ISSN 0005–1144
ATKAAF 50(1–2), 65–79 (2009)

Edouard Ivanjko, Ivan Petrović, Mišel Brezak

Experimental Comparison of Sonar Based Occupancy Grid Mapping Methods

UDK 004.896:681.518
IFAC 2.6; 2.8.1

Original scientific paper

For successful usage of mobile robots in human working areas several navigation problems have to be solved. One of the navigational problems is the creation and update of the model or map of a mobile robot working environment. This article describes most used types of the occupancy grid maps based sonar range readings. These maps are: (i) Bayesian map, (ii) Dempster-Shafer map, (iii) Fuzzy map, (iv) Borenstein map, (v) MURIEL map, and (vi) TBF map. Besides the maps description, a memory consumption and computation time comparison is done. Simulation validation is done using the AMORsim mobile robot simulator for Matlab and experimental validation is done using a Pioneer 3DX mobile robot. Obtained results are presented and compared regarding resulting map quality.

Key words: occupancy grid mapping, mobile robot, Bayesian map, Demspter-Shafer map, Fuzzy map, Borenstein map, MURIEL map, TBF map, sonar

1 INTRODUCTION

In recent years usage of mobile robots in industrial and public areas is increasing and this field of robotics witnesses an increasing amount of research in both low- and high-level mobile robot control functions. Low-level control functions include tasks like localization, path planning, mapping and obstacle avoidance. They can be referred as navigational functions. High-level control functions include tasks like human-machine interaction, action list creation needed to perform the main mobile robot duty, interactions with a higher level control system if the mobile robot is a part of an intelligent space, etc. Those functions are related to mobile robot cognitive abilities. This paper addresses the low-level mobile robot control functions part, i.e., the problems of mobile robot indoor environment mapping. Mapping is one of the important navigation modules, which obtains information about the surroundings of the mobile robot. It also includes changes that occur in the surroundings during the mobile robot motion.

Generally speaking, we can divide environment maps into three groups: metric maps, topological maps, and hybrid maps. Metric maps contain metric information about the environment and are good for localization tasks. Metric information about the modeled environment can be represented by a grid or by features (lines, corners, points). Grid maps are referred to as occupancy grid maps and represent the environment as cells of equal or unequal size, where each cell contains information if the respective environment part is occupied or empty [1]. Advantage of such a representation is that surrounding cells can be modeled as nodes in a graph and their occupancy values as costs for traversing from one environment part to another. So occupancy grid maps contain in its form also topological information about the environment and can be also effectively used for path planning. Topological maps are based on a graph like structure and contain only topological information about the environment. Such maps are good for path planning tasks but mobile robot pose can be determined only approximately. Hybrid maps consist of a topological part and metric part [15]. A typical example is that each node in a graph like representation contains a local metric map. Such maps can be successfully used for localization and path planning in high buildings, where each floor is modeled as a node, and elevator connections between floors as lines.

The mapping problem can be explained as a process of acquiring/generating spatial models of physical environments through collection of local perception sensor data. Perception sensors mostly used for the mapping process are sonars, laser range finders, and stereo-cameras [2]. Sonars or ultrasonic range finders are often used in mobile robotics (many mobile robots have them as a part of their standard equipment) due to their simplicity of operation, robustness, low price, and pretty good measurement accuracy (typically relative error is 1 [%] from the true distance) [3]. So this sensor will be used in this article for map building.

Beside the mentioned sonar advantages, there exist some drawbacks that have to be taken into account when using sonars for map building. Some of the drawbacks are wide main measurement lobe, specular reflections, outliers and crosstalk when more than one sensor is used. To alleviate the influence of these problems the occupancy grid map is mostly build with different mapping methods based on a similar probabilistic sonar sensor model [1, 5]. Other map types like feature based maps are also used but with an increase of mapping method complexity [4]. Different sonar models were also examined in [3] including enhanced filtering of obtained sonar range measurement [9, 16]. Because of higher measurement uncertainty of sonar range readings originating from larger distances, every mapping method uses threshold filtering of obtained sonar range measurements. Such basic filtering is also used in this article, where all range measurements greater than 3 [m] are discarded.

Nature of the mapping problem is dual. To accurately model an environment, exact mobile robot pose has to be known, and to accurately localize the mobile robot environment model has to be known. This relationship defines a whole field in mobile robotics named simultaneous localization and mapping (SLAM) or concurrent mapping and localization (CML). Whence the mobile robot has to move through the mapped environment a path-planning algorithm is also needed. Path-planning algorithms use the so far build map and mobile robots pose information, being so independent from the mapping and localization process. So to build an environment map three processes are needed [6]: (i) mapping, (ii) localization, and (iii) path-planning. In this paper we will concentrate on mapping methods and for the other two processes already implemented standard solution will be used.

Organization of this paper is as follows. Second chapter gives an overview of occupancy grid maps and a description of the basic sonar model. Third chapter describes implemented mapping algorithms and fourth chapter presents obtained simulation and experimental results. Paper ends with an explanation of obtained results and conclusion.

2 OCCUPANCY GRID MAPS

In mobile robotics, an occupancy grid is a two dimensional tessellation of the environment map into a grid of equal or unequal cells. Each cell represents a modeled environment part and holds information about the occupancy status of the represented environment part. Occupancy information can be of probabilistic or evidential nature and is often in the numeric range from 0 to 1. Occupancy values closer to 0 mean that this environment part is free, and occupancy values closer to 1 mean that an obstacle occupies this environment part. Values close to 0.5 mean that this particular environment part is not yet modeled and so its occupancy value is unknown. When an exploration algorithm is used, this value is also an indication that the mobile robot has not yet visited such environment parts. Some mapping methods use this value as initial value. Figure 1 presents an example of ideal occupancy grid map obtained from a small environment. Left part of Fig. 1 presents outer walls of the environment and cells belonging to an empty occupancy grid map (occupancy value of all cells

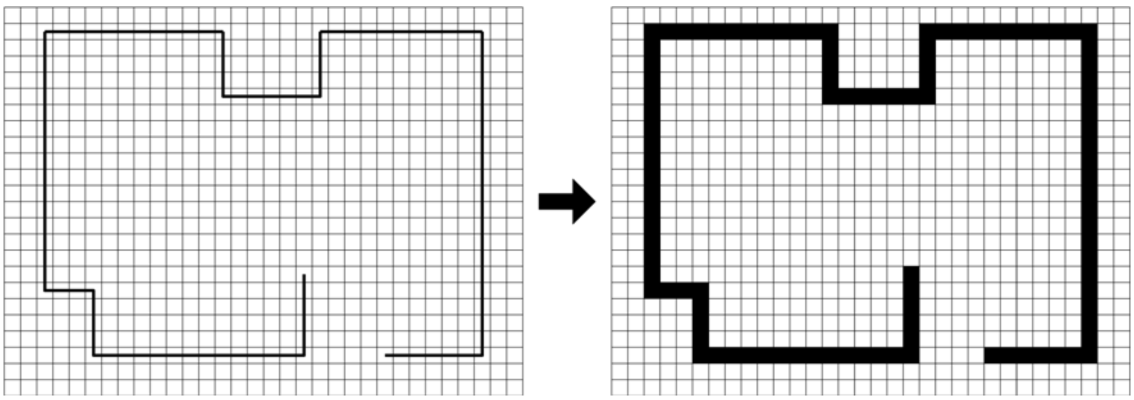


Fig. 1 Example of occupancy grid map environment

set to 0 and filled with white color). Cells that overlap with environment walls should be filled with information that this environment part is occupied (occupancy value set to 1 and filled with black color as it can be seen in the right part of Fig. 1). Result of such operation can be seen in the right part of Fig. 1. It can be noticed that cells make a discretization of the environment so smaller cells are better for a more accurate map. Drawback of smaller cell usage is increased memory consumption and decreased mapping speed because occupancy information in more cells has to be updated during the mapping process. A reasonable tradeoff between memory consumption, mapping speed, and map accuracy can be made with cell size of 10 [cm] × 10 [cm]. Such a cell size is very common when occupancy grid maps are used and will be used in this article also.

