IADIS International Journal on Computer Science and Information Systems
Vol. 6, No.3, pp. 87-103
ISSN: 1646-3692

ROBOT MOTION IMITATION BASED ON BAYES NETWORKS AND LEARNING

Andrés Gómez. *Depto. Ingeniería de Sistemas y Computación, Universidad de los Andes. Bogotá, Colombia.*

Fernando De la Rosa. *Depto. Ingeniería de Sistemas y Computación, Universidad de los Andes. Bogotá, Colombia.*

ABSTRACT

In Robotics, one of the core problems is motion planning towards a specified goal. Today, there are different solutions; however, most of them require complex models of the environment, which are not intuitive at all, and force the programmer to understand the operating mode in detail of the robot she/he is working with. Furthermore, most algorithms use inaccurate information which cannot be relied upon. This paper presents a robot motion imitation approach, using Bayes networks and machine learning, which does not require much knowledge, neither an environment model. The proposed method defines a Bayes network based on the distance detections near the robot. A learning phase provides the robot with a path for solving a motion planning problem and for processing the distance detections and velocities in order to learn how to control the robot. Finally, the robot can reproduce the learned path by using a probabilistic inference algorithm. The proposed method was tested both in simulated and experimental environments with a differential mobile robot.

KEYWORDS

Robot Imitation, Imitation learning, Machine Learning, Bayes network, Mobile Robotics.

1. INTRODUCTION

Today, a lot of powerful and accurate solutions exist for the motion planning problem, i.e. finding a path from an initial configuration to a target configuration. Although, those solutions still have some flaws. Most of the algorithms assume a given environment model, which may adapt perfectly to laboratory conditions for testing, but generally will fail to represent the complexity of real life applications. Moreover, programming a robot is not an intuitive task since there is a discrepancy in the languages used: a programmer thinks in terms of obstacles and motions, and he/she knows that, while approaching a wall, the robot should slow down a

turn away from it. However, for a mobile robot the concept of *"wall"* and *"turn"* are unknown. Such a robot works with coordinates, sensor detections and wheel velocities, processing large quantities of numerical data to adopt the best behavior. Clearly, the translation from one language to the other is far from obvious. Finally, many of the solutions involve using odometry data (internal data from the robot, generally used to estimate its position at any time), which are sometimes far from exact, preventing the robot from reaching its target. Our solution consists in using a probabilistic approach and does not have those problems.

The purpose of this paper is to explain a motion imitation method (which takes a mobile robot from an initial configuration to a final configuration) developed using a probabilistic approach instead of a more common deterministic approach. Specifically, a Bayes network is implemented, and the data needed to define the network are obtained through learning. During the learning process, a human remotely operates the robot, and the machine collects all the sensor data obtained and uses them afterwards to operate autonomously.

Thanks to the probabilistic approach, this method is quite robust since it allows uncertainties: even in case of a minor failure in a device (like the laser sensor), the robot can adopt a behavior that suits well with the situation in a natural manner, increasing the odds to success. It does require an approximated model, just as other methods do, but this model does not deteriorate passing from laboratory to real life situations. Data from simulation learning works also well with the physical robot. Moreover, the method is program friendly since it no longer needs to worry about the robot details: for instance, to program a turn to get away from an obstacle, it is enough to "*show*" the robot what it must do instead of working directly with the translation and rotation velocities. Furthermore, if one wishes to change the robot's behavior (can be a small change, such as accomplishing the same task while staying further away from obstacles, or a big change such as a robot that pushes obstacles instead of one that avoids them), it suffices to change the learning data, without modifying a single line of code. Finally, in this approach, the robot does not use odometry data, only laser detection.

The Bayes network presented in this paper is simple, but could be enlarged to resolve more difficult tasks. In this sense, this paper is a guide to use probabilistic networks in mobile robotics.

This paper is organized as follows: First, works related with motion planning and probabilistic related approaches are discussed. There is a small section explaining the Bayes network as well. Section 2 presents the main components of our proposed approach. Section 3 shows the test cases and the preliminary results obtained in laboratory with a physical mobile robot. Finally, section 4 presents some conclusions and section 5 proposes some possibilities for future work.

1.1 Related Works

To find a path between an initial point and a destination, the Dijkstra algorithm for the shortest route, or the A* algorithm, comes to mind. To apply these methods in the motion planning problem, it is required to transform the map of the environment into a graph, so we can apply these algorithms on that graph. Two of the most popular strategies for obtaining such a graph are cell decomposition and visibility graph. These methods need the complete map of the environment a priori, and while they always find a viable solution (if such solution exists, otherwise they report that there is no solution), they may be computationally heavy, dependent

of map size, and inflexible. Moreover, once a path is found, having the robot navigate through that path is a problem in itself, because odometry data must be used and may be inaccurate, causing the robot to deviate from the path and to become lost or collide with an obstacle. These algorithms are called global (i.e. they require the environment's map to make decisions) and complete (i.e. they find a solution if such solution exists).

