An adaptive fuzzy-logic algorithm for measuring and improving autonomous mobile robot sensory performance

OFIR COHEN, EPHRAIM KORACH, YAEL EDAN
Department of Industrial Engineering and Management
Ben-Gurion University of the Negev
Beer Sheva 84105
ISRAEL

Abstract: - In this paper we present an adaptive fuzzy logic algorithm (AFLA) for sensor fusion that enables also to determine unknown sensory performance based on a fast and simple analysis in controlled conditions. The algorithm deals with unknown a-priori sensory distribution and provides the autonomous mobile robot (AMR) three important characters: *efficiency, reliability* and *robustness*. The advantages of the AFLA are presented by comparing it to another adaptive algorithm using a simulation.

Key-Words: - Adaptive fuzzy logic algorithm, Sensor Fusion, Mobile robots, Logical sensors.

1. Introduction

To perform in unknown and dynamic surroundings an autonomous mobile robot (AMR) must acquire and manipulate a rich model of its internal states and operating environments. A single sensor is in general insufficient to provide complete information due to the intrinsic limitations of all sensors and due to the complex environment. Furthermore, on-line sensory and system performance must be measured so the robot can react to the changing conditions. Sensor fusion is therefore necessary to integrate data from a multitude of distinctly different sensors needed to extract parameters and states for robot self-location, map making, path computing, motion planning and motion execution [8,12]. Many algorithms for distributed sensor fusion were developed and implemented [1,3,11]. However, most of them assume known statistical parameters, namely the *a-priori* probabilities of the sensors. In unstructured environments, it is very difficult to obtain accurate statistics of the sensors or environment since in many cases knowledge is incomplete or partial [2,4,10]. Recently the idea of feedback has been employed for sensor fusion in distributed systems without memory [13] in order to optimize the fusion rule. In [13] the fusion center feeds its decision to each of the local sensors, where, each sensor make its new decision based on the feedback and its last decision. This system requires a-priori knowledge of the false alarm probabilities. In this paper we present an adaptive fuzzy logic algorithm (AFLA) for sensor fusion, which enables also to determine unknown sensory performance. This enables to measure and rank the sensors according to their on-line measured performance. By using this method the AMR can efficiently use its resources and identify when one of the sensors malfunction. The advantages of the AFLA are presented by comparing it to another adaptive algorithm [5].

2. Sensor fusion algorithm

2.1. Methodology

The AFLA was developed using three basic concepts: *logical sensors* (LSs), *grid map* paradigm and *performance measures*. The need for describing the area surrounding the AMR using sensors that have unknown noise distributions is achieved by the *logical sensor* and *grid-map* concepts. The *performance measures* are used to calculate on-line the LSs uncertainty and enable response according to changing environmental conditions.

Logical sensors enable to isolate sensor fusion evaluation from the actual semi-real world [14]. No accurate a-priori modeling of the sensors or environment is needed. This enables easy fusion of multiple and different types of sensors (e.g., vision and sonic) and algorithms (e.g., different algorithms for identical data) [14]. Sensors can be added or upgraded with no need to change the whole system's concept.

One of the common methods for modeling AMR environments is the *grid-based* method [7]. The *grid map* paradigm represents the environment by evenly spaced grids. Each grid cell may, for instance, indicate the presence of an obstacle in the corresponding rein of the environment. The LSs map size is determined as the area covered by the physical sensors in the direction in which the AMR advances.

The *performance measures* were defined in a previous work [6] and represent the difference between the final map (the outcome of the fusion process) and the LS's map. They enable to measure the quality of the different LSs at each fusion step

2.2. Information flow

The environment is represented by a local grid map constructed as a two-dimensional binary array in which 1 represents occupied cells and 0 represents free cells. Each LS reads the data from the environment and describes it in its own 2D binary array. This map contains '*noise*' that represents the probability of 'mistakes' for each LS (i.e., this noise is a parameter determined by a value of 0% to 100%; a probability of 0% implies no noise and 100% means the LS 'sees' every thing vise versa). Since sensor acquisition is a synchronized process and sensor fusion is asynchronous, whenever one of the LSs maps is updated, the adaptive fuzzy logic fuses the LSs maps into one global binary map. The domain of the GBM covers the same area covered by all the LSs. The AFLA takes as inputs all LGMs and fuses the data into a global binary map (GBM) considering the sensors performance measures (Fig. 1). This is done each time the LGMs are updated. Performance measures are calculated by comparing the global binary map with the LGM of each LS map according to equations (1-4) [6]. These measures provide the AFLA the relative number of times where both the LS map and the GBM found the cell as 'True' (e.g., TT) and correspondingly for FF, TF and FT.