Obtained occupancy grid map given in right part of Fig. 1 does not contain any unknown space. A map generated using real sonar range measurement will contain some unknown space, meaning that the whole environment was not explored or that during exploration no sonar range measurement defined the occupancy status of some environment part. An interpretation of sonar range overlapped with corresponding grid cells is given in Fig. 2. It can be seen that in 2D a sonar range measurement can be presented as a part of a circle arc. Size of circle part is defined by the angle of the main sonar lobe and is typical for of the shelf sonar's between 20 and 30 degrees. Therefore, the detected obstacle is somewhere on the arc defined by measured range and main sonar's lobe angle. Usually a one to two cells wide area around the measured range is defined as the occupied space. Space between the sonar sensor and measured range is empty space. Everything else is unaffected with a particular sonar range measurement and presents unknown space. Thus, each sonar range measurement generates a local occupancy grid map that has to be integrated into a global map. When such a local occupancy grid map is created, features of the sonar range sensor have to be considered. The sonar is a time of flight sensor, which means it sends a wave (acoustic in this case) and measures the time needed for returning the wave reflected from an obstacle back to the sonar. Generated acoustic wave has its most intensity along its axis, as denoted in Fig. 2, therefore resulting a more accurate distance measurement of obstacles that are inline and perpendicular to the sonar axis. Whence wave intensity decreases with traversed distance, absolute range measurement accuracy also decreases with wave-traversed distance. This is related with the requirement of the big range measurement which is a longer open time window to accept the reflected wave and therefore enable more specular reflections and outliers. Specular reflections and outliers present in this case false readings, which decrease the quality of the obtained map. To take this sonar range measurement features into account a stronger emphasis is given to the range measurements closer to the sonar sensor and environment parts closer to the main sonar axis. Mathematically this can be expressed with following equations [3]:

$$\alpha(\Theta)=\begin{cases}1-\left(\dfrac{\Theta}{\Theta_0}\right)^2 & 0\le\Theta\le\Theta_0\\ 0 & |\Theta|>\Theta_0\end{cases}\tag{1}$$

$$\Delta(\rho)=1-\frac{1+\tanh\left(2\left(\rho-\rho_v\right)\right)}{2}\tag{2}$$

where $\alpha(\Theta)$ presents angular modulation function i.e., main lobe pattern of the used sonar sensor, Θ angle between sonar axis and currently updated cell, Θ_0 is one half of the sonar main lobe angle, ρ distance from the sonar sensor and currently updated cell, $\Delta(\rho)$ presents radial modulation function and ρ_v presents visibility radius where less emphasis is given to the sonar range measurement. Parameter Θ_0 value depends from the used sonar sensor and for our Polaroid 6500 sonar sensor it is 12.5 [°]. Parameter ρ_v decreases influences of outlier readings and recommended value for an indoor environment is 1.2 [m].

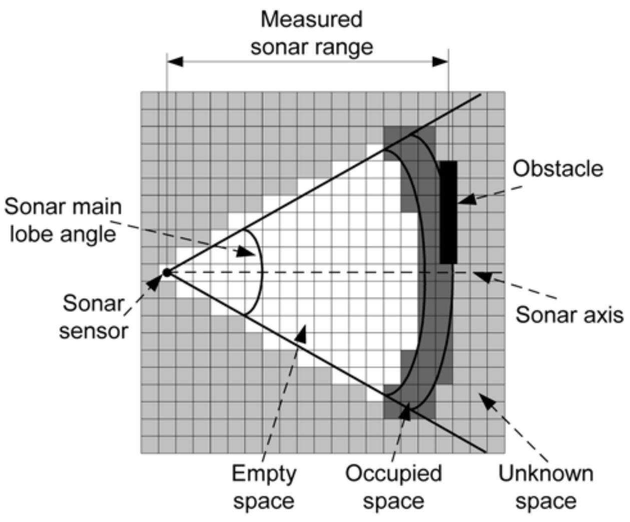


Fig. 2 Interpretation of a sonar range measurement

Described sonar sensor features are taken into account in mapping methods described in this paper. Only the Borenstein occupancy grid map uses a simplified sonar model, which considers only main sonar axis to achieve greater mapping speed sacrifying mapping accuracy [7]. The way in which empty space and occupied space membership values are computed for each cell in sonar main lobe area depends on the used mapping method and will be explained in more detail later.

3 IMPLEMENTED ALGORITHMS

In this section implemented mapping methods will be described including a required memory and computation resources allocation analysis. Also adapted sonar model for each method will be given. Implemented methods that will be described in this chapter are: Bayesian approach, Dempster-Shafer rule, Fuzzy logic map, Borenstein map, MURIEL map, and TBF map. All methods share the same specific steps in the mapping process as presented in Fig. 3.

3.1 Bayesian map

The Bayesian approach of building an occupancy grid G [1, 5] relies on Bayes rule:

$$P(A_i|B) = \frac{P(A_i \cap B)}{P(B)} \tag{3}$$

where in our case the event A_i plays the role of cell being occupied or not, and event B is the examined sonar range measurement. Each cell C_{ij} of the grid map G is therefore associated with a binary random variable S_{ij} with states (O)cupied or (E)mpty for which the following equation is true:

$$P(S_{ij} = O) + P(S_{ij} = E) = 1 \tag{4}$$

To each cell C_{ij} in the Bayesian grid map a state probability p_{ij} is so associated and is initialized as follows (usually the state probability value of 0.5 means that the cell occupancy value is unknown, for values bigger than 0.5 the cell is occupied, and for values smaller than 0.5 the cell is empty):

$$p_{ij} := 0.5 \quad \forall C_{ij} \in G \tag{5}$$

When the Bayesian grid map is updated using new sensor measurements, for each cell C_{ij}, which lies within the main sensor lobe of the reading R, the state probability value p_{ij} is updated according to:

$$p_{ij} := \frac{P(R|S_{ij}=O)\,p_{ij}}{P(R|S_{ij}=O)\,p_{ij} + \left[1 - P(R|S_{ij}=O)\right]\left[1 - p_{ij}\right]} \tag{6}$$

where $P(R|S_{ij}=0)$ represents the sensor model. As mentioned before sensor model consists of an angular and radial part. Part that computes empty and occupancy membership values is in this case augmented with the following equations:

$$P(R|S_{ij}=O) = \begin{cases} 0.5 + (p_E - 0.5)\, f_1(\Theta,\rho) & \rho < R - 2\Delta r \\ 0.5 + (p_E - 0.5)\, f_2(\Theta,\rho,R) & R - 2\Delta r \le \rho \le R - \Delta r \\ 0.5 + (p_O - 0.5)\, f_3(\Theta,\rho,R) & R - \Delta r \le \rho \le R + \Delta r' \\ 0.5 & \rho \ge R + \Delta r \end{cases} \tag{7}$$

with

$$f_1(\Theta,\rho) = \alpha(\Theta)\Delta(\rho) \tag{8}$$

$$f_2(\Theta,\rho,R) = \alpha(\Theta)\Delta(\rho)\left[1 - \left(2 + \frac{\rho - R}{\Delta r}\right)^2\right] \tag{9}$$

$$f_3(\Theta,\rho,R) = \alpha(\Theta)\Delta(\rho)\left[1 - \left(\frac{\rho - R}{\Delta r}\right)^2\right] \tag{10}$$

