A Navigation System for a High-Speed Professional Cleaning Robot

Gorka Azkune[1], Mikel Astiz[1], Urko Esnaola[1], Unai Antero[1], Jose Vicente Sogorb[2], and Antonio Alonso[2]

[1] Industrial Systems Unit, Tecnalia, Donostia, Spain
[2] Acciona R+D, Madrid, Spain
gorka.azcune@tecnalia.com

Abstract. This paper describes an approach to automate professional floor cleaning tasks based on a commercial platform. The described navigation system works in indoor environments where no extra infrastructure is needed and with no previous knowledge of it. A teach&reproduce strategy has been adopted for this purpose. During teaching, the robot maps its environment and the cleaning path. During reproduction, the robot uses a new motion planning algorithm to follow the taught path whilst avoiding obstacles suitably. The new motion planning algorithm is needed due to the special platform and operational requirements. The system presented here is focused on achieving human comparable performance and safety.

1 Introduction

A new approach for autonomous floor cleaning is presented in this paper. The goal is to achieve a high-speed robot capable of working in conventional environments, aware of obstacles. Many attempts have been made to solve this problem, but none has yet provided a suitable solution from an industrial point of view, in terms of safety, performance and cost-effectiveness. The navigation system presented in this paper addresses both challenges by combining the most appropriate methods in the literature with a new motion planning algorithm, adapted to the operational requirements of professional cleaning.

For practical reasons, the robot must work reliably with no previous knowledge of the environment. No map will be available, and no additional infrastructure is allowed. These constraints have supported the adoption of a teach&reproduce strategy: a human operator will teach how an area must be cleaned, and the robot will need to follow the same cleaning path as strictly as possible.

One of the most challenging problems consists of the physical constraints of these machines, particularly their weight (about 400Kg), their geometry (non cylindrical), and their working speed (above 1 m/s). Research efforts have been focused on obtaining a human-like behavior on both wide and narrow spaces, without compromising safety requirements. A new motion planning algorithm has been developed for this purpose, given that existing solutions do not satisfy

R. Groß et al. (Eds.): TAROS 2011, LNAI 6856, pp. 24–35, 2011.

Fig. 1. The modified Tennant T7

all requirements. The main goal has been to follow taught paths with high precision, obstacle avoidance, and performing close-to-wall cleaning with minimum performance degradation.

The perception problem has been clearly separated from the navigation system, that is, the method to merge multiple sensor sources and build a consistent representation of the world being observed. Working on this representation allows different hardware configurations to be tested with minor software changes. In the presented experiments, three laser scanners have been mounted, as a standard proposal for conventional environments. For specific functional and cost requirements, this could be scaled up or down according to application needs.

This paper is organized as follows: in section 2 related work is analysed; section 3 gives an overview of the problem description and the implemented solution; section 4 explains in detail the new motion planning approach developed for the cleaning application; the results of the implemented solution can be seen in section 5 and to finish, section 6 contains the conclusions and future work to be done.

2 Related Work

Floor cleaning automation has been one of the application domains that has attracted the research interest of many roboticist in the past two decades. Many robotic solutions have been proposed, the most relevant of each time can be found in several surveys [13] [4] [7]. Floor cleaning machines can be divided in two categories depending on their use: (i) home cleaning machines and (ii) professional cleaning machines. The work presented in this article is oriented to professional cleaning.

During the nineties many industrial cleaning robots appeared in the robotics scene, some of which were developed up to commercial product state. For localization, some needed modifications of the environment in the form of landmarks (C100 of Robosoft, or Abilix 500 of Midi Robots) or magnetic field guides (Auror, Baror and CAB-X of Cybernetix). Others were able to localize themselves without requiring any modifications of the environment (RoboKent of ServusRobots, ST82 R Variotech of Hefter Cleantech/Siemens). Siemens developed a commercially available navigation system, named SINAS, oriented to cleaning robots [9].

