A survey on Machine Learning in Mobile Robotics

João Cunha

Abstract — Learning is a prerequisite for intelligent behaviour. It is not surprising that Machine Learning, one of the most important fields of Artificial Intelligence, is becoming an increasingly hot-topic in Robotics. Machine Learning has several benefits, from enabling the use of historical data to improve future decisions, to solving problems that are difficult by hand-coding the solutions or to allow adaptive behaviours in highly dynamic environments such as learning individual preferences in human-computer interaction environments. Recent advances in algorithms and the increasing computational power available at reduced size and weight, enabled the application of Machine Learning to the field of mobile robotics. This report presents an overview of the state of the art of the different applications of Machine Learning methodologies in mobile robotics.

I. INTRODUCTION

Given the high degree of multidisciplinarity in the field of Robotics, programming a robot poses several challenges. Firstly there are no high-level programming languages to aid the development of robust algorithms. Secondly the robot sensors and actuators are complex and costly to model. Thirdly robots have been steadily moving towards more unstructured environments where robot programming becomes a tedious task. The aforementioned are just a small set of the reasons why researchers have been working on enabling robots to learn how to perform tasks by themselves. Thus the area of Robot Learning is the application of Machine Learning methodologies to the field of Robotics. According to [1], the definition of Machine Learning is:

“Any computer program that improves its performance at some task or class of tasks through experience.”

Therefore every Machine Learning problem is characterized by:

- a task or a class of tasks (ex: playing chess);
- a performance measure (ex:percentage of games won);
- experience (ex:playing practice games).

There have been several different machine learning methodologies developed over the years, however all are characterized in terms of supervised and unsupervised learning. These two classes differ in the sense that in the former there is a “teacher” supplying the learning algorithm with training data, while in the latter the algorithm has to discover the training data by itself.

Although different Machine Learning methodologies have been successfully applied to solving complex and non-trivial problems in the field of Robotics, such as the estimation of sensor noise [2], environment representations [2][4], or control policies in learning robot motion, there is still an open challenge. Factors as sensors and actuators noise impose difficulties in learning problems. Additionally [1] describes learning as a search problem of finding a policy that best fits the training examples, hence high dimensional or continuous state spaces are prohibitive factors in learning problems. Overfitting the training data is also a concern in Machine Learning since the learning program can demonstrate very high performance in the presence of training examples while failing to generalize a policy for future examples. The remainder of this paper is structured as follows. Section II presents the Credit Assignment Problem as the basic problem of any learning problem. Section III describes some of the most important learning paradigms applied to the field of Robotics, along with some illustrative examples. Finally section IV presents the conclusions.

II. THE CREDIT ASSIGNMENT PROBLEM

Robot Learning is a problem of learning a policy π in a set of sensory states S to a set of responses R. Learning a policy requires solving three credit assignment problems [6]. The temporal credit assignment involves giving credit or blame to a given response, in a sequence of responses, for a good or bad outcome. The structural credit assignment determines the range of sensor values that yield the same outcome. Finally the task credit assignment generalizes a sequence of responses to perform other similar tasks. The various learning paradigms, presented in the next section, are characterized by solving each credit assignment problem in a different manner.

III. LEARNING PARADIGMS

This section presents four different Machine Learning paradigms that have been successfully applied in field of Robotics to solve various tasks.

A. Inductive Concept Learning

The first paradigm presented is Inductive Concept Learning. In this paradigm there is a teacher providing training data along with the classification of the same data. Therefore this is a supervised paradigm where the temporal credit assignment problem is solved by the teacher. The learning problem is then reduced to inferring which values from the attributes of the training data actually affect the value of the classification.

Inductive Concept Learning is a well known paradigm not only in Robotics but in Machine Learning in general. Hence various different methodologies have been developed. Of the most important methods Vector Spaces, Decision Trees and Neural Networks are of notice. In particular Decision Trees and Neural Networks have been successfully applied in Robotics for their ability to cope with uncertainty and noisy data. An additional concept very important in learning in general and in Inductive Concept Learning in particular is the ability to generalize for unseen data. This is known as inductive bias [7], in the sense that the learner is provided a-priori assumptions on the target policy. An example of inductive bias used in Machine Learning is Occam’s Razor.