An alternative approach is using potential fields where at any given point in the environment there is a force that draws the robot to its goal, and another one repelling it from obstacles. The robot makes decisions only based on the forces at that point. This method doesn't require a map given a priori, navigation is easier and the method is more flexible, but the robot might get trapped in local minima, never reaching the destination. Such algorithms, making decisions based on the immediate surroundings without any guarantee of reaching a solution, are called local and heuristic.

As a general rule, when working in large environments, global and complete algorithms are too complex, inflexible, and for this reason they are mostly seen in situations that can guarantee both a known map to find a solution and good precision to execute its motions. Under limited conditions, the local algorithms are preferred. La Valle (2006) elaborates further on these and many others similar methods.

The field of probabilistic networks has recently seen a significant growth, with applications in distinct areas such as medicine or finances. Those networks are widely used when one wishes to program a decision making process. Binder and Koller (1997) made a comparison of learning in Bayes networks versus Neural networks, where the former gained the upper hand because it supports a priori information from experts, information that is used to give structure to the network. F. Gagliardi (1998) developed JavaBayes, a Java program for doing probabilistic inference and experimenting with Bayes networks. There are also some programs based on learning and designed to play games (Korb and Nicholson, 2004): they built a program that played a simple form of poker, and after playing against different kinds of opponents, it learned to adapt to beat those opponents. In the end, their program was able to beat most of the other programs that did not have this learning mechanism. It was able to beat novice human opponents that played predictably, but failed against advanced poker players that varied their style.

In addition, O. Lebeltel *et al.* (2004) proposed a complex robot based on a probabilistic approach, designed for firefighting. The robot is provided with probabilistic behaviors which enable him to resolve a task. However, most probabilistic distributions used are given *a priori* (instead of acquired through learning), more inputs are used alongside the laser detections we focus on, and there is no explicitly defined Bayes network. The resulting robot is composed of different behaviors, built incrementally, each one with a probabilistic component, and to solve different tasks it uses a combination of different behaviors. In this paper, we propose a robot based on one behavior alone and based on learning with very little information given a priori.

There are not many applications of Bayes networks in robotics. However, Zhou and Sakane (2007) use Bayes networks to solve the lost robot problem (i.e. given the map of an environment, the robot must find its position on that map). In their paper, they estimate the sensing costs and the degree of localization certainty (i.e. how certain a robot is about its possible location) by making inference in a Bayes network, and by using those estimates to determine the most efficient way of exploring until its location is found with a high degree of certainty.

Furthermore, a robot imitation approach combined with a probabilistic component has been used by M. Ollis *et al.* (2007) for estimating the terrain costs: the robot is remotely

navigated by a human operator, and the acquired learning data helps the robot to identify what terrain type was often used by the human, and what terrain type was avoided. This information is then used to assign a cost to the terrain which the robot is able to traverse, and with these costs a map is built and the shortest/easiest path is calculated using standard algorithms.

E. Antonelo *et al.* (2008) use Recurrent Neural Networks (RNN) with Reservoir Computing and an imitation learning process to produce a mobile robot for exploration and target seeking. The results are quite good, and the method is very robust since a noise is artificially introduced in the sensor detections, and the behavior acquired translates well when the environment is changed (despite some collisions).

1.2 Bayes Networks

Our method is based on using a Bayes network (or Bayesian network) to model the robot's decision process. Next, we will briefly describe what it is, how it operates and how it can be used.

A Bayes network can be modeled as an acyclic directed graph, modeling causes and consequences. The predecessors of a vertex x (i.e. all vertices such that there exists an edge from the vertex to x) are the causes of x, and the successors (i.e. all the vertices such that there exists an edge from x to the vertex) are the consequences of x. One of the most common examples is the burglary-earthquake network, which models the scenario with a house and an alarm, and the alarm can be set on either by a burglary or an earthquake, and if triggered, the owner may be called by his neighbors John and/or Mary (Figure 1).

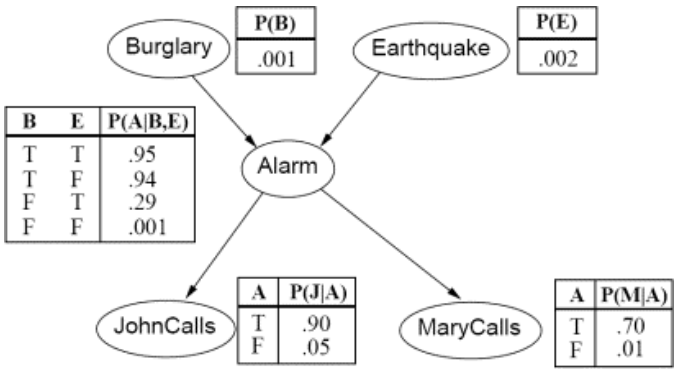

Figure 1. Bayes network for a Burglary-Earthquake scenario. Drakos *et al.* (2010)

The network is completely determined by the Conditional Probability Tables (CPTs) that determine the probability distribution of an event occurring given the values taken by his predecessors. In the example above the probability that the alarm sounds when there is both an earthquake and a burglary is 95%, the probability of John calling when the alarm sounds is 90%, and the probability of him calling when the alarm does not sound is 5%, and so on. Each successor has a probability distribution for each combination of values of his predecessors: in the example, Alarm has four probability distributions associated, while JohnCalls and MaryCalls have only two. Note that since Burglary and Earthquake have no predecessors, they just have a single probability distribution that corresponds to the probability of those events happening without any additional information.