Regarding to the operation mode, they could be operated in manual mode and in autonomous mode. For many of these solutions cleaning trajectories were programed in teach-in mode, where the robot learns the cleaning trajectories from a person guiding it (BR700 of Kärcher, C100 of Robosoft, Hako-Robomatic 80 of Hako/Anschütz).

Perception and localization techniques for robot navigation have now reached a mature state. These aspects have been important for the development of a new generation of industrial cleaning robots. Representative is DustClean, one of the robots developed under the European Community financed project DustBot [11], a robot, in prototype stage, built to autonomously clean streets. The companies Subaru, with their floor cleaning robots Tondon and RFS1, and Intellibot, with their GEN-X cleaning robot product line [8], were able to employ their robots for real everyday operation. Another example can be found in the company Cognitive Robotics, which developed the CRB100 [1] system thought to convert ordinary mobile machines into robots. They have focused their activity on professional cleaning automation.

As far as navigation concerns, a lot of obstacle avoidance methods can be found in the literature. The most similar ones to our approach are probably the Vector Field Histogram [2] (VFH+) and the Dynamic Window Approach [6] (DWA). VFH+ uses polar histograms to process sensory input similar to our approach, while DWA forward simulates the trajectories of selectable motion commands to evaluate their suitability given an environment. However, both algorithms have a lot of parameters to configure, which makes harder the set up of a robot. Additionally, none of them could match the requirements of the cleaning process identified in this paper with a unique set of configuration parameters, which suggests the implementation of a specific algorithm, as shown in section 4.

3 Problem Description and System Overview

The problem being faced consists of reducing the costs of floor-cleaning by means of automation while maintaining speed, safety and process quality (70% of the cleaning costs are attributable to labor costs according to [7]). Several key requirements have been gathered from industrial partners: the need to avoid changes in the cleaning process, hardware solution based on a well-proven cleaning machine, the compatibility with traditional operator profiles, equivalent or improved cleaning performance, and above all, absolute safety. A deeper analysis has highlighted the need of the features listed below.

- The **solution must be human-comparable** in terms of driving style and speed. A predictable behavior is essential due to the presence of humans in the working space, which suggests a deterministic control strategy. Besides, the physical constraints of these machines represent a key challenge of the problem, mainly because of the geometry and the weight of the machine and the relatively high working speed (between 1 and 2 m/s). The solution is expected to deal with such constraints while keeping absolute safety conditions.

- The **navigation system must be independent of the environment** (no previous knowledge or environment map will be available), and must require no extra infrastructure. Typical working environments include both wide areas and narrow spaces with the presence of unknown obstacles. These factors should not influence considerably the performance of the robot in terms of precision and working time, compared to a human operator. In addition, the ability to clean close-to-wall spaces is strongly requested, given their relevance in the cleaning process.
- Other industrial requirements also play an important role, particularly **robustness, ease-of-use, and cost-effectiveness**.

The development presented in this paper has been strongly influenced by these drivers. One of the goals has been to test the solution in an operational environment, as a first step in the validation process of a pre-industrial prototype which is based on a common industrial cleanning machine, the T7 of Tennant. Sensor, motion and processing hardware has been incorporated to make the machine capable to perceive the environment and to move autonomously.

The geometry of the Tennant T7 has a rectangular footprint. Its dimensions are 1.52x0.82x1.27 (length x width x height) in meters. It is a car-like vehicle, with its associated mobility restrictions. The weight of the Tennant T7 is 265 kg. and has a water deposit of 110 liters. This makes a maximum total weight of 375 kg. when the tank is full. Two Hokuyo URG-04LX have been mounted at ankle-height at both sides of the machine. Additionally a Hokuyo UTM-30LX has been mounted on the machine at chest-height (see Fig. 1).

The Tennant T7 can be operated in manual mode, where the platform is controlled by a human operator and in autonomous mode, where the platform is controlled by the PC. One relay system controlled by the Technosoft IDM640 makes this possible. A safety watchdog system has been implemented through this motor controller. If no speed commands are received in 0.5 seconds, the controller switches to manual mode.