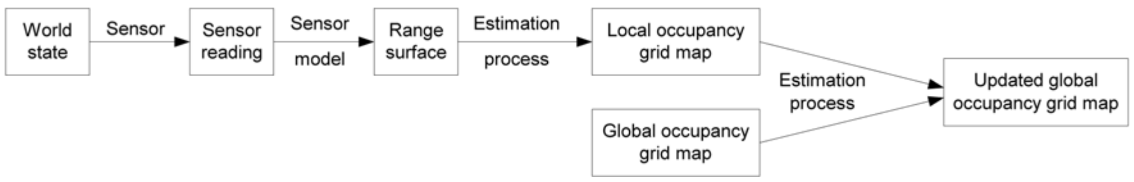


Fig. 3 Occupancy grid mapping process steps

where parameters p_E and p_O set the lower and upper bounds of $P(R|S_{ij}=0)$, $\alpha(\Theta)$ and $\Delta(\rho)$ are the angular and radial modulation functions explained in chapter 2, and parameter Δr defines the area in which the sonar range measurement is considered proximal. Used value of parameter p_E was 0.3, and for parameter p_O value 0.7 was used. Parameter Δr value was 10 [cm]. The same value will also be used for other methods that use this parameter.

It has to be noticed here that in adjustment of the Bayesian rule to the occupancy grid mapping problem two assumptions are being made. First one is related to the independence of any two cell state variables S_{ij} and S_{kl} which is not fulfilled for adjacent cells. For cells that are far apart there is no problem with this assumption. Such an assumption is made to reduce the needed computation resources because conditional probabilities between cells have not to be computed. Number of such conditional probabilities for a square occupancy grid map of $n \times n$ cells is 2^{n^2} and they have to be taken into account during every particular cell occupancy value update without such assumption. Second assumption is that subsequent sonar range measurements are also independent. This assumption is not fulfilled when measurements are taken from a similar mobile robot pose. To justify this assumption a sonar range measurement is only taken into account for a map update if the mobile robot pose change is greater than a certain threshold i.e., the pose change has to be greater than the used cell diagonal size.

Now we can examine the needed computational and memory resources for this type of map. Computational cost can be approximated with the function $O(n \times m)$ where n stands for number of collected sonar range measurements and m stands for number of cells updated by a particular sonar range measurement. Memory consumption is related to the number of cells needed to represent the modeled environment. Whence only one value has to be saved per cell, we can denote this basic amount of memory as OG. In this way we can alleviate comparison of memory consumption and numerical complexity for different mapping methods.

3.2 Dempster-Shafer map

In the Dempster-Shafer theory of evidence, a frame of discernment (FOD), denoted Θ is defined to be a finite set of mutually exclusive and exhaustive propositions. In the grid map case, each cell C_{ij} in the grid map G is characterized by two states (E)mpty or (O)ccupied, and hence the FOD of grid cell C_{ij} is given by:

$$\Theta_{ij} = (E, O) \tag{11}$$

Furthermore the Dempster-Shafer theory relies on a basic probability assignment (bpa) function:

$$m_{ij}^{G} : 2^{\Theta_{ij}} \rightarrow (0,1) \tag{12}$$

where $2^{\Theta_{ij}}$ is the power set of Θ_{ij}, or in our case:

$$2^{\Theta_{ij}} = \{0, E, O(E,O)\} \tag{13}$$

The Dempster-Shafer grid map is initialized as follows:

$$\begin{aligned} &m_{ij}^{G}(O) := 0, \quad m_{ij}^{G}(E) := 0 \\ &m_{ij}^{G}(\{O,E\}) := 1, \quad \forall C_{ij} \in G \end{aligned} \tag{14}$$

When the Dempster-Shafer grid map is updated using new sensor measurements, for each cell C_{ij} which lies within the main lobe of the sensor reading R, the bpa function values $m_{ij}^{G}(O)$ and $m_{ij}^{G}(E)$ are updated according to:

$$m_{ij}^{G}(O) := \frac{m_{ij}^{G}(O)m_{ij}^{R}(O) + m_{ij}^{G}(O)m_{ij}^{R}(\{O,E\}) + m_{ij}^{G}(\{O,E\})m_{ij}^{R}(O)}{1 - m_{ij}^{G}(E)m_{ij}^{R}(O) - m_{ij}^{G}(O)m_{ij}^{R}(E)} \tag{15}$$

$$m_{ij}^{G}(E) := \frac{m_{ij}^{G}(E)m_{ij}^{R}(E) + m_{ij}^{G}(E)m_{ij}^{R}(\{O,E\}) + m_{ij}^{G}(\{O,E\})m_{ij}^{R}(E)}{1 - m_{ij}^{G}(E)m_{ij}^{R}(O) - m_{ij}^{G}(O)m_{ij}^{R}(E)} \tag{16}$$

where $m_{ij}^{R}(O)$ and $m_{ij}^{R}(E)$ represent the sensor model [3]. It can be noticed that two sensor models are used. One model is used for the occupied space part and another is used for the empty space part. They can be expressed with following equations:

$$m_{ij}^{R}(O) = \alpha(\Theta)\Delta(\rho) f_O(\rho, R) \qquad (17)$$

$$m_{ij}^{R}(E) = \alpha(\Theta)\Delta(\rho) f_E(\rho, R) \qquad (18)$$

where evidence functions $f_O(\rho, R)$ and $f_E(\rho, R)$ are given by:

$$f_O(\rho,R) = \begin{cases} 0 & \rho < R - \Delta r \\ k_O\left[1 - \left(\frac{R-\rho}{\Delta R}\right)^2\right] & R - \Delta r \le \rho < R + \Delta r \\ 0 & \rho \ge R + \Delta r \end{cases} \qquad (19)$$

$$f_E(\rho,R) = \begin{cases} k_E & \rho < R - \Delta r \\ k_E\left(\frac{R-\rho}{\Delta R}\right)^2 & R - \Delta r \le \rho < R \\ 0 & \rho \ge R \end{cases} \qquad (20)$$

where variables Δr and ρ have the same meaning as explained above in the Bayesian algorithm description. Parameters k_O and k_E specify maximum evidence support that particular cell is occupied or empty. The value used for parameter k_O was 0.45 and for parameter k_E it was 0.25.

Since this type of map stores two values for a cell (evidence that a cell is being occupied and evidence that a cell is being empty) the memory consumption and computation cost is doubled compared to the Bayesian map approach, which can be denoted with 2OG for memory consumption and $2 \cdot O(n \times m)$ for computational complexity.

3.3 Fuzzy map

When applying fuzzy set theory to grid map building, two fuzzy sets (O)ccupied and (E)mpty are defined [8]. The defined fuzzy sets are complementary, i.e., for a given grid cell C_{ij}, partial membership to occupied cells fuzzy set O and empty cells fuzzy set E is possible. The degree of belonging to occupied cells fuzzy set O, and empty cells fuzzy set E, for each grid cell is measured by two membership functions:

$$\mu_O^G : G \rightarrow [0,1] \quad \mu_E^G : G \rightarrow [0,1] \qquad (21)$$

The Fuzzy grid map is initialized with no membership to occupied cells fuzzy set O and empty cells fuzzy set E according to the following equations:

$$\mu_O^G(C_{ij}) := 0 \quad \mu_E^G(C_{ij}) := 0 \quad \forall C_{ij} \in G \qquad (22)$$

