each update. We overcome the disadvantage of sensitivity to message ordering by applying time-triggered architecture. Time-triggered communication and computation ensures a predictable order of message tasks by avoiding race conditions.

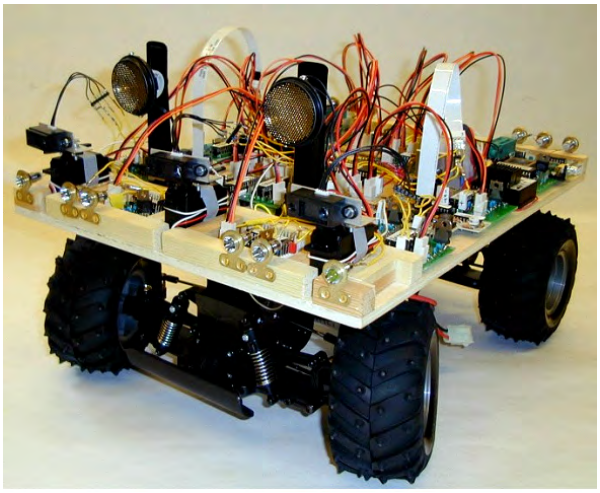
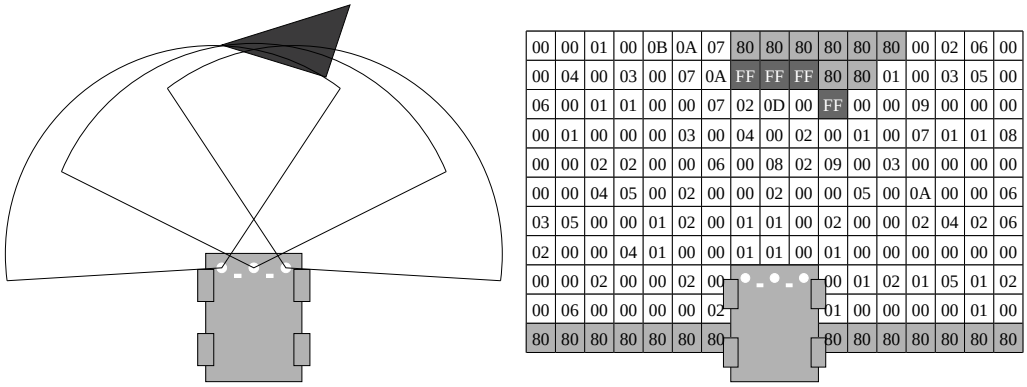


Figure 3: Smart Car: Autonomous mobile robot with pivoting sensors



00	00	01	00	0B	0A	07	80	80	80	80	80	80	00	02	06	00
00	04	00	03	00	07	0A	FF	FF	FF	80	80	01	00	03	05	00
06	00	01	01	00	00	07	02	0D	00	FF	00	00	09	00	00	00
00	01	00	00	00	03	00	04	00	02	00	01	00	07	01	01	08
00	00	02	02	00	00	06	00	08	02	09	00	03	00	00	00	00
00	00	04	05	00	02	00	00	02	00	00	05	00	0A	00	00	06
03	05	00	00	01	02	00	01	01	00	02	00	00	02	04	02	06
02	00	00	04	01	00	00	01	01	00	01	00	00	00	00	00	00
00	00	02	00	00	02	00				00	01	02	01	05	01	02
00	06	00	00	00	00	02				01	00	00	00	00	01	00
80	80	80	80	80	80	80				80	80	80	80	80	80	80

(a) Obstacle in front of car

(b) Representation in certainty grid

Figure 4: Fusion of measurements into the certainty grid

5 Case Study

We implemented the robust certainty grid in a mobile robot for demonstration purposes. The mobile robot comprises a model car (“smart car”) equipped with a suit of pivoted distance sensors, two ultrasonic sensors pointing straight forward, an electric drive, and a steering unit (see Figure 3).

In [9] we showed that time-triggered communication networks are apt to implement real-time sensor fusing applications. Therefore we used a time-triggered sensor fusion model [10] where all communication instants and computation tasks are a priori planned in a time-triggered schedule.

We used a TTP/A network to interconnect distance sensors, servo motors for sensor pivoting, driving, and steering units. Each unit is a separate TTP/A node implemented on a low-cost microcontroller and equipped with a smart transducer interface [11].

The network also contains a master node and a data processing node. The distance sensors are swivelled around by servo motors enabling them to scan the area in front of the robot. The sensors generate a value that corresponds to the distance of the object they are aimed at.

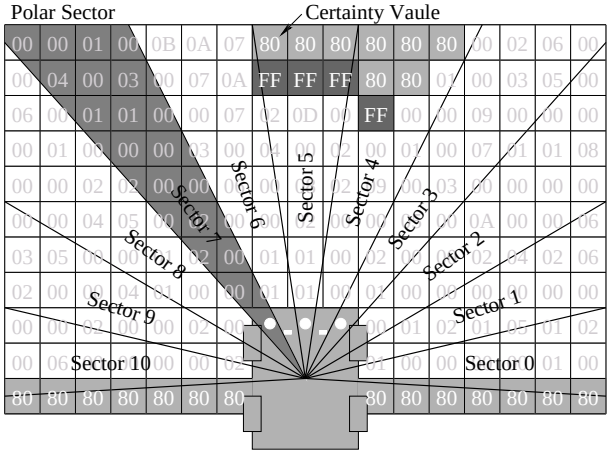


Figure 5: Dividing the certainty grid into polar sectors.

The data stream provided by the distance sensors is taken over by the data processing node that fuses the perceptions from the distance sensors with a model of the robot's environment, using the robust certainty algorithm described in the previous section. Thus, the shapes of obstacles are stored and assigned with a probability value. We added a function that moves all values up to the *uncertain* state with the progression of time. So, an object has to be re-scanned periodically.