In a Bayes networks, all events are dependent a priori. However, given some evidence, they can become independent (in the example above, the events Burglary and JohnCalls are a priori dependent, but if I know for sure the alarm sounds they become independent). Furthermore, using Bayes rule, $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$, and other probability techniques, the probability of any event given any set of evidence can be computed (for example, the probability of burglary given that John calls and Mary does not, or the probability of John calling given that there is no earthquake can be computed). This characteristic is what makes Bayes networks so useful in a decision making process, where the solution can be changed naturally as evidence is acquired. The process of computing those probabilities is called inference.

As of today, Bayes networks have two major limitations. Inference is NP-Hard, and the events' distributions can only be discrete or Gaussian (also known as Normal distribution). Nevertheless, they are a valuable tool with many useful applications, in fields such as marketing, military and medicine (Figure 2).

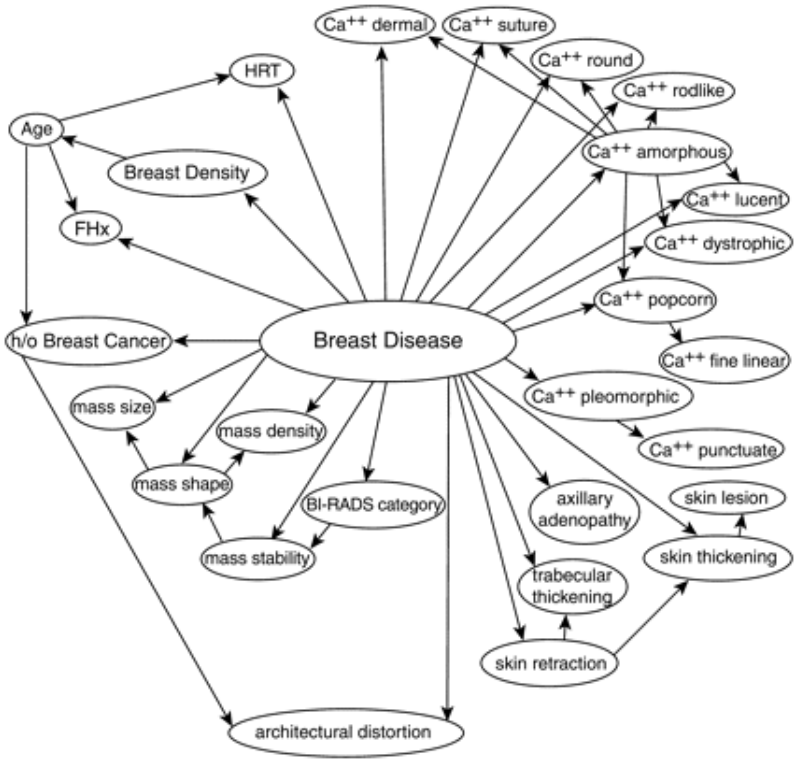

Figure 2. Bayes network for diagnosing breast disease. Burnside *et al.* (2009).

2. PROPOSED APPROACH

This section presents the main components of our approach which is composed of a designed Bayes network, a proposed learning algorithm, and a proposed probabilistic inference algorithm to control the robot.

2.1 Bayes Network Design

We must consider two kinds of variables: input variables, or sensory variables (in this case, laser detections) and output variables, or driving variables (translation speed and rotation speed). Our task is to determine a function which transforms input detections provided by the laser sensor into the output values for the robot translation and rotation speeds that best suit the situation.

Since there are many possible configurations for the detections given the laser precision, 3995^161 to be exact (e.g. the quantity of atoms in the observable universe is around 10^80), where 3995 is the laser's distance range in millimeters and 161 degrees is the laser's field of view, we must simplify the possible inputs. The proposed solution is using two new variables, named Distance and Angle. Distance indicates the robot how close it is to the obstacles. For simplicity, we only consider laser detections until 600mm. For detections farther than this distance, they are transformed to the maximum distance (600mm), since we believe such detections to be irrelevant in the robot's decision. Distance is just the minimum distance detected in all possible angles, divided by 100 and rounded to the nearest integer. It can take seven possible values, from 0 to 6, but since our robot will never approach the obstacles too closely, it should never take the value 0 and we are left with 6 values (Figure 3).

To calculate Angle, we divide the 160 degrees view laser cone that can be scanned into six smaller cones, and take the minimum distance in each of the cones (Figure 3). We then select the two cones with the smaller distance. We get thirty ways to select the two cones (order does matter). Angle then takes a value between 0 and 29, each value corresponding to a possible selection of two cones, indicating the robot the direction of the nearest obstacles.

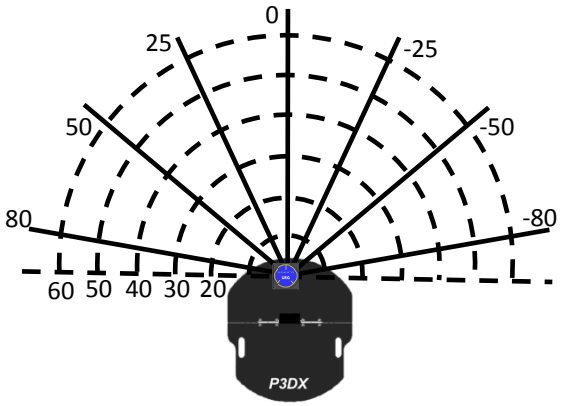

Figure 3. Intervals of distances and angles taken into account from the laser sensor

We are left with 180 possibilities for the combined values of Distance and Angle, a value that is small enough to work with, and yet rich enough to accurately model the different situations that the robot will face.

We propose the design of a Bayes network, based on the laser detections, Distance and Angle, and translation and rotation velocities. The resulting Bayes network is quite simple: Distance and Angle are consequences of laser detections, and are causes of the robot translation and rotation velocities (Figure 4).

In the previous paragraph, we explained how to compute Distance and Angle from the detections. These values can be viewed as a probability distribution, where they take the value specified above corresponding with the laser detections with probability 1, and all other values with probability 0.

We still have to determine the 180 conditional distributions of Translation Velocity and Rotation Velocity, given each combination of Distance and Angle. These distributions will be obtained through a learning algorithm.

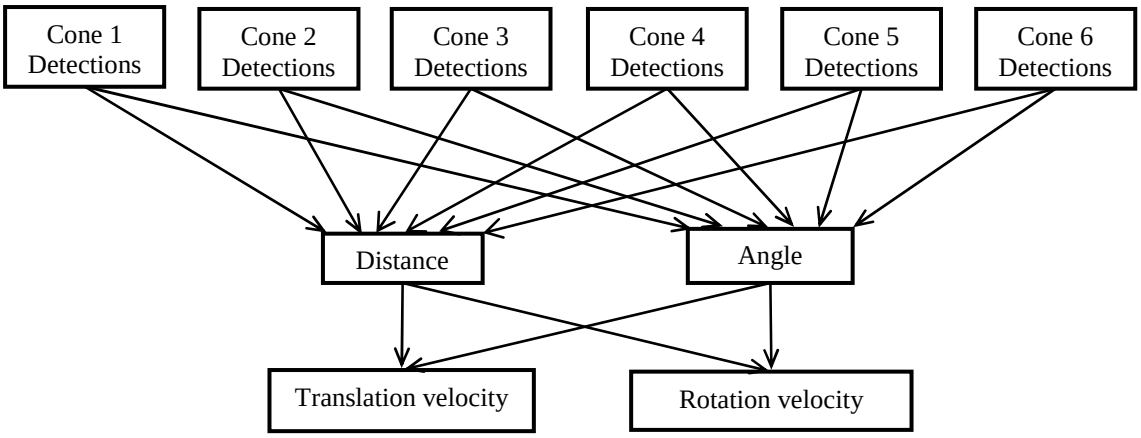

Figure 4. Bayes network based on laser detections inside angular intervals (cones) enabling to infer the translation and rotation velocities of the robot

2.2 Learning Algorithm

Prior to normal execution, the robot must go through a preliminary phase of learning, to determine the conditional probabilities of the robot's velocities. In this phase, a human operator controls the robot in a way the robot should operate, and the robot stores all the relevant data necessary, which enables it to infer the desired behavior later.

In our case, learning is done as follows. First, the operator leads the robot from an initial configuration to a target configuration using a remote operation. During this process, the robot, as fast as possible (several times per second), stores the laser detections found during one sweep, and the velocities it had at that given time. A learning data sample is shown below (Figure 5).

```
locationTypes: robot robotGlobal
scan1Id: 1
time: 0.359
velocities: 0.00 0.00 0.00
robot: 0 0 0.00
robotGlobal: 0 0 0.00
scan1: -0 -1632  28 -1631  57 -1632  86 -1633  114 -1630  143 -
1630  171 -1629  200 -1627  229 -1627  258 -1627  287 -1628  316
-1628  346 -1628  376 -1628  405 -1625  437 -1630  466 -1626  497
-1626  528 -1625  561 -1630  592 -1628  625 -1629  658 -1628  691
-1628  726 -1630  760 -1630  794 -1628  829 -1627  863 -1624  900
-1624  940 -1628  977 -1625  1017 -1628  1057 -1628  1097 -1626
1139 -1626  1183 -1628  1223 -1624  1270 -1626  1315 -1624  1363
-1625  1410 -1622  1465 -1627  1515 -1624  1568 -1624  1621 -1621
1680 -1623  1741 -1624  1798 -1619  1862 -1618  1932 -1621  1999
-1619  2075 -1621  2144 -1616  2229 -1619  2310 -1617  2397 -1617
2489 -1616  2584 -1615  2687 -1615  2797 -1615  2911 -1613  3033
-1613  3162 -1611  3304 -1611  3449 -1608  3608 -1606  3761 -1597
3763 -1521  3762 -1444  3762 -1369  3760 -1295  3759 -1221  3754
-1148  3759 -1078  3755 -1006  3755 -936  3755 -867  3758 -799
3759 -731  3757 -662  3757 -595  3757 -528  3756 -461  3755 -395
3754 -328  3756 -263  3748 -196  3748 -131  3750 -65  3752 0
3746 65  3746 131  3749 196  3744 262  3744 328  3745 394  3748
460  3430 482  3016 478  2704 477  2433 473  2219 472  2032 469
1878 468  1743 467  1625 466  1523 466  1431 465  1346 464  1275
464  1202 462  1147 463  1086 461  1032 460  988 461  941 459
902 459  862 458  831 461  795 459  762 458  732 457  708 460
682 460  655 459  627 456  607 457  585 457  566 458  547 459
527 458  505 455  489 456  470 454  455 455  439 455  424 455
410 456  396 456  380 453  366 452  353 452  343 455  332 456
316 451  305 453  295 455  283 453  272 453  260 450  250 451
239 450  230 451  221 453  212 455  201 452  193 456  183 453
173 451  165 453  155 449  148 455  138 452  130 452  122 454
113 453  105 454  95 448  87 449  79 450  72 454  63 451  55 451
48 453  39 447  32 453  23 446  16 451  8 453  -0 451
scan1Id: 2
time: 1.234
velocities: 25.00 0.00 0.00
robot: 0 0 0.00
```