A.1 Examples

There are a variety of examples of the application of Inductive Concept Learning in the field of Robotics, some dating more than 20 years. In fact as early as 1988, Pomerleau [8] was able to teach an autonomous car to drive in two-lane highways by training a neural network from previously captured images of a driven car. Such Learning from Demonstration (LfD) method was applied in the 2005 DARPA Grand Challenge winner Stanford car Stanley to perform highway [9] and car lot [10] navigation. A different application of inductive concept learning is present in [11] where the Stanford Autonomous Robot was able to determine entangled or stuck status and even the type of surface it was walking on based on accelerometer data.

Fig. 1 - The Stanford car Stanley, adapted from [12]

B. Explanation Based Learning

Explanation Based Learning is another supervised learning paradigm. However, in Explanation Based Learning the teacher doesn’t provide a classification along with the data. Instead it provides a domain theory of how the training examples are consistent with the target policy. Thus not only the temporal credit assignment problems but also the structural credit assignment problems are solved by the teacher. The domain theory can be provided in various forms, from logic rules to neural networks. The domain theory can be viewed as a priori knowledge of the task to be perform, and thus allows to speed up the search over the space of possible hypotheses.

B.1 Examples

Explanation Based Learning was a hot-topic in 1990 decade, where computational power was scarce and the domain theory was highly regarded as it enabled to speedup the learning process compared to other paradigms at the time. However, the increased require-ment is currently a derogatory factor when opposed to other learning algorithms which require less a priori requirements.

Nonetheless, [12] presents a remarkable example of an explanation Explanation Based Learning. In this case the domain theory is a neural network modelling each section of the robot. Reported results show that in 10 minutes the robot learned to navigate towards a landmark in an office environment.

C. Evolutionary Learning

Evolutionary Learning is a very distinct learning paradigm since it is not inspired on human reasoning but is a close analogy of biological evolution [14]. In Evolutionary Learning, the learner is only provided with a fitness function and a target measure. Hence this paradigm is considered a unsupervised learning paradigm. The learner searches for the optimal policy from an initial set of hypotheses. According to the provided fitness function, the most fit individuals of the population are chosen to generate the following generation. While some members pass intact to the following generation (reproduction) others are combined with each other to produce new offspring (crossover). Aside from the genetic operators of reproduction and crossover, some elements of the population maybe suffer changes resulting in different individuals (mutation).

Thus Evolutionary Learning searches for the optimal policy through search space of possible hypotheses by generating variants of the best current hypotheses. There are two major variants of Evolutionary Learning: Genetic Algorithms and Genetic Programming. The basic difference is that in Genetic Algorithms the hypotheses are encoded in strings while in Genetic Programming the hypotheses are encoded in computer programs.

C.1 Examples

Examples of Evolutionary Learning in the field of Robotics can be found in gait learning such as in [15] where the gait of a biped simulated humanoid was obtained using Genetic Algorithms. Also of notice is one of the driving forces of Evolutionary Robotics, Hod Lipson, who focuses on applying Evolutionary Learning not only to robot controllers (brain) but has also applied Evolutionary Learning to the robot hardware (body) achieving evolved morphologies from initial simple robots [16].
A survey on Machine Learning in Mobile Robotics

João Cunha

Abstract - Learning is a prerequisite for intelligent behaviour. It is not surprising that Machine Learning, one of the most important fields of Artificial Intelligence, is becoming an increasingly hot-topic in Robotics. Machine Learning has several benefits, from enabling the use of historical data to improve future decisions, to solving problems that are difficult by hand-coding the solutions or to allow adaptive behaviours in highly dynamic environments such as learning individual preferences in human-computer interaction environments. Recent advances in algorithms and the increasing computational power available at reduced size and weight, enabled the application of Machine Learning to the field of mobile robotics. This report presents an overview of the state of the art of the different applications of Machine Learning methodologies in mobile robotics.

I. INTRODUCTION
Given the high degree of multidisciplinarity in the field of Robotics, programming a robot poses several challenges. Firstly there are no high-level programming languages to aid the development of robust algorithms. Secondly the robot sensors and actuators are complex to model. Thirdly robots have been steadily moving towards more unstructured environments where robot programming becomes a arduous task.

The aforementioned are just a small set of the reasons why researchers have been working on enabling robots to learn how to perform tasks by themselves. Thus the area of Robot Learning is the application of Machine Learning methodologies to the field of Robotics.

According to [1] the definition of Machine Learning is:

"Any computer program that improves its performance at some task or class of tasks through experience."