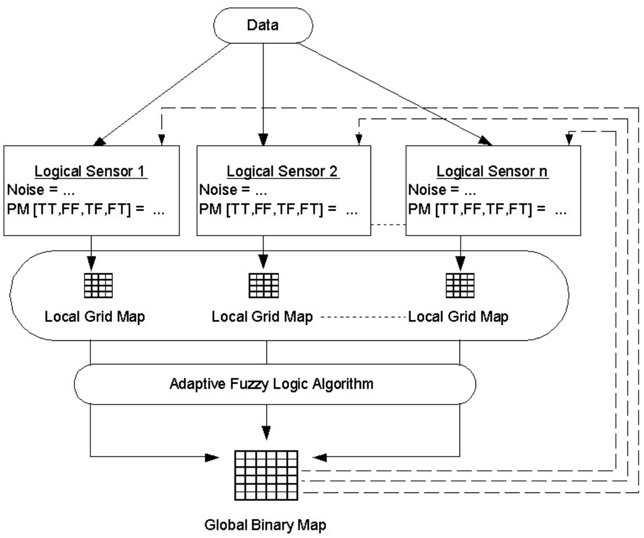


Fig. 1 Information flow

TT≡Number of cells that were signed as obstacles both on the LS and fused maps ≡

$$T_{Logical\ Sensor} / T_{GBM} = \frac{\sum\limits_{NumberOfCells} \left[T_{Logical\ Sensor} \,\&\, T_{GBM} \right]}{T_{GBM}} \tag{1}$$

FF≡Number of cells that were signed as free both on the LS and fused maps ≡

$$F_{Logical\ Sensor} / F_{GBM} = \frac{\sum\limits_{NumberOfCells} \left[F_{Logical\ Sensor} \,\&\, F_{GBM} \right]}{F_{GBM}} \tag{2}$$

TF≡Number of cells that included obstacles on the LS map but were signed as free on the fused map ≡

$$T_{Logical\ Sensor} / F_{GBM} = \frac{\sum\limits_{NumberOfCells} \left[T_{Logical\ Sensor} \,\&\, F_{GBM} \right]}{F_{GBM}} \tag{3}$$

FT≡Number of free cells on the LS map signed but was signed as obstacles on the fused map ≡

$$F_{Logical\ Sensor} / T_{GBM} = \frac{\sum\limits_{NumberOfCells} \left[F_{Logical\ Sensor} \,\&\, T_{GBM} \right]}{T_{GBM}} \tag{4}$$

According to the four measures and the AFLA a truth table is built (Fig. 2). The truth table is an array that contains all possible LS readings (True/False) and the result according to the adaptive fuzzy logic algorithm. For example, if a system contains three LSs, eight combinations of LSs (FFF, FFT, FTF…TTT) are obtained (i.e., the minimum number of LSs the AFLA can deal is three, the maximum number is unlimited). From the truth table and the algorithm, once data from the LGM is received a new GBM is rebuilt (Fig. 2). When the GBM is updated, an average value of the four logical performance measures (TT, FF, TF and FT) is returned as feedback to each LS. The AFLA uses the new values the next time it fuses the LGM's into the GBM.

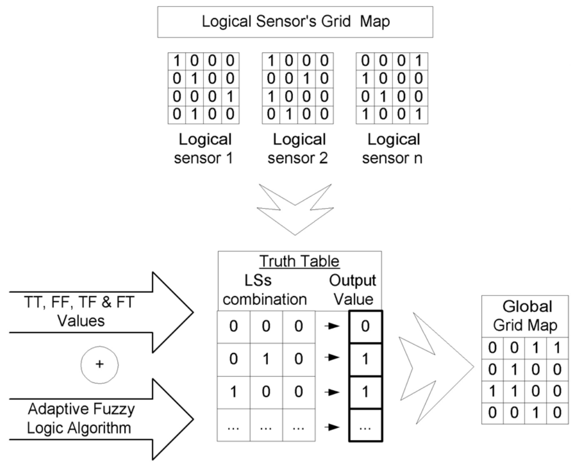


Fig. 2 Building the GBM from the LGM and the four performances measures

2.3. Adaptive fuzzy logic algorithm

The purpose of the algorithm is to reconstruct the truth table whenever the GBM is updated. The algorithm reads the four measures (TT, FF, TF and FT) of each LS and calculates for each rule the 'True' and 'False' values.

