An Introduction to the Problem of Mapping in Dynamic Environments

Nikos C. Mitsou and Costas S. Tzafestas
National Technical University of Athens, School of Electrical and Computer Engineering, Division of Signals Control and Robotics,Zografou Athens, Greece

1. Introduction

Robotic mapping comprises one of the most important problems in the field of robotics. During the past two decades, a large number of algorithms have been proposed in order to solve the problem of constructing valid models of the robot environment. As a result, highly accurate maps of large-scale indoor and outdoor environments have been constructed thus far. There are still, though, much to be done in order to achieve fully autonomous mobile robots capable of mapping any kind of environment (structured or unstructured, static or dynamic).

In this chapter, we discuss the problem of mapping dynamic environments, an issue that remains open and is extremely active nowadays. *Dynamic environments* are real world environments where moving objects (e.g. humans, robots, chairs and doors) change their positions over time. Widespread mapping algorithms developed in the past are based on the assumption that the environment remains static during the robot exploration phase. Thus, these algorithms provide imprecise results when applied in non-stationary environments. The need to map these environments has led in the development of new algorithms that are designed to exploit the dynamics of the environments towards efficient mapping. These algorithms have given so far promising results.

Through this chapter, we examine the problem of dynamic environments through the mapping point of view. Two issues that are strongly connected to mapping are (a) localization, the process of estimating the position and the orientation of the robot and (b) navigation, the generation of valid paths for the robot. They are both of great importance and remain open fields of research especially when applied on dynamic environments. However, in this chapter we concentrate on the mapping problem and refer to the other two problems only when necessary. Our effort is towards providing the ideas behind the algorithms discussed in this chapter and avoid the mathematical details and formulas. We urge the interested readers to consult the referenced papers in order to gain a better insight on the techniques discussed in this chapter.

The outline of the chapter is as follows: In Section 2, we present the mapping problem for static environments, so as to make the reader familiar with the concepts of the mapping problem. Next, in Section 3 we move to the problem of mapping in dynamic environments. More specifically, we discuss the main difficulties of the problem and present a number of methodologies that are common cases for dealing with it. In Section 4, a number of solutions are presented. We explain how artificial intelligence ideas are applied in some state of the art dynamic environment mapping algorithms. We discuss some recently published algorithms that apply statistical methodologies to identify the static and different aspects of the dynamic areas of the environment. Finally, in Section 6, we discuss the open issues and challenges of mapping in dynamic environments.

2. The problem of mapping in static environments

Robotic mapping is the problem of creating a valid model of the robot's environment. The robot explores the unknown environment, collects a number of sensor measurements and creates a spatial representation of its world. This representation might contain three different types of areas:

a. the *static areas*, which are the areas that remained occupied during the robot exploration process (e.g. walls, heavy furniture),
b. the *unoccupied areas,* which are the areas that do not contain any object (e.g. passages) and
c. the *unexplored areas,* which are the areas that we obtain no information about their occupancy.

There exist a number of different sensor types that are used during the mapping procedure and can be divided into two categories: the sensors that are used to identify the internal state of the robot and those that provide information about the robot's environment. In the first category, common examples are encoders, accelerometers, gyroscopes and GPS receivers (for outdoor navigation only) that provide information about the robot's motion and position. Ultrasonic sensors, laser range finders, bumpers, acoustic sensors and cameras are some examples of the second category which are used to detect and identify objects that lay in their field of view.

The main difficulty in mapping lies in the fact that none of the above sensors can be precisely accurate. Also, the reliability of the sensors might depend on external parameters that change over time (e.g. motor encoders accuracy depends on the slippery of the ground which is different on grass and on marble and the ultrasonic measurements will be different when the detected objects are made of glass and of wood). Thus, there will always be errors in the sensor measurements. Unfortunately, small uncertainties in some measurements can yield in huge errors in the estimation of the robot position which will lead in inaccurate maps.

In order to deal with these uncertainties, most algorithms use probabilities to describe both the robot position and the robot's environment. A number of different probabilistic approaches for the preservation of the uncertainties on the robot position and a number of world-modelling techniques can be found in the literature (more on localization and mapping can be found in (Thrun, 2002)). Some common mapping techniques are the occupancy grid (examples in (Gutmann & Konolige, 1999), (Birk & Carpin, 2006)), the geometric maps (examples in (Latecki & Lakaemper, 2006), (Zhang & Ghosh, 2000)) and the landmark representation (examples in (Larionova et al., 2006), (Suau, 2005)). They all have given successful results under specific assumptions on static environments.

Next, we describe in detail the occupancy grid technique, since it is the most common and widely used technique for modelling the robot environment. Moreover, the occupancy grid technique is the most important among the few proposed techniques that can be applied in both static and dynamic environments (there also exist a few works where the static areas of the dynamic environment are modelled using landmarks like in (Wang et al., 2007), (Andrade-Cetto & Sanfeliu, 2002) or using line segments like in (Angelov et al., 2004)).

2.1 The occupancy grid technique for modelling static environments

The occupancy grid algorithm (Moravec & Elfes, 1985) (Moravec, 1988), introduced in 1985, is the first robotic mapping technique. The basic idea of the occupancy grid is to split the environment into a finite number of cells. This way, the difficult problem of estimating the model of the environment is decomposed into a number of easier problems, those of estimating the occupancy state of every cell. So, instead of searching in the space of all possible maps (a difficult task as there exist an arbitrary number of different maps), we search in the space of the occupancy probability for the estimation of the state of every cell (a fairly easier task).

An example of an occupancy grid is depicted in Fig. 1 below, where the color denotes the probability of occupancy (black cells are occupied, white cells are free whereas in case of the grey areas we do not possess any knowledge about their occupancy).

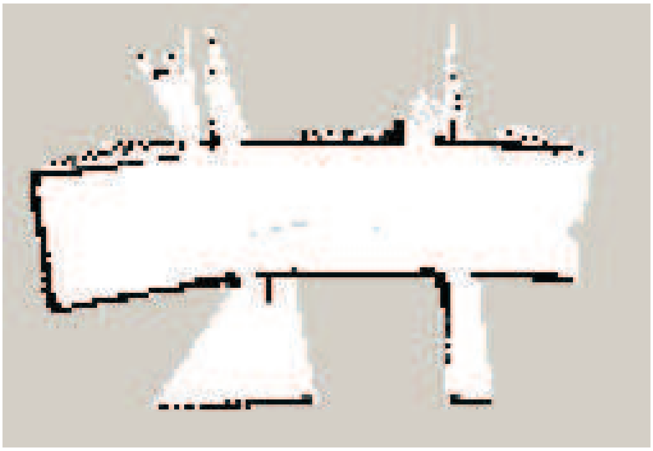

Fig. 1. Example of an occupancy grid map

In mathematical point of view, the occupancy grid algorithm can be formulated as finding the map m of the environment that maximizes the probability:

$$p(m|u_{1:t}, z_{1:t}) \tag{1}$$

where u refers to the control motion commands of the robot and z comprises the sensor measurements received during the interval [1, t].

By following the assumption that the occupancy of a cell is independent of the occupancy of its neighboring cells, the previous probability can be transformed into the product of the probabilities of the occupancy of every cell:

$$p(m|u_{1:t}, z_{1:t}) = \prod p(m_i \mid u_{1:t}, z_{1:t}) \tag{2}$$

where m_i is the i-th cell of the grid.

The assumption of the independency of the values of neighboring cells, though wrong, can be used in order to simplify the required calculations (in (Thrun, 2003), however, the authors overcome this assumption by using forward sensor models).
In order to update the occupancy probability of a cell, we can use the log-likelihood sensor model as in (Arbuckle et al, 2002):

$$R_{i,t} = \log \frac{p(z_t \mid s_i = occ)}{p(z_t \mid s_i = emp)} \tag{3}$$

where $p(z_t \mid s_i = state)$ denotes the probability of the observation z_t given that the state of the cell i equals to *state*. So, the update of the occupancy value R_i of the i-th cell of the grid can be calculated as in the following formula:

$$R_i = \sum R_{i,t} \tag{4}$$

In practice, it has been shown that by using the occupancy grid modeling of the environment, high quality maps of static environments can be generated.

3. The problem of mapping in dynamic environments

In the previous section, we presented a short introduction to the problem of mapping in static environments. However, as already stated, real world environments are dynamic rather than static. For example, people might be walking, robots might be moving, doors might be opened or closed and chairs might be repositioned. Tackling the mapping problem in dynamic environments is a more difficult and challenging task. We do not only have to determine the position of the robot and the map of the environment, but also to identify possible dynamic objects in the environment. Additionally, we should take advantage of this knowledge in order to enhance our perception of the environment and create a better model of the world.
Applying traditional algorithms from the static environments domain like the occupancy grid technique presented above will not create a precise representation of the environment. The reason is that most mapping algorithms continuously update their grids in order to adapt to the current state of the environment, so the dynamic areas will most probably be misclassified either as occupied or as free areas. The place where a chair existed, for example, might be identified as a static area even though the chair had its position changed during the mapping phase and it might be repositioned in a currently unoccupied place in the future. Even worse, there might be cases where the last observed part of an object might be assumed as static while the rest of the object might be assumed as free. A failure example for the occupancy grid technique is shown in Fig.2 below where two doors are partially identified. Their state changed while the robot was partially observing them, so some cells of the grid adapted to the new state while the others remained in the previous state of the door.
So far, we have seen that dynamic areas can lead in the creation of spurious objects in the under construction model of the environment. The existence of dynamic areas will also cause problems in the localization procedure, since matching these areas with the current sensor measurements during a scan match phase will give indeterminable results. The localisation procedure will most probably fail if, for example, the robot tries to match a range scan with the area of a door observed earlier to be closed while now is open.

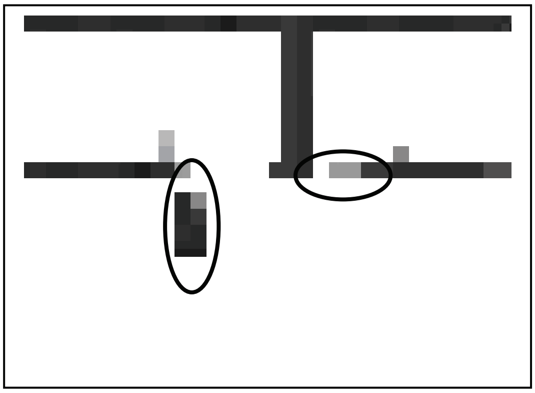

Fig. 2. Failure on mapping two doors. The doors (pointed in circles) are partially drawn.

These problems occur due to the fact that the occupancy grid algorithm does not have the ability to remember the past states of the environment. Every cell stores only one occupancy value which converges to the current observed state of the cell. If this state changes, the occupancy value will adapt to the new state, "forgetting" any past value. Moreover, the false assumption that the cells evolve independently to each other should not be used in this case as the assumption of the independency on the evolution of the cells' occupancies can lead to the problem of a partially modelled object discussed earlier (c.f. Fig. 2.).
In order to deal with these problems, several approaches have been proposed which are based on the occupancy grid structure and either make use of this structure as is or extend it and make use of the modified occupancy grid structure. By using these techniques, one can store more information about the robot's environment (including information about the past as well), which can then be processed in order to extract additional knowledge about the dynamics of the environment.
There exist three basic categories of approaches:

i. *Occupancy grids with different timescales*, which uses a number of occupancy grids where each grid is updated at a different rate.
ii. *Temporal occupancy grid* mapping, which extends the occupancy grid to efficiently preserve the history of the evolution across the time axis.
iii. The *static-dynamic grids* mapping approach which uses two grids to differentiate the static from the dynamic areas of the map.

Below we present each category in detail.

3.1 Occupancy grids with different timescales

In this approach, we use more than one occupancy grids (Arbuckle et al, 2002) (Biber & Duckett, 2005). Every occupancy grid has a particular timescale, which indicates how often it is refreshed (e.g. every 20 seconds). We can store all the grids generated so far in a list or just preserve the grids that are updated in the current step of the algorithm. By examining the occupancy values of the cells at the different occupancy grids (meaning at different timescales) we can conclude on the nature of the cell. For example, we can identify that a cell is static if it is occupied in all timescales or that it is free if it is not occupied at all the timescales. We can also conclude that a cell was occupied by a moving object if it was occupied only at one occupancy grid while on the others it remained unoccupied.

One approach that makes use of this technique is the so-called temporal occupancy grid (TOG) (Arbuckle et al, 2002). It was introduced as a way to classify cells based on their time properties of occupancy. The TOG is an extension to the common occupancy grid through the time dimension. It can be modeled as a matrix with two spatial dimensions, one time dimension and a number of additional dimensions equal to the number of different timescales being considered. So, instead of preserving only one probability value for every cell of the grid, a number of different values are preserved, where each value represents the occupancy of the cell at a specific time period and at a specific timescale. The update of the occupancy value at time t and at timescale Δt is performed using the following formula:

$$\mathrm{R}_{\mathrm{i,t,\Delta t}} = \sum_{t-\Delta t<t'\leq t} R_{i,t'} \tag{5}$$

By examining the occupancy value of a cell at the different timescales we can identify the nature of the cell. If it is occupied at every occupancy grid, then it refers to a *static object*. If it is not occupied in any of the occupancy grids, then it refers to a *free area* of the environment. If it is found to be occupied at some of the occupancy grids, then it was occupied by a *moving object*. It can be modeled as a occupancy grids under specific timescales, we can also extract the path of that moving object.
The main problem of this approach lies in the fact that the optimal number of different timescales, an important parameter of the algorithm, cannot be computed in a formal way but must be intuitively selected by the user.

3.2 Extended temporal occupancy grid

With the structure explained earlier, we do not make efficient use of the available memory, as we preserve more occupancy values than needed. For a particular occupied cell, we store the occupied probability for that cell as many times as the number of the occupancy grids in the TOG structure. In order to make a more efficient use of the occupancy values, an extended temporal occupancy grid (eTOG) has been proposed in (Mitsou & Tzafestas, 2007). Although they share the same name, the two structures are different in their nature. Instead of preserving more than one occupancy grids as in TOG, in eTOG we use only one grid. Every cell of the grid, however, contains an index structure, the so-called time index (Elmasri et al., 1990) that keeps track of the occupancy probability of the specific cell. In this way the complete history of the occupancy probability of every cell is stored in this structure. By using a grid of n x n time indexes to form a forest of time indexes, we can preserve the complete history of the changes of the environment.
Time index is a special case of B+ tree index (a widely used index structure in the database domain) (Ooi & Tan, 2002) that is used for storage and retrieval of values that are valid during specific time periods. The time index was specifically designed for indexing of temporal data. The time dimension is represented using the concept of time intervals. A time interval [t_i, t_j] is a set of consecutive (equidistant or not) time points, where t_i is the first point and t_j is the last point of the interval. A single time point t can be represented as [t, t], i.e. both start and end points are the same. An example of a time index structure appears in Fig.3.
The time index differentiates from the B+ tree index in the fact that due to the monotonic nature of time, deletions never occur while updates occur in an append mode. So, new entries will always be inserted in the rightmost node of the tree and the complexity of the

insertion will always be O(1). When the rightmost node is full, a new node is created and the changes propagate upwards, just as in a B+ tree index structure (an extensive comparison of available time indexes can be found in (Salzberg & Tsotras, 1999)).

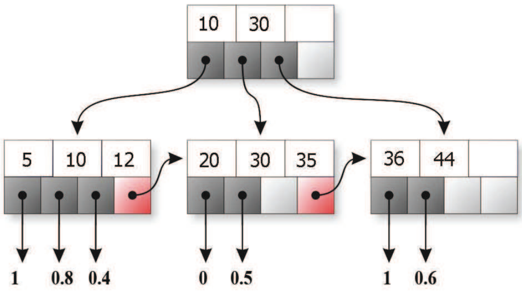

Fig. 3. A time index structure storing probabilities of occupancy

We have to keep in mind that instead of time indexes, we could use simple linked lists if we are not interested in efficiently traversing the occupancy history and we want to avoid the rebalancing cost of the tree.
When a new sensor measurement arrives, the affected cells are extracted and for every cell, the new occupancy value is calculated and inserted into the underlying time index of the cell. The insertion of a new value *v* (assuming that the current time point is t_c) follows specific rules:

- If the time interval is the rightmost node finishes at time point t_j and t_c-t_j> *thres* then a new node is added to the tree, that stores a time interval with starting and ending points at t_c and with value *v*.
- Else if the value *v* equals to the value stored on the rightmost node then update the end point of the time interval to t_c.
- Else if the value *v* is not equal to the value stored on the rightmost node then a new node is added to the tree which points to a time interval with starting and ending points at t_c and with value *v*.

In this way, we can preserve the complete history of the evolution of the occupancy of the cells of the environment. In order to identify the nature of a cell, we traverse the lead nodes of the underlying time index. If the stored probabilities are all equal to the occupied or to the free state, then the cell belongs to a *static object* or to a *free area* respectively. On the other hand, if the probabilities are mixed then the cell has been occupied at some time periods by a *dynamic object*.
The main drawback of this approach is the fact that although in static environment, the memory needed is almost equal to the memory needed by the common occupancy grid, in dynamic environments, the required memory might be extremely large. Indeed, the memory increases with the size of the dynamic effects.

3.3 The static-dynamic grids approach

The most common approach in mapping dynamic environments is the preservation of (at least) two occupancy grids (Tanaka & Kondo, 2006), (Wolf & Sukhatme, 2004), (Wang et al, 2003). In the first grid, only the static areas of the environment are stored while in the second grid, the dynamic objects are preserved. In the first grid, the occupancy probability of a cell

represents the probability of a static entity being present at that cell. In the same manner, a cell in the second grid has an occupancy probability that indicates the probability of a moving object's existence in that cell.
Such a structure is employed in (Wolf & Sukhatme, 2004). In order to specify which areas are static and which dynamic, the authors use a simple differentiation technique. The static parts of the environment never change their position so they can be used as a reference to determine which sensor readings are generated by static and dynamic objects. Two recursive formulas based on the Bayes rule for updating the probabilities of the cells of the two grids are presented. Also, two fuzzified inverse sensor models are introduced. The first models the probability of a cell being static given the current observation and the previous probability of occupancy and the second the probability of a cell being dynamic given the current observation and the previous probability of occupancy. No historical information is preserved about the occupancy of the dynamic map. This means that only the current position of the dynamic objects can be available. In (Wolf & Sukhatme, 2005), a third grid was also maintained, that contained the positions of landmarks and was used during the localization process.

4. Algorithmic implementation of mapping techniques for dynamic environments

In order to develop successful maps for dynamic environments, we should first understand the notion of a dynamic environment map and how it differentiates from the traditional map produced in stationary environments. We categorize the objects in the robot environment into the following three groups:

- *Static objects*, these are the objects that do not change their position over time (e.g. walls, beds or locked doors)
- *Low dynamic objects*, these are the objects that appear in a specific number of places (e.g. chairs or doors).
- *High dynamic objects*, these are the objects that move arbitrarily in the environment and can be found in many different positions (e.g. humans).

Ideally, to successfully map a dynamic environment, we should create different maps for every category of the objects presented above (or one map with three different layers). Firstly, we must create a map of the static objects of the robot environment. This map could be an occupancy grid map of the environment where every cell that is detected to remain occupied during the robot exploration phase is marked as static. There should also be a second map for storing the low dynamic objects of the environment. This map should contain the current state of these objects and also their other possible configurations as they are detected during the sensor acquisition phase (e.g. the chair is positioned in area A and can be found in the areas B and C). Finally, a third map can be created that will show the current state of all the high-dynamic objects of the environment. In this map, we could also mark the path of the moving objects as observed and identified by the robot. The combination of these three maps will create the current state of the environment and will also contain all the knowledge extracted by the sensor measurements.
During the last few years, three lines of research have emerged towards mapping in dynamic environments:

- algorithms that aim in the preservation of *an up-to-date map of the environment* using aging techniques

- algorithms that focus on the creation of the possible different configurations of the environment
- algorithms that map populated environments

The algorithms proposed so far are partial solutions to the problem as they deal with the problem of mapping focusing only on some of the aspects described above. In the following subsections we are going to discuss the most common approaches for every line of research.

4.1 Mapping using aging techniques

In this category, several approaches have been proposed that utilize aging techniques in order to create valid maps of the current state of the environment. Previous sensor measurements are slowly forgotten in order to preserve at any time point a valid map of the current state of the robot's environment. The goal of these algorithms is to create a number of up-to-date models of the world, rather than to differentiate between static and dynamic areas. These methods are adaptive, by means that they adapt the map of the environment so as always to reflect the current understanding of the robot about the environment.

The main difficulty of this category lies in the fact that the under construction map model must be able to adapt to changes of the environment. When a door closes, for example, the map should change the state of the particular cells that refer to the closed door configuration in a fairly short period of time. However, these algorithms should also be able to detect possible sensor errors or the existence of a high dynamic object in the environment (e.g. a human that passes by the door). This means that a false sensor measurement or a measurement of a human passing by a door should not change the state of the corresponding cells. The algorithm must be able to identify the error and ignore the measurement. So, in other words, the generated map must contain the static and the low-dynamic areas of the world but ignore the high dynamic objects.

In (Zimmer, 1995), a system that can dynamically learn and update the topology of a map is presented. The system preserves a model that is able to adapt to changes of the environment. In (Yamauchi and Beer, 1996), an adaptive place network is used to model the robot's environment. The network is able to change the confidence for every link and links with low confidence can be removed from the map. In (Andrade-Cetto & Senafeliu, 2002), landmarks are used to model the map of the environment whose positions are preserved with the use of Kalman Filters. Landmarks can disappear from the map if they are removed from the environment.

The most representative approach in this category of mapping algorithms can be found in (Biber & Duckett, 2005). The basic representation of a map m_{ti} at time point t_i is considered to consist of a set $S(t_i)$ of n measurements. The new map $S(t_{i+1})$ at time point t_{i+1} is calculated by an update rule that depends on the update rate u of the map: Remove $u*n$ randomly chosen measurements from $S(t_{i+1})$ and replace them with $u*n$ randomly chosen measurements received during the time interval $[t_i, t_{i+1}]$.

The environment is split into sub-maps and a number of different timescale Occupancy grid maps for every sub-map are used. Short-term maps, meaning maps that are updated frequently, react quickly to changes and only a few sensor measurements are required to forget an old estimation. On the other hand, long-term maps, meaning maps that have a low update ratio, are updated less frequently and do not react to temporary changes. They only adapt to consistent changes of the environment. These map models have increased accuracy

towards the static parts of the environment as sensor errors are in a way ignored since a large number of samples are required to update an already learned feature of the world.
This algorithm belongs to the same category with the TOG algorithm described earlier. It uses multiple grids of different timescales (although the number of Occupancy grids preserved by TOG is greater) with different however update schema from the TOG update schema. Also, it encounters the same problem with the TOG algorithm, which is the selection of the number of the different timescales that are needed to successfully map the robot's environment. Unfortunately, there is no clear answer to that problem. The more maps we preserve, the better we model the environment as we obtain more models of it (short-term and long-term maps). However, with the increasing number of maps, more memory is needed to store those maps.

4.2 Detecting possible environment configurations

In this section, we will present three algorithms that are used to identify all possible configurations of the environment. These approaches are motivated by the fact that many dynamic objects appear only in a limited number of different positions. Consider for example, the doors in an office environment, which are usually either open or closed. The knowledge of the possible positions of a low dynamic object can be used to enhance the localization process. If the robot identifies a door to be closed then it will expect that this door will not be found also open. A formalistic framework for localization and environment configuration selection can be found in (Stachniss & Burgard, 2005).
The three algorithms presented bellow follow the common occupancy grid assumption, that the occupancy of a cell evolves independently of the occupancy of its neighbouring cells but try to overcome this assumption by post processing the sensor measurements and searching for associations among cells. The first algorithm, *ROMA* uses an Expectation Maximization technique (Dempster et al., 1977) to identify different occurrences of the same objects in the robot environment. The second algorithm, (Stachniss & Burgard, 2005) uses a fuzzy clustering algorithm to identify common areas in a number of different occupancy grid maps. The third algorithm, (Mitsou and Tzafestas, 2007) uses an extended Temporal Occupancy grid and exploits the temporal behaviour of the cells to create group of cells that behave in the same manner. An extended survey on clustering can be found in (Jain et al., 1999).
The first two algorithms use the shape similarity to detect low dynamic objects while the third one makes use of the temporal similarity. The last two algorithms use clustering to extract knowledge from the environment. The former one uses the collected occupancy grids as instances of the clustering algorithm while the last one uses the occupancy evolutions of the cells for the same reason.
Apart from these three algorithms, other algorithms also exist that deal with the same problem. For example, in (Schulz & Burgard, 2001), a probabilistic algorithm is presented to identify the state of dynamic objects in the environment. In (Avots et al, 2002), the authors use particle filters and conditional binary Bayes filters to estimate the state of doors in the environment. In both these works, the environment is assumed to be predefined. Also, in (Anguelov et al, 2004), an Expectation Maximization algorithm is used to detect and model doors. Additionally to the laser range device used in the previous algorithms, images from a camera mounted on the robot are used. Every object in the world has a specific shape and color. The advantage of this algorithm is that an object can be identified to be a door even if it did not move during the experiment (assuming that all doors bare the same color).

4.2.1 The ROMA algorithm

The *Robot Object Mapping Algorithm* (ROMA), also found as *Dynamic Occupancy Grid Mapping Algorithm* (DOGMA), (Biswas et al., 2002) aims at the identification of moving objects and the extraction of their models, at learning in other words, the models of the low dynamic objects of the environment. To learn these models, we assume that the robot maps the environment at different points in time, between which the configuration of the environment may have changed. Each map is represented as a static occupancy grid map.

By using a simple map differencing technique, we can detect the dynamic areas of the environment. The result is a list of “snapshots” of low dynamic objects, each represented by a local occupancy grid map. Two such snapshots of the same low dynamic object (e.g. a chair) might be completely different from each other. The object might be translated and rotated in a different way in the two snapshots. In order to group the snapshots that correspond to the same object, a modified Expectation Maximization (EM) algorithm is applied.

The algorithm uses the following two steps:

a. Step 0: The robot observes the dynamic environment and generates snapshots of the low dynamic objects.
b. Step 1: The EM algorithm is applied:
    a) E step: Create correspondences between the different snapshots based on the estimated models of the objects.
    b) M step: Create the models of the objects based on the correspondences.

The analysis of these steps follows below.

Step 0. Create initial snapshots

The robot explores the environment (which is assumed to remain static or to change slowly) and acquires sensor values. Based on these measurements, the areas of the environment that contain dynamic objects are identified with the use of a simple differencing technique. The areas that are occupied in some maps and free in other indicate low dynamic objects. From these areas, snapshots of these objects are extracted.

Step 1. Apply the EM algorithm

The EM algorithm is applied in order to associate snapshots on different occupancy maps with specific objects.

In the E step of the EM algorithm, correspondences between different snapshots at different points in time are established. Based on the models found on the M step, we associate snapshots at different time points and we estimate the best rotational and translational parameters so that the snapshot will match with the estimated model of the moving object.

In the M step, these probabilistic correspondences are used to regenerate new estimates for the object models. We combine the snapshots that correspond to the same object and we create the most probable model of the object.

By iterating between the E and the M step, we will finally converge to the correct models of the low dynamic objects and their correspondences in the occupancy grid maps. However, as we do not know in advance the number of moving objects in the robot environment, we have to run EM in a number of times with different number of possible moving objects, starting from the lower bound of the number of objects (the maximum number of objects identified in a single map). The number of objects that maximizes the probability of the detected snapshots given the models of the moving objects while minimizes a penalty factor (large numbers of objects are assumed to be less possible) is selected as the best choice. An advantage of the *ROMA* is the fact that it can identify objects that can take any arbitrary position in the environment (assuming that they move slowly). It does not require the objects to be positioned in specific areas, since it is based on the shape similarity of the moving objects. However, the fact that it identifies an object's occurrences based on its shape can lead to incomplete results. Objects that are partially observed (if for example placed next to walls or to other objects) might not be successfully correlated to their other occurrences in the environment.

The Hierarchical Object Mapping algorithm

An extension of the *ROMA* algorithm was presented in (Anguelov et al., 2002). The basic purpose of this work were to extract not only the model of the moving objects but also possible object templates. This approach is able to generalize across different object templates, as long as they model objects of the same type. It uses again an extension of the EM algorithm. It has been shown that this approach performs better than *ROMA*.

4.2.2 Stachniss & Burgard approach

In this approach (Stachniss & Burgard, 2005), the information about changes in the environment during robot exploration was used to estimate possible spatial configurations (examples in Fig. 4). A number of snapshots of the environment at different time intervals were collected and clustering was used to create groups of common snapshots.

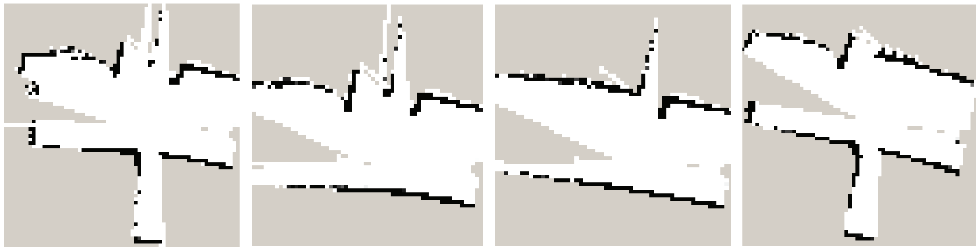

Fig. 4. Different configurations of an environment with three doors.

With this approach, low dynamic objects that lay in the robot environment can be identified. This algorithm assumes that the environment remains static during every time interval of exploration.

The algorithm uses the following two steps:

a. Step 0: The robot observes the dynamic environment at different time intervals and creates a number of sub-areas.
b. Step 1: For each sub-area, we apply a clustering technique to create similar configurations into groups.

The analysis of these steps follows below.

Step 0. Data Acquisition – Sub-areas creation

The robot explores the environment (which is assumed to remain static or to change slowly) and acquires sensor values. Based on these measurements, the areas of the environment that contain dynamic objects are identified. The environment is then segmented into local areas, called sub-maps, so that each sub-map contains a small number of dynamic areas.

Step 1. Clustering occupancy grids
For every sub-map created in the previous step, the occupancy grids are generated from the sensor values collected earlier. Each occupancy grid captures the state of the sub-map at a given time period. By clustering the occupancy grids for every sub-map separately, we can create groups of similar occupancy grids. Each such group denotes a different configuration of the sub-map.
It is a necessity to create sub-maps of the environment. If we did not, we would have to store a number of maps that would be exponential in the number of dynamic objects. This means that we would need a huge number of occupancy maps of the whole environment to successfully cluster them into groups. Instead, by splitting the environment into sub-areas, we need a smaller number of smaller occupancy grid maps.
In order to cluster the occupancy grids, we transform the grids into a vector of probability values from 0 to 1 with the additional value of ξ. The ξ value represents an unobserved cell. When comparing two such vectors *a* and *b*, the following similarity measure has been shown to give good results:

$$\mathrm{d}(\alpha, b) = \sum_i \begin{cases} (a_i - b_i), a_i \neq \xi \wedge b_i \neq \xi \\ 0, a_i = \xi \wedge b_i = \xi \\ e, otherwise \end{cases} \tag{6}$$

where *e* is a number close to zero.
Then, a fuzzy k-means algorithm is used (Duda et al, 2001). In order to estimate the correct number of clusters (not known in advance), we iterate over the number of clusters and compute in each step a model using the fuzzy k-means algorithm. In each iteration, we create a new cluster initialized using the input vector which has the lowest likelihood under the current model. We then evaluate every model with the Bayesian Information Criterion (BIC) (Schwarz, 1978). The model with the biggest BIC is selected as the best representation of the environment.

4.2.3 Extended temporal occupancy grid algorithm
The key idea of this approach is to use an extended Temporal Occupancy Grid to preserve the occupancy history of the environment. The values stored in the grid are processed to extract information about the possible configurations of the environment.
With this approach, low dynamic objects can be detected without any prior knowledge of the object shape or motion. This algorithm does not assume that the environment remains static during the robot exploration. On the contrary, in order to correctly detect the dynamic objects, it demands that these objects move fairly enough in order to be distinguished from the static or the high dynamic objects.
The algorithm uses three steps:

a. Step 0: The robot explores the dynamic environment and fills the extended Temporal Occupancy grid with occupancy values.
b. Step 1: Find possible object configurations by grouping neighbouring cells that follow the same pattern of occupancy evolution
c. Step 2: Identify which configurations found in the previous step belong to the same object.

The analysis of these steps follows.

Step 0. Data Acquisition
The robot explores the environment and acquires sensor values. These values are stored in the eTOG. The environment is assumed to be active during the exploration, meaning that low dynamic objects must change their positions thus different configurations of the environment are observed. One single pass of the environment is sufficient to identify all the low dynamic objects that were observed to move.
Step 1. Find possible configurations
In order to identify different configurations of the environment, we safely assume that a low dynamic object covers more than one cell in the environment (a door, for example, might cover three or more cells). Such an object falls into different configurations/ states (in the previous example, the states could be open and closed). In order to detect these configurations, we search through the history of all cells to find neighbouring cells that change with the same motif. These cells correspond to one of the possible configurations of the object. For example, in the case of a door, all cells of the closed door configuration would have the same values, regardless of the door's state (occupied if the door is closed, free if it is opened). To find cells that change with the same motif, we treat the values in the leaf nodes of the cells as Time Series (collections of observations made sequentially in time) that describe the cells occupancy. A single time series describes the evolution of the occupancy of the corresponding cell over time. Similar time series indicate similar cell occupancies. Thus, in order to find a single configuration of a moving object, we aim in finding neighbouring cells with similar time series. To do so, a clustering algorithm is applied.
To cluster time series, we need to define an appropriate distance function. There exist various similarity measures in the data mining community. A simple yet effective function is the Minkowski distance:

$$\mathrm{D}_{\text{Mink}}(T_1, T_2) = \sqrt[p]{\sum_{i=1}^{n} | t_{1,i} - t_{2,i} |^p} \tag{7}$$

Any other distance measure can be applicable.
At first, an agglomerative hierarchical clustering algorithm (Jain et al., 1999) is applied. In hierarchical clustering the data is not partitioned into a particular cluster in a single step. Instead, a series of partitions takes place. In each partition, two clusters are combined according to their distance to create a new cluster. The number of clusters is not given as an input. The algorithm keeps merging clusters until a given threshold (in the distance of two just merged clusters) is reached. The choice of an appropriate threshold is intuitive and depends on the data to be clustered.
The agglomerative clustering will generate a clustering where every cluster contains similar time series. Most probably each cluster will contain only one moving object. However, in the extreme case that two dynamic objects have similar change rate, the cluster will contain time series of more than one dynamic object. To avoid this case, we post process the generated clusters in order to create sub-clusters of both similar and neighbouring time series. To do so, we apply a second clustering algorithm, a DBSCAN algorithm (Ester et al., 1996) in order to group neighbouring time series. DBSCAN is a density based clustering that creates groups of data objects. Each of the objects in a group has at least one other object with distance less than a given threshold. DBSCAN does not require the number of clusters as an input to the algorithm and can create groups of arbitrary shapes. So, the DBSCAN algorithm

gets as input the cell coordinates of the time series of every agglomerative cluster extracted by the first clustering and groups them into dense clusters. Eventually, after the two clustering algorithms, the low-dynamic object configurations will have been found.

Step 2. Associate Configurations

In this step, we search for correlations among the configurations found in the previous step, for patterns that represent different configurations of the same object. The simpler case is the case of two-state objects, i.e. objects that can appear in two different positions. In such objects, their configurations are complementary; when the one is occupied the other one is free. Thus, in order for two different patterns to belong to the same object, their combined Time Series must contain the occupied value at any time.

Following the same rationale, we can search for objects with three, four, five etc different configurations. The difference is that we have to combine and evaluate more than two Time Series. In order for the algorithm to correctly associate the different patterns, the moving objects must change their positions with different time rates. If, for example, two doors follow the same motion pattern, the algorithm will be confused in the pattern association step. We have to keep in mind that the quality of the associations drops with the increase of the number of the states of the objects.

Experimental results can be found in Fig.5.

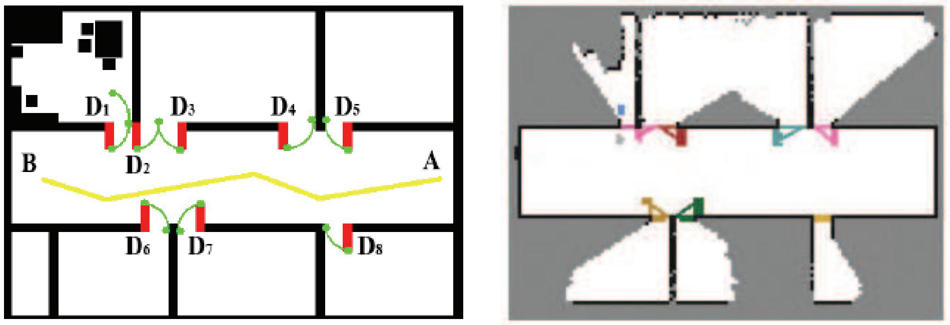

Fig. 5. The environment (yellow line indicates the robot path) and the experimental results of the eTOG algorithm. Associations between different configurations are indicated with a line that connects them.

4.3 Mapping in populated environments

Populated environments are highly dynamic environments where a lot of people move within them, e.g. crowded museums, metro stations at rush hour, supermarkets. The main difficulty in mapping those environments is the fact that many sensor measurements correspond to dynamic objects rather than static areas of the environment. So, the extraction of the model of the environment might contain spurious objects that reduce the quality of the generated maps.

There exist two categories of methods for dealing with populated environments both aiming at discovering the high dynamic objects and omitting them during the mapping phase. Their difference lies in the fact that the algorithms in the first category apply filters in order to track the moving objects while those in the second category apply statistical methods to identify the measurements that correspond to dynamic objects. There exist a variety of tracking techniques in the literature. In this section, we present a short introduction on techniques that have been applied successfully in the problem of mapping populated environments.

Regarding the tracking category, a number of filtering techniques have been used. When a single object is being tracked, its state can be estimated using algorithms such as the Kalman Filter, Extended Kalman Filter and Particles Filter (Bellotto & Hu, 2007).

Multiple objects tracking can be seen as the same problem with higher complexity. With multiple objects there is the additional difficulty of distinguishing one target from the other (data association). The complexity of solving the data association problem grows exponentially with the number of targets. Moreover, the number of different objects in the environment is also unknown and has to be derived from the observations. Once the number of objects is known, the data association allows the evolution of the filter associated to each object.

A common approach consists of using one Kalman Filter per target and solves data association independently for each track. This method is simple but as it examines each moving object separately it can lead to multiple associations of the single measurement with different moving objects. To deal with this problem, several approaches have been proposed, including Joint Probabilistic Data Association Filters (JPDAFs) (Hähnel et al., 2003a), Multi Hypothesis Tracking (MHT) (Mucientes & Burgard, 2006) and Particle Filtering. Also, combinations of these methods have been proposed such as the SJPDAFs that combine particle filters with JPDAF. Further information can be found in (Hähnel et al., 2003a), (Hähnel, 2004).

A framework for the solution of the detection and tracking of moving objects (*DTMO*) problem is presented in (Wang & Thorpe, 2002). The authors use the static-dynamic grids approach (c.f. Section 3.3) in order to detect the moving objects in the robot's environment. In order to track the objects, they apply a matching technique between the current state of the environment and the previous one stored in the grid. When the best match between the two states is found, information about the moving object can be extracted. In their next work (Wang et al., 2003), they present a Bayesian formulation of the Simultaneous Localization and Mapping problem (*SLAM*) with the *DTMO* problem. In order to track moving objects, a motion based detector is used. Then a Multiple Hypothesis Tracking method is applied to find the associations between current and previously found dynamic objects. When the associations are found, we can conclude on the motion of this moving object and predict its future position.

Regarding the second category of solutions, filtering techniques are used in (Fox et al., 98) to detect humans in the vicinity of the robot (on a known environment). A distance filter is applied to filter out all measurements that are found to be shorter than expected. Also, an entropy filter is implemented to measure the uncertainty of the measurements. Those measurements that are found to increase the uncertainty of the system are ignored.

In (Hähnel et al., 2003c), an EM algorithm was used to learn which measurements correspond to static objects. In the expectation step, an estimation is made on which measurements might correspond to static parts of the environment while in the maximization step, the position of the robot is calculated with respect to the considered static areas of the estimation step. By iterating between the two steps we can generate high quality maps with significantly less dynamic objects that if we used a common algorithm for static environments. This technique can be also applied in dynamic environments with the same results.

5. Conclusions and outlook

In this chapter, we have presented, in short, the major algorithms in the field of mapping dynamic environments. We categorize the objects in the robot environments into three categories (static, low dynamic and high dynamic objects) and for each category, we present a number of solutions proposed so far in the literature. Special attention is paid on the occupancy grid structure and its variations that have been applied on dynamic environments giving promising results. Of course, it was not possible to cover all the available techniques or examine relative to mapping issues, such as localization and navigation.

Summarizing, we can state that there exist a number of promising algorithms for mapping in dynamic environments. These algorithms are able to create valid maps of the static areas of the world and detect and model (low or high) dynamic objects in the robot environment. Nevertheless, a large number of challenging issues still remains to be solved. First of all, many algorithms that are presented in this work are not *real-time*. They make use of iterative techniques (e.g. EM, clustering) with execution times that depend on the size of the data. The necessity for real-time algorithms is obvious. A robot must be able to identify its environment rapidly in order to react on time. This need is more urgent while navigating outdoors. *Outdoor mapping* is an issue that remains open. Streets, parks and garages are some examples of places where a mobile robot can be of extreme usefulness. However, these environments pose new challenging issues. Their rate of change, complexity and size make many existing algorithms inapplicable and impose the need for new algorithms suitable for the special characteristics of such environments (some examples can be found in (Nuchter et al., 2007) and (Yoon et al., 2007)).

In order to deal with these problems, an interesting direction that has to be investigated is the combination of computer vision with laser data. So far, only a few algorithms have applied techniques borrowed from the computer vision field e.g. (Anguelov et al., 2004), (Yoon et al., 2007). Vision, however, can provide a wealth of information about the state of the environment (such as humans walking by the robot or chairs in the middle of a room) and it can be really helpful in the task of object identification. The combination of vision and laser data can provide important information about the robot environment.

Another way to gain more information about the objects that surround the robot is to acquire 3D laser range data of them e.g. (Hahnel et al., 2003b), (Ryde & Hu, 2006), (Harati & Siegwart, 2007). Humans possess a complete prior knowledge of the model of the objects in the environment and can easily distinguish two different objects even when they are partially observed in contrast to modern robots. To bridge the gap between human perception and robot perception of the environment, we can utilize 3D laser data that provide more information including information about the objects' shapes.

To conclude, the problem of mapping in dynamic environments is a really challenging and extremely active research field in robotics. In the future, we will be able to design robots that will be capable of moving among humans in their own environments without interrupting human activities. In order to do that, a milestone that has to be reached is a complete solution to the problem of mapping dynamic environment.

6. Acknowledgments

This work was supported by the European Commission and the Hellenic General Secretariat for Research and Technology, under Measure 3.3 of the Operational Programme

”Information Society” in the 3rd Community Support Framework (Project name: DIANOEMA, ID:35).

7. References

Andrade-Cetto, J. & Sanfeliu, A., (2002), Concurrent Map Building and Localization on Indoor Dynamic Environments, International Journal on Pattern Recognition and Artificial Intelligence, Vol 16, pages 361-374

Anguelov, D.; Biswas, R.; Koller, D. Limketkai, B. & Sebastian Thrun (2002), Learning Hierarchical Object Maps Of Non-Stationary Environments With Mobile Robots, Eighteenth Conference on Uncertainty in Artificial Intelligence (pp. 10-17)

Anguelov, D.; Koller, D.; Parker, E. & Thrun, S. (2004), Detecting and modeling doors with mobile robots, In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)

Arbuckle, D.; Howard, A. & Mataric. M. J. (2002). Temporal occupancy grids: a method for classifying spatio-temporal properties of the environment, IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 409–414, Lausanne, Switzerland

Avots, D.; Lim, E.; Thibaux, R. & Thrun. S.(2002) A probabilistic technique for simultaneous localization and door state estimation with mobile robots in dynamic environments, Proceedings of the Conference on Intelligent Robots and Systems (IROS), Lausanne, Switzerland

Bellotto, N. & Hu, H. (2007), People Tracking with a Mobile Robot: a Comparison of Kalman and Particle Filters, 13th IASTED International Conference on Robotics and Applications (RA 2007), Germany

Biber, P. and Duckett, T. (2005) Dynamic maps for long-term operation of mobile service robots, In Robotics Science and Systems

Birk, A., Carpin, S., (2006) Merging Occupancy Grid Maps From Multiple Robots, PIEEE(94), No. 7, pp. 1384-1397

Biswas, R.; Limketkai, B.; Sanner, S. & Thrun, S. (2002) Towards Object Mapping in Dynamic Environments With Mobile Robots, Conference on Intelligent Robots and Systems (IROS), Lausanne, Switzerland

Dempster, A.P.; Laird, N.M. & Rubin. D.B., (1977), Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, Series B, pages 1–38

Duda, R.; Hart, P.; & Stork, D. , (2001), Pattern Classification. Wiley-Interscience

Elmasri, R.; Wuu, G. T. & Kim. Y. J. (1990). The time index: An access structure for temporal data, 16th VLDB, pages 1–12

Ester, M.; Kriegel, H.-P.; Sander, J. & X. Xu, (1996), A density-based algorithm for discovering clusters in large spatial databases with noise. In Second International Conference on Knowledge Discovery and Data Mining, pages 226– 231, Portland, Oregon

Fox, D.; Burgard, W.; Thrun, S. & Cremers, A.B, (1998), Position estimation for mobile robots in dynamic environments, Proceedings of the AAAI Fifteenth National Conference on Artificial Intelligence

Gutmann, J. & Konolige, K., (1999), Incremental mapping of large cyclic environments, the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), pages 318–325, Monterey, California

Hähnel, D. (2004), Mapping with Mobile Robots. PhD thesis, Fakultät für Angewandte Wissenschaften, Universität Freiburg

Hähnel, D.; Schulz, D. & Burgard, W., (2003a) Mobile robot mapping in populate environments, Advanced Robotics, Volume 17, pages 579-597(19)

Hähnel, D.; Thrun, S. & Burgard, W. (2003b), An Extension of the {ICP} Algorithm for Modeling Nonrigid Objects with Mobile Robots, Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), Mexico

Hähnel, D., Triebel, R., Burgard, W. & Thrun, S., (2003c), Map Building with Mobile Robots in Dynamic Environments, In Proceedings of the International Conference on Robotics and Automation (ICRA), pages 1557-1563

Harati, A. and Siegwart, R. (2007), Orthogonal 3D-SLAM for Indoor Environments Using Right Angle Corners, The 3rd European Conference on Mobile Robotics (ECMR) 2007, Freiburg, Germany

Jain, A. K.; Murty, M. N. & P. J. Flynn, (1999), Data clustering: a review, ACM Computer Surveys, pages 264–323

Larionova, S.; Marques, L. & de Almeida, T. , (2006) Detection of Natural Landmarks for Mapping by a Demining Robot, IEEE International Conference on Intelligent Robots and Systems (IROS), pages 4959-4964

Latecki, L. J. & Lakaemper, R. (2006), Polygonal Approximation of Laser Range Data Based on Perceptual Grouping and EM, IEEE International Conference on Robotics and Automation (ICRA), Orlando, Florida

Mitsou, N. & Tzafestas, C. (2007), Temporal Occupancy Grid for mobile robot dynamic environment mapping, in the 15th IEEE Mediterranean Conference on Control and Automation, MED'07, Athens, Greece

Moravec, H. P. (1988). Sensor Fusion in Certainty Grids for Mobile Robots, AI Magazine, Vol. 9, No. 2, pp. 61-74

Moravec, H. P. & Elfes, A. E. (1985), High Resolution Maps from Wide angle Sonar, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'85), pp. 116-121, St. Louis, Missouri

Mucientes M. & Burgard W., (2006), Multiple Hypothesis Tracking of Clusters of People, 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China

Nuchter, A.; Lingemann, K.; Hertzberg J. & Surmann, H., (2007), 6D SLAM—3D mapping outdoor environments, Journal of Field Robotics

Ooi, B. C. & Tan. K. L. (2002). B-trees: Bearing fruits of all kinds, Thirteenth Australasian Database Conference (ADC2002), Melbourne, Australia

Ryde, J. & Hu, H. (2006), Mutual localisation and 3D mapping of cooperative mobile robots, The 9th International Conference on Intelligent Autonomous Systems (IAS-9), Tokyo, Japan, pages 217-224

Salzberg, B. & Tsotras, V. (1999), Comparison of access methods for time-evolving data, ACM Computing Surveys (CSUR)

Schulz, D. & Burgard, W., (2001), Probabilistic state estimation of dynamic objects with a moving mobile robot, Robotics and Autonomous Systems, pages 107–115

Schwarz, G., (1978), Estimating the dimension of a model, The Annals of Statistics 6(2)

Stachniss, C. & Burgard, W. (2005). Mobile robot mapping and localization in non-static environments, Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, USA

Suau, P., (2005), Robust Artificial Landmark Recognition Using Polar Histograms, Intelligent Robotics (IROBOT 2005) Chapter 7 - pages 455-461

Tanaka, K. & Kondo, E., (2006) Towards Real-Time Global Localization in Dynamic Unstructured Environments, Special Issue on Advanced Technology of Vibration and Sound pp.905-911

Thrun, S. (2002), Robotic Mapping: A Survey, Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann

Thrun, S. (2003), Learning Occupancy Grid Maps with Forward Sensor Models, Journal of Autonomous Robots, vol. 15, pages 111-127

Wang, C.-C. & Thorpe. C, (2002), Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects. In IEEE International Conference on Robotics and Automation (ICRA'02)

Wang, C.-C.; Thorpe. C, & Thrun. S, (2003), Online Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects: Theory and Results from a Ground Vehicle in Crowded Urban Areas. In IEEE International Conference on Robotics and Automation (ICRA'03)

Wang, H.; Hou, Z & Tan, M., (2007), Mapping Dynamic Environment Using Gaussian Mixture Model, 6th IEEE International Conference on Cognitive Informatics (ICCI'07)

Wolf, D. & Sukhatme, G. (2004), Online simultaneous localization and mapping in dynamic environments, IEEE International Conference on Robotics and Automation, pages 1301–1306

Wolf, D. & Sukhatme, G. (2005), Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments, Journal of Autonomous Robotics, Volume 19, pages = 53-65

Yamauchi, B. & Beer, R. (1996), Spatial learning for navigation in dynamic environments. IEEE Transactions on Systems, Man and Cybernetics, Special Issue of Learning Autonomous Robots, pages 496– 505

Yoon, S.; Sung-Kee, P.; Choi, H. D.; Kim, S. & Kwak, Y. K. (2007), ViSion-Based Outdoor Simultaneous Localization and Map Building Using Compressed Extended Kalman Filter, European Control Conference (ECC07), Kos, Greece

Zhang, L. & Ghosh, B.K., (2000), Line segment based map building and localization using 2D laserrangefinder, IEEE International Conference on Robotics and Automation, ICRA '00, pages 2538-2543

Zimmer, U. (1995), Adaptive Approaches to Basic Mobile Robot Tasks. PhD thesis, University of Kaiserslautern