Therefore every Machine Learning problem is characterized by:
- a task or a class of tasks (ex: playing chess);
- a performance measure (e.g. percentage of games won);
- experience (e.g. playing practice games).

There have been several different machine learning methodologies developed over the years, however all are characterized in terms of supervised and unsupervised learning. These two classes differ in the sense that in the former there is a "teacher" supplying the learning program with training data, while in the latter there is no feedback.

Although different Machine Learning methodologies have been successfully applied at solving complex and non-trivial problems in the field of Robotics, such as estimation of sensor noise [2], environment representation [3, 4], or control policies for robot learning is still an open challenge. Factors as sensors and actuators noise impose difficulties in learning problems. Additionally [1] describes learning as a search problem of finding a policy that best fits the training examples, hence high dimensional or continuous state spaces are prohibitive factors in learning problems. Overfitting the training data is also a concern in Machine Learning since the learning program can demonstrate very high performance in the presence of training examples while failing to generalize a policy for future examples.

The remainder of this paper is structured as follows. Section II presents the Credit Assignment Problem the basic problem of any learning problem. Section III describes some of the most important learning paradigms applied to the field of Robotics, along with some illustrative examples. Finally section IV presents the conclusions.

II. THE CREDIT ASSIGNMENT PROBLEM
Robot Learning is a problem of learning a policy \( \pi \) from a set of sensory states \( S \) to a set of responses \( R \).

Learning a policy requires solving three credit assignment problems [6]. The temporal credit assignment involves giving credit or blame to a given response, in a sequence of responses, for a good or bad outcome. The structural credit assignment determines the range of sensor values that yield the same outcome. Finally the task credit assignment generalizes a sequence of responses to perform other similar tasks.

The various learning paradigms, presented in the next section, are characterized by solving each credit assignment problem in a different manner.

III. LEARNING PARADIGMS
This section presents four different Machine Learning paradigms that have been successfully applied in field of Robotics to solve various tasks.

A. Inductive Concept Learning
The first paradigm presented is Inductive Concept Learning. In this paradigm there is a teacher providing training data along with the classification of the same data. Therefore this is a supervised paradigm where the temporal credit assignment problem is solved by the teacher. The learning problem is then reduced to inferring which values from the attributes of the training data actually affect the value of the classification.

Inductive Concept Learning is a well known paradigm not only in Robotics but in Machine Learning in general. Hence various different methodologies have been developed. Of the most important methods are the Version Spaces, Decision Trees and Neural Networks which are of note.

In particular Decision Trees and Neural Networks have been successfully applied in Robotics for their ability to cope with uncertainty and noisy data. An additional concept very important in learning in general and in Inductive Concept Learning in particular is the ability to generalize for unseen data. This is known as inductive bias [7], in the sense that the learner is provided a-priori assumptions on the target policy. An example of inductive bias used in Machine Learning is Occam's Razor.

B. Explanation Based Learning
There are a variety of examples of the application of Inductive Concept Learning in the field of Robotics, some dating more than 20 years. In fact as early as 1988, Pomerleau [8] was able to teach an autonomous car to drive in two-lane highways by training a neural network from previously captured images of a driven car. Such Learning from Demonstration (LID) method was applied in the 2005 DARPA Grand Challenge winning car Stanley [9] to perform highway [9] and car lot [10] navigation. A different application of inductive concept learning is presented in [11] where the Sensor AIFo algorithm was used to determine entangled or stuck states and even the type of surface it was walking on based on accelerometer data.

B. Explanation Based Learning
Explanation Based Learning was a hot-topic in 1990 decade, where computational power was scarce and the domain theory was highly regarded as it enabled to speedup the learning process when compared to other paradigms at the time. However, the increased requirement is currently a derogatory factor when opposed to other learning algorithms which require less a priori requirements.

Nonetheless, [12] presents a remarkable example of an application Explanation Based Learning. In this case the domain theory is a neural network modelling each section of the robot. Reported results show that in 10 minutes the robot learned to navigate towards a landmark in an office environment.

C. Evolutionary Learning
Evolutionary Learning is a very distinct learning paradigm since it is not inspired on human reasoning but is a close analogy of biological evolution [14].