We used 8-bit values to express the probability values between 0 and 1. Thus 0x00 corresponds to the $free$ state, 0x80 means the *uncertain* state while 0xFF is used to express the *occupied* state of a grid cell. Fig. 4 depicts an example of a grid allocation.

Movement decisions about direction and speed are made based on a vector field histogram derived from the certainty grid [12]. The vector field histogram is a method to determine the direction with the lowest obstacle density. The algorithm uses the grid representation of the environment which is provided by the cluster level.

The vector field histogram approach divides the certainty grid into disjoint polar sectors S_k (see Figure 5).

For each cell C_{ij} in a give sector an obstacle vector m_{ij} is calculated. The magnitude of m_{ij} depends on the certainty value occ_{ij} of the cell and also of the distance d_{ij} between the vehicle and the respective cell:

$$m_{ij} = (occ_{ij})^2(a - b \cdot d_{ij}) \tag{3}$$

with $a - bd_{max} = 0$, a and b are positive constants and d_{max} is the distance between the farthest cell and the vehicle.

Hence, the sum over all obstacle vectors m_{ij} in sector S_k forms an obstacle density entity h_k.

$$h_k = \sum_{C_{ij} \in S_k} m_{ij}, \qquad k = 1, \ldots, n. \tag{4}$$

At this point the entities $h_1, h_2, \ldots, h_n$ are used to form a histogram (see Figure 6), which can be used for obstacle avoidance.

High magnitudes in the histogram indicates regions with high obstacle density, while areas with low magnitudes indicate regions with low obstacle density. By applying a threshold to the histogram it is possible to localize regions with low obstacle density, which can be used for obstacle avoidance. The car will then be moved towards the most promising direction. Moving or turning

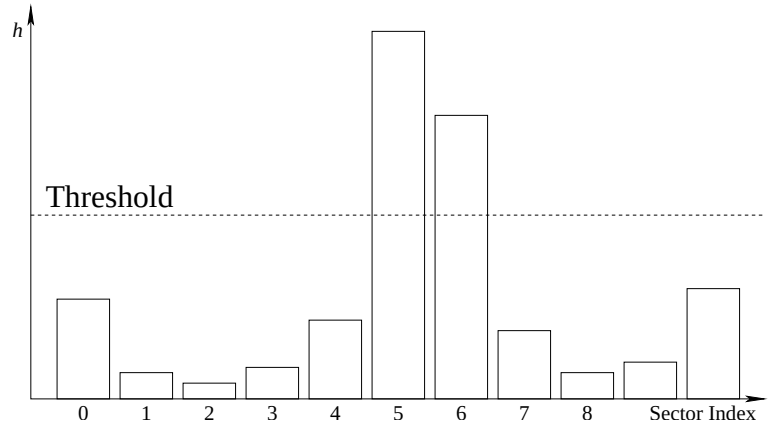


Figure 6: Histogram of the obstacle density.

of the car affords a correction of the grid values. Since the grid contains only 17×11 values the shift and rotate operations can be applied to the grid in real-time.

The smart car is able to move autonomously through its environment. Additionally, it has an interface to a service point for monitoring and configuration access. Via this monitoring interface it is possible to watch snapshots of the certainty grid or the sensor confidence values. The non time-critical service communication is routed concurrently to the real-time system that receives the sensor values and maintains the grid and confidence values. It is possible to route the service access over different networks, e. g. the internet. We plan to implement a wireless connection between the car and the service point. Currently the car is connected via an RS232 serial interface for monitoring and configuration access.

6 Conclusion

The class of certainty grid algorithms are qualified for mapping sensor information of mobile robots into a concise description of the environment.

The previously published certainty grid algorithms can tolerate occasional transient sensor errors and crash failures but will fail for permanent sensor faults.

We developed a method for sensor validation that detects abnormal sensor measurements and adjusts a weight value of the corresponding sensor. Recovered sensors are reintegrated automatically. This robust certainty grid approach supports also sensor maintenance, because it provides a measurement for the operability of a sensor.

The robust certainty grid will be implemented in a mobile autonomous robot with a time-triggered communication architecture.

Acknowledgments

This work was supported in part by the Austrian Ministry of Science, project TTSB and by the European IST project DSoS under contract No IST-1999-11585.

References

[1] H. Kopetz. *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[2] A. Elfes. A sonar-based mapping and navigation system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1986.

[3] M. C. Martin and H. P. Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, The Robotics Institute, Carneghie Mellon University, Pittsburgh, Pennsylvania, 1996.

[4] T. Bayes. Essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53:370–418, 1763. Reprinted in Biometrika (1958) 45, pp. 293-315.

[5] A. Hoover and B. D. Olsen. Sensor network perception for mobile robotics. In *Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA*, pages 342–347, April 2000.

[6] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, 22(6):46–57, 1989.

[7] L. Matthies and A. Elfes. Integration of sonar and stereo range data using a grid-based representation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 727–733, 1988.

[8] J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 43–98. Princeton University Press, 1956.

[9] W. Elmenreich and S. Pitzek. Using sensor fusion in a time-triggered network. In *Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society, Denver, Colorado*, volume 1, pages 369–374, Nov.-Dec. 2001.

[10] W. Elmenreich and S. Pitzek. The time-triggered sensor fusion model. *Proceedings of the 5th IEEE International Conference on Intelligent Engineering Systems, Helsinki, Finland*, pages 297–300, Sept. 2001.

[11] H. Kopetz, M. Holzmann, and W. Elmenreich. A universal smart transducer interface: TTP/A. *International Journal of Computer System Science & Engineering*, 16(2), March 2001.

[12] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, June 1991.