Figure 5. File showing the relevant robot information stored during the learning phase

The relevant data are robot velocities (first number indicates translation velocity in mm/sec, the second one indicates rotation velocity in deg/sec, and the third one indicates vertical velocity- since our robot cannot fly, this value will always be 0), and the scan detection to the obstacles inside the laser's field of view (each couple of integers represents the XY coordinates from the robot perspective of the nearest obstacle point in each angle, so there are 161 couple of integers). The figure shows only one scan, but a learning phase of five minutes provides around 5000 scans.

Afterwards, the information must be processed. The algorithm finds the specific situation where the robot is between the 180 possibilities, thanks to the laser detections, and stores the robot velocities for that situation.

In the Bayes networks shown before, every probability distribution was discrete. However, it seems more natural to think of a velocity as a continuous variable, with an unknown distribution. Since after a normal learning phase there are typically around 100 velocities stored for any given situation, by the Central Limit Theorem of probability we can assume that those distributions are normal (or Gaussian).

Those distributions are uniquely determined by the mean value and variance. When a remote operation ends, the robot will estimate those values using their maximum likelihood estimators, which are the sample mean and the sample variance.

2.3 Autonomous Execution

There is a general process the robot must follow during execution. Each laser scan is the input of the Bayes network which is transformed into the Distance and Angle variables. These

variables are used to make a classification into a "discrete" situation which has associated a normal/Gaussian distribution modeling the robot translation and rotation velocities that are the results of the probabilistic inference algorithm proposed. Although inference is in general a NP-hard problem in a Bayes network, in our solution the inference is done efficiently thanks to the simplicity of our network. For clarity, the process is detailed as follows. After doing a complete 160 degree laser scan, the Bayes network determines the current situation for the robot by computing Distance and Angle. That situation has a corresponding Gaussian distribution associated with each velocity, and it randomly generates a value for each distribution. Then it simply adapts the velocities of rotation and translation to match those values. This simple process is repeated several times each second, so the robot adapts quickly when the situation changes. The robot was taught to stop when it reaches the goal configuration, meaning that the goal configuration must be different from the configurations the robot can find along the path (otherwise the robot might stop halfway, or continue advancing when the goal is reached). In our test cases, the goal configuration was chosen to be near a wall or corner, facing the wall.

The robot's behavior mimics the behavior that was shown during the learning phase. So in order to change the desired behavior, for example passing from a behavior that simply dodges obstacles to a wall-following behavior, an entirely new learning data must be provided. Furthermore, to change the robot's environment (the map) without changing the desired behavior may also require a new learning pattern, depending on the behavior. In our case, the behavior, going to a desired configuration without colliding with obstacles, is highly map dependent, so for each different environment there will be an independent learning phase. However, some behaviors such as wall-following could have the same learning data independent from the map. Furthermore, independently of the map and the task, some learned behaviors maintain their validity. For example, when the robot learns to turn when it is near a corner, this knowledge transfers from map to map. In any case, note that the control execution itself does not need any modification, and once the proposed approach was implemented, the only thing needed to "*program*" a new behavior is remotely operating the robot to gather data needed by the learning phase. It is a task that anybody can do (and not just robot experts).

3. TEST CASES AND RESULTS

The following subsections present the mobile robot used to make tests, the test cases designed to validate the proposed method, and the results obtained with these test cases under an experimental environment using the available mobile robot.

3.1 Mobile Robot Specifications

The robot used to test the proposed method is a differential mobile robot (i.e. it can be controlled either by setting the velocities to each wheel or by setting the instant translation and rotation velocities) Pioneer P3DX (MobileRobots, 2008) (Figure 6): a 30x30 cm^2 robot, equipped with an URG laser capable of doing a 160 degree sweep in front of the robot, and returns the distance from the nearest obstacle in each angle scanned. This distance is an integer between 0 (collision) and 3995 (obstacle far away), and each number represents the distance in

millimeters. Note that if an obstacle is farther than 3995 mm, the distance given by the laser is still 3995 mm.

Figure 6. P3DX mobile robot equipped with an URG laser rangefinder

3.2 Test Cases

Three scenarios were designed to test the motion learning method. In the first and second scenarios, learning consisted in an operator commanding the robot from an initial configuration to a target configuration, where a configuration defines position and heading. Then, the robot was launched, and when it got lost, the operator took over from that configuration to the target configuration. Only the data acquired in this process are used, and in particular no odometry queries are done. In all cases, the learning phase was done in simulation.

In the first scenario (Figure 7), the robot starts at the initial configuration q_0 (upper left corner heading to the south) and must go to the final configuration q_f (lower right corner heading to the south). Many tests were done, varying the robot's initial coordinates and initial heading (but the robot is never placed while directly facing a wall). The robot never received its initial configuration (position/orientation), and it used exactly the same learning data code to solve the problem from all the different configurations.

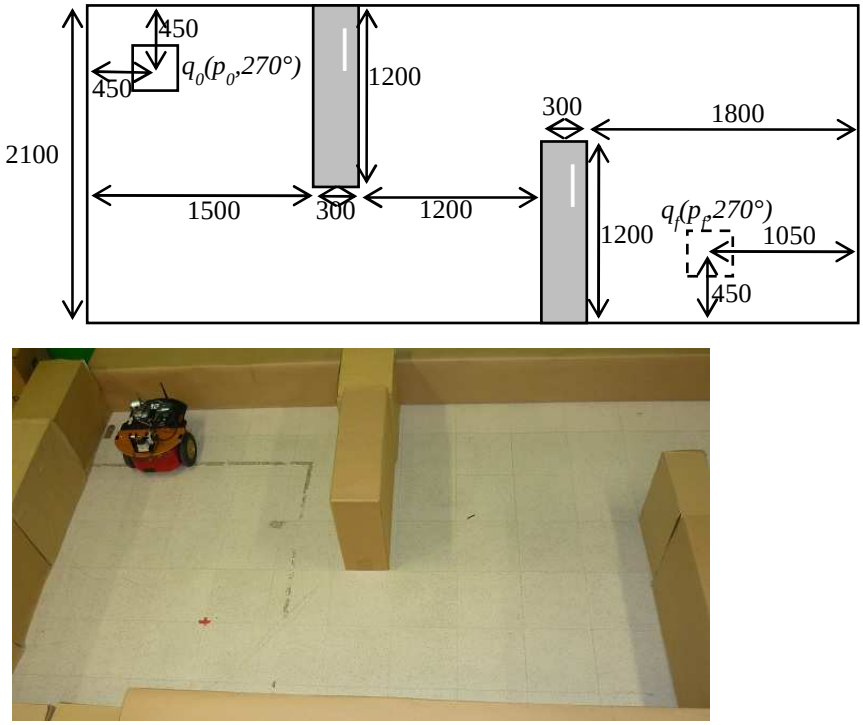

Figure 7. (Top) Scenario 1 (5100 x 2100 mm^2) where the robot starts at q_0 and must finish at q_f. (Bottom) Experimental scenario in the laboratory to make tests.

In the second scenario (Figure 8), the robot starts at q_0 (lower left corner heading to the east) and must arrive at q_f (upper right corner heading to the north). There are two ways to arrive (going below or above the central obstacle), but during learning, the robot was always commanded to take the lower path. So, during an autonomous execution it is expected to take that same path.

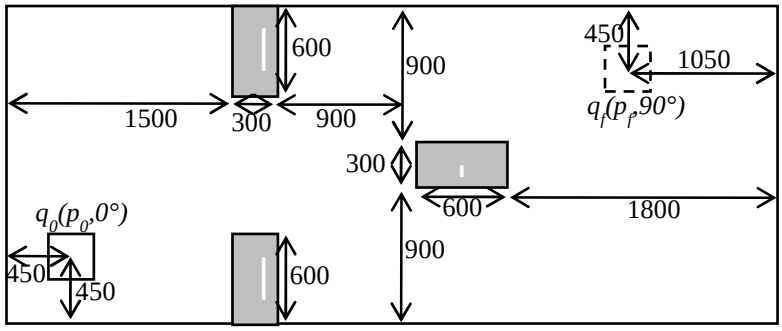

Figure 8. Scenario 2 (5100 x 2100 mm^2) where the robot starts at q_0 and must finish at q_f

In the third scenario (Figure 9), the robot's task is not to reach a destination, but to follow the wall indefinitely. Learning was done similarly to the previous cases, but instead of leading

the robot to the target configuration as done before, the operator just leads the robot until it reaches the initial configuration again. During the learning phase, the robot was only taught how to follow a wall on its left.

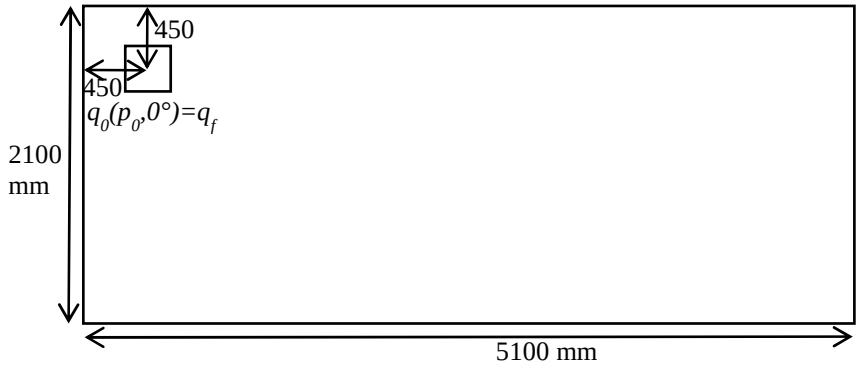

Figure 9. Scenario 3 (5100 x 2100 mm^2) without obstacles where the robot starts and ends at q_0

3.3 Results for Test Cases

In this section we present the method's results. All results are all obtained from execution with the physical robot, despite that learning was done in simulation. They vary depending on the time spent in the learning phase.

Table 1 shows results for the first scenario. Results are quite satisfactory, since the robot manages to arrive at the target configuration almost every time due to the learning process, even after varying its initial configuration. Also worth mentioning is that, despite no odometry data are used, when the robot is lifted and put back at the start without shutting it down, it does not get lost but instead just begins its path anew.

Table 1. Results for scenario 1

Learning time	Behavior
5 minutes	Robot gets lost very quickly.
20 minutes	Robot goes past first obstacle but gets lost afterwards.
40 minutes	Robot goes past first obstacle, turns correctly in order to overcome the second one, but gets lost shortly before doing so.
60 minutes	Robot arrives at destination 95% of the time. If the initial configuration is modified, the robot arrives at destination 80% of the time.
80 minutes	Robot arrives at destination 100% of the time. If the initial configuration is modified, the robot arrives at destination 90% of the time.

Table 2 shows results for the second scenario. We see that the results are again quite good, despite small mistakes. It is interesting to see that robot occasionally takes the upper path, although during learning that path was never explored. What's even more interesting is that it still manages to find its way around half of the times while taking said path.

Table 2. Results for scenario 2

Learning time	Behavior
5 minutes	Robot gets lost very quickly.
15 minutes	Robot manages to exit left side of scenario, but gets lost afterwards.
35 minutes	Robot manages to exit left side of scenario, and 50% of the time reaches destination. The other 50% it gets lost along the way.
60 minutes	Robot manages to exit left side of scenario, and 80% of tries it takes the lower path and reaches destination. 10% of times it takes the upper path and still reaches destination, and 10% of times it takes the upper path and gets lost along the way.

Table 3 shows results for the third scenario. Surprisingly, this apparently "easier" behavior required more learning than the previous ones. After 100 minutes of learning, the robot manages to follow the wall successfully, but the optimal behavior of following the wall in a straight line could not be obtained, and the robot follows the wall in zigzag. The reason is that the model used to describe the laser detections (the way we compute Distance and Angle) was thought to resolve another problem, and it is not the best model to follow the wall. Instead, a new model that takes more into account detections at the sides of the robot, with little focus on the front, would be more adequate. Nevertheless, it is worth noting that despite using the same model we used for path planning, the robot does manage to follow the wall.

Table 3. Results for scenario 3

Learning time	Behavior
5 minutes	Robot turns away from wall.
15 minutes	Robot follows the wall* before arriving at corners. However it stops when it reaches corners.
40 minutes	Robot follows the wall* before arriving at corners. When it reaches a corner, it stops and begins turning. 50% of times it turns correctly and continues following the wall, 50% of times it stops halfway through the turn.
100 minutes	Robot follows the wall* before arriving at corners. When it reaches a corner, the robot stops and turns correctly, and continues following the wall.

* Robot does not follow the wall in a straight line, but it alternatively approaches then moves away from the wall

4. CONCLUSION

In this paper a motion imitation approach was designed and implemented for a differential mobile robot using a Bayes network and learning techniques. It was tested with the physical robot in different scenarios of average complexity, and results were quite good.

In comparison with other alternatives, our method's strengths are:

- Only two algorithms, inference and learning, are used to solve different kinds of problems. These algorithms are general and do not depend on the problem.
- Once the probabilistic component is implemented, it is enough to modify the learning data in order to change the robot behavior, and no code changes are need in the system.
- To solve a problem that generally requires coordinates and geometric space location, we use only local sensory information (laser detections). In particular, no odometry data is required.
- Robot does not need memory. It is never instructed to remember what it was doing previously (like turning) to determine its best action.
- The method is robust against inaccuracies (like varying the robot's initial configuration without notifying it, or manually changing its configuration during execution).
- To "teach" the robot a behavior, no programming knowledge is required. Remotely operating is enough.
- Programming the robot is a more intuitive task, being easier to show the robot the desired behavior instead of having to translate that behavior into the robot's language.
- It is a generic method for solving different problems. The robot can be configured to wall following or to go to the target configuration using the same program and the same Bayes network.

However, using this method also has drawbacks that cannot be ignored:

- A significant learning period could be required for complex tasks.
- The method is not complete. Occasionally the robot fails to solve the task.
- It is difficult to find mistakes. If the robot's behavior is not as desired, it is hard to detect what fragment of the learning data is responsible, and the operator must restart the learning process from scratch.
- This method depends heavily on both the Bayes network used and the data simplifications. Moreover, there is no tool that determines for sure what network and simplifications will have the best results. Instead, one must proceed by trial and error until a suitable model is found.
- In this case, the Bayes network used is very simple and is not capable of solving more complex problems which may arise in mobile robotics. However, it is possible to design much richer networks (for example, one that includes odometry data) to cope with those problems.

We would like to insist again on one of the most significant features of our solution: the robot does not use odometry data. This is really surprising, because the robot manages to reach a destination without really "knowing" its current position, or where the target position is located. We will use an analogy with human behavior: in traditional motion planning solutions, the robot's thoughts are "I am at GPS position (24.56, 30.32) and I need to arrive at GPS position (27.40, 25.32), but I detect an obstacle at (26.77, 28.09)"; in our solution, it looks more like "when I find the tall building, I will turn left, and when I find the dead end, I will know that I have reached the goal". The latter is readily seen to be much more natural (for a human).

5. FUTURE WORK

The proposed method in this paper is not directly applicable to real life robotics problems, since its purpose was to explore the possibilities that probabilistic networks offer in Robotics. However the method can be built upon to implement it in a complex application. In this section, we provide a few ideas for extending the work presented.

The basis and most important aspect for the method is the underlying Bayes network. We presented just one possibility, but it is by no means the only "correct" one. Other Bayes networks should also be tested, using different simplifications and variables, and perhaps even including odometry data. An alternative would be to extend the network we proposed with two new variables that can be estimated during execution using odometry. One is Current Configuration, having no predecessors and with the robot velocities and the detections in each cone as its successors. The other one is Goal Configuration, a predecessor for the robot's velocities. The resulting network is more difficult to handle, but it provides more accuracy and power.

In this paper, there are only two problems tackled: motion planning and wall-following. There are many more problems in Robotics that can be solved with a probabilistic approach as shown in this paper. Different Bayes networks for different problems should be designed and implemented.

While in this paper a pure probabilistic approach was used, totally based on machine learning, this method is compatible with others alternatives. In particular, a hybrid approach could be developed, using both deterministic and probabilistic behaviors, some given a priori and some acquired through learning. Our method is able to apply only a behavior to perform a task (e.g. motion planning), but a solution produced by mixing behaviors (e.g. pushing obstacles, object recognition, doing localization among many other possibilities) could equally be constructed.

In any case, we do believe that there is plenty of room to use probabilistic networks in Robotics in an innovative way. Those networks are powerful tools for reasoning under uncertainty, a characteristic of most situations. Their applications should therefore be fully developed.

REFERENCES

Antonelo, E.A. et al, 2008. Imitation Learning of an Intelligent Navigation System for Mobile Robots using Reservoir Computing, *Proceedings of the IEEE 10th Brazilian Symposium on Neural Networks*, Salvador, Brazil, pp. 93-98.

Binder, J. and Koller, D., 1997. Adaptive Probabilistic Networks with Hidden Variables. *Machine Learning*, Vol.29, pp 213-244.

Burnside, E. et al, 2009. Probabilistic Computer Model Developed from Clinical Data in National Mammography Database Format to Classify Mammographic Findings, *Radiology*, Vol. 251, No. 3, pp. 663-672 [Online] Available at: http://radiology.rsna.org/content/251/3/663.full

Drakos, N. and Moore, R., 2010. Machine Learning B - Approximate Inference in Bayesian Networks [Online] Available at: http://www.igi.tugraz.at/lehre/MLB/WS10/MLB_Exercises_2010/node15.html

Gagliardi, F., 1998. JavaBayes v0.346. [Online] Available at: http://www.cs.cmu.edu/~javabayes/Home/

Hogg, R. et al, 2005. *Introduction to Mathematical Statistics (sixth edition)*. Pearson, New Jersey, USA.

Knuth, D., 1997. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Reading, Massachusetts, USA.

Korb, K.B. and Nicholson, A.E., 2004. *Bayesian Artificial Intelligence*, CRC Press, Florida, USA.

La Valle, S., 2006. *Planning Algorithms*. Cambridge University Press, Cambridge, USA. [Online] Available at: http://planning.cs.uiuc.edu/

Lebeltel, O. et al, 2004. Bayesian Robots Programming. *Autonomous Robots*, Vol. 16, No. 1, pp 49-79.

MobileRobots, 2008. Adept MobileRobots Inc. [Online] Available at: http://www.mobilerobots.com

Ollis, M. et al, 2007.A Bayesian Approach to Imitation Learning for Robot Navigation, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, USA, pp. 709-714.

Uffe, B. and Anders, L., 2008. *Bayesian Networks and Influence Diagrams. A Guide to Construction and Analysis*. Springer, Spring Street, New York, USA.

Zhou, H. and Sakane, S., 2007. Mobile Robot Localization using Active Sensing based on Bayes Network Inference. *Robotics and Autonomous Systems*, Vol. 55, No. 4, pp. 292-305.