A general overview of the implemented solution presents several parts that will be roughly explained. The *perception for obstacle avoidance* merges sensor data coming from different sources in a probabilistic occupancy grid, which provides a consistent observation of the space surrounding the robot (see [3] and [12]). For *localization* purposes a Monte Carlo localization algorithm (see [5]) has been used, combining odometry and laser data to track the position of the robot. The *teaching by showing* module records the taught path using a sequence of odometry poses –position and orientation, without speed–, each of which will have an associated local map of the environment consisting of a set of geometric features, namely corners and planar walls, extracted from the laser scanner readings. During teaching, the machine will be driven manually, just like under ordinary operation. Finally, for the reproduction of the taught path, a new motion planning algorithm has been implemented. Section 4 contains a deep explanation of the mentioned algorithm.

4 Motion Planning

In the context of a teach&reproduce approach, the problem of motion planning consists in calculating the suitable motion commands given a taught path and some special constraints due to (i) the platform, (ii) the operation process and (iii) the user requirements.

Platform constraints cover geometric, kinematic and dynamic constraints of the Tennant T7. The most relevant ones are the rectangular footprint of the platform, the low angular speed of the steering wheel and the low braking rate due to the weight. **Operational constraints** refer to a set of navigation features that must be fulfilled while cleaning: navigation on slippery surfaces, fidelity to the taught path, reasonable speed during close-to-wall navigation, navigation through narrow spaces and avoidance of non-expected obstacles. **User constraints** refer mainly to safety and ease-of-use, which implies an intuitive, deterministic and easily configurable motion behavior.

The requirements of the problem described were found to demand a new approach for motion planning. The new approach presented here is founded in a two layer architecture: the first layer is a path follower that generates motion commands without considering any obstacle; the second layer is a new reactive obstacle avoidance algorithm. At a first stage, a standard PID controller for the steering angle has shown to be enough for path following. The steering angle generated by the path follower is used by the obstacle avoider to calculate new speed and steering angle commands that ensure a safe path through obstacles, fulfilling the platform, operational and user constraints enumerated above.

To understand how the motion planning algorithm deals with those requirements, three different solution sets are identified inside the space of all possible solutions. A solution set is a continuous range of steering angles θ and velocities v. Solution sets can be distinguished based on the following conditions: (i) **safety conditions** refer to the minimum and necessary requirements for navigation; if safety conditions are not guaranteed, the platform will no longer move, (ii) **navigable conditions** are those which guarantee conventional navigation requirements, whereas (iii) **desirable conditions** are those conditions that would like to be fulfilled under ideal circumstances. Three solution sets emerge from those conditions: the safe solution set, the navigable solution set and the desirable solution set. The so called solution sets are defined using very intuitive parameters. For example, a (v, θ) solution will be a navigable solution if the trajectory generated by (v, θ) has a collision distance which is greater than a certain value ($nav\_frontal\_clearance$), for a certain lateral clearance ($nav\_lateral\_clearance$). Both values can be configured and set by the user.

The key idea behind the algorithm is to use solution sets to evaluate the safety, navigable and desirable conditions, given any situation. The algorithm will try to find a solution inside a desirable solution set. But if it is not possible, requirements will be brought down to find solutions inside navigable solution sets or even safe solution sets. This way, the algorithm guarantees to find a solution which is as close as possible from the desirable conditions, providing safe navigation when required by environment conditions.

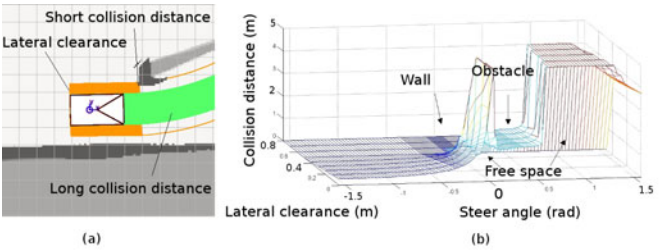

Fig. 2. (a) The occupancy grid generated by the perception component, where a curve trajectory has been drawn for two lateral clearances. (b) PCPH example for the occupancy grid of (a). Positive steer angles represent the left hand side of the platform. Valleys in the diagram represent small collision distances, hence obstacles, as shown by the arrows.

To be able to divide the whole solution space in the three sets defined above, a concrete data structure has been adopted called *platform clearance polar histogram* (PCPH). A PCPH is a data structure that represents the frontal clearance distance to obstacles (collision distance) for each precalculated trajectory and for each lateral clearance distance. So in other words, the idea of PCPH is to represent the collision distance for all the steering angles of the platform. However, this is done not only for the exact footprint of the platform but for different footprint inflations, resulting on a useful world representation, where lateral clearance can be handled easily.

To compute a PCPH an occupancy grid is required. As explained in section 3, sensor information is used to compute such an occupancy grid as the one shown in Fig. 2.a. Additionally, a grid of precalculated trajectories is generated at initialization time. This last grid stores the curve trajectories defined by each steer angle of the platform. So using the occupancy grid and the precalculated trajectories, a PCPH is calculated. Fig. 2.b shows a graphical representation of a PCPH.

One of the major challenges of PCPH is to calculate collision distances in real time at high rates. For obstacle avoidance to work efficiently, a minimum rate of 10 Hz is desirable. An efficient library has been implemented that can perform PCPH calculations at 20 Hz, with a grid-cell resolution of 5 cm, a maximum collision distance of 10 m and a steering angle resolution of 1 degree.

4.1 Steer Angle Selection

The process of steer angle selection is based on the segmentation of PCPH to find safe, navigable and desirable solution sets. Segmentation is the process of finding sets of steering angles that fulfill specific frontal and lateral clearance conditions. The pseudo-code for the algorithm is depicted in listing 1.1.

The pseudo-code shows how the algorithm, at its first stage, tries to find navigable sets. If there are no navigable sets, frontal and lateral clearance are iteratively decreased until a so called safe set is found or safety conditions are

reached without any solution. In that case, the algorithm stops the robot, because safety requirements cannot be fulfilled in the environment. However, if the first stage is successful, i.e. if a navigable set exists, the algorithm continues selecting the best navigable set.

```
float selectSteerAngle(pcph, previous_set)
{
  if (!navigableSetSelection(nav_sets, nav_lateral_clearance,
       nav_frontal_clearance, pcph))
  {
    // There is no navigable set. Find safe sets
    return findSafeSteerAngle(nav_lateral_clearance,
        nav_frontal_clearance, safety_dist, pcph);
  }
  // Some navigable sets exist. Select the best one
  best_nav_set = selectBestNavSet(nav_sets, previous_set,
      target_steer_angle);
  // Increase frontal and lateral clearance to find desirable
       sets
  desirable_sets = selectDesirableSets(best_nav_set,
      previous_set, nav_lateral_clearance,
      nav_frontal_clearance, des_lateral_clearance,
      des_frontal_clearance);
  best_des_set = selectBestDesSet(desirable_sets,
      target_steer_angle);
  previous_set = best_des_set;
  return selectClosestSteerAngle(best_des_set,
      target_steer_angle);
}
```

Listing 1.1. Pseudo-code for steer angle selection

The selection of the best navigable set deserves further explanations. As it can be seen in the pseudo-code of listing 1.1, this is done by the function *selectBestNavSet*. The function uses the list of navigable sets, the set selected in the previous iteration of the algorithm and the target steer angle sent by the path follower. A two step process is run in the function:

(i) Initial selection: there is a key concept in this step: set matching. The algorithm stores in each iteration the set selected in the end of the whole process. In the next iteration, the stored set *previous_set* is used for the initial selection of navigable sets. The algorithm only considers those navigable sets which match *previous_set*. Matching is defined as the overlap of sets and it has been introduced to avoid oscillations in the trajectories of the platform. This solution has shown to be a very effective method in the tests, despite of its simplicity.

(ii) Best set selection: from all the sets that pass the initial selection, choose the one which is closest to target steer angle.

The process described above ensures that the selected set is navigable based on the frontal and lateral clearance parameters. Besides, it also removes oscillations

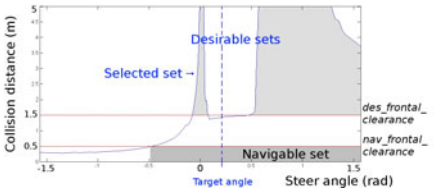

Fig. 3. Segmentation process for a PCPH slice corresponding to Fig. 2; previous selected set is supposed to overlap with both desirable sets

and selects a set that is close to the target steer angle, hence close to the path which is being followed by the path follower.

As navigability requirements are fulfilled, the algorithm attempts to find whether desirable conditions can also be fulfilled. For that purpose, *selectDesirableSets* function iteratively increases frontal and lateral clearance values. The criterion used to select a set is to match both, the selected navigable set and the last selected set. Hence, the function will increase clearance values until no sets can be found or desired clearance values are reached. The sets that survive this process will be called desirable sets. All of them will be as close as possible from fulfilling the desirability conditions and they will match the last selected set, preventing any oscillations.

Finally, the best set of all desirable sets is chosen inside the *selectBestDesSet* function. To select the best set as well as to select the best steer angle inside that set, a simple criterion has been implemented: choose the set and steer angle which are closest to the target steer angle. This simple criterion has shown to work very well keeping trajectories close to what the path follower commands. The most important concepts of the motion planning algorithm and their relations are illustrated in Fig. 3.

4.2 Speed Calculation

There are mainly four aspects that influence speed calculation: (i) the steering wheel latencies, (ii) the dynamic model of the platform, which specifies how the platform behaves under a speed command, (iii) frontal clearance and (iv) lateral clearance. Some assumptions have been done to simplify the dynamic model. The algorithm assumes that the platform acceleration is constant and that the latencies generated by the speed controller are also constant. With those assumptions, all the aspects commented above except (iv) are considered in equation 1:

$$v = \sqrt{2(col\_dist_{ref} - v_c t_L)a_b} \tag{1}$$

where v is the output speed calculated by the algorithm, v_c is the current speed of the platform, t_L is the estimated latency for the speed controller and a_b is the deceleration capacity of the platform. The variable col_dist_{ref} deserves further explanations, since it is the key to handle steering wheel latencies. The idea is to store in col_dist_{ref} the smallest collision distance found between current and

selected steer angle and calculate the speed respect to that value. This strategy has shown to work well for the low angular speed of the wheel, which is around $\pi/4\ rad/s$.

As said above, equation 1 handles all the aspects except lateral clearance, which is important to control the platform speed in narrow openings, close to walls or any other obstacle which is not in front of the platform. A simple and effective way to deal with it is to limit the maximum speed depending on the lateral clearance. Any relation between lateral clearance and maximum speed can be implemented. A linear relation has shown to work very well and it is very easy to configure.

5 Experimental Results

The experimental results section has been divided in two main sub-sections: the first one to evaluate the general performance of the system compared to human operators and the second one to show the performance of the new motion planning algorithm explained in section 4.

General performance of the cleaning machine has been measured by two parameters: (i) the comparison between human operator cleaning time and autonomous cleaning time and (ii) the error between taught path and reproduced path. For the solution to be acceptable from an industrial point of view, autonomous cleaning time and fidelity should be as close as possible from the taught ones. For the environment layout 3.5 meters wide corridors were used. The Tennant T7 is not well suited for narrower corridors due to its size and its poor maneuverability. Respect to the taught path, near optimal cleaning paths for corridors have been taught to the system, composed by long straights followed by 180 degree curves at the end of the corridor that cover the whole cleaning surface. Finally, for the experiments to be fair, the environment is exactly the same at reproduction time as in the teaching stage.

To measure the reproduction time against the teaching time, a total number of 50 paths were taught and reproduced. The shortest path was 50 meters long and the longest one 200 meters. After performing time measurements for each path teaching and reproducing, an average increment of 13.56% has been observed for reproduction time. However, the time increment measured for this kind of environments is remarkable. As a reference, the average speed of the machine while teaching was 1.3 m/s, which is a fast speed for cleaning. Fig. 4.a shows a graphic with all the tests performed.

The second parameter is the fidelity to the taught path. This can be measured as the reproduction error respect to the taught path. The teaching and reproduction tests done in Fig. 4.a have been also used for fidelity tests. After measuring the errors of all the paths reproduced, an average error of 0.09 m was found. The maximum error measured for the motion planning module was 0.32 m. Fig. 4.b shows the error measurements obtained during a reproduction of one of the taught paths. As it can be seen, up to the 85% of the measurements lie between [0.0, 0.2] meters, while only the 5% are bigger than 0.3 meters. These results are encouraging.

Motion planning performance: There are several aspects of motion planning that have to be tested for cleaning: mainly how the algorithm performs very close to walls, whether it avoids oscillations and its behavior in narrow spaces and crowded environments. As far as close to wall behavior concerns, a wall following path was taught to the platform, as close to the wall as possible. After teaching mode, the platform reproduced the taught path autonomously 50 times. The results show an average distance to wall of 0.13 meters (safety distance was set to 0.1 meters), while the average speed still keeps high: 0.7 m/s. Additionally, the resultant trajectory was smooth, very straight and free of oscillations.

As an example of a crowded environment, a slalom was prepared (Fig. 5). Wooden panels were set at 2.5 meters of distance (the Tennant T7 is 1.52 meters long). In a previous stage, the platform was taught a straight path (represented as the red dashed line in Fig. 5.b). The trajectory executed in reproduction mode can also be seen in Fig. 5.b. Localization poses were stored for visualization and analysis purposes. Fig. 5.b shows the superposition of the platform along the trajectory described in the environment of 5.a. The trajectories described by the platform are smooth and free of oscillations. Additionally, the average speed during the slalom was of 0.4 m/s, which, given the conditions, is a suitable speed.

Additionally, some trajectories were taught to enter narrow spaces and navigate trough them. Those situations are not very common for cleaning scenarios with such machines, but the algorithm showed good results. It could approach and cross narrow spaces of 1 meter wide (the Tennant T7 is 0.82meters wide), both in straight trajectories and curved trajectories.

To complete the tests of the motion planning algorithm, comparative tests were performed between the presented algorithm and DWA [6]. DWA was selected because it considers dynamic and geometric constraints of platforms (crucial for our application) and it is widely used. When DWA was configured to strictly follow the path, big obstacles could hardly ever be avoided (notice that tests were performed without any global planner). Besides, close to wall behavior was not satisfactory. The trajectories were not as smooth as desired and the

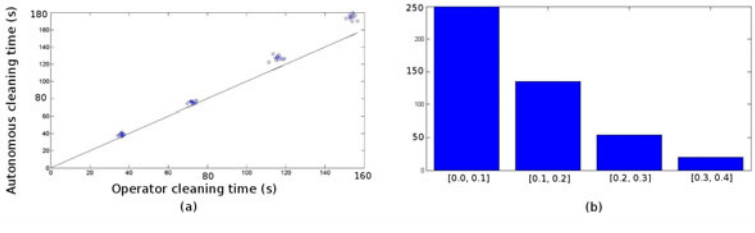

Fig. 4. (a) Comparison between operator cleaning time and autonomous cleaning time; black line represents equality between both variables. (b) Fidelity to the taught path. The histogram shows the number of pose measurements that fall into the error gaps depicted in the x axis. Nearly the 85% of the poses stored in a reproduction are closer than 0.1 meters to the taught pose.

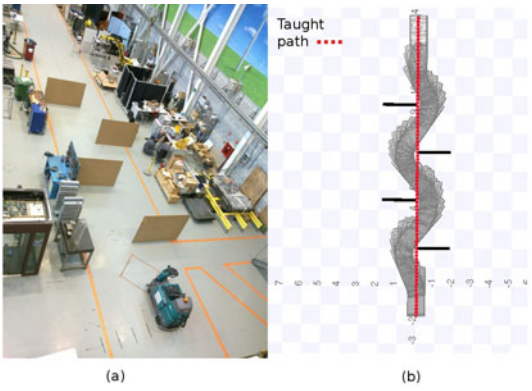

Fig. 5. Motion planning slalom test. (a) shows the environment used to perform the tests: wooden walls wich are located 2.5 meters far from each other. (b) shows the execution of the trajectory recorded in the real environment. Localization poses were stored for visualization and analysis purposes. Average lateral distance to wooden walls is 0.16 m.

average speed could not match our algorithm (0.4 m/s for DWA and 0.7 m/s for our algorithm). In general, it was almost impossible to find a set of parameters that could fulfill all the requirements of the cleaning process.

6 Conclusions and Future Work

An industrial autonomous floor cleaning machine has been presented in this paper. A teach&reproduce approach has been adopted to solve the problem, driven by the requirement that the machine should work in any environment with no extra infrastructure and no previous knowledge of the environment. Additionally the motion planning problem given platform restrictions and cleaning requirements has been addressed in depth. To overcome the problems arisen from those restrictions a new motion planning approach has been introduced, which has shown to work reliably at high speeds without trading off safety and enhancing the cleaning performance of the platform.

As shown in section 5, a lot of experiments have demonstrated that the approach presented here can provide a good solution for industrial cleaning, since autonomous reproduction parameters keep quite close in terms of performance to those of human operators. This human comparable performance and high speed motion does not affect obstacle avoidance. The platform behavior while avoiding obstacles is smooth and suitable.

Additionally, the system proposed in this paper can easily work on top of different hardware configurations. That means that the cost of the hardware can be reduced, for example replacing several laser sensors by ultrasound arrays. As perception is detached from motion planning, sensor replacement should not be a major problem. Indeed, some tests have been done, where ultrasound sensors

were merged with a laser to compute an occupancy grid. The motion planning algorithm does not have to be changed to work on top of this configuration. Besides, the incorporation of a tilting laser can also be approached, to improve obstacle detection and achieve a higher security level. 3D perception is becoming a key feature for navigation applications (see [10]) and does not require any significant change in the system. Those improvements are believed to have important roles to definitely bring professional cleaning robots to market.

References

1. Cognitive robots (2010), http://www.c-robots.com/en/index.html
2. Borenstein, J., Koren, Y., Member, S.: The vector field histogram - fast obstacle avoidance for mobile robots. IEEE Journal of Robotics and Automation 7, 278–288 (1991)
3. Elfes, A.: Using occupancy grids for mobile robot perception and navigation. Computer 22, 46–57 (1989)
4. Elkmann, N., Hortig, J., Fritzsche, M.: Cleaning automation. In: Nof, S.Y. (ed.) Springer Handbook of Automation, pp. 1253–1264. Springer, Heidelberg (2009)
5. Fox, D.: Kld-sampling: Adaptive particle filters. In: Advances in Neural Information Processing Systems, vol. 14, pp. 713–720. MIT Press, Cambridge (2001)
6. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robotics and Automation Magazine 4(1), 22–23 (1997)
7. Hägele, M.: World Robotics, Service Robots 2009 (2009)
8. Intellibot robotics (2010), http://www.intellibotrobotics.com/
9. Lawitzky, G.: A navigation system for cleaning robots. Autonomous Robots 9, 255–260 (2000), 10.1023/A:1008910917742
10. Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B.P., Konolige, K.: The office marathon: Robust navigation in an indoor office environment. In: International Conference on Robotics and Automation (May 2010)
11. Mazzolai, B., Mattoli, V., Laschi, C., Salvini, P., Ferri, G., Ciaravella, G., Dario, P.: Networked and cooperating robots for urban hygiene: the eu funded dustbot project. In: Proceedings of the Fifth International Conference on Ubiquitous Robots and Ambient Intelligence, pp. 447–452. MIT Press, Cambridge (2008)
12. Moravec, H.: Sensor fusion in certainty grids for mobile robots. AI Mag. 9(2), 61–74 (1988)
13. Prassler, E., Ritter, A., Schaeffer, C., Fiorini, P.: A short history of cleaning robots. Auton. Robots 9(3), 211–226 (2000)