In Evolutionary Learning, the learner is only provided with a fitness function and a target measure. Hence this paradigm is considered a unsupervised learning paradigm. The learner searches for the optimal policy from an initial population of hypotheses. According to the provided fitness function, the most fit individuals of the population are chosen to generate the following generation. While some members pass intact to the following generation (reproduction) others are combined with each other to produce new offspring (crossover). Aside from the genetic operators of reproduction and crossover, some elements of the population maybe suffer changes resulting in different individuals (mutation).

Thus Evolutionary Learning searches for the optimal policy through search space of possible hypotheses by generating variants of the best current hypotheses. There are two major variants of Evolutionary Learning: Genetic Algorithms and Genetic Programming.

The basic difference is that in Genetic Algorithms hypotheses are encoded in strings while in Genetic Programming the hypotheses are encoded in computer programs.

C.1. Examples
Examples of Evolutionary Learning in the field of Robotics can be found in gait learning such as in [15] where the gait of a biped simulated humanoid was obtained using Genetic Algorithms.

Also of notice is one of the driving forces of Evolutionary Robotics, Hod Lipson, who focuses on applying Evolutionary Learning not only to robot controllers (brain) but has also applied Evolutionary Learning to the robot hardware (body) achieving evolved morphologies from initial simple robots [16].
D. Reinforcement Learning

The last learning paradigm presented in this paper is Reinforcement Learning [17].

The basic framework of Reinforcement Learning is a Markov Decision Process (MDP). MDPs are characterized by a set of states, S, a set of actions, A, a state transition function $\delta: S \times A \rightarrow S$ and an immediate reward function $r: S \times A \rightarrow R$. A fundamental scenario of MDPs is that a state $s$ only depends on a finite number of past states.

The learning problem is to find a policy $\pi: S \rightarrow A$ which produces the greatest cumulative reward over time, $r(s)$. The greatest cumulative reward gained in a given state is given by $r(s) = \sum_{\gamma} \gamma^\gamma$, where $\gamma$ is a discount factor of the delayed rewards in the future.

Reinforcement Learning methodologies are impacted by factors such as delayed rewards, since the robot might only receive a positive reward when it reaches the goal state, which may take some time to achieve. On the other hand while learning the robot must choose between exploiting a previously learned policy or to explore unknown states and actions. Finally, the robot sensors may not be enough to observe the entire surrounding environment.

An optimal policy is then a policy that, for a given state $s$ chooses the action $a$ that maximizes the immediate reward of applying $a$ in state $s$ plus the cumulative reward of the successor state $\delta(s, a)$. This policy is given by $\pi(s) = \max_a \{r(s, a) + \gamma \sum_{s' \in S} \delta(s, a) P(s', s, a)\}$.

Hence a robot with the perfect knowledge of the state transition and the immediate reward function can determine the optimal policy by applying the value-iteration algorithm which is proved to converge to the optimal value.

Here we assumed a deterministic state transition function. However the state transition function is commonly probabilistic given the sensors and effectors errors and noise. Thus the optimal MDP can be extended to accommodate probabilistic state transitions, $P(s'|s, a)$ which are used in the field of Robotic systems. Robotic Learning is robot control.

One example of such application is a classical control problem, the inverted pendulum problem. Biedermann [9] was able to balance a single and double inverted pendulum using only Reinforcement Learning.

A prolific environment for Reinforcement applications is the RoboCup robotic soccer competitions which have numerous examples of Reinforcement Learning.

For instance tasks from low-level control, such as motor control behaviors, such as ball interception and dribbling, to cooperation skills, such as attacking strategy have all been solved using Reinforcement Learning [19] [20].

Another example of Reinforcement Learning in particular in the RoboCup competitions can be seen in the RoboCup Multi-Agent Special Interest Group site [21] where it is shown that the vast majority of the learning methods applied is Reinforcement Learning. In particular the now extinct four legged league (4LL) is a great example of the application of Reinforcement Learning methods to complex-modeled systems such as the Sony AIBO platform.

This approach is a great example of the application of Reinforcement Learning algorithms in the field of Robotics. Robot learning offers several advantages for developing robotic systems while replacing hand-coded methodologies which assume the programmer has a perfect knowledge of the system model in order to produce optimal algorithms.