When the Fuzzy grid map is updated using new sensor measurements, for each cell C_{ij} which lies within the main lobe of the sensor reading R, the membership functions $\mu_O^G(C_{ij})$ and $\mu_E^G(C_{ij})$ are updated using the following algebraic sums:

$$\mu_O^G(C_{ij}) := \mu_O^G(C_{ij}) + \mu_O^R(C_{ij}) - \\ -\mu_O^G(C_{ij})\mu_O^R(C_{ij}) \qquad (23)$$

$$\mu_E^G(C_{ij}) := \mu_E^G(C_{ij}) + \mu_E^R(C_{ij}) - \\ -\mu_E^G(C_{ij})\mu_E^R(C_{ij}) \qquad (24)$$

where $\mu_O^R(C_{ij})$ and $\mu_E^R(C_{ij})$ represent the appropriate sensor models [3, 8]. Just like the Dempster--Shafer method, two sensor models are needed and the same equations are used to represent the needed models. Only values of the parameters k_O and k_E are different and were in this case 0.55 and 0.55, respectively. To track safe areas for the mobile robot to operate in, a refined fuzzy set S is formed. The membership function $\mu_S^G(C_{ij})$ of the safe fuzzy set can be defined in two ways. Original approach computes the safe set by subtracting the occupied, ambiguous and intermediate cells from the very empty cells. Ambiguous cells are obtained as an intersection of the occupied (O) and empty (E) cells. Intermediate cells are obtained as an intersection of cells that are neither empty (E) nor occupied (O). It's a more conservative approach where an additional factor, $\mu_C^G(C_{ij})$, weights down membership function $\mu_S^G(C_{ij})$ if contradicting information between $\mu_O^G(C_{ij})$ and $\mu_E^G(C_{ij})$ is present. Mathematically this approach is defined as:

$$\mu_S^G\left(C_{ij}\right)=\left[\mu_E^G\left(C_{ij}\right)\right]^2\left[1-\mu_O^G\left(C_{ij}\right)\right]\mu_C^G\left(C_{ij}\right),\forall C_{ij}\in G \qquad (25)$$

with $\mu_C^G(C_{ij})$ defined as:

$$\mu_C^G\left(C_{ij}\right)=\left[1-\mu_E^G\left(C_{ij}\right)\mu_O^G\left(C_{ij}\right)\right]\cdot\left\{1-\left[1-\mu_E^G\left(C_{ij}\right)\right]\left[1-\mu_O^G\left(C_{ij}\right)\right]\right\} \qquad (26)$$

When smaller number of sonar range measurements is available from the mobile robot, the conservative approach can result in a slow map convergence to sufficient environment representation quality. For this reason a less conservative approach can be used for resulting safe set computation. It can be mathematically expressed as follows:

$$\mu_S^G\left(C_{ij}\right)=\mu_O^G\left(C_{ij}\right)+\mu_E^G\left(C_{ij}\right)- -\mu_E^G\left(C_{ij}\right)\cdot\mu_O^G\left(C_{ij}\right),\forall C_{ij}\in G \qquad (27)$$

It has to be noticed that cells in the safe set S containing higher numerical values have also a higher probability to be empty. This is opposite to other described maps notations, so before it can be used as an occupancy grid map, the obtained safe set has to be complemented. So the final resulting occupancy grid map is in this case the complemented safe set.

This type of map requires two values per cell, and with the computation of the safe fuzzy set $\mu_S^G(C_{ij})$ one more value is needed. This makes this type of map more computationally complex and more memory consuming than the Dempster-Shafer map. In terms of the above used notation, computational complexity can be expressed as $3\cdot O(n\times m)$ and memory consumption can be expressed as 3OG.

3.4 Borenstein map

The Borenstein occupancy grid map is intended for fast mapping with a less emphasis on mapping accuracy [7]. Original development was done for real time fast obstacle avoidance. In this method each cell C_{ij} in the occupancy grid G holds an integer value from the interval [0, 15]. Higher numeric values imply a higher occupancy probability. Map initialization is as follows:

$$g_{ij}:=5 \quad \forall C_{ij}\in G \qquad (28)$$

where g_{ij} denotes numeric value that each cell C_{ij} holds.

With every acquired sonar range measurement R, only cells that lie along the center sonar beam axes are being updated. So when an object is detected, only the cell closest to the midpoint of the sonar range measurement arc becomes a new higher occupancy value. In all other methods all cells that lie on the sonar range arc get a higher occupancy value. An object is so detected by the Borenstein method and can be avoided by path-planning algorithm but obstacle representation in the obtained map can be sparse. Occupancy grid map update is in this case done using the following rules:

$$g_{ij}:=\begin{cases}\max\left(0,g_{ij}-1\right) & \text{if} \quad C_{ij}\in L, C_{ij}\neq C_{ij}^{mid}\\ \min\left(15,g_{ij}+3\right) & \text{if} \quad C_{ij}=C_{ij}^{mid}\end{cases} \qquad (29)$$

where L denotes the center line of the sonar range reading R main lobe, and C_{ij}^{mid} denotes the grid cell closest to the midpoint position of the arc formed by the sonar range reading R.

Whence only one map with one integer type information per cell is created, memory consumption corresponds to the Bayesian map, and is equal to OG. Computational complexity can be presented with the function $O(n\times m)$ but it has to be noticed that the number of updated cells m is in this case significantly smaller then for all other methods. Taking this feature into account, this method has the smallest computational complexity.

3.5 MURIEL map

The MURIEL (MUltiple Representation, Independent Evidence Log) method is designed to deal with outliers in sonar range measurements [9]. Cell occupancy value is determined using a diffuse/ /specular sonar model and a log of all sonar range readings that affect occupancy value of a particular cell. Each sonar range reading defines so an area that has a strong freespace hypothesis and an area that has a strong surface hypothesis. Surface area denotes occupied environment parts. Sonar range readings logs are saved in the so called pose buckets, which consist of a set of angles and distances to the sonar sensor. Used sensor model is based on the following assumptions: (i) occupancy probability distribution function is Gaussian with maximum at the measured distance R, (ii) measurement error increases with measured distance, (iii) obstacle detection probability decreases with distance ρ, and (iv) inside the sonar measurement cone more than one obstacle can be located. Occupancy values are in this model computed using a set of linear beams traversed from the sonar sensor origin that cover the whole sonar range sensor cone. Sensor model can be expressed using following equations:

$$p_{2m}\left(\rho=R|C_i\right)=\frac{\alpha\left(\rho_i\right)}{2\pi\delta\left(\rho_i\right)\sigma}e^{\frac{-\theta_i^2}{2\sigma^2}}e^{\frac{-\left(R-\rho_i\right)^2}{2\delta\left(\rho_i\right)^2}}+F \quad (30)$$

$$P_{2m}\left(\rho\not\subset R|Q\right)=1-\int_0^R\int_{-\pi}^{\pi}p_{2m}\left(\rho=x|Q\right)\mathrm{d}\theta\mathrm{d}x,$$
$$\text{with } Q=C_i \text{ or } Q=\bar{C}_i \quad (31)$$

$$\delta\left(\rho\right)=0.01+0.015\rho \quad (32)$$

$$\alpha\left(\rho\right)=0.6\left[1-\min\left(1,0.25\rho\right)\right] \quad (33)$$

where σ denotes angular sonar lobe width, Θ is the angle between the sonar sensor direction and the cell C_i direction, function $\delta(\rho)$ describes with distance growing measurement error, and $\alpha(\rho)$ presents sonar sensor detection attenuation, F is a small constant that has to be added to indicate existence of more than one obstacle in the sonar range measurement cone. Equations (30) and (31) can be approximated using a lookup table to speed up the mapping process.

MURIEL algorithm can be summarized into following steps [9]:

1. Collection of sonar range readings where each new reading is checked if it duplicates a previous reading. This is done with the use of pose buckets, i.e. if a range reading is done from a same angle and from a same distance as a previous reading it is discarded.
2. A log likelihood ratio for all surface hypothesis readings is computed.
3. A specularity probability is computed for an assumed occupied cell using all freespace readings for that particular cell.
4. A freespace log likelihood ratio is computed for each freespace reading.
5. Final odds value of particular cell occupancy is computed and cell occupancy value is updated.