The fuzzy associated variables, attributes and linguistic variable, of each fuzzy set (Table 1, Table 2 and Fig. 3) membership functions (MF) values and linguistic variables (LV) were defined empirically and were not optimized. System performance can be improved by changing the rules and/or the membership function values. The MF and the rules were determined using two assumptions. The first is symmetry for 'True' and 'False' values. A 'True' value has equal importance as 'False' value (Table 1 and Fig. 3). The second assumption assumes two groups of rules that refer to the LS decision, 'support' and 'contradict'. These rules correspond to TT, FF (rules 1-6, Table 2) and TF, FT (rules 7-12, Table 2) respectively. After defuzzyfication and based on the sensors combination in the truth table (Fig. 2), the final 'True' and the 'False' values were summed. Based on the following rule, a final result for the specific sensors combination (i.e., 'True' or 'False') is obtained.

$$\begin{aligned}
&\textit{if} \quad \sum_{Sensor=1}^{n} Sensor.True < \sum_{Sensor=1}^{n} Sensor.False \ \ Then \\
&\qquad Cell.Value = False \\
&\textit{else} \\
&\qquad Cell.Value = True
\end{aligned}$$

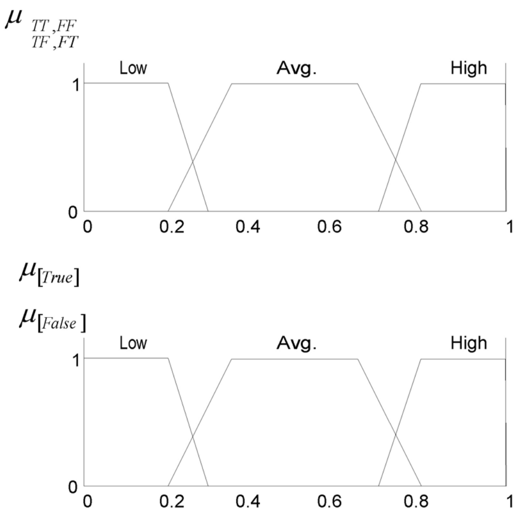


Fig. 3 Membership functions of the system variables.

Table 1 The FSs, their associated variables, attributes and the LVs of each FS

	Fuzzy Sets Values	Fuzzy Set	Attribute	Fuzzy Variable
Inputs	High, Avg., Low	Trend	TT	Low (0,0,0.2,0.3) Avg. (0.2,0.35,0.65,0.8) High (0.7,0.8,1,1)
	High, Avg., Low	Trend	FF	
	High, Avg., Low	Trend	TF	
	High, Avg., Low	Trend	FT	
Outputs	High, Avg., Low	Trend	True	
	High, Avg., Low	Trend	False	

Table 2 Rules for calculating the reliability of the LS units

	Fuzzy Variable Input				Fuzzy Variable Output	
Rule	TT	FF	TF	FT	True	False
1	High				High	
2	Avg.				Avg.	
3	Low				Low	
4		High				High
5		Avg.				Avg.
6		Low				Low
7			High		Low	
8			Avg.		Avg.	
9			Low		High	
10				High		Low
11				Avg.		Avg.
12				Low		High

2.3.1. Algorithm Inputs and Outputs

The algorithm inputs and outputs for each LS are the performance measure parameters and two values 'True' and 'False' respectively. These outputs represent the LS's certainty for identifying signed and unsigned cells within its grid-map. These two outputs values are used as an input for the rule (section 2.3), which with the truth table (Fig. 2) determine the final value (i.e., '0' or '1') for each LSs combination.

3. Sensors performance determination

3.1. General

The purpose of this system is to determine unknown LS's noise.

The procedure for calculating this value contains two parts: (a) *Simulation* (details in section 4.2) that calculated the performance measures. (b) *Semi-real world experiment* (details in section 4.3) to calculate the unknown LS's noise.

The aim of the *simulation* is to build the transition matrices, which represent a unique state of the LS performance (defined in section 3.3) [9] for the unknown LS for different noise conditions. The aim of the *semi-real world experiment* is to calculate the unknown LS's noise based on one of the transition matrices generated in the simulation part.

3.2. Assumptions

- The number of LSs in the simulation and semi-real world experiments should be the same because the TM of each LS combination is unique (section 3.3).
- The number of LSs in the simulation and semi-real world experiments should be odd. This assumption is necessary since in case of disagreement between the LSs the system cannot converge into a stable solution.
- TT and FF have identical values, and TF and FT have another identical value. By definition the sum of TT and FT is equal to one, and the sum of FF and TF is equal to one. This assumption does not influence system performance, however is essential to simplify system analysis due to the infinite number of performance measures combinations.

3.3. Transition matrix

Each LS has four unique transition matrixes (TM) [9], one for each performance measure. Each cell in the matrix defines the probability to transfer from state i to state j (Fig. 4). There is a different transition matrix for each combination of noise level of the other sensors and the sensor itself (i.e., for sensor A of 10%, sensor B of 20% and sensor X unknown). The transition matrix is made of 10201 cells (i.e., from '0' to '1' with steps of 0.01). The shape of the TM depends on three parameters: the total number of LSs in the system, the LSs noise parameters and initial performance measures values for each LS. These parameters enable to define the noise characteristic of each LS precisely.

3.4. Procedure

The procedure for calculating and building the transition matrix is as follows:

i. The noise for each LS is defined.
ii. Each LS builds its map (according to its noise).
iii. Performance measure values for the four parameters are defined (TT, FF, TF and FT), the values runs from '0' to '1', as described in 3.1.
iv. The AFLA creates the fused map.
v. A comparison between the fused and the original map is made. Four new performance measures are recalculated using equations (1-4).
vi. The average value, of the old and the new performance measure is calculated, and stored in an array.
vii. The above procedure, (from creating the noised map (i) until saving it (vi)), runs for 100 times. It is important to run this procedure many times because the generating noise process of the LGMs is random.
viii. After summing data from the array and normalizing each line separately in it, the transition matrix can be determined using linear regression equation (5) and Fig. 4.

The procedure for calculating the performance measures for the unknown LS is:

i. Find two LSs with maximum noise of 10% each.
ii. Determine the initial performance measures for all three LSs.
iii. Make an experiment in semi-real world and calculate the performance measure for the unknown LSs.
iv. The performance measure is returned as feedback and calculated as the average value of the last and the new performances measures. Therefore, the line's slope in the transition matrix is constant an equal to 0.5. One point is necessary to determine the correspond TM (more in section 4.3).

The above procedure, (from (i) to (iii)), runs several times for different initial performance measure values to increase the results reliability.

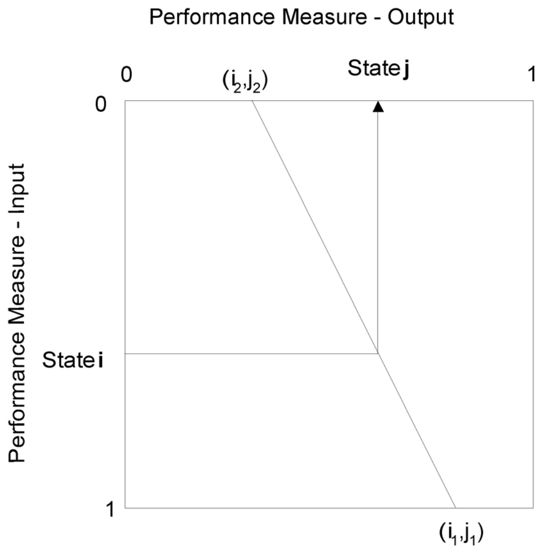


Fig. 4 Transition matrix: Input, Output and straight-line equation.

$$State_j = a \cdot State_i + b$$
$$a = \frac{j_1 - j_2}{i_1 - i_2} \tag{5}$$
$$b = State_j - a \cdot State_i$$

4. System analysis

4.1. General

To analyze the AFLA performance a simulation was implemented with three LSs. The simulation consists of two off-line programs: the *first* is a program in VC++ that generates the data 100 times; the *second* finds the TM using Matlab.

4.2. Transition matrix generation

The simulation's first part is implemented off-line and consists of three sensors in which two of them are of known performance and one is the unknown sensor to be determined. The original map consists of a two dimensional binary array with 100 cells in which half of the cells are set '1' and half of them set '0'. Each LS reads this original map. Noise is added randomly according to the predefined noise parameter. The initial values of the performance measures for the first and the second sensors remain constant during the process, and were defined as TT=FF=0 and TF=FT=1. For the unknown sensor, the noise is constant during the process but the performance measures changed from 0 to 1 with step of 0.01. The noise of the two other sensors changes from 0% to 100% noise with steps of 10%. All noise combinations were checked without repetitions (i.e., noise combination of LS1=0%, LS2=30%, LS3=60% is identical to the case of LS1=30%, LS2=0%, LS3=60%).

The simulation's second part reads the repeated data (i.e., each LSs condition checked for 100 times) and calculates an average value for each input state of the TM (more details in section 3.4).

An example how the algorithm works can be seen in Fig. 5. In this example the TT performance measure is for 3 sensors with two of them with 0% noise, and the third has 0% noise in case (a) and 100% noise in case (b). Case (a) represents an example where the third sensor (with 0% noise) has in the beginning a TT value equal 0 (Pi0), the reliability of the two other sensors is low, and in four iterations reaches a high reliability (Po4=0.9375). An opposite example can be seen in case (b) where the third sensor has a TT value of 1 (Pi0); since the other two sensors are very reliable (with 0% noise) this sensor's reliability decreases to 0.0625 (Po4) within four iterations.

4.3. Noised calculation of the unknown sensor

The noise is calculated using a semi-real world experiment. In the semi-real world experiment the system has three LSs, where for two of them the noise is unknown but must be less than 10%. The operator determines the initial performance measures for all three LSs.

The experiment is operated once each of the LSs reads the **same area** and builds its local grid map. The AFLA then fuses the data from the local maps into one global binary map and calculates for the unknown LS its new performance measures. The experiment is implemented once and a new set of performance measures is accepted for the third LS. By having the line slope (i.e., constant and equals to 2) and a point, which is determined by the input and output performance measures, the TM is be built.

From these TMs calculated in the simulation part, the most suitable TM can be determined and the LSs noise can be found.

Some methods exist to increase results reliability:

i. Implementing the procedure (section 3.4: (i) to (iii)) several times for identical or different initial performance measure values.
ii. Causing the algorithm to converge by implementing the algorithm for three or four times without initiating the performance measure values each time, then using linear regression to pass the best line between the points.
iii. Combining the first and second methods.

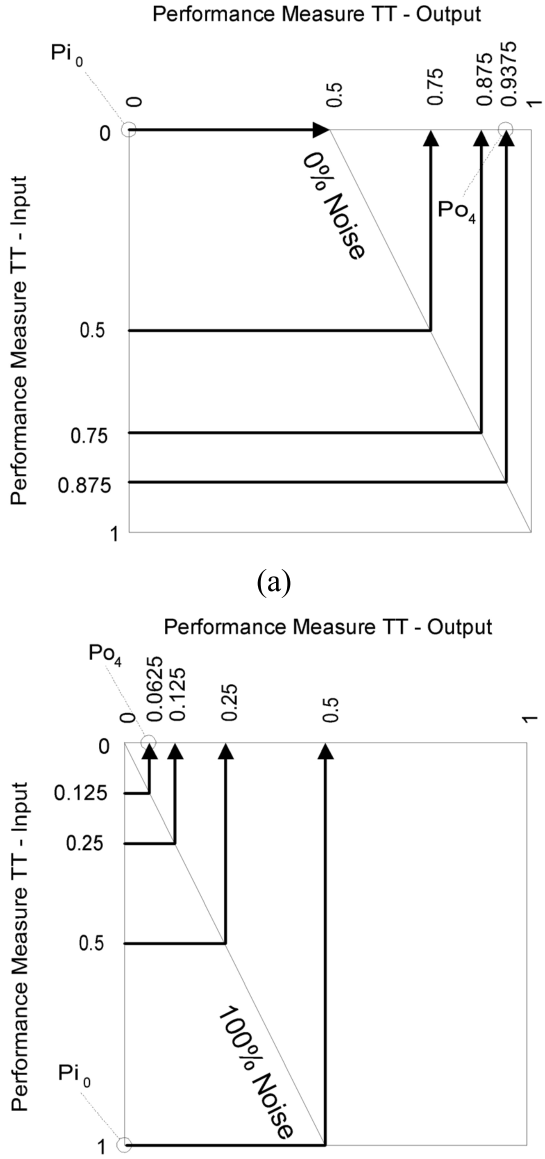


Fig. 5 The third sensor TT measures.
The third sensor has in case (a) 0% noise and in case (b) 100% noise. The other two sensors have 0% noise.

5. Results and discussion

The simulation analysis exposed the *majority* rule. The *majority* rule is similar to the 'MOST' algorithm [5], which means that when the majority of the LSs are 'good', (i.e., each LS has less than 20% noise) the amount of noise for the third LS does not influence the final result (even if it has 100% noise). Accepted results are always good. This phenomenon works in the opposite direction as well, when the majority of the sensors are 'bad' (90% noise) it does not matter how 'good' is the third LS, the accepted results are always bad. An example for this rule can be seen in Fig. 6. In this case, the first and second LSs have 0% noise and the third sensor has different noise values. In Fig. 6(a) the TT and FF performance measures are presented and in Fig. 6 (b) the TF and FT performance measures are presented. In each subfigure eleven simulations are sequentially presented according to the noise of the third LS (the numbers below the figure represent its noise).

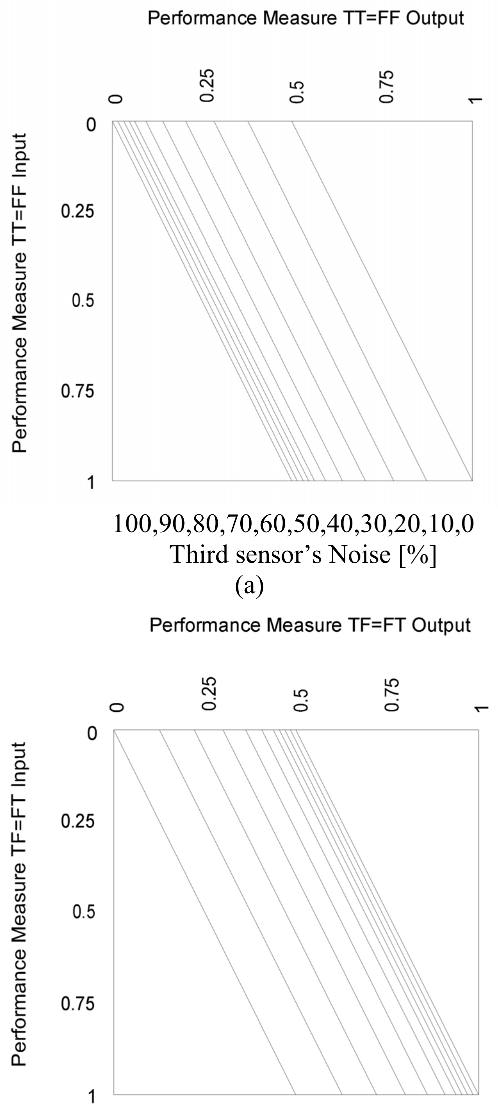
Analysis of the *majority* rule under different noise conditions exposed three phenomenons:

i. The *sequential noise* phenomenon: the noise of the third LS moves sequentially from one side of the matrix to the other.

ii. The *noise density* phenomenon: for the unknown LS the distance between the lines for the TT and FF measures decreases as the noise ratio increases and for the TF and FT measures the distance between the lines increases as the noise ratio decreases (Fig. 6).

iii. The *loop* phenomenon: the noise of the first and the second sensors are within the 20% to 80% interval noise and the third sensors noise is changing. When combining the eleven transition matrices of the third sensor into one image, a loop phenomenon is exposed. First the line goes in one direction and then changes direction as shown in Fig. 7. This means that the reliability of the third LS has its lowest values when the noise is close to limits (0% and 100%). When the noise of the third LS is in between, its reliability increases.

To calculate the LS's noise value, the sensor fusion algorithm (i.e., in this case the AFLA) must have two basic characteristics: *continuity* and *dissimilarity*. The lines in the transition matrices must be continuous but differ in location for each sensor combination. In the Adaptive Dempster-Shafer algorithm [5], which does not have these characteristics, the *dissimilarity* phenomena interfere when part of the lines within the transition matrix is the same for all sensors combinations (Fig. 8, area 'A' and Figure 9). The *continuous* phenomenon interferes by the differences between areas 'A' and 'B', in Fig. 8. The differences between the two figures (Fig. 8 and Figure 9) are not only with the LS's noise but also with their PM's definitions, which differ in the LS initial PM values. For Fig. 8 the initial LSs are changeable but the same for all the three LSs. The initial values for Figure 9 are constant values for the first and the second LS (i.e., TT and FF equals to 1 and TF and FT equals to 0).



100,90,80,70,60,50,40,30,20,10,0
Third sensor's Noise [%]
(a)

0,10,20,30,40,50,60,70,80,90,100
Third sensor's Noise [%]
(b)

Fig. 6 TMs of three sensors, the first and the second sensors with 0% noise, the third sensor's noise change from 0% to 100%.
(a) TT and FF transition matrix.
(b) TF and FT transition matrix.

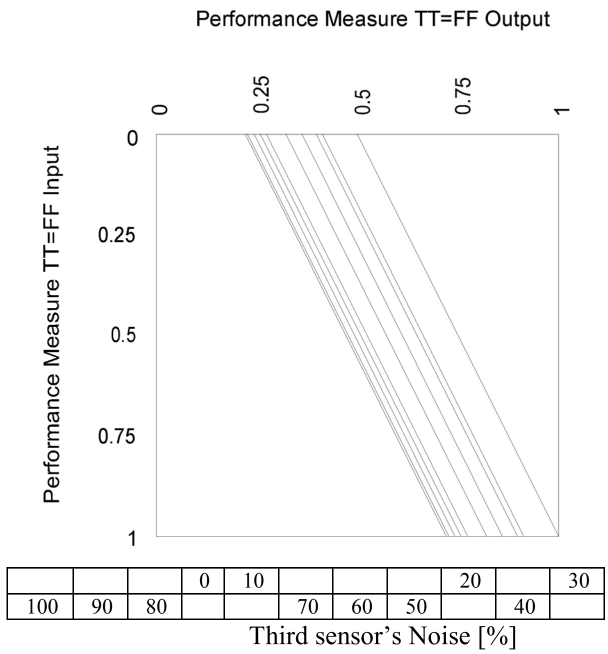


Fig. 7 Simulation results of three sensors, two of them with 30% noise; the third sensor change from 0% to 100% noise.

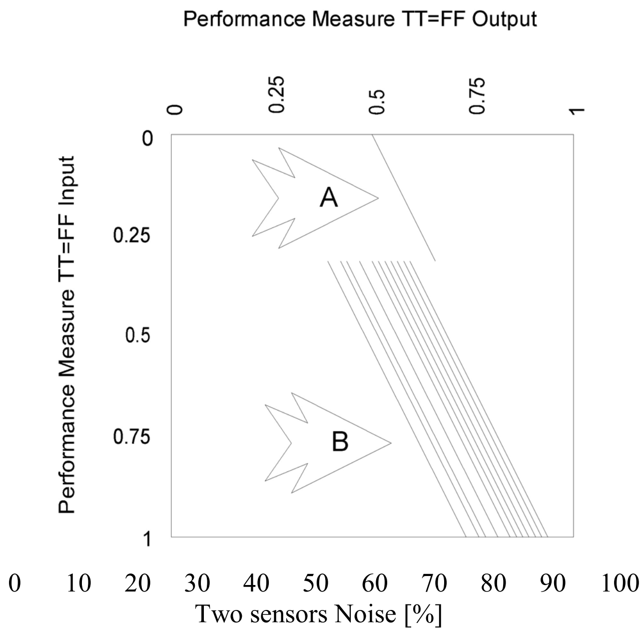


Fig. 8 TMs for ADS algorithm with three sensors, the first and the second sensors with 30% noise.
The third sensors change from 0% to 100% noise.
The initiated PM values for the three LS are the same.

6. Summary and Conclusions

The ability to measure sensory performance under real conditions and characterize them provides the AMR three important parameters: *efficiency, reliability* and *robustness*.

Maintaining the *continuity* and *dissimilarity* characteristics while combining them with the three concepts of *logical sensors*, *grid map* and *performance measures* together in one paradigm has not been implemented. The unique concept and the characteristics of the AFLA make it appropriate for measuring, calibrating and ranking LSs. This enables to react for environment changes and choose the most suitable LS automatically. Practically, malfunctioning or poorly behaved sensor can be detected on-line and suspended until their performance improve. At each step the simplest LS can be employed thus reducing system complexity and providing fast response. Since the LS are analyzed on-line more complex LS can be introduced whenever needed. This ensures high equity performance.

Acknowledgements

This project was supported by the Ministry of Defense MAFAT Grant No. 2647 and partially supported by the Paul Ivanier Center for Robotics Research & Production Management, Ben-Gurion University of the Negev.

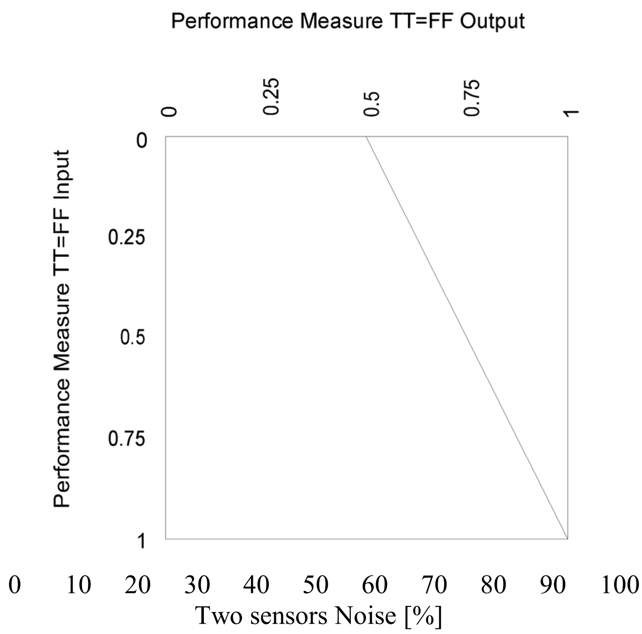


Figure 9 TMs for ADS algorithm with three sensors, the first and the second sensors with 50% noise.
The third sensors change from 0% to 100% noise.
The initiated PM values for two of the LS are the constant (1 for TT and FF, 0 for TF and FT), and the PM for the third LS is changeable.

References

[1] Abidi M. A. and Gonzales R. C., *Data fusion in robotics and machine intelligence*, Academic Press, San Diego, CA, 1992.

[2] Alhakeem S. and Varshney P.K., Decentralized Bayesian detection with feedback, IEEE Trans. On System, Man, and Cybernetics – Part A: Systems and Humans, Vol. 26, No.4, 1996, pp. 503-513.

[3] Brooks, R.R. and Iyengar S.S., *Multi-sensor fusion*. Prentice Hall, New York, NY, 1998.

[4] Chang W. and Kam M., Asynchronous distributed detection, IEEE Trans. on Aerospace and Electronic Systems Vol. 30, No.3, 1994, pp. 818-826.

[5] Cohen O., Multi-sensor fusion for an autonomous vehicle, Ph.D. proposal, Dept. of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer Sheva, Israel, 2000.

[6] Cohen O., Schechtman E. and Edan Y., Sensor fusion quality evaluation tools for autonomous vehicles, Proceedings of the IASTED International Conference, Robotics and Automation 2000, Honolulu, Hawaii, USA, 2000, pp. 192-196.

[7] Elfes A., A sonar based mapping and navigation system, IEEE Journal on Robotic and Automation Vol. 3, No.3, 1987, pp. 249-265.

[8] Estela B., Pierre M. and Gregor S., Target representation on an autonomous vehicle with low-level sensors, The Intel. J. of Robotics Research Vol. 19, No. 5, 2000, pp. 424-447.

[9] Gunter B., Stefan G., Kishor S. T., and Hermann de M., *Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications*, John Wiley & Sons, 1998.

[10] Kushner H. J. and Pacut A., A simulation study of a decentralized detection problem, IEEE Trans. on Automatic Control Vol. 27, No.5, 1982, pp. 1116-1119.

[11] Luo R. C. and Kay M. G., Multisensor integration and fusion in intelligent systems, IEEE Trans. on Systems, Man. and Cybernetics Vol. 19, No.5, 1989, pp. 901-931.

[12] Minoru A., Map building for a mobile robot from sensory data, IEEE Trans. on Systems, Man. and Cybernetics Vol. 37, No.6, 1990, pp. 1326-1336.

[13] Srinivasan R., Distributed detection with decision feedback, IEE Proceedings, Vol. 137, pt. F, No. 6, 1990, 427-432.

[14] Zheng F. Y., Integration of multiple sensors into a robotic system and its performance evaluation, IEEE Trans. on Robotics and Automation Vol. 5, No.5, 1989, pp. 658-669.