REFERENCES
D. Reinforcement Learning

The last learning paradigm presented in this paper is Reinforcement Learning [17]. The basic framework of Reinforcement Learning is a Markov Decision Process (MDP), which are characterized by a set of states $S$, a set of actions $A$, a state transition function $\delta : S \times A \rightarrow S$ and an immediate reward function $r : S \times A \rightarrow \mathbb{R}$. A fundamental feature of MDPs is that a state $s$ only depends on a finite number of past states. The learning problem is to find a policy $\pi : S \rightarrow A$ which produces the greatest cumulative reward over time, $w(s)$. The greatest cumulative reward gained in a given state is given by $w(s) = \sum \gamma^t r(s', t)$, where $\gamma$ is a discount factor of the delayed rewards in the future. Reinforcement Learning methodologies are impacted by factors such as delayed rewards, since the robot might only receive a positive reward when it reaches the goal state, what may take some time to achieve. On the other hand while learning the robot must choose between exploiting a previously learned policy or to explore unknown states and actions. Finally, the robot senses may not be enough to observe the entire surrounding environment.

An optimal policy is then a policy that for a given state $s$ chooses the action $a$ that maximizes the immediate reward of applying $a$ in state $s$ plus the cumulative reward of the successor state $\delta(a, s)$. This policy is given by $\pi(s, a) = \max_a \left( r(s, a) + \gamma \sum_{s' \in S} \pi(\delta(s, a)) P(r(s', s)) \right)$. Hence a robot with the perfect knowledge of the state transition and the immediate reward function can determine the optimal policy by applying the value-iteration algorithm which is proved to converge to the optimal value.

Here we assumed a deterministic state transition function. However the state transition function is commonly probabilistic given the sensors and effectors errors and noise. Thus the optimal policy can be extended to accommodate probabilistic state transitions, $\pi(s) = \max_a \left( E[r(s, a)] + \gamma \sum_{s' \in S} P(r(s', s)) \pi(\delta(s, a)) \right)$. However having a complete and perfect knowledge of the state transition function is often an unrealistic scenario, compared to having a perfect domain theory in explanation based learning.

To overcome this limitation a model-free Reinforcement Learning methodology is usually used, Q Learning. Q Learning is based on learning the Q function, $Q(s, a)$, which represents the maximum cumulative discounted reward obtained from state $s$ and applying action $a$ [18].

$$Q(s, a) = E[r(s, a)] + \gamma \sum_{s' \in S} P(r(s', s)) \pi(\delta(s, a))$$

On the other hand $w(s) = \max_a Q(s, a)$ which rewriting the previous equation yields $Q(s, a) = E[r(s, a)] + \gamma \sum_{s' \in S} P(r(s', s)) \pi(\delta(s, a))$. To learn the value of the Q function the robot starts at a random initial state $s$ and applies an action $a$ while observing the resulting state $s'$ and the obtained reward $r$. Hence Q-Learning can be viewed as a robot acting randomly upon the environment and analyzing the outcome of its actions. This is usually done using a table with an entry for each state-action pair. This is a major constraint since the robot estimation of the Q function will only converge in every pair state-action is visited sufficient times. This is an unrealistic assumption for very large dimensional or continuous space. On the other hand the learned policies would not be capable of generalizing to unseen examples. An alternative approach is to use neural networks instead of explicit tables as neural Q function. While this alternative has advantages and has been successfully applied to various robotic systems, classic algorithms for training neural networks to accommodate a new pair state-action pair, may change the Q estimates for other state-action pairs. This fact affects the convergence towards the optimal value and explains the large learning times and several thousand experience needed to achieve a good policy. However, variant algorithms were proposed in order to minimize the number of training examples required. Of notice is the method Neural Fitted Q Iteration [18] which stores the previous training examples as tuples $(s, a, r, s')$ that are considered when the neural network is updated for new experience data. This method is reported to be able to achieve a close to optimal policy in just a few hundreds examples.

Reinforcement Learning is very important in robotics as a framework for autonomous learning given the very few requirements with respect to other paradigms. Thus Reinforcement Learning is an unsupervised learning paradigm.

IV. CONCLUSIONS

This paper presented an overview of Machine Learning applications in the field of Robotic. Robot learning offers several advantages for developing robotic systems while replacing hand-coded methodologies which assume the programmer has a perfect knowledge of the system model in order to produce optimal algorithms. Reinforcement Learning [9] was able to balance a single and double inverted pendulum using only Reinforcement Learning.