In this method, for each cell occupancy value, freespace hypothesis value and surface hypothesis value has to be saved. Memory consumption is so equal to 3OG. In addition for each cell pose buckets for freespace and surface range readings have to be saved. Number of pose buckets for each cell depends on the range (variable C_r) and angle (variable C_a) classes' number. This part can so be denoted as $2\cdot C_r\cdot C_a\cdot OG$. Summating this two part follows that this method has the biggest memory consumption. Regarding the computational complexity for this method it is also bigger than the other methods complexity. Here sonar range readings have to be checked for duplicates and logged, then appropriate log likelihood ratios and specularity values have to be computed before the final cell occupancy value update is done, so that computational complexity can be approximated as $4\cdot O(n\times m)$. In an implementation without approximations in the sonar models, computational complexity is even bigger.

3.6 TBF map

The TBF occupancy grid map is developed using the triangulation based fusion (TBF) algorithm [14]. To understand the basic triangulation principle, one needs to consider two sonar readings originating from the same point in the environment. Due to bad angular resolution, after receiving a single sonar reading we can only assume there is an object somewhere along the beam arc defined by sensor direction, its main lobe angle and the range reading. However, by combining two readings that have hit a mutual target we can obtain more precise information about the target's position by finding the intersection point of their associated beam arcs. The beam arc intersection point is the first triangulation point estimate. As the mobile robot moves along its path, triangulation points are found and further refined by combining subsequent measurements originating from the same target. The algorithm can be implemented as a sliding time window, where each column represents a complete scan of readings from all sonars [3]. Used sliding time window size was 25 sonar range readings samples. Each time a new scan is inserted at the beginning of the window, the oldest scan is discarded and the window is then swept backwards searching among all past readings for triangulation partners for every reading from the most recent scan. The output of the algorithm is a set of triangulation points n_t, defined by their position and number of triangulations performed in order to obtain them. Triangulation points supported by a high number of triangulations are highly likely to originate from solid objects in the environment, while sonar readings without triangulation partners are discarded as outliers.

A fairly simple stochastic model, previously presented in [3], was used during the course of this work. This model assumes the axial and radial component of a sonar reading to be independent of each other. It then models the axial component with a Gaussian distribution around the returned range reading $\bar{r}$:

$$r \sim N(\bar{r}, 0.01\bar{r} + 0.01\mathrm{m}) \tag{34}$$

and the radial component with a uniform distribution within the main lobe angle:

$$f_{ij} = \frac{1}{2\pi \det(\mathbf{P_T})} \exp\left\{ -\frac{1}{2}\left[\left(x_{ij}, y_{ij}\right)^T - \hat{\mathbf{T}} \right]^T \cdot \mathbf{P_T^{-1}} \cdot \left[\left(x_{ij}, y_{ij}\right)^T - \hat{\mathbf{T}} \right] \right\} \tag{39}$$

$$\theta \sim U(-12.5°, 12.5°) \tag{35}$$

The given model gives a fairly accurate qualitative description of sonar range readings and has the big advantage that it requires absolutely no modeling of the environment. As shown in [3], it can be successfully used for modeling uncertainty when building occupancy grid maps.

By applying the sonar reading probability model, described by equations (34) and (35) to triangulation point computation, we can obtain the uncertainty associated with each triangulation point [3]. Such a refined triangulation point T, defined by its position (x_T, y_T), triangulation number n_t and covariance matrix $\mathbf{P_T}$ is suitable for creating an occupancy grid model of the mobile robot's environment.

When creating the map, we start off with a grid G, consisting of cells $C_{ij} \in G$ ($i, j = 1, 2, \ldots$), and assign a variable $g_{ij} \geq 0$ to each cell. Initially, all cells are assumed to be unoccupied:

$$g_{ij} := 0, \ \forall C_{ij} \in G \tag{36}$$

When updating the map with triangulation point T:

$$\left\{ n_t, \ \hat{\mathbf{T}} = (\hat{x}_T, \hat{y}_T)^T, \ \mathbf{P_T} \right\} \tag{37}$$

we consider an ellipse E, three standard deviations around the triangulation point and a line L, connecting the triangulation point with the last reading that supports it. The described map-updating algorithm is formally expressed as [3]: $\forall \hat{T}$, such that

$$\sqrt{\rho(\mathbf{P_T})} < 0.1[\mathrm{m}],$$

update all the cells

$$C_{ij} \in \{E \cup L\}$$

according to the formulas:

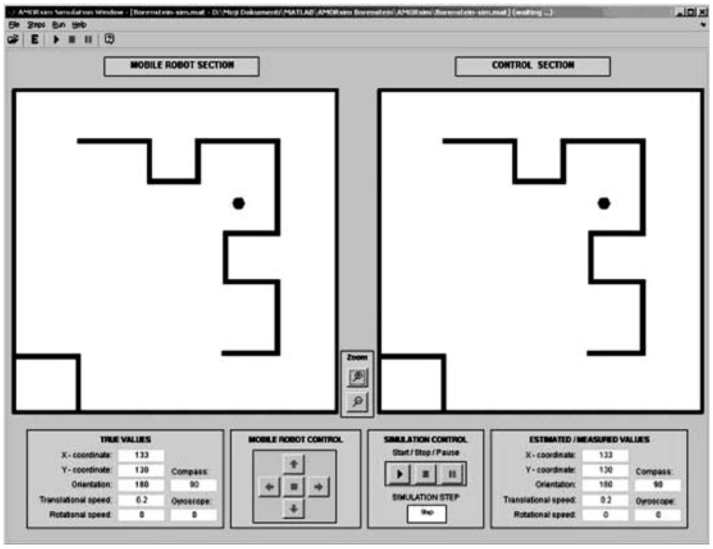
$$g_{ij} = \begin{cases} \max\left(g_{ij}, A|n_t| f_{ij} \ \forall C_{ij} \in E \right. \\ \dfrac{g_{ij}}{2} \ \forall C_{ij} \in L, C_{ij} \notin E \end{cases} \tag{38}$$

where $\rho(\mathbf{P}_\mathrm{T})$ is the spectral radius of the covariance matrix and A is a scaling factor that keeps grid values within the desired interval.

Since a relatively small number of cells need to be updated for every triangulation point, the computation cost of the algorithm is very reasonable. It can be denoted as $O[n_c^2(n_t-1)]$ where n_c denotes number of occupancy grid cells that have to be updated during a particular sonar range reading evaluation and n_t denotes number of triangulation points used for a particular cell occupancy value update. Regarding the memory consumption one occupancy grid map has to be maintained and additionally triangulation points and past sonar readings in the sliding time window have to be saved. So the memory consumption can be denoted as OG + TP + PS where TP denotes memory needed for saving of extracted triangulation points and PS denoted memory for saving of past sonar readings in the sliding time window.

4 EXPERIMENTAL RESULTS

In order to test implemented mapping algorithms, simulation and real world experiments were done. A Matlab based mobile robot simulator was used for simulations and a Pioneer 3DX mobile robot was used for experiments. Simulation evaluation was done primarily for implementation as-



TRUE VALUES
X - coordinate: 133
Y - coordinate: 130
Orientation: 180
Translational speed: 0.2
Rotational speed: 0
Compass: 90
Gyroscope: 0

ESTIMATED / MEASURED VALUES
X - coordinate: 133
Y - coordinate: 130
Orientation: 180
Translational speed: 0.2
Rotational speed: 0
Compass: 90
Gyroscope: 0

Fig. 4 AMORsim simulator graphical interface

pects testing of mapping algorithms, whence the simulator enables creation of various scenarios to thoroughly verify implemented program code but doesn't model all sonar sensor features, such as outliers and specular reflections. Real world experiments were done in a corridor environment that generates a larger number of outliers. Corridor environment also includes a few door niches where for good mapping results it is essential to distinguish regular sonar range measurements from outliers. It has to be noticed that some outliers are filtered out by rejecting sonar range measurements greater than 3 [m]. Locations traversed by the mobile robot during the mapping procedure are denoted as pure empty space, i.e. occupancy value is set to zero. So experimentally obtained maps contain a wide empty space in the middle of the corridor.

4.1 Simulation and experimental setups

Before testing the implemented mapping algorithms on a real mobile robot, the AMORsim mobile robot simulator in Matlab was used for simulation testing [10]. This simulator was developed for research and educational purpose enabling modeling various mobile robot types, environment configurations, and fast implementing of algorithms in the Matlab script programming language. Its graphical interface is shown in Fig. 4. Graphical interface gives a view on the simulated mobile robot in the left »Mobile Robot Section« and a teleoperator view in the right »Control Section«. Under each section relevant mobile robot data are given like mobile robot pose, velocities and proprioceptive sensor measurements during the simulation. Mobile robot control buttons and simulation control buttons are placed in the middle. For simulation purposes a Pioneer 3DX model was made and combined with a navigational algorithm that included path planning and a mapping method. This mobile robot model has also sonar sensors irregularly placed on its body like the real one, where the sonars are condensed on front and rear side of the robot to obtain good forward and backward perception, but on the left and right side of the robot there are parts without any sonar sensor. The goal of the simulated mobile robot was to go around the whole simulated environment and to build a map using collected sonar range measurement. Whence this simulator doesn't model any dynamics, in every simulation step new measurements from every sonar sensor are available. In our case, there are 16 new sonar range measurements in every simulation step. To ensure independence between two subsequent sonar range measurements, new sonar range measurements are taken into account only if the mobile robot displacement was greater that one cell's diagonal or orientation change greater than 10 degrees. Of course such a check is not necessary in case of the MURIEL mapping method. A localization algorithm was not used, because the simulator gives access to the true mobile robot pose. Used simulation world model can be seen in Fig. 5. Whence used simulator does not simulate outliers in sonar range readings a detailed algorithm comparison cannot be done. Therefore, only implementations check and related complexity and memory consumption analysis were done.

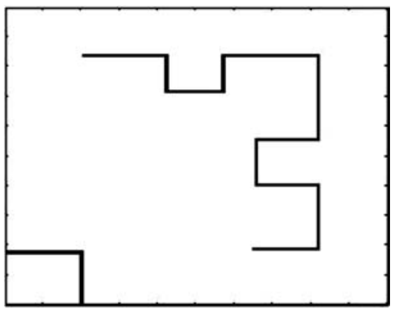


Fig. 5 Used simulation world model

Simulation environment was so chosen to enable a more complex mobile robot mapping trajectory to thoroughly verify implemented mapping framework (measurement data collection, its processing, and final map generation).

After successful simulation in the AMORsim mobile robot simulator all above mapping methods where implemented on a Pioneer 3DX mobile robot (presented in Fig. 6) and tested experimentally in a corridor like environment (presented in Fig. 7). Navigational algorithm is consisted from a path planning, obstacle avoidance, localization and mapping part. Localization was done using the LRF

Fig. 6 Pioneer 3DX indoor mobile robot

sensor measurements and a Monte Carlo algorithm [11]. For path planning and obstacle avoidance a gradient algorithm was used [12]. All examined mapping algorithms where implemented in the ARIA environment obtained from the mobile robot manufacturer MobileRobots Inc. [13]. Implementation was made so that all navigational algorithms worked parallel and after the mobile robot finished traversing from one corridor end to the other, generated map was saved for Matlab evaluation. Mapping part used the estimated mobile robot pose and sonar range measurements. An adjustment in the mapping algorithms implementation is needed to be done because all sonar range measurements are not available in every time sample. New measurements are available every 100 [ms] and that for 2 or 3 sonars only. To ensure subsequent sonar range measurement independence a pose change check was performed like in the simulation case. Also only sonar range readings smaller than 3 [m] were used in the mapping process. Size of the experimental environment was about 6 [m] $\times$ 27 [m].

4.2 Simulation results

Obtained results using the AMORsim mobile robot simulator are given in Fig. 8. Resulting maps are presented with the corresponding grid. Influence of the wide sonar lobe can be observed in walls modeled with a width of more than one cell. Methods that best deal with that are the Borenstein and MURIEL approaches thereby the Borenstein method has a lower mapping quality of environment areas that were positioned at a greater distance from the mobile robot in the moment of sonar range readings collection. This can be explained with the used sonar model in the Borenstein method, which uses only center beam axis of the main sonar lobe. Best results are here achieved using the MURIEL method. TBF method did not manage to map all walls of the environment and some walls are segmented.

4.3 Experimental results with real robot

Results obtained using the Pioneer 3DX mobile robot are presented in Fig. 9. Each figure contains

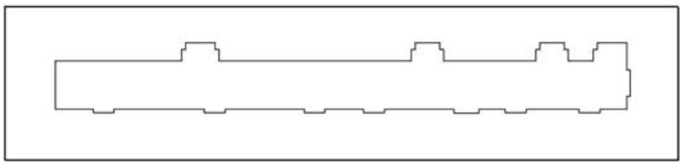


Fig. 7 Model of the experimental environment

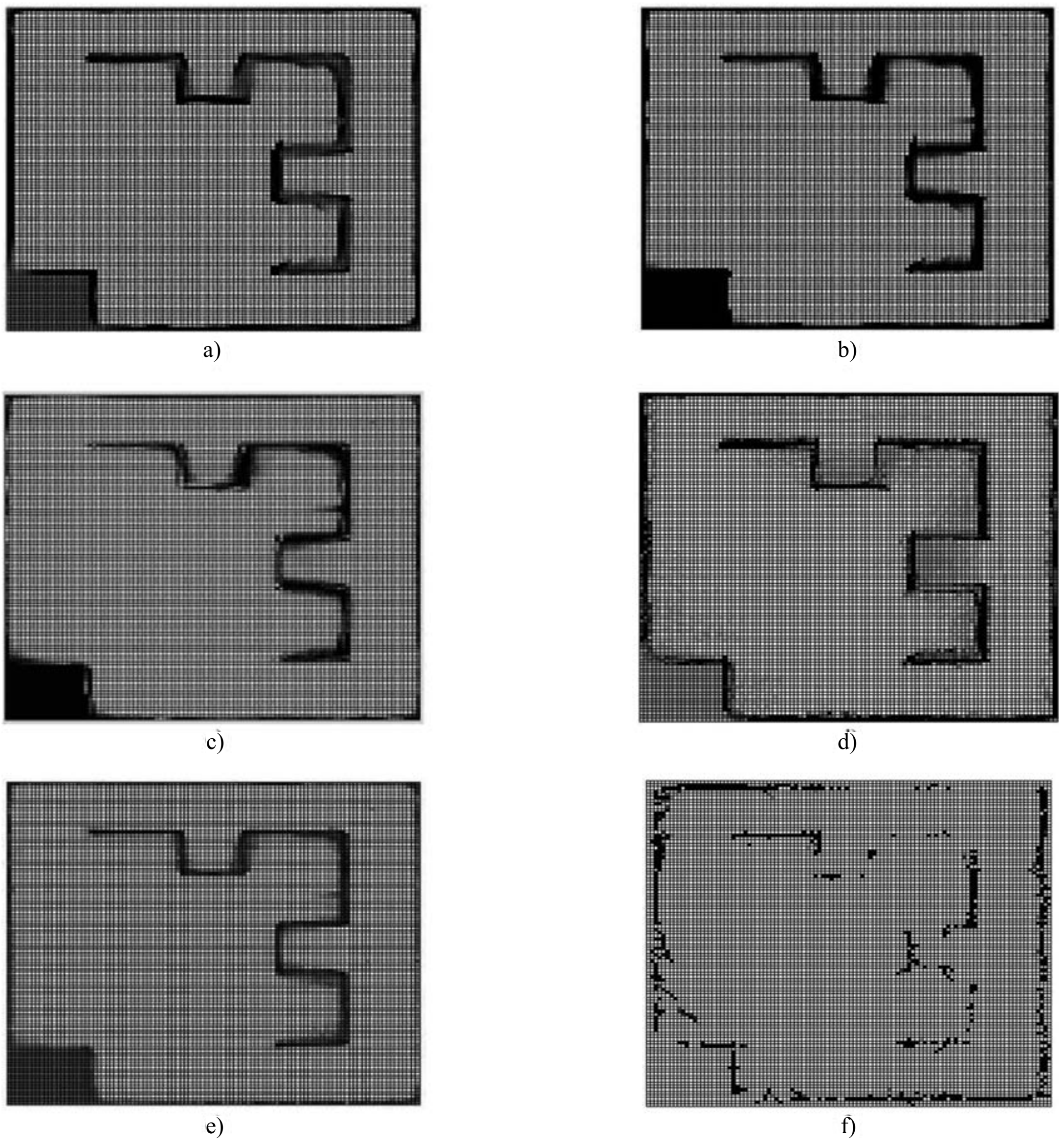


Fig. 8 Simulation results: a) Bayesian map, b) Dempster-Shafer map, c) Fuzzy map original approach, d) Borenstein map, e) MURIEL map, and f) TBF map

obtained occupancy grid model and CAD drawing of the experimental environment. Experimental environment is given to alleviate evaluation of obtained occupancy grid map and its color is adapted to each presented mapping algorithm. In all maps the influence of a much smaller number of taken sonar range readings and the sonar sensor irregular placement on the mobile robot body can be seen. Some areas inside the corridor are not properly modeled as free space. This is especially notable in the case of the Fuzzy map conservative approach (a part of the empty space is not modeled as empty space) and the Borenstein map (gives only a rough contour of the mobile robot environment). These maps are good enough for obstacle avoidance but for a detailed environment map additional data processing is required, like region growing or similar algorithms. Bayes map is of inferior quality in door niches mapping, which is especially notable in the door niches at the bottom of the map. The Dempster-Shafer map gives a good contour of the environment with a few artifacts in the middle of the corridor. The MURIEL map gives also a good contour of the environment, with a few artifacts more in the middle of the corridor. Therefore, Dempster-Shafer and MURIEL maps can be taken

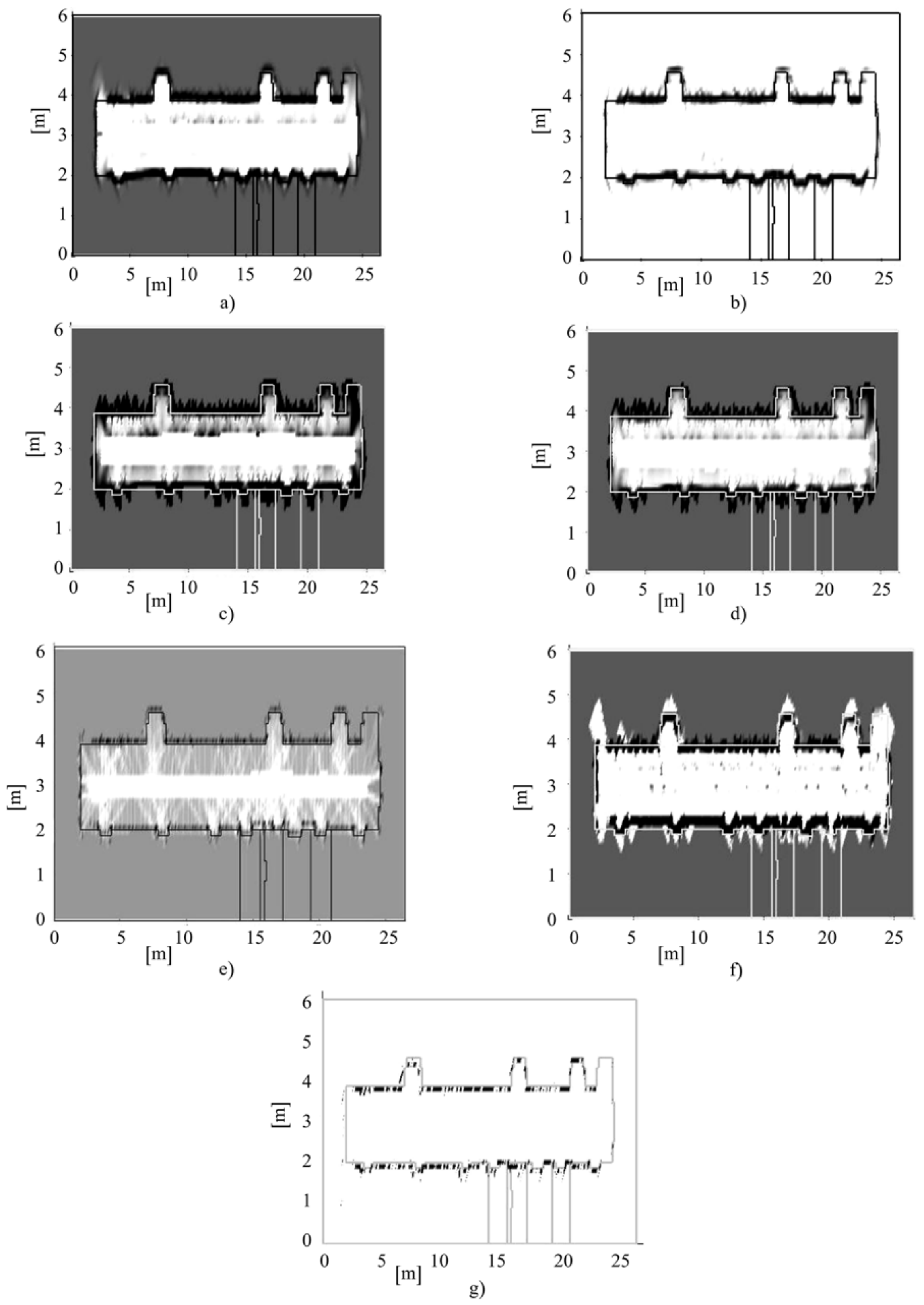


Fig. 9 Experimental results with real robot: a) Bayesian map, b) Dempster-Shafer map, c) Fuzzy map with original conservative approach, d) Fuzzy map with less conservative approach, e) Borenstein map, f) MURIEL map, and g) TBF map

out as the best two approaches. The Dempster--Shafer method has a lower memory consumption and computational complexity but it creates a map of inferior quality when a greater number of outlier measurements are present. This can be observed at the corridor ends. MURIEL method gives good results also at the corridor ends where outlier and contradicting readings influence is the strongest. In the map obtained using this method, a strong influence of the outlier readings can be observed. Around several occupied areas a narrow free space area is presented. That indicates that a false readings firstly marked this area as free space, but measurement from other poses indicated an occupied area. MURIEL mapping algorithm successfully added this new information into the map without creating significant artifacts of occupied space behind the detected walls like in the case of other mapping methods. If a usable map with minimum memory consumption and numerical complexity is needed, then the Bayesian map would be the choice. Fuzzy maps are conservative in integrating new sonar range measurements and are better for mobile robots with more perceptive sensors that can make faster measurements than in the case of the mobile robot used in our experiments. In fuzzy map, environment contour is visible, but around the mapped walls significant number of artifacts that do not correspond to real world environment can be seen. TBF map produced segmented mapped walls like in the simulation case. Also the corridor ends are poorly mapped which can be explained by insufficient sonar readings.

Table 1 summarizes typical average map update time for obtained new sonar range readings set and memory consumption for every implemented mapping method. As expected, the Borenstein method is the fastest and MURIEL method is the slowest. MURIEL method has also the largest memory consumption. Data are taken from the real world experiments. A 2.4 [GHz] Pentium IV computer with 512 [MB] RAM memory running under a RedHat Linux operating system was used. The maximum average update time of the mapping process is 1 [ms], so there is enough computational time left for other algorithms, i.e. robot navigation system. In the case of the used mobile robot, maximum update time for the whole navigational system is 100 [ms]. Regarding the memory consumption, also the types of the variables have to be considered. A floating point variable type was used for the occupancy grid map and an integer variable type for flag notations in the pose buckets and for past sonar range saving in the sliding time window. It's assumed that floating point variable type uses 4 bytes in memory and the integer variable type 2 bytes.

5 CONCLUSION

This article presents a comparison of most common mapping methods for creating occupancy grid maps. Each method was implemented and tested in the AMORsim mobile robot simulator for Matlab. After successful simulation verification of the implemented code in a more complex environment model, the algorithms were experimentally tested using a Pioneer 3DX mobile robot. The experimental environment was a corridor with several door niches. In the case of a sonar sensor, such environment can generate several false readings and outliers. Discarding readings greater than a threshold, some of these readings can be filtered out, but the used mapping algorithm must handle the rest.

Regarding accurate empty and occupied space modeling, two methods can be separated, Demspter-Shafer and MURIEL algorithm. Thereby MURIEL method automatically sorts out redundant readings, but on the cost of a higher memory consumption and numerical complexity. It is also better in filtering outlier measurements. Demspter-Shafer method gives good results in applications where sonar crosstalk measurements are filtered out before the mapping process. If the mobile robot has to fulfill very strict safety demands regarding safe obstacle avoidance, fuzzy logic is the adequate modeling algorithm. Its conservative nature creates good free space model with adequate expansion of obstacles for increasing safety distance.

6 REFERENCES

[1] A. Elfes, **Sonar-Based Real-World Mapping and Navigation**, IEEE Journal of Robotics and Automation, Vol. 3, No. 3, pp. 249–265, 1987.

Table 1 **Comparison of occupancy grid methods resource requirements**

Method	Bayes	Dampster-Shafer	Fuzzy	Borenstein	MURIEL	TBF
Update time [ms]	0.43	0.6	0.65	0.04	1.0	0.6
Memory consumption [kb]	63.3	126.6	189.9	63.3	24000	82.6

[2] Sv. Noykov, Ch. Roumenin, **Occupancy grids building by sonar and mobile robot**, Robotics and Autonomous Systems 55, pp. 162–175, 2007.

[3] O. Wijk, **Triangulation Based Fusion of Sonar Dana with Application in Mobile Robot Mapping and Localization**, PhD Thesis TRITA-S3-REG-0101, 2001.

[4] J. D. Tardos, J. Neira, P. M. Newman, J. J. Leonard, **Robust Mapping and Localization in Indoor Environments using Sonar Data**, International Journal of Robotics Research, Volume 21, number 4, pp. 311–330, April, 2002.

[5] M. Ribo, A. Pinz, **A comparison of three uncertainty calculi for building sonar-based occupancy grids**, Robotics and Autonomous Systems (35), pp. 201–209, 2001.

[6] J.-A. Meyer, D. Filliat, **Map-based navigation in Mobile robots – II. A review of map-learning and path-planning strategies**, Journal of Cognitive Systems Research, Vol. 4, No. 4, pp. 283–317, 2003.

[7] J. Borenstein, Y. Koren, **Histogramic In-motion Mapping for Mobile Robot Obstacle Avoidance**. IEEE Journal of Robotics and Automation, Vol. 7, No. 4, pp. 535–539, 1991.

[8] G. Oriolo, G. Ulivi, M. Vendittelli, **Fuzzy maps: A new tool for mobile robot perception and planning**, Journal of Robotic Systems, vol. 14, no. 3, pp. 179–197, 1997.

[9] K. Konolige, I**mproved Occupancy Grids for Map Building**, Autonomous Robots 4 (4), pp. 351–367, 1997.

[10] T. Petrinić, E. Ivanjko, I. Petrović, **AMORsim – A Mobile Robot Simulator for Matlab**, Proceedings of 15th International Workshop on Robotics in Alpe--Adria-Danube Region, Balatonfüred, Hungary, June 15–17, 2006.

[11] K. Konolige, K. Chou, **Markov Localization using Correlation**, International Joint Conference on Artificial Intelligence, Stockholm, Sweden, July 1999.

[12] K. Konolige, **A Gradient Method for Realtime Robot Control**, SRI International, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000), Kagawa University, Takamatsu, Japan, October 30 – November 5, 2000.

[13] ***, **ARIA Reference manual**, ActivMedia Robotics, LLC., 2000.

[14] O. Wijk, P. Jensfelt, H.I. Christensen, **Triangulation based fusion of ultrasonic sensor data**, Proceedings of 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium, pp. 3419–3424, 1998.

[15] Nicola Tomatis, **Hybrid, Metric-Topological**, Mobile Robot Navigation, PhD Thesis No. 2444, Ecole Polytechnique Federale de Lausanne, 2001.

[16] Kyoungmin Lee, Il Hong Suh, Sang-Rok Oh, Wan Kyun Chung, **Conflict Evaluation Method for Grid Maps using Sonar Sensors**, Proceedings of 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, pp. 2908–2914, September, 2008.

Eksperimentalna usporedba metoda izgradnje mrežastih karata prostora korištenjem ultrazvučnih senzora. Za uspješnu primjenu mobilnih robota u radnim prostorima s ljudima potrebno je riješiti različite probleme navigacije. Jedan od problema navigacije jest kreiranje modela i uključivanje novih informacija o radnoj okolini mobilnog robota u model radne okoline ili kartu. Članak opisuje često korištene tipove mrežastih karata prostora zasnovanih na očitanjima ultrazvučnih osjetila udaljenosti. Obrađeni modeli prostora su: (i) Bayesova karta, (ii) Dempster-Shaferova karta, (iii) neizrazita karta, (iv) Borensteinova karta, (v) MURIEL karta i (vi) TBF karta. Osim opisa, u članku je dana i usporedba implementiranih algoritama prema memorijskim i računskim zahtjevima. Simulacijska provjera napravljena je korištenjem AMORsim simulatora mobilnog robota za programski paket Matlab, a eksperimentalna provjera napravljena je korištenjem Pioneer 3DX mobilnog robota. Također su prikazani dobiveni rezultati uz usporedbu njihove kakvoće.

Ključne riječi: izgradnja mrežastih karata okoline, mobilni robot, Bayesova karta, Demspter-Shaferova karta, neizrazita karta, Borensteinova karta, MURIEL karta, TBF karta, sonar

AUTHORS' ADDRESSES:

Edouard Ivanjko, Ph. D., assistant
Ivan Petrović, Ph. D., associate professor
Mišel Brezak, B. Sc. E., assistant

Faculty of Electrical Engineering and Computing
Department of Control and Computer Engineering,
University of Zagreb, Unska 3, HR-10000 Zagreb

Received: 2009-02-09
Accepted: 2009-04-14