High level assisted control mode based on SLAM for a remotely controlled robot

Jean Clement Devaux, Paul Nadrag, Etienne Colle, Philippe Hoppenot

High level assisted control mode based on SLAM for a remote controlled robot

Jean-Clément Devaux, Paul Nadrag, Etienne Colle and Philippe Hoppenot[1]

***Abstract*— Our aim in ambient assistive technologies is to reduce long term hospitalization for elderly people, especially with pathologies such as Mild Cognitive Impairment (MCI). The smart environment assists these people and their families for safety and cognitive stimulation so they stay as long as possible at home. The originality comes from using the robot in the elderly person's home. This robot is remote controlled by a distant user, a therapist or a relation, for determining alarming situations or for participating in stimulation exercises. Several modes are possible for controlling the robot. This paper deals with an assisted control mode in which the remote user gives to the robot one goal then the robot reaches the goal by itself. During the robot movement, the user can dynamically change the current goal. An important hypothesis is that the robot has no *a priori* knowledge on its environment at the beginning. The knowledge will increase with time and the planned trajectory will be refreshed at two levels: a local one -faster but not always sufficient- and a global one -slower but which always finds a path if one exists-. The idea is to work only with local information, using the robot sensors, the operator keeping the high level control. To assure that control, the remote operator uses a video feedback and information from a laser range.**

I. Introduction

THE increasing number of elderly people, especially with pathologies such as Alzheimer disease, is becoming an important issue in Europe. It is more and more difficult and expensive to assure long term hospitalization for these people, so they stay at home as long as possible. There are two main advances to make this possible: safety of the person and cognitive stimulation. The aim of the European CompanionAble project is to assist the people with Mild Cognitive Impairment (MCI) and their families in those situations, in the context of ambient assisted living. The purpose of the robot isn't to remove the human presence around the person, but to ease his caring. Decreasing this presence is a clearly expressed wish, in a quite comprehensible will of intimacy for the person and in a will of assistance for the family. Concerning safety, the idea is to detect alarming situations by monitoring the person. The principle is to measure certain physiological data and the activity of the person. Other sensors are placed in the environment which can also give information on the person activity. With regard to cognitive stimulation, different kinds of exercises can be proposed, to limit as much as possible the progression of the person's disease.

The originality comes from using the robot in the person's home. This robot can be used either on safety aspects or on cognitive stimulation aspects. Several control modes are possible for the robot. According to Sheridan [1,2], there is a continuum from the robot being completely controlled by a human to the robot being completely autonomous. Between these two control modes, a large panel of assisted modes can be developed.

This paper deals with an assisted control mode in which the remote operator gives to the robot a goal and the robot reaches the goal by itself. During the movement of the robot, the operator can dynamically change the current goal. In fact, the idea is to give to the remote operator a high interaction level with the robot. An important hypothesis in this work is that the robot has no a priori knowledge on its environment. The idea is to work only with local information, using only the robot's sensors, the operator keeping the high level control. To assure this control, the remote operator has video feedback and information from a laser range scanner.

For the robot, reaching the goal given by the operator requires three capabilities: obstacle avoidance, planning and localization. The idea is to use a SLAM technique in our specific situation of medium term navigation without a priori knowledge on the environment for planning and with an odometer not sufficiently precise. We are not proposing a new SLAM algorithm but using one of them.

The next section (II) deals with related works on SLAM and planning in unknown environments. Section III presents our algorithm and section IV gives experimental results. A video is also given.

II. Related works

For our needs, the robot must create a plan of the environment while keeping track of its coordinates in it. It must also reach the point specified by the user, in the minimum amount of time possible. Because the target point can be outside the acquired plan (initially, a single laser scan), the robot must be able to plan in an unknown environment. When a dead end is detected, it must re-plan a new trajectory to the target point, if such a trajectory exists, or inform the user that the target point is outside the reachable space.

A. SLAM

In order to complete the navigation task, or, on the

Manuscript received September 15, 2010. The research leading to these results has received funding from the European Community's Seventh Framework Program ([FP7/2007-2013]) under grant agreement n° 216487 (CompanionAble: http://www.companionable.net/).

[1]J.-C. Devaux, P. Nadrag, E. Colle and P. Hoppenot are with IBISC, University of Evry, France (e-mail: name.surname@ibisc.univ-evry.fr)

contrary, to detect that it is impossible to, the robot needs to have a plan of the environment and to be able to pinpoint its location in it. As our starting hypothesis is that we don't have a plan of the environment, the robot must construct one. Together with the localization part, this is the SLAM problem. In the literature, there are many practical solutions to solve this problem, coming under various domains. They can be outdoors or indoors, on the ground, in the air or underwater. In [3] Durrant-Whyte and Bailey present a brief history of the SLAM problem and the ways tried for solving it. They also present the established solutions: one based on the extended Kalman filter (EKF-SLAM), the other on the Rao-Blackwellised particle filter (FastSLAM). A list of SLAM algorithms with open source implementations in various programming languages is given. Datasets available online, on which one can test its own implementation or algorithms, are also mentioned. A practical solution is proposed by Thrun in [4]. The robots used are mobile platforms, dedicated to navigation tasks indoors. Distance sensors (ultrasonic or laser) are used to detect walls, used in the localization of the robot. A reasonable hypothesis used by the author is that the walls have a global orientation that is a multiple of 90°, or their orientation is different with at least 15° from the global wall orientation. A wall is labeled as such is five or more adjacent sonar measurements form a straight line. The same idea is also applied for a laser range finder, with more accurate result, thanks to the sensor used. A museum guide robot is presented in [5] by Thrun et al. It uses a laser range finder to create an occupancy map of the museum and a vertical black and white camera to create a texture map of the museum's ceiling. The robot performs Markov localization and occupancy grid mapping, as in [4]. This takes the form of an expectancy maximization cycle. In [6] the SLAM algorithm fuses the laser with the sonar sensors, in order to provide more reliable results (such as corner detection). The SLAM is based on an EKF and implemented in Matlab, with all the calculations done offline. For validation purposes, the robot was manually driven. In [7] quadtrees are used for storing the acquired map. By using a weighted scan matcher, the latest scan is compared against all the past scans stored as a quadtree. A comparison with SLAM approaches (Carmen) is also given, and the results for three datasets show that the scan matching with quadtrees gives promising results (small offset). The SLAM algorithm presented in [8] retained our attention because it was designed to be run in real time and the source code is publicly available. It is designed to use the measurements from a laser range scanner. It resides on the use of a particle filter. Because of its assets, we decide to use it as a starting base for our assisted control mode.

B. Planning in unknown environments

Our proposed assisted control mode can be invoked at any moment by the remote operator and the robot doesn't have a plan of the environment recorded Thus, it must start by planning a trajectory to its destination in an unknown environment. An earlier work on this subject is by Elfes [9]. In it, an integrated approach to map construction (represented as an occupancy grid) and navigation is presented. The path planning is thought of as a dual objective function that needs to be minimized. The objectives are the path length and the occupancy probabilities of the cells that are to be traversed. In [10], the authors' robot is an unmanned surface vessel (a boat), which has a general knowledge of its environment. For safely guiding the robot to its destination two obstacle avoidance components are used: a deliberative component (far field) and a reactive component. The deliberative component is responsible for planning a path using an occupancy grid that follows the original planned path as close as possible. The reactive component projects a number of arcs in front of the robot and chooses the one that best combines the following criteria: follow the path given by the deliberative component, avoid the obstacle that are nearby and go towards a free space. The same layered approach (plan globally and react locally) to navigation in unknown environments is found in [11], used with a rover-type robot. The global planner uses A* to compute a new path, when needed. The local planner uses potential fields to act on the steering of the robot (move away from the obstacles and closer to the goal). Instead of the distance between the robot and the obstacles, the time to collision is used by the local planner. In the event that the local planner can't give a result, a local A* planning is employed to guide the robot out of that hazardous area. In [12] test results with a real robot are presented. The robot is able to navigate in its environment even in the absence of an initial plan. Two planners are present, a global and a local one. D* is employed for rapid planning. The possible future position of the robot (as arcs) is evaluated by the local planner and a choice is expressed. A trial run of 1.4 km is presented, with satisfactory results. [13] is a development of [12]. The improvement is that the map is no longer stored as a regular grid, but as a framed quadtree, which results in memory gains. Test results with a real robot are also presented.

III. Proposed solution

The aim of this paper is to propose an algorithm capable of planning and navigation in an initially unknown environment. After the goal is selected by the remote operator, the algorithm computes an initial global path. The main idea is that the knowledge of the environment will increase over time up to the completion of the mission and the planned trajectory will be refreshed at two levels: a local one (faster but not always sufficient) and a global one (slower but always finds a path if one exists). The smoothing of the trajectories is also included. This section

describes this algorithm. Obstacle avoidance is supposed to be active at navigation time.

Our navigation strategy involves two levels of action. First, we try to reduce the complexity of the map to reduce computational time and then we supervise planning from the start position to the goal at a high level.

A. Preliminary work - path planning

As described before, we use SLAM technique both to improve the knowledge of robot's position and to get a relatively clean map to be able to find a path to goal. We've chosen a very simple and specific algorithm called CoreSLAM because of its simplicity and efficiency, especially in terms of CPU cost. Initially, the map is empty (or more exactly filled with non-visited space). To define a goal, the operator has just to set a target to reach and the robot plans its path and executes it. In the first step, the map is very poor, only built with the first laser scan. Meanwhile, the SLAM program is running in parallel, updating the robot's position and producing new maps. These maps are certainly partially different than the one at planning time, while moving the robot is discovering areas which could either be free or occupied. It is possible that the initial trajectory becomes invalid and the point is to know how to re-compute a new path saving as much computational time as possible.

First we had to prepare the map to be compatible with the graph algorithm of path planning. As maps can be considered as pictures, we chose to reduce the number of areas of interest regrouping pixels according to their state (free, occupied or unidentified). A well-known algorithm for that type of compression is QuadTrees. Rather than a simple occupancy grid, we chose to reduce the number of nodes and edges of our 8-neighborhood graph computing the QuadTree algorithm on our map. Sure, quality of simplified map with occupancy grid or QuadTrees depends a lot on the minimal size of the nodes. But after a spatial median filter and a dilatation morphology transform, the resulting map is really clean and simple and we can obtain an 8-neighborhood graph, concentrated around important areas like obstacles, reducing again the number of nodes. That decrease of the number of nodes using QuadTree allows us to raise the resolution if needed. Framed QuadTrees are widely used in path planning to improve quality of found path, preventing the problem of a non-shortest and very schematic trajectory (path has to go from a center of a cell to a center of another one). Because framed Quadtrees increase the number of nodes in the neighborhood graph, we've decided to only consider the main and simple Quadtree algorithm and to solve that problem on-line in our higher level algorithm.

The initially chosen path planning algorithm was D* but as we developed our navigation supervisor we progressively neutralized the dynamic part, replacing the updates of edge costs by the action of our high level algorithm without any disadvantage. Thus, we preferred to keep control of the re-planning functionality by considering map updates at a higher level than the dynamic path planner.

B. Planning and navigation supervision

Our algorithm depends on a 2-level planning: global and local. This permits to dramatically reduce the computational time because of the size of the needed map to compute local paths and offers some interesting results according to the resulting trajectory. After all the above preliminary steps, the first thing is to get an initial path to goal, executing the global path planner (path planning algorithm on full map). Thus, the nodes we have at our disposal are equivalent to points on the map, which we have to reach successively to achieve the goal. In fact, we consider these points as areas and because we don't force angles to be exact, the trajectory will be really smoothed (simple and cost-less).

We assume that the operator gives to the robot a reachable goal. If not, the algorithm just has to stop the robot and wait for another order. When a global path is found, because the environment is constantly changing and computing a full path takes several seconds, we chose to compute a local path to try to reach the checkpoint which is the closest to the goal but below the range of detection of our robot. To compute this local path, we first try to reach it by drawing a line between the start point and the checkpoint. If it's not possible (because of the presence of obstacles), we use the same algorithm as for global path but computing only on a sub-map of which the size is equivalent to the range of view. This sub-map is a kind of trusted-area where the trajectories found are pretty certain because directly viewed by the rangefinder and obtained in real time (without stopping the robot). Cutting the map to extract a smaller window and then executing all the process to get a path hardly reduces the time needed to get the graph and to find the path. In that condition this can be realized in real time.

If there is no way to reach any global checkpoint in local window then there are two possibilities. First, the initial trajectory crosses a wall or any obstacle. In this case, the only thing to do is to re-compute a global path using a better-known map. This is detected when the trajectory can't be executed. Second, the next checkpoint is far enough to exceed local window size and then, we just have to follow carefully the global trajectory until we get a checkpoint in the range of view or an obstacle over the predicted trajectory. This occurs only if areas are larger than the local window and then, the areas are free at least for the length of the sub-map so we assume that the robot can move without coming across any obstacle, following the initial trajectory. If an obstacle is finally detected, local resolution all around will be higher and then there will be areas of smaller size, making possible to compute a new

path (locally or globally depending on the situations described above).

Keeping full control on trajectory allows us to add any constraint we decide on the way (how the trajectory has to look like), on the part of the map involved in research, on sequences of sub-goals or user-defined checkpoints... and above all, it gives us the possibility to recover easily and cleverly from all kinds of planning problems. Thus, it would be possible to add as many levels of maps as we want, considering other ranges of sources of information like ultrasound rangefinder or camera for example.

IV. Experimental results

To improve our algorithm, we first used a simulated robot in several situations to illustrate all the capabilities of the algorithm. Three parts of the algorithm are presented in detail (paragraphs A, B and C), and its behavior in the case of an unreachable goal is shown in paragraph D. Experiments using a real robot are also presented at the end of this section (E).

A. Global planning and re-planning

When a goal is set by the operator, the first thing to do is to look for a global path. In most cases if there are some obstacles to avoid, the algorithm is going to find a solution crossing unknown areas of the map because we begin the search in a clear map with just a scan at the starting position. In all cases, the global planner always finds a trajectory, if one exists. In Fig. 1, the robot starts without any previous knowledge of its environment. The first laser scan offers only a small part of map (we only consider as free space the areas between robot and a wall: if no obstacle has been detected, range value would be at maximum value which is also the error constant) like in Fig. 1, left, and then the operator sets a goal behind the wall. The computed trajectory crosses unknown parts of the map just on the right of the obstacle. While robot is following the computed trajectory, it is progressively discovering that the wall is expanding on the right to finally encounter another obstacle which forms an impassable corner. At this stage, it is decided to re-compute the whole trajectory from the current position. As we can see on Fig. 1, right, the computed trajectory is now going on the left of the wall because of the shortest path finder.

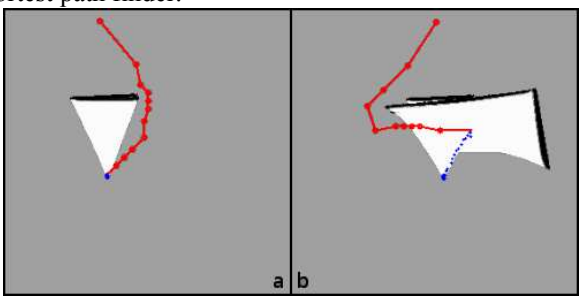

Fig. 1 : Global re-planning situation

B. Local re-planning

As we've said before, our algorithm has the ability to modify a local part of the path to correct an erroneous global trajectory which could cross a wall because of having been computed before knowing there was a wall. This is very useful because global path finder is very costly in terms of CPU time due to the size of the considered map.

Let's turn our attention on Fig. 2. The global path is represented by the zigzagging line while the trajectory of the robot is represented by dots. The robot has already moved in the environment before planning (upper right of the figure), but the map was not totally known. Free areas are in white, occupied ones are in black and not visited ones are in grey.

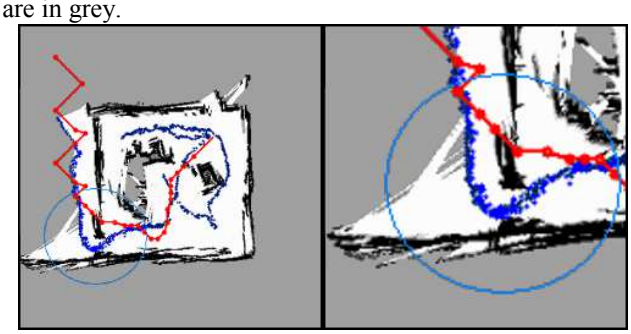

Fig. 2 : Local re-planning situation

In this paragraph, we only focus on the second part of the trajectory, starting on the upper-right corner of map. Considering the real trajectory before the operator targets the goal, we easily can assume that two-thirds (upper right) of the map are visited and the rest is unknown when global trajectory is computed. As we can see, the real path follows quite precisely the computed one during the first part of the trajectory because it corresponds to the well known area where the operator indicated the goal. But while the robot reaches the going up part of the trajectory (after the corner of the gray and black mass in the center of Fig. 2.a), the laser rangefinder discovers a wall just under the global path, involving the search of a local solution to by-pass the problem (Fig. 2.b). We can see the real trajectory strating from the computed path to cross a free space (a door) before going back to the global path. Thus, the robot was able to find a local solution because there was an alternate path near the original one. The robot can avoid every kinds of obstacles as long as another way is in sight (sufficiently near to enter the robot's range of view), without re-computing the whole trajectory. This local search often offers good results when the operator sets the goal behind a wall (if a door exists) or an obstacle of moderate size (most of obstacles in a person's home: chair, table, person…). The main effect is a simple and automatic avoidance keeping in mind the initial goal.

In Fig. 1, there was no way to compute a local solution to avoid the wall and get back to the initial trajectory because the range of view of the robot wasn't containing any free

space to get over the wall. Re-planning is then taking a significant time, forcing the robot to stop, waiting the end of computing. The algorithm needs to re-compute the global path as soon as there is no path to goal in the local window. This often occurs when the robot reaches crowded or highly dynamical areas where it has never been before.

C. On-line research of a better path

Fig. 3 is the same configuration than Fig. 2 but we focus on other parts of the path to reveal trajectory improvements (encircled ones). As we've explained in section III, our supervisor has the ability to improve the computed path before executing it. This is very useful in several cases.

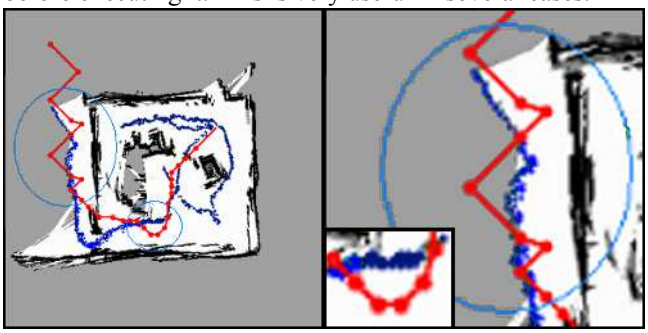

Fig. 3 : Smoother, shorter, better

Firstly, because we choose to use simple quadtrees algorithm instead of framed quadtrees to limit the computational time, the global trajectory looks pretty strange in large areas. The computed trajectory is always between centers of cells but there is no need for the real trajectory to be so restricted. Our algorithm allows us to get a better real path like in the last part of trajectory in Fig. 3. The robot's path is directly avoiding some of the checkpoints to short its path to the goal. The point is to follow a line from robot's position to the farthest checkpoint in range of view. Finally, one can observe another improvement just before the corner where the real path is avoiding a useless bend. Because the rangefinder and our SLAM are not perfect, the computed trajectory was probably trying to avoid an obstacle there but while the robot was approaching, the map was corrected and it wasn't any longer useful to follow that path in that part of the map: the robot could simply move straight forward.

This functionality and our tolerance about positions around a checkpoint to clear it and the angle of robot to start its trajectory allow the robot to move in a smoother way. A complicated path is equivalent to close checkpoints and many little angle corrections. Linear and angular speeds are linked to distance and angle to checkpoint. So not only the path is globally shorter but also the robot is quicker because of its less complicated trajectory.

D. Closed environment

Now, let's examine the case of an unreachable goal. The operator could indicate an erroneous target (when a door is closed for example). So our algorithm has to deal with some unattainable areas. In case of a goal situated out of the bounds drawn by walls all around robot, there are two possibilities, depending on whether or not the robot knows the map sufficiently to determine there is no way to go.

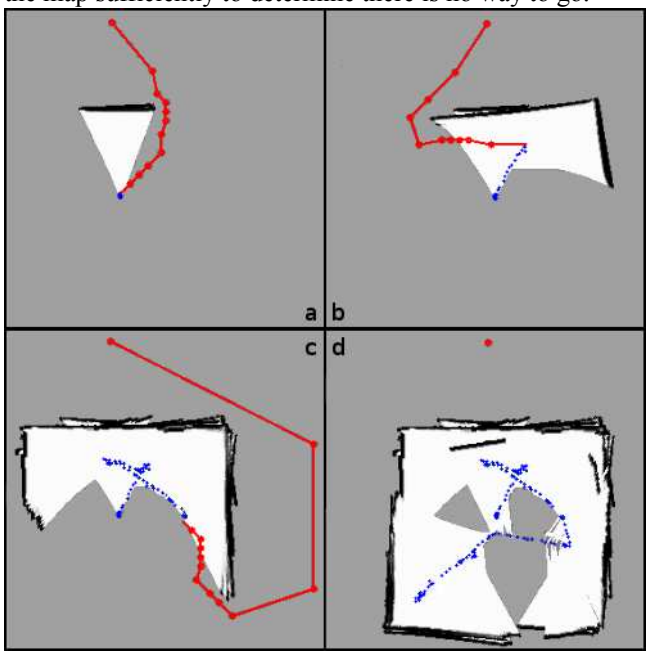

Fig. 4 : Successive trajectories in closed environment

In the first case, the robot is just going to stop, waiting for a new order. In the second case, robot is going to look for a global path which has to cross some unidentified areas as explained in section A and shown in Fig. 4 a, b and c, which represent several of the global computations needed to finish on the situation shown in Fig. 4.d. Those global path searches have to be done again and again until the robot has discovered enough parts of the map to determine that they draw an impassable boundary (because the goal was set outside the closed environment) like in Fig. 4.d (the isolated dot is the unreachable objective) where the walls are closed. It works perfectly to detect a closed environment but if a person opens a door after the robot has visited that area there is a risk of not finding an existing solution until the open door enters in its field of view, which could never happen. In that case, the presence of the human operator can palliate the problem by proposing a new goal near the open door to give the robot the possibility to update the map.

E. Real robot execution

Even if map quality has dropped a little between simulation and reality, these situations can still be observed on a real robot (with odometry drift, rangefinder noise and more constraints on CPU work). Fig. 5 presents two of the four situations described in section IV: shorter and smoother path on the first part of the trajectory and local re-planning on the last part. The quality of the map is

obviously a bit reduced in comparison with simulated maps as demonstrated by the few artifacts behind walls and the thickness of the objects in the map.

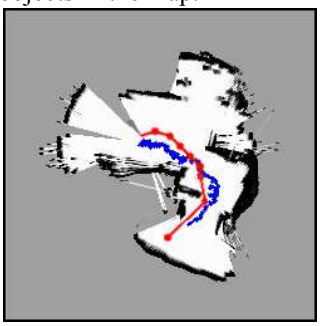

Fig. 5 : Local re-planning & smoother path on real robot

Nevertheless, a global path to goal is easily found from the start position (up and left from center of figure 5). On the first part of the trajectory, one can observe that the robot isn't following the global trajectory because a more distant checkpoint can be reach moving along a line as it was described in section IV.C. The second part of the trajectory is showing a local re-planning to avoid a wall corner which was crossed by the initially computed trajectory because that part of map wasn't discovered at that time. After this little bend, the robot is returning to the global trajectory to reach the goal.

V. Conclusion

The aim of this paper was to propose an algorithm for planning and navigation in an initially unknown environment. A goal being given, the system has to plan the trajectory and navigate up to it. We have proposed a multistage algorithm, taking into account local and global planning, on-line research of better path and detection of impossible planning. All the functionalities of the algorithm have been demonstrated, illustrated on simulation and implemented on a real robot.

Future work on this topic will be on the one hand to deal with other cost functions to minimize (for example the fastest trajectory or the easiest to perform by the robot). A second direction will be to delete from the model given by the SLAM the oldest information. Indeed, it is difficult to keep the map coherent in time. In the case of remote control, we want to give to the operator a control mode in which he just has to supervise the displacement of the robot, but for one mission. A third amelioration will be to deal with quadtree resolution. Indeed, it could be interesting to reduce it for global planning to save time and to make it higher for local planning, which is more rapid, to have a better knowledge on the obstacles around the robot.

References

[1] T. B. Sheridan, Telerobotics, Automation, and Human Supervisory Control. Cambridge, MIT Press, 1992.

[2] T. B. Sheridan, Humans and Automation: System Design and Research Issues. John Wiley & Sons, 2002.

[3] H. Durrant-Whyte, T. Bailey, Simultaneous localisation and mapping (SLAM): Part I the essential algorithms, *Robotics and Automation Magazine*, vol. 13, n°2, pp. 99-110, 2006.

[4] S. Thrun, Learning metric-topological maps for indoor mobile robot navigation, *Artificial Intelligence*, vol. 99, no. 1, pp. 21-71, 1998.

[5] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, D. Schulz, MINERVA: A tour-guide robot that learns, *KI-99: Advances in Artificial Intelligence,* pp. 14-26, 1999.

[6] A. Diosi, L. Kleeman, Advanced sonar and laser range finder fusion for simultaneous localization and mapping, in *Proc. of 2004 IEEE/RSJ Intl Conf on Intelligent Robots and Systems*, pp. 1854-1859.

[7] A. Visser, B. A. Slamet, M. Pfingsthorn, Robust weighted scan matching with quadtrees, in Proc. of the 5th Int. Workshop on SRMED 2009.

[8] B. Steux, O. El Hamzaoui: CoreSLAM : a SLAM Algorithm in less than 200 lines of C code, Mines ParisTech - Center of Robotics, 2009.

[9] A. Elfes, Robot navigation: Integrating perception, environmental constraints and task execution within a probabilistic framework, *Reasoning with Uncertainty in Robotics*, pp. 91-130, 1995.

[10] J. Larson, M. Bruch, J. Ebken, Autonomous navigation and obstacle avoidance for unmanned surface vehicles, in *SPIE Proc. 6230: Unmanned Systems Technology VIII*, pp. 17-20, 2006.

[11] B. Hamner, S. Singh, S. Roth, T. Takahashi, An efficient system for combined route traversal and collision avoidance, in *Autonomous Robots*, vol. 24, no. 4, pp. 365-385, 2008.

[12] A. Stentz, M. Hebert, A complete navigation system for goal acquisition in unknown environments, in *Autonomous Robots*, vol. 2, no. 2, pp. 127-145, 1995.

[13] A. Yahja, S. and Singh, S. and Stentz, An efficient on-line path planner for outdoor mobile robots, in Robotics and Autonomous systems, vol. 32, no. 2, pp. 129-144, 2000.