A prolific environment for Reinforcement applications is the RoboCup robotic soccer competitions which have numerous examples of Reinforcement Learning. For instance tasks from low-level control, such as motor control behavior, such as ball interception and dribbling, to cooperation skills, such as attacking strategy have all been solved using Reinforcement Learning [19] [20].

Another example of Reinforcement Learning in particular in the RoboCup competitions can be seen in the RoboCup Multi-Agent Special Interest Group site [21] where it is shown that the vast majority of the learning methods applied is Reinforcement Learning. In particular the now extinct four legged league (ALL) is a great example of the application of Reinforcement Learning methods to complex-modeled systems such as the Sony AIBO platform.

Fig. 3 - A game of the RoboCup Four Legged League, adapted from [22].

REFERENCES

Localization techniques for autonomous mobile robots

João Silva, Nuno Lau, António J. R. Neves

Abstract — Mobile autonomous robotics is nowadays an area of study much addressed by research teams worldwide. One of the main challenges to create robots that can be really autonomous is the self-localization problem. For a robot to be able to plan a motion and move in a useful way, it should know where it is in the environment. Otherwise, it would just move randomly, probably not being useful at all. Then, there is also the case when one does not have a map to provide to the robot. In those situations, the robot should be able to build a map and localize itself relatively to it on runtime. This document aims to provide a brief presentation of these problems and some of the currently used solutions.

I. INTRODUCTION

Self-localization and mapping are classic problems of intelligent mobile robotics, over which research is still extremely active. These are part of the more general challenge of defining, managing and updating the robot internal world model. Sensor and information fusion techniques are widely used for these tasks. Generally, robots have access to partial and uncertain information through a set of multi-modal sensors. In dynamic environments, information fusion (of historical information and information coming from different sensors) is essential for the world model to be as precise as possible. Information fusion for localization is usually addressed through the use of probabilistic techniques such as Kalman or particle filters, sometimes conjugated with maximum likelihood techniques.

The integration of information over time in order to filter sensor noise is essential to get better estimates. This type of integration may be performed using Kalman filter based approaches, Monte-Carlo methods or Markov approaches. Generally, Monte-Carlo [1] approaches have better performance in cases where great discontinuities of the output values are expected, as the assumption of Gaussian probability density functions of the Kalman filter [2] is usually less accurate. However, Kalman filtering is a very effective method if the assumptions of Gaussian noise can be met and the system can be linearized. Other common approaches are the use of the Extended and Unscented Kalman filters [3], which are prepared to deal with non-linear systems at the cost of more computational weight.

When working with mobile autonomous robots, there are typically two scenarios. The environment can be known or partially known and the robot needs to localize itself, or the environment is unknown and the robot needs to build the map as it runs. This is addressed as Simultaneous Localization And Mapping (SLAM) and it is another common application of sensor fusion techniques [4], [5].

This document presents the main issues of the localization problem in Section II. A brief summary of some of the most commonly used localization algorithms is presented in Section III. Section IV briefly presents the SLAM problem and common approaches to the mapping and SLAM forms. Some remarks are presented in Section V.

II. LOCALIZATION PROBLEM

The problem of mobile robots localization is to identify where a robot is, given a map of the environment around it. The localization of a robot is usually defined as a pose, which contains a position (given in some coordinate system) and an orientation (relative to the defined coordinate system). When a robot is running and performing localization by its own means, it is not sure where it really is and how really is oriented. The pose that it keeps is thus called belief, as the robot believes it has a particular pose, but there is no guarantee that it is really correct; it is an estimate. This belief can be presented as a probability distribution function, \( p(\theta) \), where \( \theta \) is the pose of the robot, whereas coordinates are used (it can be a 1D, 2D or 3D situation, both for position and orientation).

At some moments of the run, the robot gets information from sensors that are at its disposal. These sensors can get information about the surroundings and provide information with some degree of accuracy about the pose at that instant. A robot can be equipped with a variety of sensors to help localization purposes, but this will not be subject to analysis. Let us just consider that the sensors provide measurements \( \hat{r} \) that are also not 100% accurate, they have some noise associated. The observations are represented by an observation model written as \( p(\hat{r} | \theta) \), which is a probability function.

Thus, the localization problem is mostly a probabilistic problem. Robots must have models for their movement (motion models) that are capable of providing the beliefs. These beliefs can then be reinforced or not by sensor measurements.

The localization problems are usually divided considering different aspects that are not equally difficult to solve. There are four main aspects to consider [4]: